Universidad de las Ciencias Informáticas

Facultad 3

Centro CEIGE

Herramienta para la evaluación de la usabilidad en los prototipos de Interfaz de Usuario para los Sistemas de Gestión

Trabajo final presentado en opción al título de

Ingeniero en Ciencias Informáticas

Autores: Irianny Pupo Leyva

Leydis Rojas Elissalt

Tutor: Ing. Sandy Suárez Jiménez

Ing. Yisel Niño Benítez

Co-Tutor: Msc. Iliannis Pupo Leyva

La Habana, junio, 2015

"Los usuarios dedican treinta segundos a leer la página de inicio de una web. En ese tiempo y en pocas palabras, la web debe mostrar lo que ofrece."

Jakob Nielsen

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año_____.

Irianny Pupo Leyva

_____

Firma del Autor

Leydis Rojas Elissalt

_____

Firma del Autor

Ing. Sandy Suárez Jiménez

_____

Firma del Tutor

Ing. Yisel Niño Benítez

_____

Firma del Tutor

Msc. Iliannis Pupo Leyva

_____

Firma del Tutor

Datos del contacto

Datos del tutor:

Nombre: Ing. Sandy Suárez Jiménez

Correo electrónico: ssjimenez@uci.cu

Datos del tutor:

Nombre: Ing. Yisel Niño Benítez

Correo electrónico: ynino@uci.cu

Datos del Co-Tutor:

Nombre: Msc. Iliannis Pupo Leyva

Correo electrónico: ipupo@uci.cu

Autor: Irianny Pupo Leyva

Correo electrónico: ipleyva@estudiantes.uci.cu

Autor: Leydis Rojas Elissalt

Correo electrónico: lelissaet@estudiantes.uci.cu

Dedicatoria

A mis padres por ser las personas que más amo en el mundo .Tomada de sus manos inicie mi aprendizaje en la vida, y les doy gracias por su amor y estar siempre a mi lado cuando más la necesito, todo lo que soy se lo debo por su ejemplo de perseverancia, valor y abnegación.

A mi segundo padre Oscarito, por preocuparse por mí en estos 11 años, por guiarme y apoyarme en esta vida y darme todo sin pedir nada a cambio, por darme su amor y cariño incondicional.

A mi hermana por ser parte importante en mi existencia y brindarme su apoyo durante el tiempo de estudio. Me convertí en su más fiel seguidora. Gracias mi Hermana.

A mi familia por el apoyo que me han dado en todo momento de mi vida, en especial a mis tías Ana y Consi por darme su eterno amor y considerarme más que una hija, a mi tío Ramón por brindarme su comprensión, a mi abuelito Pacheco por preocuparse y quererme tanto y mis primitos hermanos Anita y Tin, y mi sobri Eyli.

A Nelson por brindarme su amor, respeto y amistad, por haber compartido dificultades y alegrías durante todos los años de la universidad, y superando obstáculos para alcanzar un objetivo en común: "conformar nuestra familia".

A la persona que más extraño en este mundo y hoy no está con nosotros, a mi abuelita Mireya la gracia de Dios está sobre ti y me gozo por ello, agradezco tus oraciones, por preocuparte y amarme con tanta ternura y devoción. Donde estés mi abuelita yo sé que me bendices, guarda y protege todos los días de mi existir. Te amo y te amaré siempre.

Iri.

Ha culminado una etapa de mi vida y quiero dedicarla a Mi madre, por ser un ejemplo para mí, a mi familia por estar siempre ahí, a mis abuelos que han sido más que mis padres, han sido mis amigos.

A los amigos que han estado en las buenas y en las malas. Por último a mi novio, por haberme guiado estos 4 años.

Leydis.

Resumen

Un atributo de calidad primordial es la usabilidad de software, puesto que esta influye directamente en la eficacia, eficiencia y satisfacción con que un producto permite alcanzar los objetivos a un usuario. Existen herramientas que evalúan la usabilidad de un sistema, sin embargo contienen un conjunto de limitaciones que atentan contra un resultado final satisfactorio. El objetivo del presente trabajo es construir una herramienta web que evalúe la usabilidad en los prototipos de interfaz de usuarios para sistemas de gestión. La herramienta desarrollada es capaz de devolver un reporte final por: prototipo, funcionalidad, módulo, evaluación y proyecto, donde se muestra un conjunto de elementos relacionados con los mismos y el grado de usabilidad que alcanzan los prototipos pertenecientes a un proyecto específico. Con el desarrollo de la propuesta de solución se garantizan beneficios referentes a la reducción de los costes de soporte a la aplicación. Además al no evaluarse el atributo de calidad en los prototipos de IU de un sistema de forma rápida, podrían generarse riesgos que atenten contra la sostenibilidad y costos de estos, viéndose así afectada la calidad e integridad de los mismos.

Palabras clave: métricas, prototipo, solución, usabilidad, usuario.

ÍNDICE

Introducción **1**

Capítulo 1: Fundamentación teórica **6**

1.1 Introducción **6**

1.2 Conceptos fundamentales **6**

1.2.1 Definiciones de usabilidad 6

1.2.2 Usabilidad en el diseño de Interfaz de Usuario (IU) 6

1.2.3 Usabilidad como elemento funcional 7

1.2.4 Atributos, propiedades y patrones de usabilidad 8

1.2.5 Principios de la usabilidad 10

1.3 Evaluación de la usabilidad **11**

1.3.1 Métodos y técnicas de evaluación de la usabilidad 11

1.4 Estudio de las tendencias actuales **13**

1.4.1 Sistemas internacionales 13

1.4.2 Sistemas nacionales desarrollados en la UCI 14

1.4.3 Resultado del estudio de las tendencias actuales 14

1.5 Metodología de desarrollo de Software **16**

1.6 Herramientas y Tecnologías **17**

1.6.1 Marco de trabajo 17

1.6.2 Mapeo de Objeto Relacional (ORM) 18

1.6.3 Marco de trabajo de creación de diseños web 18

1.6.4 Herramientas para el modelado 18

1.6.5 Lenguaje de modelado 19

1.6.6 Lenguajes de programación del lado del servidor 19

1.6.7 Lenguaje de programación del lado del cliente 19

1.6.8 Entorno de Desarrollo Integrado (IDE) 20

1.6.9 Gestor de base de datos 20

1.6.10 Servidor web 21

1.7 Conclusiones del Capítulo **21**

Capítulo 2: Descripción de la propuesta de solución 22

2.1 Introducción 22

2.2 Descripción de la propuesta de solución 22

2.2.1 Métrica para evaluar usabilidad en prototipo de IU de acuerdo a la importancia de las heurísticas 22

2.2.2 Métrica para evaluar prototipo de IU de acuerdo a los patrones de usabilidad 25

2.2.3 Métrica para evaluar prototipo de IU de acuerdo a la relevancia de incumplimiento de los criterios 26

2.2.4 Criterio de aceptabilidad según el estándar ISO 14598 29

2.3 Modelado de la propuesta de solución 29

2.3.1 Modelo conceptual 29

2.4 Requisitos de la herramienta 30

2.4.1 Requisitos funcionales 30

2.4.2 Requisitos no funcionales 33

2.5 Técnicas de validación de requisitos 34

2.5.1 Prototipado 34

2.6 Análisis y diseño de la propuesta de solución 36

2.6.1 Patrones de diseño 36

2.6.2 Arquitectura del sistema 41

2.6.3 Patrones arquitectónicos 43

2.6.4 Diagrama de clases del diseño 46

2.6.5 Diagrama de interacción 47

2.6.6 Modelo de datos 48

2.6.7 Validación del diseño del sistema 49

2.7 Conclusiones del Capítulo 52

Capítulo 3: Implementación y prueba 53

3.1 Introducción 53

3.2 Modelo de implementación 53

3.2.1 Diagrama de componentes 53

3.2.2 Diagrama de despliegue 54

3.3 Estándares de codificación 55

3.4 Validación de la solución 56

3.4.1 Método de Caja Negra. Pruebas funcionales 56

3.4.2 Método de Caja Blanca 58

3.4.3 Evaluación heurística 61

3.4.4 Evaluación del placer de uso 63

3.5 Conclusiones del capítulo 65

Conclusiones 66

Recomendaciones 67

Referencias bibliográficas 68

Índice de Tablas

Tabla 1. Conceptos fundamentales del Modelo Conceptual. *30*
Tabla 2. Requisitos funcionales. *31*
Tabla 3. Descripción de requisito funcional Realizar evaluación. *32*
Tabla 4. Requisitos No funcionales. *34*
Tabla 5. Descripción de las tablas de la Base datos. *49*
Tabla 6. Clases evaluadas según la métrica TOC. *49*
Tabla 7. Caso de Prueba para el camino básico #1. ..... *60*

Índice de Figuras

Figura 1.Iceberg de la usabilidad propuesto por Dick Berry (D.Mayhew, 1999). *8*
Figura 2. Relación entre atributos, propiedades y patrones de usabilidad (Moreno, 2001). *10*
Figura 3. Resultados del parámetro: modo de evaluación. *15*
Figura 4. Resultados del parámetro: licencia. *15*
Figura 5. Resultados del parámetro: resultados. *16*
Figura 6. Expresión matemática para obtener el total de puntuación de la heurística (ALCALA, 2007). *23*
Figura 7. Algoritmo de cálculo para la heurística (ALCALA, 2007). *23*
Figura 8. Expresión matemática para obtener el peso de una heurística (ALCALA, 2007). *24*
Figura 9. Expresión matemática para obtener el peso de una heurística (ALCALA, 2007). *24*
Figura 10. Expresión matemática para obtener la usabilidad (ALCALA, 2007). *25*
Figura 11. Expresión matemática para obtener la evaluación parcial (Leyva, 2012). *26*
Figura 12. Expresión matemática para obtener la usabilidad (Leyva, 2012). *26*
Figura 13. Expresión matemática para obtener el factor de corrección (Torrente, 2011). *28*
Figura 14. Expresión matemática para obtener el porcentaje de usabilidad (Torrente, 2011). *28*
Figura 16. Modelo conceptual. *29*
Figura 17. Prototipo de IU: Realizar Evaluación. *33*
Figura 18. Patrón controlador utilizado en el escenario "Realizar Evaluación". *37*
Figura 19. Patrón experto utilizado en el escenario "Realizar Evaluación". *38*
Figura 20. Patrón Creador utilizado en el escenario "Realizar Evaluación". *38*
Figura 21. Patrón bajo acoplamiento utilizado en el escenario "InsertarProyecto". *39*
Figura 22. Patrón Decorador utilizado en la generación de plantillas Twig. *40*
Figura 23. Esquema simplificado de la arquitectura interna del sistema. *42*
Figura 24. Organización propuesta por Symfony2. ..... *43*
Figura 25. Clases Repository. *44*
Figura 26. Entidades. *44*
Figura 27. Plantilla Base. *45*
Figura 28.Capa de presentación Datos Principales. *45*
Figura 29. Controlador en Datos Principales. *46*
Figura 30. Diagrama de clases de diseño. Escenario Gestionar Evaluación. *47*
Figura 31. Diagrama de interacción. Escenario Realizar Evaluación. *48*
Figura 32. Modelo de Datos. *48*
Figura 33. Representación de la Cantidad de clases agrupadas en intervalos según la cantidad de procedimientos. *50*
Figura 34. Representación en porcientos (%) de los datos obtenidos de los atributos a medir. *50*
Figura 35. Intervalos de las clases agrupadas según las dependencias entre ellas. *51*
Figura 36. Representación en porcientos (%) de los datos obtenidos de los atributos a medir. *51*
Figura 37. Diagrama de componentes del módulo Evaluaciones. *54*
Figura 38. Diagrama de Despliegue. *55*
Figura 39. Caso de prueba: Escenario Realizar Evaluación. *57*

Figura 40. Descripción de las variables. 57
Figura 41. Resultados generales de las pruebas. 58
Figura 42. Método insertarPrototipo(). 59
Figura 43.Grafo del flujo correspondiente al método. 59
Figura 44. Gráfico de resultados finales de la primera iteración de la prueba (Leyva, 2012). 62
Figura 45. Tabla de resultados finales de la primera iteración de la prueba. 62
Figura 46 .Tabla de resultados finales de la segunda iteración de la prueba. 63
Figura 47. Conjunto de tarjetas EmoCard (Valdivia, 2012) 64
Figura 48. Resultados de la aplicación de EmoCard en la primera etapa. 64
Figura 49.Resultados de la aplicación de EmoCard en la primera etapa. 64
Figura 50. Resultados de la aplicación de EmoCard en la segunda etapa. 65
Figura 51. Resultados de la aplicación de EmoCard en la segunda etapa. 65

Introducción

En la actualidad, a nivel mundial la Industria del Software se ha convertido en un gigante, el cual crece y se desarrolla a ritmo acelerado. Todas las empresas quieren producir aplicaciones informáticas con alta calidad, en el menor tiempo posible y a costos mínimos. Aumentando, así la competitividad entre ellas debido a que los clientes son cada vez más exigentes (Bevan, 2009).

La calidad del software juega un papel importante dentro del desarrollo de aplicaciones informáticas, influyendo positivamente en la decisión de un cliente a la hora de escoger el producto que necesita (Pressman, 2010).

El usuario es una parte esencial en el desarrollo de una aplicación, por lo que la usabilidad es un componente básico de la calidad del software. A pesar de que no existe un conjunto aceptado de atributos críticos de calidad, la usabilidad ha estado presente como atributo de calidad del software desde los años 70 (Boehm, 1978). También es definida como el atributo de calidad que mide la facilidad de uso de las interfaces web (Nielsen, 1993). “La gran importancia de las interfaces gráficas de usuario actualmente dice mucho de la importancia de la usabilidad" (Glass, 2002).

En contraste con la relevancia otorgada a la usabilidad, existe una falta de atención del atributo en la educación y práctica de la Ingeniería de Software (Larman, 2003).

Tradicionalmente se defiende a nivel internacional la idea de medir y mejorar la usabilidad al finalizar el ciclo de vida del software. Muchas investigaciones han afirmado que no es cierto pues han evidenciado que la mayoría de los proyectos han fracasado por no tener en cuenta la usabilidad al inicio de la creación del sistema (Casanovas, 2004).

Los estudios de autores como Pressman y Landauer plantean que una importante parte de los costos del mantenimiento de los sistemas de software se gastan en resolver problemas de usabilidad (Pressman, 2010).

A continuación se muestran ejemplos de las causas que afectan estos costos:

- Las técnicas de la ingeniería presentan una capacidad limitada para la captura de los requisitos de la usabilidad.
- Los requisitos de usabilidad son a menudo pobremente especificados.
- Los requisitos de usabilidad pueden cambiar durante el desarrollo del producto.

A nivel internacional muchas empresas han tenido problemas con la usabilidad de sus productos. Primeramente porque durante el desarrollo de estos, el enfoque recae más sobre el sistema y las tecnologías y no tanto en el usuario. Este error se debe a la suposición que los humanos son más

flexibles, y que es más fácil adaptar a la persona al producto que viceversa. Una segunda razón para la existencia de un producto no usable es que muchas empresas consideran el diseño de sistemas usables como “sentido común”, cuando en realidad se trata de un esfuerzo difícil e imprevisible, cuyas actividades forman una disciplina científica llamada la Ingeniería de Usabilidad (INU). Otro problema y no menos importante es que las empresas tienen dificultades para integrar las actividades de la INU con el desarrollo de software (Vos, 2006).

La Universidad de las Ciencias Informáticas (UCI), no está exenta de lo anteriormente mencionado. La UCI ha creado varios centros para el desarrollo de software con el propósito de insertar en el mercado nacional e internacional productos de gran calidad. Uno de ellos es el Centro de Informatización de Entidades (CEIGE) y como parte de la estructura del mismo la Subdirección de Producción conformada por un grupo de calidad del software.

Dicho grupo tiene entre sus objetivos la validación, verificación y mejora de los productos informáticos que se desarrollan en el centro. Internamente se encuentra el grupo de pruebas internas que tiene entre sus principales metas llevar a cabo el proceso de pruebas de liberación a los productos del CEIGE que soliciten el servicio. El proceso de pruebas tiene como objetivo evaluar características de calidad, tales como: la fiabilidad, eficiencia, usabilidad, mantenibilidad, portabilidad y funcionalidad. Para ello se aplican los diferentes tipos de pruebas que verifican su cumplimiento, entre las que se encuentran las pruebas de mantenibilidad, funcionalidad, recuperación, tolerancia a fallas, carga y estrés.

Actualmente se hace engorrosa la medición de estos atributos, porque en el grupo solo se realizan pruebas funcionales, y para el caso de la usabilidad se tiene establecido que se realicen a través de listas de chequeo, por lo que no existe un sistema automatizado idóneo para medir la usabilidad en los prototipos de Interfaz de Usuario (IU), lo que provoca la pérdida de información por estar en formato duro; la seguridad también se puede ver comprometida pues personas no autorizadas pueden acceder; se hace más difícil la obtención de reportes estadísticos que ayuden a la toma de decisiones importantes; la disponibilidad de la información no es la óptima; la organización y la búsqueda de la información en formato duro lleva consigo un trabajo adicional, difícil y extenuante; además de existir un desgaste físico por parte de las personas que realizan esta evaluación. Por todas estas razones y las muchas facilidades y potencialidades que brindan la digitalización de los procesos es que se decide automatizar el mismo.

Se han realizado investigaciones con el fin de conocer el estado actual de la usabilidad en la UCI, y así poder trazar soluciones para resolver estos problemas. Sin embargo actualmente se siguen encontrando deficiencias en este aspecto. Ejemplo de esto son las entrevistas que se realizó a 10 centros de la universidad y en específico al centro CEIGE (Leyva, 2012). A continuación se muestra el resultado que se obtuvo en el diagnóstico y se puede afirmar que todavía siguen vigentes estos problemas:

Se percibe que la usabilidad tiene un gran impacto en la satisfacción del cliente, la calidad del producto y el éxito del proyecto y un menor impacto en la funcionalidad y la arquitectura del software, esto denota el escaso conocimiento que existe en torno a la usabilidad (Leyva, 2012).

- La evaluación de la usabilidad solo en un caso se identificó que debía realizarse durante el diseño y análisis arquitectónico, el resto defendió la idea que debía medirse en el levantamiento de requisitos, las pruebas de software y en la aceptación del producto, esta idea ha fracasado, la usabilidad debe medirse en todo el proceso de desarrollo (Leyva, 2012).
- Durante el proceso de captura de requisitos, se confunden los escenarios de usabilidad con requisitos funcionales del sistema, cuando se debería tener en cuenta, que los requisitos funcionales son el centro de las necesidades del cliente, mientras que los de usabilidad, giran alrededor de estos para garantizar que el usuario los realice de una forma más cómoda y sencilla (Leyva, 2012).

Otros de los factores que influye en una mala gestión de la usabilidad en el centro, es que los usuarios finales son mayoritariamente excluidos del desarrollo del producto de software y en ocasiones su participación es nula. Los usuarios finales son los que determinan si un sistema es fácil de usar o no, estableciendo realmente si tiene o no calidad, puesto que la usabilidad es considerada como un atributo de la calidad del software. Debe existir una interacción usuario-grupo de desarrollo-cliente desde las etapas iniciales del proyecto para así lograr una mejor calidad en las aplicaciones desarrolladas.

La aplicación de los métodos de la ingeniería de la usabilidad al desarrollo de aplicaciones informáticas aporta importantes beneficios referentes a la reducción de los costes de soporte a la aplicación ya que resulta un producto fácil de instalar, de aprender y de usar, a la calidad del producto y la satisfacción del cliente. Por lo que al no evaluarse el atributo de calidad en los prototipos de IU de los productos desarrollados en el CEIGE de forma rápida, podrían generarse riesgos que atenten contra la sostenibilidad y costes de estos, viéndose así afectada la calidad e integridad de los mismos.

Considerándose lo analizado anteriormente, se define el siguiente **problema a resolver:** la insuficiencia en el proceso de medición de los prototipos de IU de los sistemas que se prueban en el grupo de pruebas internas de CEIGE, dificulta estimar el grado de usabilidad en los prototipos de IU de los productos del centro. A partir del problema planteado se define como **objeto de estudio:** la usabilidad en el desarrollo del software, y como **campo de acción** de la presente investigación comprende la usabilidad en el proceso de evaluación de los prototipos de IU de los sistemas de gestión, teniendo como **objetivo general:** implementar una herramienta que a través de métricas realice la evaluación en los prototipos de IU de los sistemas de gestión que se prueban en el grupo de pruebas internas de CEIGE, para estimar el grado de usabilidad de dichos productos.

Para dar cumplimiento a lo anteriormente planteado se trazaron los siguientes **objetivos específicos:**

Diagnosticar las herramientas que realizan la evaluación de la usabilidad para comparar sus características e identificar deficiencias.

1. Analizar tendencias para la evaluación de la usabilidad, métodos de evaluación, métricas, técnica, entre otras.
2. Implementar una herramienta para la evaluación de las IU que permita a los especialistas determinar el grado de usabilidad.
3. Validar la herramienta propuesta a partir de pruebas de software y técnicas de evaluación.

Se define la siguiente **idea a defender**: Si se desarrollara una herramienta para la evaluación de la usabilidad en los prototipos de IU para sistemas de gestión, se garantizaría estimar el grado de usabilidad en los productos del centro.

Como **tareas de la investigación** las siguientes:

1. Caracterización del tratamiento de la evaluación de la usabilidad a nivel nacional e internacional. Conceptos fundamentales.
2. Comparación de herramientas existentes para la evaluación de la usabilidad.
3. Caracterización de las técnicas y métodos de evaluación de la usabilidad.
4. Desarrollo del marco teórico (exploración de herramientas, marco de trabajo y lenguajes, estado del arte).
5. Diseño de los prototipos de baja fidelidad y validarlos con usuarios potenciales del centro.
6. Definición de los requisitos que deberá cumplir la herramienta.
7. Elaboración de los artefactos propios de la metodología como por ejemplo: el modelo conceptual, y de clases del diseño.
8. Implementación de los requisitos que permita el diseño de la interfaces de usuario.
9. Validación de la herramienta propuesta a partir de pruebas de software, técnicas de evaluación de usabilidad como: listas de chequeo, además de evaluación del placer de uso a través de la técnica «*EmoCard*».

Para la realización de las tareas antes mencionadas se emplearon métodos científicos, los cuales permiten abordar la realidad, estudiar la sociedad y el pensamiento con el propósito de descubrir su esencia. Se clasifican en teóricos y empíricos (Alfredo, 2011).

Los **métodos teóricos** posibilitan las condiciones para buscar las características triviales de la realidad, permiten explicar los hechos y profundizar en las principales relaciones y cualidades de los fenómenos, hechos y procesos.

A nivel teórico se emplearon los siguientes métodos

Analítico – Sintético: sintetiza las ideas y conceptos empleados en el campo de la usabilidad de software, expresando de manera resumida la posición del investigador. **Histórico - Lógico:** para realizar un estudio crítico de los sistemas existentes en este contexto y utilizarlos como punto de referencia y comparación, además de constatar teóricamente cómo ha evolucionado el tema en el tiempo. El método **Hipotético – Deductivo**: para el análisis y la definición de la idea a defender que será verificada o probada con el desarrollo de la investigación, arribando a conclusiones particulares. La **Modelación**: para la realización de los diagramas en el proceso de desarrollo de software, haciendo una representación abstracta de la solución que facilite el desarrollo de la solución.

Los **métodos empíricos** describen y explican las características del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia (Alfredo, 2011). Se emplearon como métodos empíricos:

La **Entrevista** al asesor de calidad del grupo de Calidad del centro CEIGE de la Facultad 3, con el objetivo de obtener información de interés para comprender el funcionamiento de los procesos de calidad. Como resultado de la entrevista se llega a la conclusión que actualmente no existe ninguna herramienta que evalúe la usabilidad. La **Observación** al proceso de medición de la usabilidad en los prototipos de IU en sistemas de gestión para lograr un mejor entendimiento de los mismos. La **Medición** a través del uso de métricas y técnicas de calidad para validar la herramienta.

Para lograr la claridad y comprensión de los contenidos de la investigación realizada se ha estructurado el documento en tres capítulos.

En el **capítulo 1** se realiza el estudio del estado del arte, se hace énfasis en los conceptos asociados a la investigación así como las soluciones existentes a nivel nacional e internacional. Además se caracteriza la metodología de desarrollo de software, se explican las principales tecnologías y herramientas que se utilizarán para la construcción de la solución propuesta, así como las ventajas de su utilización.

En el **capítulo 2** se realiza un análisis de la arquitectura y de los patrones de diseño. Se especifican los requisitos funcionales y no funcionales; se representan los diagramas de clases de diseño y de secuencia, así como la relación existente entre las entidades a través del modelo de datos.

En el **capítulo 3** se abordan todos los elementos relacionados con la implementación y las pruebas, como son los diagramas de componentes, de despliegue y los tipos de prueba. Además se realiza la validación funcional de la solución desarrollada, mediante la realización de pruebas de calidad.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En el capítulo se traza como objetivo realizar el estudio del estado del arte correspondiente a las herramientas que evalúen la usabilidad. Se incluyen además, definiciones de términos significativos para lograr una mejor comprensión del contenido, se mencionan y describen las tecnologías, metodologías y herramientas seleccionadas para desarrollar la solución propuesta, con el principal objetivo de obtener, a través de estas, un sistema que responda a las necesidades de los clientes.

1.2 Conceptos fundamentales

Los conceptos que se definen en el capítulo están relacionados con el dominio del problema, aportando un breve entendimiento de todos los aspectos que se tratan posteriormente.

1.2.1 Definiciones de usabilidad

Según la ISO/IEC 25010 la usabilidad es definida como: *capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones* (Bevan, 2009).

La usabilidad tiene que ver con crear sistemas fáciles de usar. *Por lo que un sistema usable es: fácil de aprender, fácil de recordar, efectivo, eficiente, y seguro de usar* (Nielsen, 1993).

Otro concepto importante asociado a este término es que la usabilidad se define como *"el atributo de calidad que mide la facilidad de las interfaces web"* (Nielsen, 1993).

Después de realizar un análisis de las diferentes definiciones de la usabilidad, las autoras se identifican con el concepto que describe a la usabilidad como "*un atributo de calidad que logra medir la facilidad de uso que brinda cualquier sistema."*

1.2.2 Usabilidad en el diseño de Interfaz de Usuario (IU)

La usabilidad está en el interior del sistema, depende del concepto en el que se basa este. Esta es la razón por la que es necesario ocuparse de la usabilidad del sistema desde el inicio de su desarrollo. No es razonable ocuparse de la usabilidad de un sistema cuando la mayor parte de los recursos dedicados a la construcción del mismo han sido ya consumidos. Por el contrario, cuando se pretende construir un sistema usable, es necesario pensar en el uso que va a tener desde el inicio, y hay que llevar a cabo evaluaciones de usabilidad desde las primeras etapas del proceso de desarrollo. Para tener elementos que evaluar será necesario construir prototipos que den forma

al diseño en el que se esté trabajando. El ciclo de vida usado deberá ser iterativo con prototipado (Ferré, 2000).

La interfaz no es sólo el programa o lo que se ve en la pantalla. Es todo cuando se comienza a interactuar con el sistema. El diseño de interfaces es una disciplina que estudia y trata de poner en práctica procesos orientados a construir la interfaz lo más usable posible, dadas ciertas condiciones de entorno. (Ferré, 2000).

La usabilidad está estrechamente ligada al desarrollo del software, esencialmente en la fase de diseño. Este atributo de calidad ayuda en el desempeño de los diseñadores, los cuales deben pensar siempre en la persona a la cual será dado el producto. En los procesos que serán descritos, diseñados, implementados y validados siempre se debe tener presente la usabilidad como un elemento importante (Ferré, 2000).

Según Nielsen, uno de los teóricos más destacado del tema, el diseño de las interfaces ya ha adquirido autonomía y estatus disciplinar. Sintéticamente puede considerarse como el arte y la ciencia de gestionar, procesar y comunicar la información, de modo que pueda usarse por los hombres devenidos en usuarios, con eficacia y eficiencia. En particular, estima que la usabilidad es un rasgo destacable de las interfaces (Nielsen, 1993).

1.2.3 Usabilidad como elemento funcional

Hasta hace poco, se asumía que la usabilidad era una propiedad exclusiva de la presentación de la información. Se creía que, encapsulando la capa de presentación y separándola del resto, se podía desarrollar la aplicación y, de forma iterativa, pasar las pruebas de usabilidad. Tras cada prueba, tan sólo sería necesario resolver los problemas modificando la presentación y, gracias a esta separación, la funcionalidad no quedaría afectada (Casanovas, 2004).
En realidad, el modelo de desarrollo ha fallado a menudo. Dick Berry, en su analogía del Iceberg de la usabilidad, explica que los aspectos relacionados con la presentación, es decir, lo que normalmente se entiende como mirar y sentir, sólo afectan en un 40% a la usabilidad. El 60% restante está influenciado por lo que él llama modelo del usuario, que está constituido por los objetivos que el usuario quiere alcanzar con sus tareas (Casanovas, 2004).

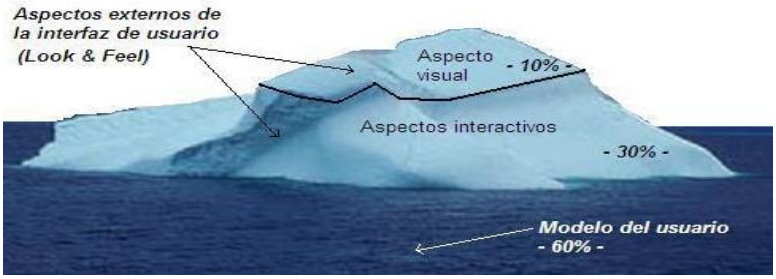

Figura 1.*Iceberg de la usabilidad propuesto por Dick Berry* **(D.Mayhew, 1999)**.

Berry relaciona el modelo del usuario con el modelo de objetos propio de la interfaz de usuario, en el sentido estricto de la POO (programación orientada a objetos), que incluye, entre otras cosas, los objetos y las metáforas de la interfaz. El modelo de objetos, siempre según Berry, es el que permite al usuario relacionar sus objetivos con la funcionalidad del sistema (Casanovas, 2004).

Es cierto que la interfaz gráfica es una parte importante del sistema, y un buen diseño de la misma puede hacer que un sistema aumente su nivel de usabilidad, pero un sistema con un diseño de la interacción pobre no puede mejorar su nivel de usabilidad tan solo cambiando la interfaz gráfica. En cuanto a usabilidad, la parte más crítica es la lógica del sistema (el concepto en base al cual funciona) (Ferré, 2000). Por tanto, la interacción debe diseñarse junto con la lógica de negocio, para asegurar de que la lógica del sistema es usable. No es posible diseñar la lógica de negocio independientemente de la interacción y luego unirlas. Esta relación entre diseño de la interacción y el diseño de software tradicional obliga a modificar este último para acomodar al primero.

Por lo tanto, para conseguir una buena usabilidad, no basta con tener en cuenta la capa de presentación, sino que es preciso que la usabilidad se contemple también en el momento de la definición de la funcionalidad de la aplicación (Casanovas, 2004).

1.2.4 Atributos, propiedades y patrones de usabilidad

La usabilidad se suele evaluar sobre el sistema finalizado, intentando asignar valores a los atributos de usabilidad clásicos (C. Newman, 2002).

Sin embargo, el nivel de abstracción de estos atributos de usabilidad es demasiado alto como para poder estudiar los mecanismos que deben aplicarse a la arquitectura del software para mejorarlos. Por tanto, la filosofía que se sigue es descomponer estos atributos en dos niveles intermedios de conceptos que se acercan más a la solución de software: las propiedades de usabilidad y los patrones de usabilidad (Moreno, 2001).

El primer nivel consiste en relacionar los atributos de usabilidad con las propiedades de usabilidad específicas que determinan las características de usabilidad que han de ser mejoradas en el sistema. Las propiedades de usabilidad pueden verse también como los requisitos que debe satisfacer un sistema software para que sea usable (Moreno, 2001).

El segundo nivel se planteó para identificar mecanismos concretos que podrían incorporarse a la arquitectura de software para mejorar la usabilidad del sistema final. Estos mecanismos se han denominado patrones de usabilidad y abordan necesidades indicada por una propiedad de usabilidad. Cabe mencionar que los patrones de usabilidad no proporcionan una solución de software concreta para ser incorporada a la arquitectura del software, simplemente sugieren algún mecanismo abstracto que podría utilizarse para mejorar la usabilidad (Moreno, 2001).

La aplicación de un patrón de usabilidad es una modificación que puede resolver un determinado problema de usabilidad en un contexto específico, pero que puede ser muy difícil de poner en práctica después, porque el patrón puede tener implicaciones arquitectónicas. Un "patrón de usabilidad arquitectónicamente sensibles “se refiere a una técnica o mecanismo que debe aplicarse al diseño de la arquitectura de un sistema de software con el fin de hacer frente a una necesidad identificada por una propiedad de la usabilidad en la fase de requisitos (Folmer, 2005).
Para obtener un mayor entendimiento de la relación entre atributo, propiedad y patrón, se debe mostrar algunos de los patrones de usabilidad los cuales forman parte de la presente investigación. Las listas de chequeo que contendrá la herramienta a desarrollar estará constituida por un conjunto de indicadores y entre ellos se encontrarán dichos patrones. Existen diferentes clasificaciones de patrones de usabilidad (Leyva, 2012) entre los que se encuentran:

- **Diferentes lenguajes:** capacidad del software para interactuar con los usuarios en diferentes idiomas.
- **Indicación del estado:** los usuarios estén informados de la situación actual del sistema.
- **Forma de validación de campo:** cuando un usuario rellena en varios campos, el sistema puede validar estos campos por completo cuando se envían los datos (validación de formularios) o de forma individual cada vez que se llena un campo (validación de campo).
- **Deshacer:** la capacidad del sistema para deshacer una acción y volver al estado anterior.
- **Asistente:** presenta al usuario una secuencia estructurada de pasos para llevar a cabo una tarea, proporcionando al usuario una guía para cada paso. El usuario puede volver atrás y cambiar cualquiera de las etapas anteriores en el proceso.
- **Vista previa:** el sistema permite a los usuarios ver los resultados de una acción antes de ejecutar esta acción.
- **Perfil de usuario:** el sistema crea y guarda perfiles de usuario para poder recuperar los atributos específicos del sistema asociado con un usuario cada vez que el mismo utiliza el sistema.
- **Cancelación:** los usuarios pueden cancelar lo que están haciendo, si se dan cuenta que ellos han hecho algo mal antes de un estado de error se alcance.
- **Reutilización de la información:** el patrón permite a los usuarios mover o copiar datos de forma manual o automática de una parte del sistema, lo que permite su reutilización
- **Apoyo de búsqueda completa:** proveerá la capacidad de permitir a los usuarios buscar datos de una manera integral y coherente por parte de los criterios pertinentes en el sistema.

Los patrones de usabilidad pueden abordar una o varias propiedades de usabilidad, y las propiedades de usabilidad pueden o no, mejorar varios atributos de usabilidad (Moreno, 2001).

Figura 2. *Relación entre atributos, propiedades y patrones de usabilidad* **(Moreno, 2001).**

1.2.5 Principios de la usabilidad

Como se ha explicado anteriormente la usabilidad hace referencia a la facilidad de uso de un sistema por parte del usuario. Su manejo es claro, intuitivo y transparente, lo que facilita que el usuario alcance sus objetivos de una manera rápida y sencilla. La usabilidad es un aspecto básico del diseño de las interfaces. También está relacionada con campos como: la visualización de la información, la comunicación y la programación (la operativa del sistema).

El padre de la usabilidad Jakob Nielsen propuso un listado de diez principios heurísticos para evaluar la usabilidad del diseño de interfaces (Nielsen, 1993).

Estos principios se utilizarán en la presente investigación con el objetivo de obtener una evaluación de la forma más robusta y precisa posible. Los indicadores presentes en las listas de chequeo cumplirán con los principios mencionados a continuación.

1. **Visibilidad del estado del sistema:** el sistema debe mantener al usuario informado sobre el estado de los procesos que se están llevando a cabo.
2. **Grado de consistencia entre el sistema y el mundo real:** el sistema debe dirigirse al usuario en términos que le resulten familiares, y siguiendo una estructura lógica.
3. **Control del usuario y libertad de uso:** el usuario debe poder explorar el sistema con libertad y abandonarlo en el momento que desee. Las acciones deberían poder anularse y repetirse con facilidad.
4. **Consistencia y mantenimiento de estándares:** para cada acción o situación idéntica, la representación debe ser la misma. Por otra parte, el sistema debe tener en cuenta los estándares del contexto, de manera que el usuario no tenga que reaprender rutinas que acostumbraba a ejecutar de otra manera.
5. **Prevención de errores:** el diseño de la interfaz debe prevenir las posibilidades de error.
6. **El reconocimiento es mejor que la memorización:** todas las opciones deben ser visibles. No debe obligarse al usuario a memorizarlas para poder ejecutarlas.

7. **Flexibilidad y eficiencia:** el sistema debe prever su utilización tanto por parte de usuarios expertos como por parte de novatos. Para los expertos, es aconsejable preparar atajos de teclado que permitan agilizar el proceso de trabajo.
8. **Diseño minimalista:** las ventanas de diálogo no deben contener más información que aquella que resulta relevante; cualquier exceso de información sirve solamente para confundir al usuario y ralentizar el proceso.
9. **Ayuda al usuario en la corrección de errores:** los mensajes de error deben ser breves y claros, describiendo el problema en términos inteligibles y presentando sugerencias para su solución.
10. **Ayuda y documentación:** la ayuda del sistema debe ser fácilmente accesible y tiene que exponer la descripción de los procesos paso a paso y de manera concreta.

1.3 Evaluación de la usabilidad

1.3.1 Métodos y técnicas de evaluación de la usabilidad

Un método de evaluación de usabilidad es un procedimiento sistemático para grabar datos, relacionados con la interacción del usuario final con un producto de software o sistema (R.Fitzpatrick, 2001).

Respecto a los nombres de los métodos se observa que la mayoría de los autores usan algunos nombres de modo consistente (por ejemplo, observación, cuestionario, heurísticas). Por otro lado existen otros nombres de métodos (por ejemplo, indagación, pruebas, inspección) que son usados de acuerdo a la preferencia o prejuicio del autor (A.Dix, et al., 1998).

Una vez revisados los principales métodos de evaluación de usabilidad, se encontró que en su mayoría se apoyan en el uso de técnicas que ayudan en la recolección de información que facilitará la detección de problemas.

Las técnicas de evaluación de usabilidad definen un conjunto de actividades a ejecutar por los evaluadores. (Obeso, 2005)
A continuación se presentan los diferentes tipos de métodos y dentro de estos las técnicas (Obeso, 2005) que ayudan en la recolección de información para la evaluación de la usabilidad:
Indagación: se realiza mediante la recogida de información; ya sea observando, preguntando o interactuando con los usuarios. Se utiliza a menudo en las primeras etapas del proceso de desarrollo de software que es donde se necesita una mayor información con respecto al trabajo que se va a realizar.

- **Entrevista:** se realiza con el objetivo de obtener datos que servirán para realizar la evaluación.

- **Cuestionario:** su función es muy parecida a la entrevista, a diferencia que este es un poco más específico a la hora de realizar las preguntas pertinentes.

Pruebas: se realiza con los usuarios cuando se les asigna tareas concretas y así poder obtener los resultados, como hasta qué punto se cumplió o no el objetivo de dichas tareas.

- **Pensando en voz alta:** técnica formal cuyo objetivo es recoger comentarios u observaciones del usuario. Para ellos se le hace preguntas relacionadas con la lista de tareas de la prueba de usabilidad.
- **Card Sorting (clasificación de tarjetas):** es una técnica de ayuda en la toma de decisiones para realizar una organización de categorías centrada en el usuario. Esta técnica la lleva a cabo el arquitecto de información.

Inspección: es un método que emplea expertos; que son personas que conocen del tema en cuestión, tienen una clasificación especial para poder evaluar diferentes productos.

- **Heurística:** proceso de aplicar reglas o principios de diseño de interfaces a un sistema o prototipo, identificando problemas de usabilidad. Evalúa la consistencia, mensajes de error, el lenguaje, la existencia de ayuda en línea y elementos como íconos, entre otros.
- **Recorrido cognitivo:** se centra en evaluar en un diseño su facilidad de aprendizaje, básicamente por exploración y está inspirado en la observación que muchos usuarios prefieren aprender software a base de explorar las posibilidades que el mismo ofrece.

Se concluye que estos métodos de evaluación de usabilidad aunque existan diferentes clasificaciones pueden ser usados durante una evaluación de usabilidad, dependiendo de los costos y el ciclo de vida, y que son utilizados para asegurar referencias que mejoren la usabilidad o establezcan si esta es suficientemente buena. Del estudio realizado se puede resaltar que los diferentes métodos de evaluación de usabilidad tienen fortalezas y debilidades que están enfocadas a evaluar aspectos específicos de usabilidad, por lo que es recomendable combinarlos en una evaluación para complementarlos entre sí. Por tanto en la presente investigación se utilizará el método de inspección y dentro de él la técnica evaluación heurística ya que esta es en donde se juzgan si cada elemento de la interfaz de usuario cumple con los principios de la usabilidad.

Además dentro de sus ventajas se puede encontrar primeramente su bajo costo, en realidad el tipo de evaluación puede tener el costo que se desee. Un número mínimo de tres evaluadores permite realizar una evaluación por criterios. Los costos son por tanto mucho menores que cualquier otro método de evaluación. Otra ventaja es que en comparación con otras técnicas de evaluación donde el observador debe interpretar las acciones del usuario, en la evaluación heurística no es necesaria la interpretación externa, porque las ideas, comentarios e información elaborada por los evaluadores están contenidos en sus informes.

1.4 Estudio de las tendencias actuales

Para la realización de la presente investigación se consultaron numerosas fuentes bibliográficas. Se identificaron las principales herramientas que evalúan la usabilidad en sistemas de software tanto en el ámbito mundial, así como en los centros de desarrollo de la universidad. Se visitaron sus sitios oficiales para lograr una mayor veracidad en la información mostrada. Además de consultar libros de autores que están relacionados con la publicación de contenidos sobre usabilidad y prototipado en los últimos 5 años.

1.4.1 Sistemas internacionales

WAMMI: es una herramienta creada por Jurek Kirakowski y Nigel Claridge para la evaluación de la calidad de uso de sitios web. Es un cuestionario que tiene como objetivo medir la satisfacción del usuario con sitios web. WAMMI al realizar encuestas en línea se hacen menos costosas que las encuestas por correo y teléfono (González, 2009).

UserPlus: es una aplicación que permite registrar los diseños y responder a preguntas relacionadas con la usabilidad, dependiendo de los resultados se obtendrá una u otra puntuación. Se obtiene una lista de verificación y una puntuación de usabilidad que se basan en las normas ISO por tanto el diseño debe ser compatible con el estándar definido (Rodríguez, 2008).

Usabillla: analiza los sitios y páginas destacando mapas de calor, notas y entradas del usuario. Se seleccionan las páginas a evaluar, seleccionan el tipo de preguntas del test de usabilidad de una lista de pre-generada o agregando las suyas. Usabilla es gratuita para el análisis de hasta 5 páginas web, si se necesita evaluar más, se debe recurrir a planes de análisis con precios prefijados (Mascheroni, 2012).

Five Second Test: es una herramienta para realizar encuestas sobre la web y para comprobar con que idea se quedan los visitantes durante los 5 primeros segundos una vez que acceden al sitio. Dentro de las pruebas que realiza es la prueba de memoria, test para realizar alguna tarea o llegar a algún elemento en concreto (Mascheroni, 2012). Su utilidad es gratuita para 5 pruebas (Rodríguez, 2008).

SIRIUS: es una herramienta creada en el 2011 la cual se basa en heurísticas, no sólo aporta un marco claro y concreto de evaluación si no que se proporciona una medida porcentual del grado de usabilidad de un sitio web (Torrente, 2011). SIRIUS a pesar de ser una herramienta muy utilizada para realizar la evaluación, no se encuentra automatizada, es decir la medición se realiza a través de un Excel. De esta herramienta se tomará la métrica que propone el Excel para realizar la evaluación.

1.4.2 Sistemas nacionales desarrollados en la UCI

SEUSGOW: es un sistema evaluador basado en el procedimiento de evaluación de la usabilidad en el Software de Gestión sobre Plataforma Web en la Facultad 2 de la Universidad de las Ciencias Informáticas creado en junio del 2010. Documentan todos los resultados obtenidos en cada evaluación para tomarlo en cuenta para la siguiente etapa (Lago, 2010). Es una herramienta multiplataforma y gratuita.

Herramienta para la evaluación de la usabilidad a aplicaciones Web en el Departamento de Señales Digitales: es una herramienta que permite evaluar la usabilidad de una aplicación web a través de indicadores o criterios, como resultado contiene las no conformidades detectadas en el proceso. Es creada por la Facultad 2 de la Universidad de las Ciencias Informáticas en junio del 2012 (Valdivia, 2012). Es una herramienta multiplataforma, y gratuita.

A pesar de los elementos antes mencionados para ambas herramientas las mismas no son empleadas en la universidad, además no tienen en cuenta los patrones de usabilidad.

1.4.3 Resultado del estudio de las tendencias actuales

Para realizar la comparación de las herramientas se debe tener en cuenta una serie de parámetros. Estos parámetros responden a características genéricas presentes en cualquier herramienta de evaluación de la usabilidad, con el objetivo de determinar en qué medida incorpora la herramienta estos elementos y así demostrar la necesidad de la implementación de una nueva herramienta.

- **Modo de Evaluación (ME):** especifica el modo de evaluación que utiliza la herramienta, es decir si es a través de cuestionarios, patrones de usabilidad, métricas y heurísticas.
- **Licencia (L):** relaciona si la herramienta es gratuita o no, factor determinante en tanto a la posibilidad de acceso que tendrán los usuarios.
- **Ambiente (A):** especifica si la herramienta es web o de escritorio, aspecto determinante en cuanto a su capacidad para el acceso remoto y diseño colaborativo
- **Principio de Usabilidad (PU):** define si la herramienta se basa en el uso de evaluación heurística. Por lo que esta característica propicia conocer por cada heurística cuáles son los elementos de usabilidad que se evaluarán en el prototipo de interfaz de usuario.
- **Etapa de Evaluación (EE):** muestra en qué fase del ciclo de desarrollo del sistema se debe aplicar la evaluación de la usabilidad.

- **Resultado (R):** especifica el modo de efectuar la evaluación de la usabilidad, es decir si el resultado es cuantitativo o cualitativo- subjetivo u objetivo.

- **Gestión de la Evaluación (GE):** define si la herramienta es capaz de gestionar los elementos que incorpore el sistema. Esta característica garantiza eficacia y eficiencia en el resultado obtenido y disminución en los tiempos de realizar la evaluación.

La tabla comparativa de las herramientas con sus especificaciones se puede encontrar en el anexo 1.

De acuerdo a las características de las herramientas y de su posterior comparación se identificó lo siguiente:

De las herramientas analizadas cuatro herramientas realizan la evaluación de la usabilidad a partir de la técnica de cuestionarios y test de usuarios, tres de ellas solo utilizan la heurística, en comparación con otras técnicas de evaluación donde el observador debe interpretar las acciones del usuario, en la evaluación heurística no es necesaria la interpretación externa, porque las ideas, comentarios e información elaborada por los evaluadores está contenida en sus informes. Además solo dos herramientas emplean una métrica, no la combinación de ellas por lo que se limita en el análisis cuantitativo de la evaluación. Ninguna de las herramientas analizadas emplea patrones por lo que no evalúan elementos funcionales en el sistema relacionado con la usabilidad, como se muestra en la siguiente figura.

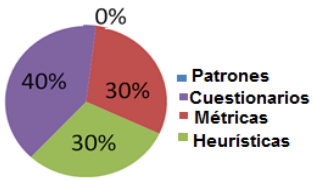

Figura 3. *Resultados del parámetro: modo de evaluación.*

Como se muestra en la anterior figura, dos de estas herramientas son privativas lo cual contraviene con la política de uso de aplicaciones del país y limita su acceso a empresas o equipos de desarrollo de pocos recursos.

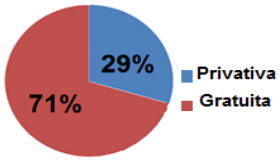

Figura 4. *Resultados del parámetro: licencia.*

Se detectó en las sietes herramientas estudiadas, que la evaluación de la usabilidad no se aplica en las primeras etapas del ciclo de desarrollo del sistema, precisamente cuando se realiza la

construcción del prototipado; esto trae consigo que podría generarse riesgos que atenten contra la sostenibilidad y coste del producto, lo cual debe evaluarse en las fases tempranas del desarrollo.

El modo de efectuar la evaluación de la usabilidad de tres herramientas analizadas es de forma subjetiva esto trae como consecuencias que el resultado depende de lo que el usuario es capaz de analizar y observar, y no de lo que realmente muestra el sistema. En las cuatro restante es cuantitativo pero no muestran reportes estadísticos del resultado.

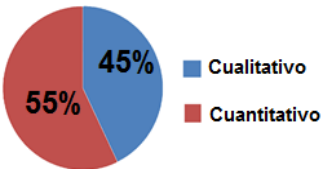

Figura 5. *Resultados del parámetro: resultados.*

De las herramientas solo tres gestionan la evaluación, lo que provoca que disminuyan el tiempo de la evaluación de la usabilidad y posibilitando la introducción de errores.

En el desarrollo de la nueva propuesta se tendrán en consideración las mejores características de las herramientas estudiadas, como el trabajo con métricas y la evaluación de la usabilidad. También se trabajará con el objetivo de solucionar las principales limitaciones de las herramientas como: la integración de varias métricas, la incorporación de los principios y patrones de usabilidad, la generación de reportes estadísticos y la gestión de la evaluación en etapas tempranas del desarrollo donde los problemas de usabilidad son menos costosos de solucionar. Por lo planteado anteriormente surge la necesidad de implementar una herramienta que resuelva estas inconveniencias específicamente para sistemas de gestión.

1.5 Metodología de desarrollo de Software

Durante el desarrollo de software, pueden aparecer diversos cambios en el negocio que impactan significativamente el proceso de ejecución de todas las actividades y la creación de los artefactos (Alfredo, 2011) .Por esto existen diversas metodologías clasificadas en dos categorías: ágiles y tradicionales para guiar el proceso de desarrollo de software.

La UCI, opta por utilizar la metodología AUP (Proceso Unificado Ágil) en su versión AUP-UCI para desarrollo de sistemas informáticos, debido al programa de mejora en que la universidad está inmersa.

AUP es una versión simplificada del Proceso Unificado de Rational (RUP). Además describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. (Sanchez, 2014)

Fases de AUP-UCI

AUP-UCI está forma por tres fases, (Inicio, Ejecución y Cierre) para el ciclo de vida de los proyectos de la universidad, las cuales contienen las características de las cuatro fases (Inicio, Elaboración, Construcción y Transición) propuestas en AUP (Sanchez, 2014).Las características de las fases de la metodología de la universidad son:

- Inicio: durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- Ejecución: en esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el software es transferido al ambiente de los usuarios finales o entregado al cliente junto con la documentación. Además, en esta transición se capacita a los usuarios finales sobre la utilización de la aplicación.
- Cierre: en el cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Disciplinas de AUP-UCI

AUP propone 7 disciplinas: Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno; para el ciclo de vida de los proyectos de la UCI se decidió contar con 8 disciplinas, pero a un nivel más atómico que el definido en AUP, ellas son: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación, Pruebas de Aceptación y Despliegue.

Teniendo como premisa las exigencias del programa de mejora utilizado por la universidad, se selecciona la metodología AUP para regir el proceso de desarrollo de la investigación, porque genera de forma ágil, artefactos que brindan al usuario mayor autonomía para mantener el sistema.

1.6 Herramientas y Tecnologías

1.6.1 Marco de trabajo

Symfony 2: es un marco de trabajo diseñado para optimizar el desarrollo de las aplicaciones web. Está construido para cumplir los principios bases de crear herramientas que permitan desarrollar y construir rápidamente robustas aplicaciones, pues está construido sobre la idea de las mejores

tecnologías. Se escoge además el marco de trabajo Symfony 2 por su abundante documentación, el potencial de trabajo que presenta para el desarrollo de aplicaciones web de forma rápida, robusta y segura. Es uno de los marcos de trabajo más maduros en cuanto a desarrollo web, que incluye varios bundles de terceros que constantemente están mejorando temas de seguridad, rendimiento y otras características (Eguíluz, 2012).

1.6.2 Mapeo de Objeto Relacional (ORM)

Doctrine 2.3.2: se escoge el mismo por proporcionar una persistencia transparente de objetos PHP hacia una base de datos relacional. Se sitúa en la parte superior de una poderosa capa de abstracción de base de datos (DBAL por Data Base Abstraction Layer). Otra de las características de Doctrine es la opción de escribir las consultas de base de datos en un lenguaje de consulta estructurado (SQL por Structured Query Language) propio, orientado a objetos llamado lenguaje de consulta doctrine (DQL por Doctrine Query Language) (Pacheco, 2011) (Doctrine , 2013).

1.6.3 Marco de trabajo de creación de diseños web

Bootstrap 3.0: es un framework (marco de trabajo) desarrollado por Twitter que simplifica el proceso de creación de diseños web combinando CSS y JavaScript. Se selecciona por su mayor ventaja que se pueden crear interfaces que se adapten a los distintos navegadores apoyándose en un marco de trabajo potente con numerosos componentes webs que ahorrarán mucho esfuerzo y tiempo. Es un proyecto de código abierto con licencia Apache v2.0 (Eguíluz, 2012).

jQuery: es una librería de JavaScript rápida y concisa que simplifica el uso del documento HTML, el manejo de eventos, la animación, y las interacciones Ajax para el desarrollo web rápido (Project, 2010). Se escoge esta librería ya que manipula una gran cantidad de estándares para varios navegadores web. En la actualidad esta librería es ampliamente utilizada en disímiles aplicaciones web. Su reputación ha crecido exponencialmente al igual que la comunidad de desarrolladores que comparten sus experiencias así como complementos que mejoran el trabajo con la librería (Eguíluz, 2012).

1.6.4 Herramientas para el modelado

Visual Paradigm for UML 8.0: se hará uso de esta herramienta pues proporciona un conjunto de ayudas para el desarrollo de programas informáticos desde la planificación, transitando por el análisis y el diseño hasta la generación del código fuente de los programas y la documentación. Es de gran ayuda en la modelación del diseño, ya que permite visualizar el flujo central y detallado de cada proceso mediante diagramas. La relación que se establece entre sus componentes y la creación de cualquier tipo de diagramas la convierte en una herramienta amigable, que permite a

múltiples usuarios trabajar sobre el mismo proyecto. Otra de sus características es que permite generar todos los diagramas que se necesitan para construir y documentar el sistema a desarrollar (Visual Paradigm).

1.6.5 Lenguaje de modelado

Lenguaje de Modelado Unificado (UML): es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Incluyendo funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Como lenguaje para el modelado se seleccionó UML ya que permite organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan y convengan con él. Además de ser el lenguaje con el que trabaja el Visual Paradigm (Schmuller, 2000).

1.6.6 Lenguajes de programación del lado del servidor

PHP 5.4: es un lenguaje de programación de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones web dirigidas a bases de datos. Tiene la capacidad de conexión con la mayoría de los manejadores de base de datos, destacando su conectividad con MySQL y PostgreSQL. Es un lenguaje libre, por lo que se presenta como una alternativa de fácil acceso para todos. Ya que el marco de trabajo seleccionado es Symfony2 es necesario utilizar el lenguaje para la creación del sistema, ya que el marco de trabajo, el motor de plantillas y el marco de trabajo doctrine necesita la versión 5.3 o superior de este lenguaje (Welling, 2003).

Twig 2.1: es un motor de plantillas que posee un amigable ambiente para el diseñador y desarrollador apegado a los principios de PHP, añadiendo funcionalidades a los entornos de plantillas. Posee una sintaxis corta y concisa, similar a la de otros lenguajes de programación, además, implementa un mecanismo de herencia de plantillas. Twig es seleccionado por la integración que posee con el marco de trabajo escogido. Además, permite agilizar la construcción de plantillas de la vista y permitió compilar las plantillas hasta código PHP optimizado, lo que proporcionó rapidez durante la implementación. Brindó seguridad, pues permite evaluar el código de plantilla que no es confiable (Pacheco, 2013).

1.6.7 Lenguaje de programación del lado del cliente

JavaScript 1.6: es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Pérez, 2009). Para facilitar el uso JavaScript se propone utilizar jQuery, una rápida y concisa librería de JavaScript que simplifica la

construcción de vistas HTML, el manejo de evento y las animaciones e interacciones Ajax permitiendo un desarrollo web rápido (Pérez, 2009).

HTML 5.0: en inglés de HyperText Markup Language (en español se traduce como lenguaje de marcado de hipertexto). Es un lenguaje para escritura de hipertexto, es decir, documentos de texto estructurados, incluye enlaces que conducen a otros documentos o a otras fuentes de información y permite la inclusión de información por formularios, entre otros.

CSS 3.0: es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. CSS3 es seleccionado por las ventajas y facilidades que aporta a la hora de trabajar las plantillas que se van a mostrar en la aplicación, permitiendo ahorro de código y tiempo a los desarrolladores en la elaboración de varios trucos para lograr confeccionar las interfaces como son requeridas por el cliente (Css3, 2014).

1.6.8 Entorno de Desarrollo Integrado (IDE)

NetBeans 8.0: el IDE es un galardonado ambiente de desarrollo integrado disponible para los sistemas operativos Windows y Linux. El proyecto NetBeans consiste en un IDE de código abierto y una aplicación de plataforma que posibilita a los desarrolladores crear páginas web rápidamente y aplicaciones móviles usando la plataforma Java, así como PHP, JavaScript, Ajax. El IDE NetBeans 8.0 provee una significativa mejora de rendimiento y experiencia codificando, con una nueva capacidad de análisis de código estático en el editor de PHP y un escaneo de proyectos más inteligentes. Por todas estas razones que mejoran la integración con el marco de trabajo y el rendimiento de la aplicación se decide utilizar el entorno de desarrollo NetBeans 8.0 (Corporation, 2014).

1.6.9 Gestor de base de datos

PostgreSQL 9.1: es un sistema de gestión de bases de datos objeto-relacional, con su código fuente disponible libremente. PostgreSQL se caracteriza por ser un sistema estable, de alto rendimiento, gran flexibilidad ya que funciona con la mayoría de los sistemas Unix, además tiene características que permiten extender fácilmente el sistema. PostgreSQL puede ser integrada al ambiente Windows permitiendo de esta manera a los desarrolladores, generar nuevas aplicaciones o mantener las ya existentes (Denzer, 2002).

1.6.10 Servidor web

Apache 2.0: está diseñado para ser un servidor web potente y flexible, continuamente actualizado y adaptado a los nuevos protocolos (HTTP) y puede funcionar en la más amplia variedad de plataformas y entornos. Es un software de libre distribución, que publica su código fuente, lo que permite que cualquiera pueda modificarlo y colaborar así a su desarrollo (Mohammed, 2003).

1.7 Conclusiones del Capítulo

En el capítulo se realizó un análisis de las principales herramientas que realizan el proceso de medición de la usabilidad, concluyendo que los mismos no pueden ser utilizados porque contienen un conjunto de limitaciones como son: no integran varias métricas, no incorporan los principios y patrones de usabilidad, no generan reportes estadísticos, no gestionan la evaluación y no incorporan la misma en etapas tempranas del desarrollo donde los problemas de usabilidad son menos costosos de solucionar.

Partiendo de los resultados anteriores se propone la metodología, herramientas y tecnologías que permitirán desarrollar la herramienta para la evaluación de la usabilidad en los prototipos de IU para sistemas de gestión.

Capítulo 2: Descripción de la propuesta de solución

2.1 Introducción

El presente capítulo tiene como objetivo la presentación de la herramienta para la evaluación de la usabilidad en los prototipos de IU para sistemas de gestión, la misma lleva por nombre “UsaProt”. Para su desarrollo se empleó la metodología AUP para guiar el proceso de desarrollo de software. Se describe la arquitectura, así como los requisitos funcionales y no funcionales a tener en cuenta para el desarrollo del nuevo sistema. Se realizan los artefactos planteados por la metodología seleccionada y se especifican las tablas de la base de datos, con el objetivo de brindar un mejor entendimiento de la propuesta de solución.

2.2 Descripción de la propuesta de solución

Teniendo en cuenta los objetivos que persigue el presente trabajo de diploma y en aras de utilizar la información recopilada en el capítulo anterior, se propone como solución desarrollar una aplicación web que posibilite realizar la evaluación de la usabilidad en los prototipos de IU.

La herramienta UsaProt debe ser capaz de contribuir al proceso de medición de los prototipos de interfaz de usuario de los sistemas que se prueban en el grupo de pruebas internas de CEIGE. Para estimar el grado de usabilidad de los prototipos, se utilizarán métricas de usabilidad las cuales permitirán decidir si un prototipo es fácil de usar o no. Además de generar un reporte final por: proyecto, módulo, funcionalidad, prototipo y evaluación, en donde se muestra un conjunto de elementos relacionados con los mismos y el grado de usabilidad que tomará el proyecto, módulo y funcionalidad; que es la media ponderada del grado de usabilidad de los prototipos que pertenecen a estos. La herramienta contendrá tres métricas para realizar la evaluación las cuales se analizarán a continuación.

2.2.1 Métrica para evaluar usabilidad en prototipo de IU de acuerdo a la importancia de las heurísticas

La métrica está basada en las heurísticas de J. Nielsen y cada una de ellas está formada por una lista de elementos verificables. Estos elementos se puntuarán con un porcentaje en el intervalo de 0 a 100, en función de su grado de certeza (0, 25, 50, 75,100) en el prototipo a medir (ALCALA, 2007). Con la expresión matemática que a continuación se muestra se obtendrá su valor, esta devuelve un valor para cada heurística para poder estudiar cada una por separado y recomendar su mejora con consejos concretos.

$$t = \sum_{k=1}^{5} \left[\sum_{i=1}^{n} v_i \; p_k \right]$$

Figura 6. *Expresión matemática para obtener el total de puntuación de la heurística* **(ALCALA, 2007)**.

Siendo:

k: el número de grados de certeza **i:** el número de fila

n: el número total de filas **s:** la suma de unos, {1}, de cada columna

t: total de puntuación de la heurística en función del grado de certeza de cada fila

v: valor de cada cuadro. Se escribe un 1 en el lugar correspondiente en función del grado de certeza de la fila

p: grados de certeza: {0, 25, 50, 75, 100}

Algoritmo de cálculo para cada heurística

```
t = 0
para cada K = 1 hasta 5
        sk = 0
        para cada i = 1 hasta n
                sk = sk + vi
        fin para
        t = t + sk * pk
fin para
```

Figura 7. *Algoritmo de cálculo para la heurística* (**ALCALA, 2007)**.

La métrica final se obtendrá al realizar la media ponderada de los valores obtenidos.

Los puntos a seguir para hallar un valor de usabilidad son los siguientes

1- Para cada heurística, y dentro de esta para cada fila, marcar con un “1” el grado de certeza que se considere más conveniente, siendo “0%” el total desacuerdo y “100%” el total acuerdo. En el caso de no ser aplicable al prototipo a medir, marcar con un “1” la columna N/A”.
2- Después de rellenar cada heurística, se aplica la fórmula 1 anteriormente mencionada.
3- Luego se asignan a cada una los criterios de asignación de pesos (2 criterios).
4- Después de aplicar los dos criterios a cada heurística y obtener los resultados correspondientes a los criterios, se obtiene la media para cada heurística.
5- Como resultado de la obtención de los pesos y pasando estos a números enteros para un manejo más fácil y dividiendo por 100 para la entrada en la escala de 0 a 100, se obtiene la expresión como métrica de la usabilidad.

6- Una vez obtenida la usabilidad se convierte a escala de porciento (%) multiplicando el resultado obtenido por 100 y se mide en la escala de criterios para la aceptabilidad del nivel de usabilidad definido en el estándar ISO 14598.

Criterios de asignación de pesos a las heurísticas

Para la medida última de usabilidad se ha dado peso a cada heurística considerándose que no todas tienen la misma importancia. Se han considerado dos criterios para hallar los pesos definitivos y poder aplicarlos a las medias de las observaciones calculadas con la expresión del punto 1.

1er Criterio

Para el primer criterio se ha utilizado la mayor o menor cantidad de elementos verificables de cada heurística según la expresión:

$$w_h = \frac{\sum_{i=1}^{n} v_i}{\sum_{h=1}^{10}\sum_{i=1}^{n} v_i}; \qquad \text{siendo } \sum_{h=1}^{10} w_h = 1$$

Figura 8. *Expresión matemática para obtener el peso de una heurística* **(ALCALA, 2007)**.

2do Criterio

Los pesos se han obtenido midiendo la contribución de cada heurística a su media:

$$w_i = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{h_i}{\sum_{k=1}^{10} h_k}\right)$$

Figura 9. *Expresión matemática para obtener el peso de una heurística* **(ALCALA, 2007)**.

Siendo:

Wi: los pesos a hallar **n:** el número de prototipos

k: el número de heurísticas, 10 **hk:** el valor de la observación de cada heurística

Resumen de los pesos y su media

Los pesos definitivos obtenidos es la media de los pesos según los dos criterios.

Expresión matemática de la métrica de usabilidad

Como resultado de la obtención de los pesos y pasando estos a números enteros para un manejo más fácil y dividiendo por 100 para la entrada en la escala de 0 a 100, se ha obtenido la siguiente expresión como métrica de la usabilidad**:**

$$U = \sum_{i=1}^{10} P/100$$

Figura 10. *Expresión matemática para obtener la usabilidad* **(ALCALA, 2007)**.

2.2.2 Métrica para evaluar prototipo de IU de acuerdo a los patrones de usabilidad

La métrica está basada en diagnosticar los patrones de usabilidad en una lista de elementos verificables. Estos elementos para obtener la evaluación tendrán la **frecuencia** con que aparece el problema y el **impacto** del mismo (Leyva, 2012).

El criterio utilizado para estimar la evaluación de cada indicador de la lista de chequeo es el siguiente:

-1: el sistema no satisface la directriz +1: el sistema sí satisface la directriz

NA: no aplica

El criterio utilizado para estimar **el impacto** de cada una de las directrices es el siguiente:

1**. Bajo**: aunque es recomendable que se cumpla la afirmación su incumplimiento no implica confusión ni error en el usuario. No daría problemas de usabilidad importantes.

2**. Medio**: el incumplimiento puede provocar problemas no muy graves de usabilidad aunque conviene resolverlos ya que se facilitaría el funcionamiento del sistema.

3**. Alto**: produce problemas altos de comprensión y funcionalidad en el sistema por lo que es primordial que el problema sea solventado. Puede provocar problemas graves de usabilidad.

La **columna frecuencia** puede tener los siguientes valores: siempre (s) si siempre se cumple la afirmación, no (n) si nunca se cumple y a veces (a) si se cumple alguna vez.

La respuesta afirmativa de la pregunta implica la satisfacción del criterio a evaluar, mientras que una negativa implica que el criterio no se cumple. El resultado de la evaluación heurística será de forma cuantitativa arrojando el por ciento de usabilidad del prototipo.

Para evaluar el prototipo, se realizará mediante la lista de chequeo de usabilidad para sistemas de gestión aplicada por los evaluadores.

Se obtienen el grado de usabilidad mediante la métrica, para calcular la misma se debe determinar la evaluación parcial (Ep), que consiste en promediar los indicadores de la lista de chequeo evaluados de “1”, para ello se utiliza la siguiente ecuación:

$$Ep = \frac{I}{N},$$

Figura 11. *Expresión matemática para obtener la evaluación parcial* **(Leyva, 2012)**.

Siendo:

I: cantidad de indicadores evaluados de “1” que es el caso que el elemento revisado no presente errores

N: total indicadores de la lista de chequeo

Luego se calcula el grado de usabilidad (U), calculando el promedio de la Ep de cada evaluador, mediante la siguiente ecuación:

$$U = \frac{\sum_{i=0}^{k} Ep}{K}$$

Figura 12. *Expresión matemática para obtener la usabilidad* **(Leyva, 2012)**.

Siendo:

U: evaluación total del sistema **K:** total de evaluadores que participa en la evaluación

Ep: evaluación parcial dada por cada evaluador

2.2.3 Métrica para evaluar prototipo de IU de acuerdo a la relevancia de incumplimiento de los criterios

La métrica está basada en las heurísticas de J. Nielsen y cada una de ellas está formada por una lista de elementos verificables. Permite obtener una medida cuantitativa en rango 0-100 que cuantifique el nivel de usabilidad obtenido en un sistema de gestión considerando los prototipos a evaluar. Estos elementos para obtener la evaluación tendrán **relevancia del incumplimiento de los criterios** (Torrente, 2011).

Relevancia del incumplimiento de los criterios

Para establecer la relevancia de incumplimientos de los criterios se define lo siguiente:

- **Crítica (CR):** el problema identificado es severo. El usuario no podrá completar la tarea.
- **Mayor (MA):** es posible que el usuario complete la tarea pero tendrá mucha dificultad, frustración o incluso tendrá que ejecutar muchos pasos innecesarios.
- **Media (ME):** en la mayoría de casos, el usuario podrá completar la tarea, realizando un moderado esfuerzo para evitar el problema.
- **Moderada (MO):** el problema ocurre de manera intermitente y puede ser fácilmente superado, aunque es irritante para el usuario. Debido principalmente a problemas estéticos.

Una vez establecida la relevancia del incumplimiento de los criterios, se procede a determinar la métrica de evaluación. Para ello se determinan, en primer lugar, los elementos de valoración de cada criterio, que serán los que tenga que asignar el evaluador en el proceso de revisión del prototipo. Se asignan a continuación los valores numéricos correspondientes a cada uno de los valores textuales propuestos para la evaluación. Con el valor de evaluación y el factor de relevancia de cada criterio se formula, finalmente, la métrica de evaluación.

Con este objetivo, se establecieron los siguientes elementos de valoración:

- 0…10: donde si es 0 no se cumple en absoluto y si es 10 se cumple totalmente
- NTS: no se cumple en todo el prototipo
- S: se cumple el criterio
- N/A: criterio no aplicable en el prototipo

Luego se hará una correspondencia entre estos valores textuales por numéricos:

NTS: equivale a 0 S: equivale a 10

Ponderación del incumplimiento de criterios

Factores de relevancia que serán considerados en la fórmula asociada a la métrica de evaluación:

Crítica: equivale a 8 Mayor: equivale a 4

Media: equivale a 2 Moderada: equivale a 1

Se considera entonces como factor determinante del nivel de usabilidad del prototipo, la relevancia de incumplimiento de los criterios.

Cálculo del porcentaje de usabilidad

Finalmente, se hace necesario obtener un valor cuantitativo que refleje el nivel de usabilidad obtenido tras la evaluación heurística. Para ello se propone una fórmula en la que se consideran los siguientes conceptos:

- **Factor de corrección**: valor de ajuste que se aplica a cada uno de los criterios evaluados con el fin de obtener diferentes niveles de usabilidad dependiendo de la relevancia de los mismos en función del prototipo en evaluación. Los valores resultantes al aplicar el factor de corrección debieran llevar a obtener valores de usabilidad comprendidos siempre entre 0 y 100, por tratarse de un porcentaje.
- **Cálculo del factor de corrección:** partiendo de los valores correspondientes a los diferentes niveles de relevancia de un criterio, el factor de corrección se calcula dividiendo cada valor de relevancia entre la suma de todos los valores de relevancia de los criterios evaluados.

El valor del factor de corrección de cada uno de los criterios evaluados se obtiene de la siguiente manera:

$$fci = \frac{rci}{\sum_{j=1}^{j=nce} rcj}$$

Figura 13. *Expresión matemática para obtener el factor de corrección* **(Torrente, 2011)**.

Siendo:

rc: valor de relevancia que corresponde a un criterio

La fórmula propuesta para la determinación del porcentaje de usabilidad es la siguiente:

$$PU = \frac{\sum_{i=1}^{i=nce} (fci * vci)}{\sum_{i=1}^{i=nce} (fci * 10)} * 100$$

Figura 14. *Expresión matemática para obtener el porcentaje de usabilidad* **(Torrente, 2011)**.

Siendo:

nce: número de criterios evaluados. Se considera que alguno de los criterios pueda no ser aplicable al prototipo en evaluación y, por lo tanto, no se evalúe

vc: valor de evaluación de un criterio (0 ó 10)

fc: factor de corrección aplicado al criterio evaluado

2.2.4 Criterio de aceptabilidad según el estándar ISO 14598

Se aplicará el criterio para todas las métricas explicadas anteriormente. Una vez obtenida la usabilidad se convierte a escala de % y se mide en la escala de criterios para la aceptabilidad del nivel de usabilidad definido en el estándar ISO 14598, en tres regiones de un rango de 0 a 100%:

- **Satisfactoria:** cuando U es mayor que 60%.
- **Aceptable:** cuando U está entre 40% y 60%.
- **Insatisfactoria:** cuando U es menor que 40%.

2.3 Modelado de la propuesta de solución

2.3.1 Modelo conceptual

Un modelo conceptual puede entenderse como un mapa de conceptos y sus relaciones, incluyendo suposiciones acerca de la naturaleza tanto de los fenómenos que esos conceptos representan como sus relaciones e implican un alto nivel de abstracción. (Larman, 1999) Se pretende obtener el mapa conceptual que contenga los principales conceptos relacionados con la evaluación de la usabilidad en los prototipos de interfaz de usuario.

El modelo conceptual comprende los principales conceptos relacionados con la evaluación.

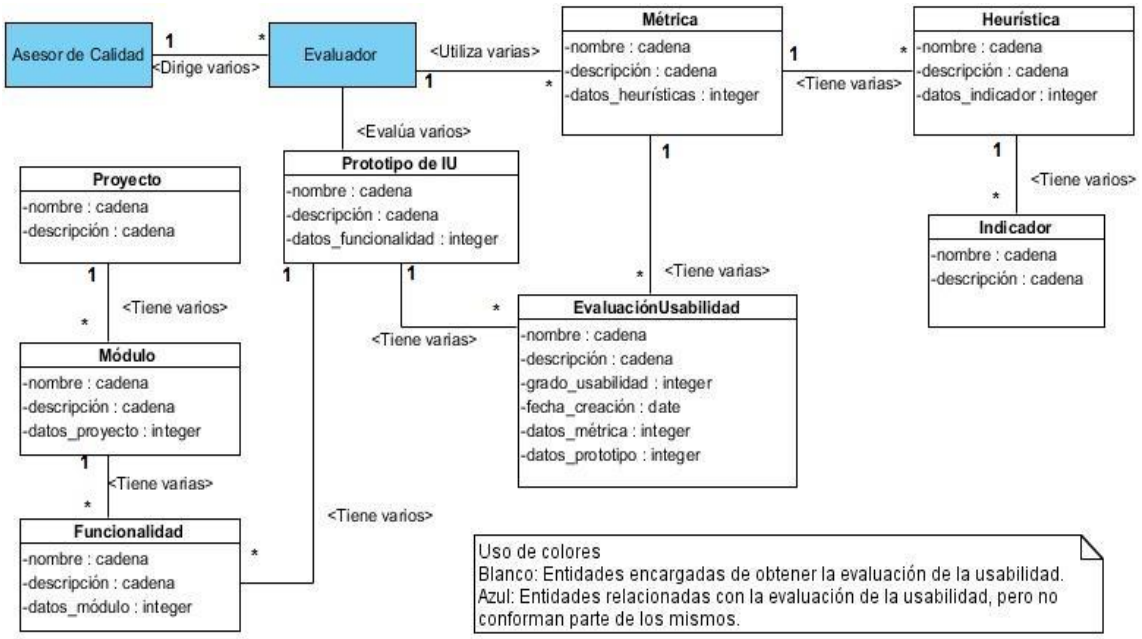

Figura 15. *Modelo conceptual.*

Descripción del modelo de conceptual

El asesor de calidad dirige a los evaluadores que tienen la misión de evaluar el prototipo los cuales son sometidos al proceso de evaluación, para ello se procesan principios o criterios de evaluación, donde se medirán un conjunto de características que el sistema debe cumplir para

obtener la evaluación del prototipo, el cual se obtendrá aplicando métricas, técnicas y métodos de usabilidad dando como resultado un informe que contiene las no conformidades detectadas en el proceso y un reporte estadístico de los mismos. A partir de estos informes se pueden registrar las recomendaciones asociadas a los problemas detectados.
A continuación se muestra los conceptos que formarán parte en el desarrollo del sistema.

Tabla 1. *Conceptos fundamentales del Modelo Conceptual.*

Asesor de calidad: persona encargada de dirigir el proceso de evaluación de la usabilidad.
Evaluador: persona encargada de realizar la evaluación en los prototipos de IU.
Prototipo de IU: prototipo al que se le realiza la evaluación.
Evaluación Usabilidad: es el proceso automatizado que se lleva a cabo para evaluar los prototipos de IU.
Proyecto: producto informático al que se le realizará la evaluación de usabilidad a los prototipos pertenecientes a él y se devolverá una evaluación final del mismo.
Módulo: pertenece a un proyecto, se realizará la evaluación de usabilidad a los prototipos pertenecientes a él y se devolverá una evaluación final del mismo.
Funcionalidad: pertenece a un módulo, se realizará la evaluación de usabilidad a los prototipos pertenecientes a él y se devolverá una evaluación final al mismo.
Indicadores: características que debe cumplir el prototipo para que obtenga un grado de usabilidad aceptable.
Heurística: principios de usabilidad que se debe de tener en cuenta para el diseño de una interfaz gráfica de usuario.
Métricas: expresiones matemáticas para obtener el grado de usabilidad.

2.4 Requisitos de la herramienta

Los requisitos representan las condiciones o capacidades que debe tener un sistema o componente, para satisfacer un contrato, estándar o documento impuesto formalmente (Sommerville, 2007).

La definición y captura de los requisitos del sistema se basó en los resultados del estudio de herramientas homólogas y la aplicación de técnicas de captura de información como la entrevista. En este último caso, la atención estuvo centrada en sujetos claves del negocio que por su dominio operaron como porteros en el estudio. Para más detalles consultar el anexo 2.

2.4.1 Requisitos funcionales

Los requerimientos funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que debe reaccionar a entradas específicas y de cómo se debe comportar en situaciones particulares (Sommerville, 2007).

En la tabla siguiente se muestran los requisitos definidos para la realización e implementación del sistema que se propone. Los mismos se han especificado a partir de las características y funcionalidades que debe tener el sistema.

Tabla 2. *Requisitos funcionales*.

Requisitos Funcionales		
RF_1 Consultar métrica	RF_20 Consultar funcionalidad	RF_39 Generar reporte por módulo
RF_2 Buscar métrica	RF_21 Buscar funcionalidad	RF_40 Generar reporte por funcionalidad
RF_3 Consultar heurística	RF_22 Insertar prototipo	RF_41 Generar reporte por prototipo
RF_4 Buscar heurística	RF_23 Modificar prototipo	RF_42 Autenticar usuario
RF_5 Consultar indicador	RF_24 Eliminar prototipo	RF_43 Insertar usuario
RF6 Buscar indicador	RF_25 Consultar prototipo	RF_44 Modificar usuario
RF_7 Insertar proyecto	RF_26 Buscar prototipo	RF_45 Eliminar usuario
RF_8 Modificar proyecto	RF_27 Realizar evaluación	RF_46 Consultar usuario
RF_9 Eliminar proyecto	RF_28 Modificar evaluación	RF_47 Buscar usuario
RF_10 Consultar proyecto	RF_29 Eliminar evaluación	RF_48 Asignar roles
RF_11 Buscar proyecto	RF_30 Consultar evaluación	RF_49 Insertar rol
RF_12 Insertar módulo	RF_31 Buscar evaluación	RF_50 Modificar rol
RF_13 Modificar módulo	RF_32 Insertar no conformidad	RF_51 Eliminar rol
RF_14 Eliminar módulo	RF_33 Modificar no conformidad	RF_52 Consultar rol
RF_15 Consultar módulo	RF_34 Eliminar no conformidad	RF_53 Buscar rol
RF_16 Buscar módulo	RF_35 Consultar no conformidad	RF_54 Asignar permisos
RF_17Insertar funcionalidad	RF_36 Buscar no conformidad	RF_55 Buscar permisos
RF_18 Modificar funcionalidad	RF_37 Generar reporte por evaluación	RF_56 Consultar permisos
RF_19 Eliminar funcionalidad	RF_38 Generar reporte por proyecto	

Los artefactos de descripción de requisitos del software generados como resultado del análisis efectuado, constituyen un elemento primordial para el diseño y la posterior implementación, pues al registrar las características y condiciones que debe cumplir cada requisito funcional,

propician la comprensión de los procesos a implementar, mediante la identificación de las funcionalidades a desarrollar. Se muestra la descripción del requisito *Realizar evaluación* y el prototipo asociado al mismo. Los restantes requisitos descritos se encuentran en el anexo 3.

Tabla 3. *Descripción de requisito funcional Realizar evaluación.*

Precondiciones	El usuario debe estar autenticado con los permisos necesarios. El usuario debe haber seleccionado en el módulo Evaluaciones la opción Gestionar Evaluación. El usuario debe haber seleccionado la opción Realizar Evaluación.
Flujo de eventos	
Flujo básico Realizar Evaluación	
1	El usuario selecciona los siguientes datos: métrica, proyecto, módulo, funcionalidad y prototipo que será evaluado.
2	El usuario llena los campos de la lista de chequeo según la métrica seleccionada: **1-Métrica de acuerdo a la importancia de la heurística** En dependencia del grado de certeza que toma el indicador que puede ser (0,25, 50,75, 100 o N/A), la columna toma valor de "1" de acuerdo al grado de certeza que posea el indicador. **2-Métrica de acuerdo a los patrones de usabilidad:** El criterio utilizado para estimar la evaluación de cada indicador de la lista de chequeo es el siguiente: -1: no satisface la directriz. +1: sí satisface la directriz. **NA**: no aplica. 3- **Métrica de acuerdo a la relevancia de incumplimiento de los criterios:** Los elementos de valoración de cada criterio, que serán los que tenga que asignar el evaluador en el proceso de revisión son los siguientes: **NTS:** equivale a 0. **S:** equivale a 10. Además de los factores de relevancia que serán considerados en la fórmula asociada a la métrica de evaluación: **Crítica:** equivale a 8. **Mayor:** equivale a 4. **Media**: equivale a 2. **Moderada:** equivale a 1.
3	El usuario selecciona la opción Aceptar.
4	El sistema realiza la evaluación correctamente.
Pos-condiciones	
1	Queda realizada la evaluación correctamente y devuelve el valor del grado de usabilidad de acuerdo a la métrica seleccionada.
Flujos alternativos	
Flujo alternativo 1	
1	El usuario selecciona la opción Cancelar.
2	El sistema cierra la interfaz sin realizar la evaluación.
Flujo alternativo 2	
1	El usuario introduce datos incorrectos.
2	El sistema muestra un mensaje de error indicando que hay datos incorrectos o campos obligatorios vacíos.
Pos-condiciones	

1	N/A

Validaciones

1	

A continuación se muestra la vista correspondiente al prototipo de alta fidelidad de la funcionalidad de la aplicación "*Realizar evaluación*".

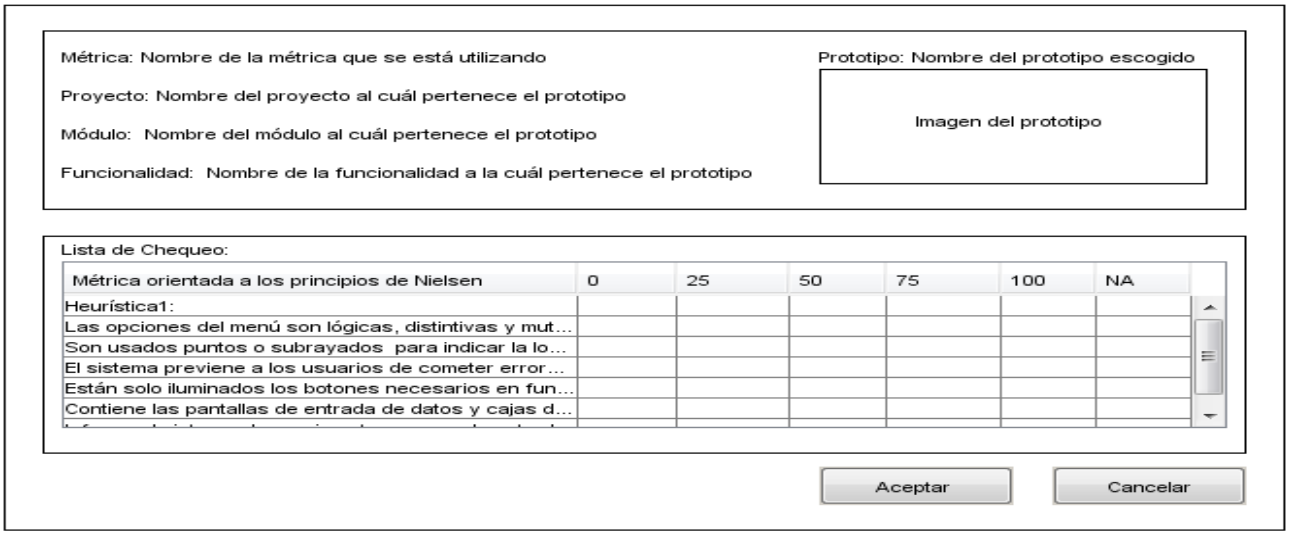

Figura 16. *Prototipo de IU: Realizar Evaluación.*

Descripción del escenario a desarrollar

La interfaz muestra el área de trabajo para la funcionalidad Realizar evaluación. Está dividida en dos regiones principales: la primera región se encuentra en la parte superior izquierda contiene el nombre de la métrica a utilizar, proyecto, módulo y funcionalidad a la que pertenece el prototipo a evaluar, en la parte superior derecha muestra el prototipo; la segunda región en la parte inferior muestra la lista de chequeo, la cual contiene 10 heurísticas con sus indicadores con los que se analizarán el prototipo seleccionado. Luego de haber llenado los campos se prosigue a insertar la evaluación la cual mostrará la medida de usabilidad para el prototipo seleccionado. El escenario contiene las funcionalidades genéricas del sistema, es decir es el que muestra todo el proceso de evaluación del atributo de usabilidad.

2.4.2 Requisitos no funcionales

Los requisitos no funcionales (RNF) son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. A menudo son aplicados en su totalidad al sistema y normalmente apenas se aplican a características o servicios individuales del sistema (Sommerville, 2007).

Tabla 4. *Requisitos No funcionales.*

Requisitos no funcionales:	
Software:	RNF_1. Servidores de base de datos locales PostgreSQL versión 9.1
Hardware:	RNF_4. Hardware mínimo para oficinas: ➢ 1 GB de memoria RAM ➢ Procesador Pentium IV ➢ 120 GB de disco duro RNF_5. Hardware mínimo para servidores: ➢ 2 GB de memoria RAM ➢ Procesador Dual Core o superior 500 GB de disco duro
Soporte:	RNF_6. Adaptabilidad. Se debe diseñar para ser desarrollado sin basarse en características propias de algún sistema operativo, para lograr un producto multiplataforma.
Seguridad:	RNF_7. La autenticación estará basada en el uso de credenciales. RNF_8. Los usuarios estarán autorizados a realizar las acciones que se encuentran definidas para el rol al cual pertenece. RNF_9. Solo se podrá acceder a la aplicación desde las direcciones IP autorizadas. RNF_10. Los documentos que exporta el sistema deben estar dotados de las configuraciones necesarias para que sean de solo lectura.
Usabilidad:	RNF_11. Mínimo impacto de los errores. RNF_12. Facilidad de aprendizaje: necesidad de un fácil aprendizaje en cuanto a instalación, configuración y uso del sistema. Proveer manuales de instalación, configuración y uso del sistema. RNF_13. Facilidad de uso: las interfaces de usuario indican los campos. Los menús permitirán una navegación sencilla, tanto a los usuarios con conocimientos avanzados de informática como a los usuarios más inexpertos.

2.5 Técnicas de validación de requisitos

2.5.1 Prototipado

Mediante la utilización de esta técnica, se logra describir una primera versión del sistema, demostrando cómo serán dispuestos posteriormente los conceptos que intervienen en el mismo (Bradshaw Venzant, y otros, 2010) .Los prototipos ofrecen la oportunidad al cliente de apreciar una primera muestra del producto, y posteriormente hacer ver sus puntos de vista con respecto a los mismos, con el objetivo de encauzar la aplicación rumbo al correcto cumplimiento de las funcionalidades necesarias (Barros, 2008) .La utilización de los prototipos de interfaz permite corregir la aparición de errores en fases tempranas del software, evitando gastos innecesarios (Barros, 2008).

Criterios para la verificación de los requisitos

Luego de realizar la validación de los requisitos, se deben verificar los mismos como parte de la propuesta de solución, con el objetivo de obtener numéricamente una evaluación de dichos requisitos, la cual determinará su aceptación. Para ello fueron utilizados los criterios de

correctitud, completitud y consistencia del "modelo de McCall" por ser criterios de verificación reconocidos y fáciles de aplicar (Bradshaw Venzant, y otros, 2010).

1- Correctitud: la correctitud propone determinar si las especificaciones de requisitos cubren todas las necesidades del negocio y expectativas del cliente. Para aplicar esta métrica es necesario llevar un registro de observaciones del negocio en tiempo de validación, que es el registro encargado de llevar especificidades del negocio tales como la omisión de validaciones, atributos erróneos, restricciones de negocio y omisión de un requisito. La Correctitud **(C)** se obtiene dividiendo el Total de Observaciones (TO) de negocio realizadas mediante la validación, entre el Total de requisitos revisados (T). La misma es aceptable si el valor obtenido es menor estricto que 0.10.

Teniéndose un total de 56 requisitos generados en el presente trabajo y que superaron el proceso de revisión. Teniéndose entonces que:

C = TO/T **C** = 4/56 **C** = 0.071

El resultado obtenido de correctitud es menor que 0.10, por lo tanto, se aceptan las especificaciones de requisitos.

2- Completitud: el criterio es utilizado con el objetivo de comprobar que la especificación de requisitos comprenda descritos todos los aspectos necesarios para la implementación del sistema, a través de un registro de las Observaciones de Formato (OF) ejecutadas; las mismas son generadas en caso de que se hayan omitido secciones requeridas de la plantilla.

La Completitud **(CO)** es aceptable si el valor obtenido es mayor que 0.90 y se obtiene mediante la sustracción a 1 de la división entre el total de observaciones de formato y el Total de Secciones del documento (TS).

CO = 1 - OF/TS.

Al tenerse cero observaciones en el registro de formato, y un total de 6 secciones obligatorias por cada documento de especificación de requisitos, se obtiene la siguiente completitud:

CO = 1 – 0/6 **CO** = 1.

La completitud obtenida es mayor que 0.90, por lo que las especificaciones de requisitos se consideran completas.

3- Consistencia: la consistencia se encarga de determinar que todas las especificaciones de requisitos se guíen por un mismo patrón, para lo cual se crea un registro de observaciones de consistencia, donde se almacenan las observaciones realizadas por contradicciones existentes entre los requisitos. La Consistencia **(CI)** es aceptable si el valor obtenido es menor que 0.20 y se obtiene de la división entre el total de Observaciones de inconsistencia (O) y el Total de requisitos revisados (T).

CI = O/T

Al no detectarse observaciones de inconsistencias durante la etapa de validación de requisitos, para un total de 56 especificaciones de requisitos revisados:

CI = 0/56 **CI** = 0

El resultado obtenido de consistencia es menor que 0.20, por lo tanto, las especificaciones de requisitos se consideran consistentes.

Las métricas aplicadas a las especificaciones de requisitos funcionales, con el objetivo de verificar la validación de los mismos, arrojaron resultados satisfactorios que evidenciaban el cumplimiento de la completitud, la correctitud y la consistencia en los mismos.

2.6 Análisis y diseño de la propuesta de solución

2.6.1 Patrones de diseño

Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifica y describe maneras de solucionar problemas que ocurren frecuentemente en el desarrollo. Se agrupan de la siguiente manera: Patrones de Software para Asignación de Responsabilidades, del inglés General Responsability Assignment Software Patterns (GRASP) y patrones de la banda de los cuatro, del inglés Gang of Four (GOF) (Larman, 2003).

Ambos grupos son usados en la implementación de la solución propuesta y se ilustrarán a través de algunos escenarios a implementar.

Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman, 2003). Dentro de este grupo se encuentran los siguientes patrones:

Patrón controlador: un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa .El patrón controlador es el encargado de asignar la responsabilidad de controlar el flujo de eventos del sistema, a una clase específica. (Larman, 2003) .Cuando un evaluador presiona el botón "*realizarEvaluación*", está generando un evento sistémico que indica que “se debe realizar una evaluación”. La mayor parte de los sistemas reciben eventos de entrada externa, los cuales generalmente incluyen una interfaz gráfica para el usuario operado por una persona. La misma clase controlador debería utilizarse con todos los eventos sistémicos de un escenario, de modo que se conserve la información referente al estado del mismo.

A continuación se muestra en el siguiente diagrama de clases, el flujo de eventos para realizar la evaluación aplicando el patrón Controlador.

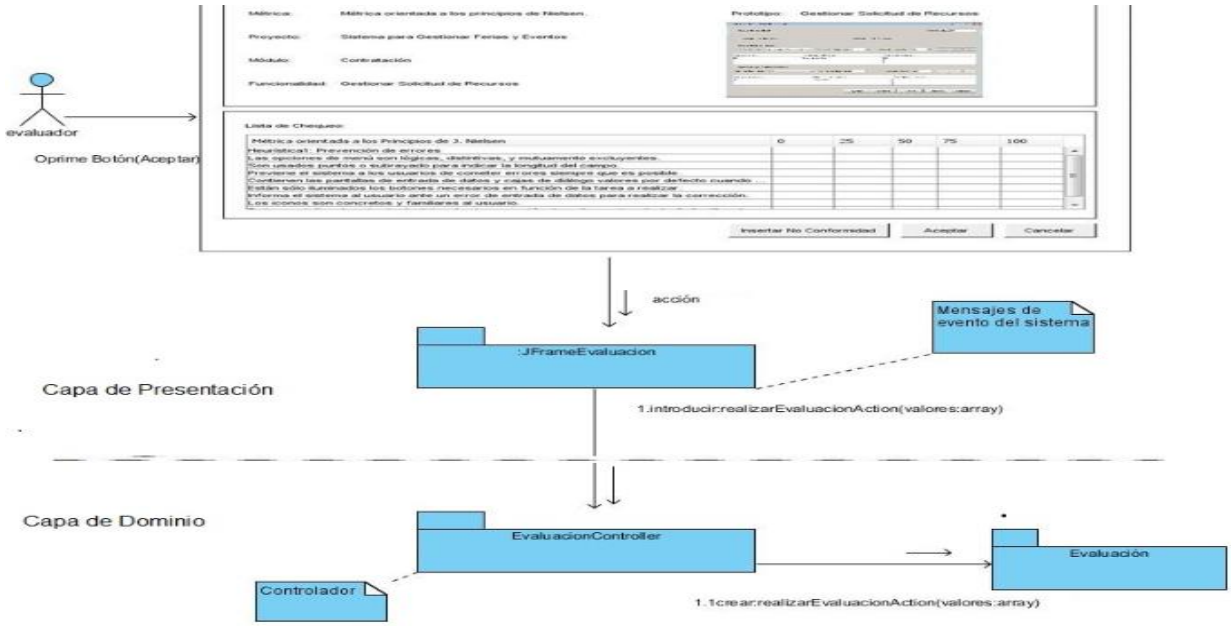

Figura 17. *Patrón controlador utilizado en el escenario "Realizar Evaluación".*

Patrón experto: un Modelo de Diseño podría definir cientos o miles de clases software, y una aplicación podría requerir que se realicen cientos o miles de responsabilidades (Larman, 2003). Durante el diseño de objetos, cuando se definen las interacciones entre los objetos, se toman decisiones sobre la asignación de responsabilidades a las clases de software con el propósito de que los sistemas tiendan a ser más fáciles de entender, mantener y ampliar, y así existen más oportunidades para reutilizar componentes en futuras aplicaciones. Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele ser útil en el diseño orientado a objetos (Larman, 2003)**.**

En el escenario 'Realizar Evaluación', se comienza asignando las responsabilidades con una definición clara de ellas. A partir de esta recomendación se plantea la pregunta: ¿Quién es el responsable de proveer el grado de usabilidad de un prototipo? Desde el punto de vista del patrón Experto, busca la clase de objetos que posee la información necesaria para obtener la evaluación del prototipo, en este caso es la clase entidad Evaluación, a la cual se le agregaría el método *"realizarEvaluacion"*. El patrón trae como beneficio que el comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase "sencillas" y más cohesivas que son más fáciles de comprender y de mantener así se brinda soporte a una alta cohesión y además se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide (Larman, 2003). Se muestra en la Figura 18, la entidad que contiene la información de la evaluación de la usabilidad.

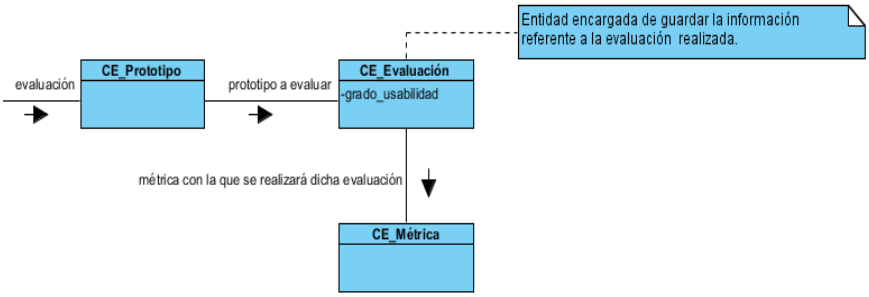

Figura 18. *Patrón experto utilizado en el escenario "Realizar Evaluación".*

Patrón creador: la creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación (Larman, 2003). Si se asignan bien, el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización. El patrón ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La intención básica del patrón Creador es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Eligiéndose como el creador se favorece el bajo acoplamiento (Larman, 2003). En la aplicación UsaProt, ¿quién debería ser el responsable de la creación de una instancia de *Evaluación*? según el patrón Creador, busca las clases que agregan y contienen, instancias de *Evaluación.* El patrón Creador sugiere que la clase EvaluacionController es buen candidato para tener la responsabilidad de la creación de la instancia *Evaluación* puesto que este contiene (de hecho, agrega) el objeto de *Evaluación.*

A continuación se evidencia el cumplimiento del patrón en la Figura 19.

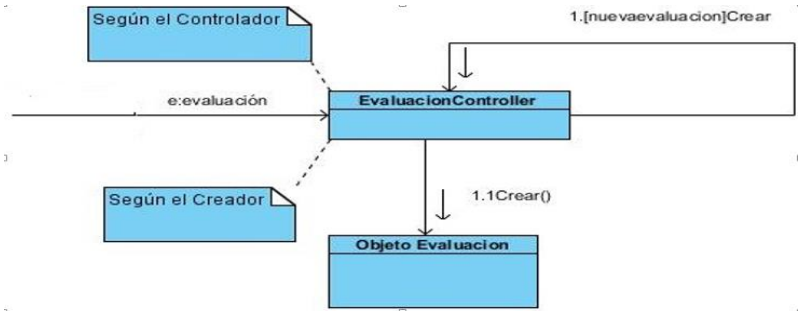

Figura 19. *Patrón Creador utilizado en el escenario "Realizar Evaluación".*

Patrón bajo acoplamiento: el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Acoplamiento bajo significa que una clase no depende de muchas clases. Acoplamiento alto significa que una clase recurre a muchas otras clases por lo que presenta los siguientes problemas: los cambios de las clases afines ocasionan cambios locales, difíciles de entender cuando están aisladas y difíciles de reutilizar puesto que dependen de otras, para resolver estos problemas se asigna una responsabilidad para mantener bajo acoplamiento.

El patrón plantea la baja dependencia que debe existir entre las clases. Además el grado de acoplamiento no puede considerarse aisladamente de otros principios como Experto y Alta Cohesión. Sin embargo, es un factor a considerar cuando se intente mejorar el diseño (Larman, 2003). Se tiene la necesidad de crear una instancia de *Proyecto*, el patrón Creador sugiere la creación de esta instancia en la clase ProyectoController, por los elementos anteriormente mencionado cuando se analizó este patrón, aunque esta clase tiene una relación con ModuloController y FuncionalidadController, la asignación de responsabilidad que se sugiere no aumenta acoplamiento entre las clases. Desde el punto de vista puramente del acoplamiento, es preferible realizar la instanciación en la clase ProyectoController ya que se mantiene el acoplamiento global más bajo entre estas tres clases. Se muestra lo anteriormente mencionado.

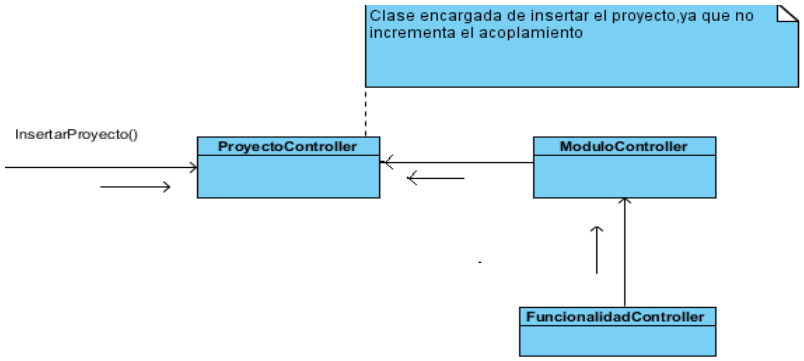

Figura 20. *Patrón bajo acoplamiento utilizado en el escenario "InsertarProyecto".*

Patrón alta cohesión: la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una baja cohesión hace un conjunto de elementos no afines o realiza trabajo excesivo. Presenta los siguientes problemas: son difíciles de comprender, difíciles de reutilizar, difíciles de conservarlas, afectan constantemente los cambios, para resolver estos problemas asignar una responsabilidad de modo que la cohesión siga siendo alta. El patrón tiene como propósito asignar responsabilidades de manera que la relación entre las clases siga siendo alta. (Larman, 2003). El ejemplo que se analizó anteriormente soporta tanto bajo acoplamiento como alta cohesión ya que el diseño delega la responsabilidad de "*InsertarProyecto"* a la clase ProyectoController que favorece una cohesión muy alta a entre las tres clases. Igual que el Bajo Acoplamiento, el patrón de alta cohesión es un principio a tener en cuenta durante todas las decisiones de diseño. Es un principio evaluativo que aplica un diseñador mientras evalúa todas las decisiones de diseño.

Patrones GOF

Los patrones de diseño del grupo de GoF clasifican en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Larman, 2003).

- *Categoría estructurales(a nivel de clases como a nivel de objetos):*

Decorador: añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender funcionalidad, de tal modo que sea transparente a sus clientes (Larman, 2003). El patrón viene implícito en el marco de trabajo Symfony2 haciendo uso del motor de plantillas Twig. La parte más poderosa de Twig es la herencia entre plantillas, que permite crear un "esqueleto" de plantilla base que contenga todos los elementos comunes del sistema y define los bloques que las plantillas descendientes pueden sustituir. Esta plantilla es el archivo base.html.twig, que define un simple documento donde se almacena el código HTML que es común a todas las páginas del sistema, para no tener que repetirlo en cada una de ellas.

En los conjuntos de utilidades para interfaces gráficas de usuario se permite añadir a los componentes visuales propiedades (por ejemplo, un borde) o comportamientos (por ejemplo, una barra de desplazamiento). Se muestra a continuación la utilización del patrón en el sistema haciendo uso de la plantilla twig en el archivo base.html.twig.

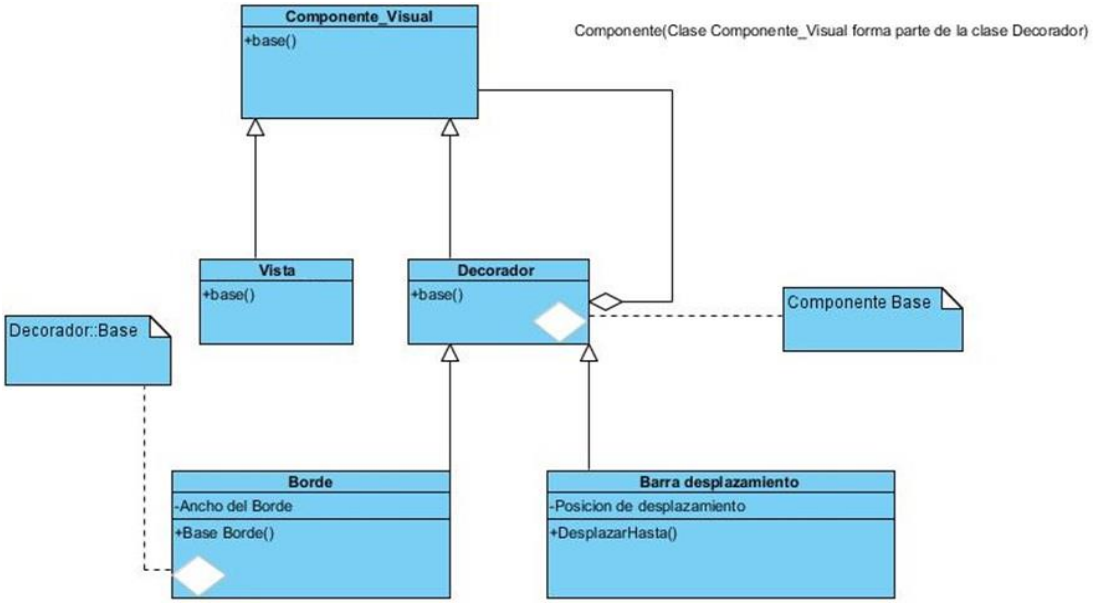

Figura 21. *Patrón Decorador utilizado en la generación de plantillas Twig.*

- *Categoría creacionales(a nivel de objetos):*

Fábrica abstracta: su propósito es definir una interfaz para la creación de familias de objetos relacionados o dependientes sin tener que especificar la clase concreta (Larman, 2003) .La fábrica abstracta se utiliza para la creación de controladores dentro del mecanismo de rutas del marco de trabajo Symfony 2. Las páginas del sistema UsaProt, que pertenece al escenario "GestionarEvaluacion" se corresponden de forma directa con el objeto de tipo *Page* del modelo, que define el título y el contenido que se muestran.

Para asociar la URL con su correspondiente objeto *Page*, se crea una nueva ruta de objetos de tipo *sfDoctrineRoute* y que hace uso del campo *slug* esta variable es la que utiliza el controlador para buscar en la base de datos la *Evaluación* solicitada. Luego se busca

automáticamente en la base de datos un objeto de tipo *Page* cuyo campo *slug* coincida con el que incluye la URL. La ruta que creada se asocia correctamente a la página http://localhost/AppUltima/web/App.dev.php/evaluacion. Se potencia el encapsulamiento, puesto que se aísla a los clientes de las implementaciones y se obtiene un incremento de la flexibilidad del diseño (Larman, 2003).

- *Categoría de comportamiento(a nivel de objetos):*

Observador: permite a los objetos captar dinámicamente las dependencias entre objetos, de tal forma que un objeto notificará a los objetos dependientes de él cuando cambia su estado, siendo actualizados automáticamente (Larman, 2003). El patrón observador se utiliza en jQuery librería de JavaScript este utiliza la programación orientada a objetos de JavaScript para crear nuevos objetos, los cuales tienen a su vez propiedades y métodos que son los que crean una nueva manera de programar. En la vista que pertenece a la funcionalidad "realizarEvaluacion" se necesita modificar dicha vista, internamente sucede lo siguiente: El principal objeto utilizado es el objeto jQuery () el cual actúa directamente sobre los elementos de la página (elementos del DOM (Modelo Objeto Documento)), seleccionándolos y modificándolos con sus métodos y propiedades. El objeto le notifica a los objetos dependientes de el sobre la modificación que ocurrió y estos automáticamente también cambiarán su estado.

2.6.2 Arquitectura del sistema

La Arquitectura de Software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema (Larman, 2003). Tomando en consideración los principios y la arquitectura que propone el marco de trabajo Symfony2, para el desarrollo del sistema se define una arquitectura en capas basada en el patrón MVC (Modelo-Vista-Controlador).

El siguiente esquema muestra de forma más simplificado del funcionamiento interno de Symfony2, con algunos de los componentes que conforman cada una de las capas de esta arquitectura. El escenario que se muestra es relacionado con la funcionalidad "Realizar Evaluación".

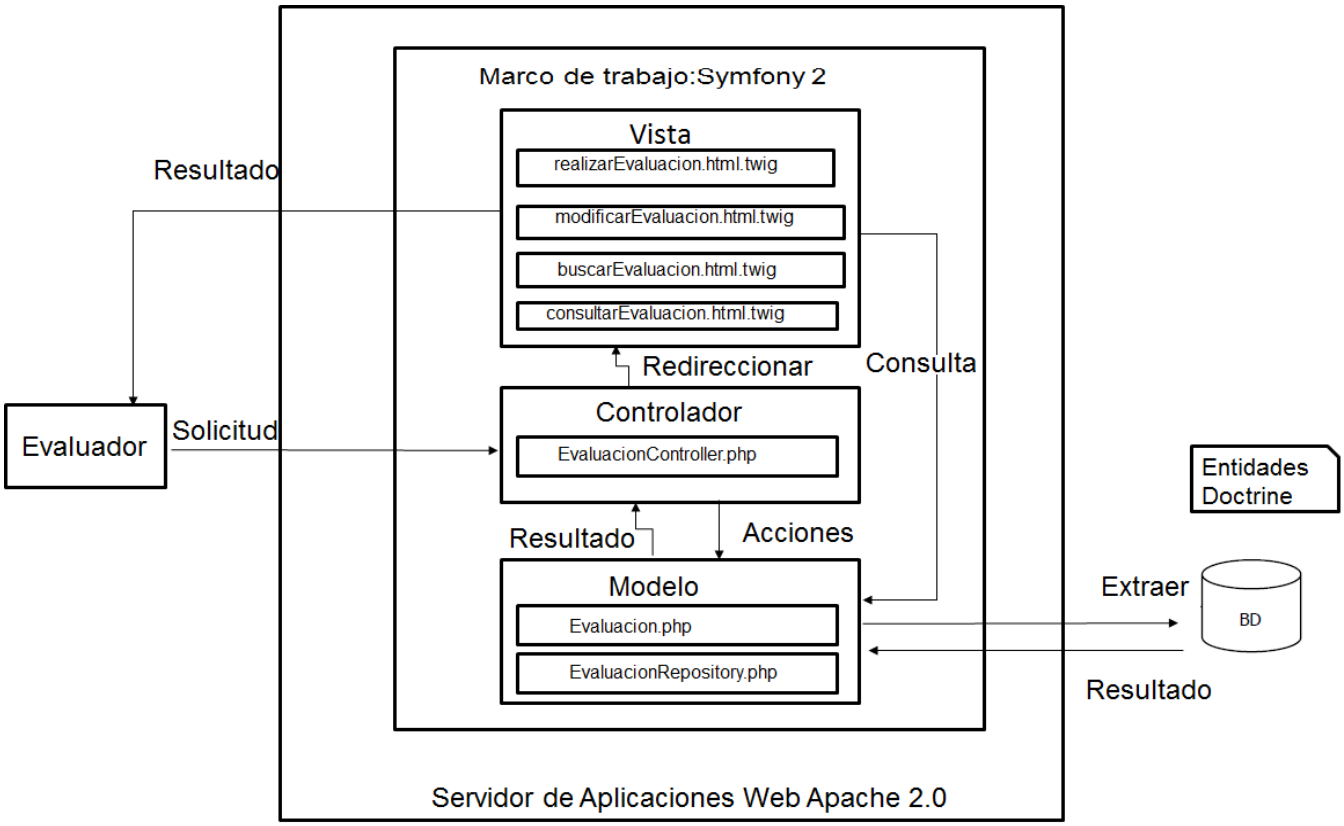

Figura 22. *Esquema simplificado de la arquitectura interna del sistema.*

A continuación se describe cuando el usuario realiza la evaluación, internamente sucede lo siguiente:

1. El cliente (Evaluador) realiza una solicitud sobre la funcionalidad que se desea ejecutar "Realizar Evaluación", esta solicitud es procesada por el servidor de aplicaciones quien posee un enrutador para determinar el controlador responsable (controlador: EvaluaciónController.php).

2. El sistema de enrutamiento determina el controlador en este caso es EvaluacionController el cual está asociado con la página del sistema. Symfony2 ejecuta el **Controlador** asociado a la página. El controlador puede contener código donde se instancie la entidad Evaluación.php que es la forma en que Symfony2 define los **Modelos**.

3. El **Controlador** solicita al **Modelo** los datos. El modelo es una representación de los atributos y estados de algún Objeto, en Symfony2 generalmente son alguna clase PHP especializada en obtener información, normalmente de una base de datos (el modelo está formado por las entidades de Doctrine).

4. Con los datos devueltos por el **Modelo**, el **Controlador** solicita a la **Vista** que cree una página mediante una plantilla y que gestione los datos del **Modelo**.

5. El **Controlador** entrega al servidor la página creada por la **Vista**.

2.6.3 Patrones arquitectónicos

Resuelven problemas arquitectónicos, adaptabilidad a requerimientos cambiantes, modularidad y acoplamiento. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones y una serie de recomendaciones para organizar los distintos componentes. (Larman, 2003)

Patrón MVC

El patrón separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Frecuentemente se evidencian en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es la capa del dominio y el controlador representa la lógica de negocio (Larman, 2003). A continuación se muestra en la Figura 23 la organización propuesta por Symfony 2.

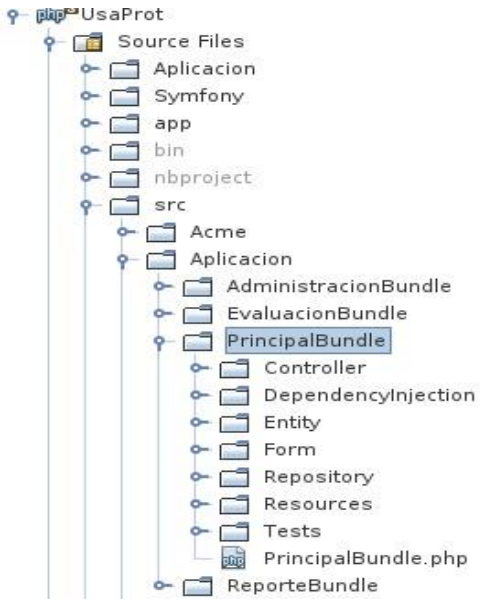

Figura 23. *Organización propuesta por Symfony2.*

Modelo

Representa la información con la cual el sistema opera. La capa Modelo está dividida en: capa de acceso a datos y capa de abstracción de base de datos. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos (Larman, 2003) .A **la capa de acceso a datos** pertenece la claseRepository que se encuentran en el paquete "src/Aplicacion/PrincipalBundle/Repository/". La clase Repository es donde se encuentran las consultas a la base de datos, aislándolas del modelo. En esta capa también se encuentra el ORM Doctrine como marco de trabajo que, mediante su lenguaje propio de consultas DQL (Lenguaje de Consultas Doctrine), permite la independencia de la aplicación respecto al gestor de base de datos. Se muestra a continuación la estructura anteriormente mencionada:

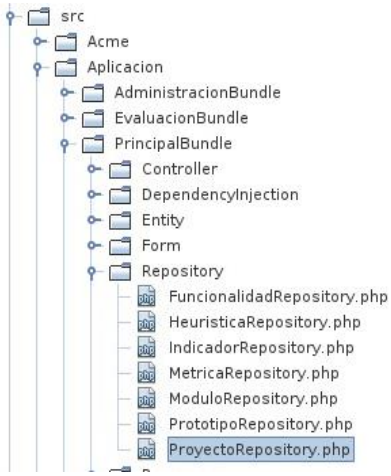

Figura 24. *Clases Repository.*

A la **capa de abstracción de base de datos** pertenecen las entidades, que representan las tablas de la base de datos previamente mapeadas por Doctrine; estas se encuentran en el paquete "Entity" donde se colocan las dependencias de terceros, en "src/Aplicacion/PrincipalBundle/Entity/".

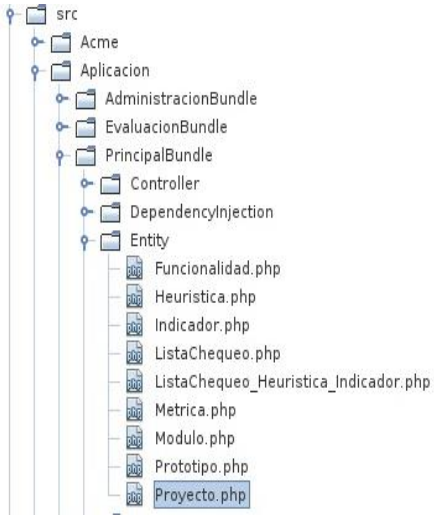

Figura 25. *Entidades.*

Vista

Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario (Larman, 2003). La plantilla base.html.twig que se encuentran dentro del paquete"/app/Resources/views", en conjuntos con las plantillas que se encuentran en "/src/Aplicacion/PrincipalBundle/Resources/views/" son las que responden a esta capa.

Estas transforman los datos obtenidos del modelo en vistas que son las que permiten al usuario interactuar con la aplicación. Para hacer esto se apoyan en la utilización del motor de plantillas Twig y la librería Bootstrap y J-Query, que facilita la construcción de interfaces bien acabadas y agradables a la vista del usuario.

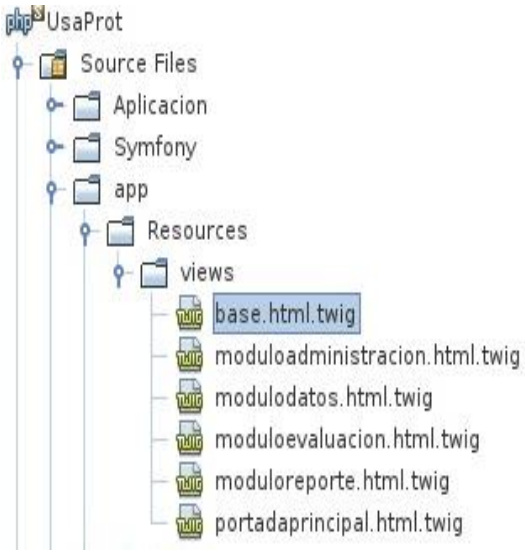

Figura 26. *Plantilla Base.*

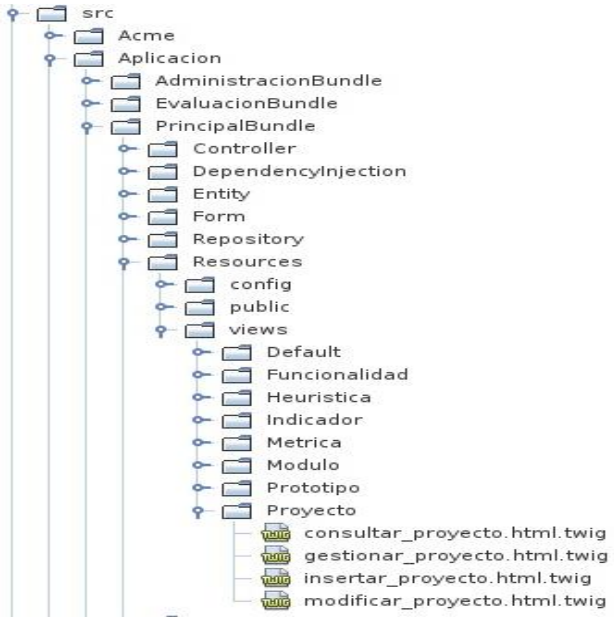

Figura 27.*Capa de presentación Datos Principales.*

Controlador

Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. El Controlador recibe las solicitudes o peticiones realizadas por el usuario desde la vista, ejecuta las acciones pertinentes y devuelve una respuesta a la interfaz con los resultados de las operaciones realizadas (Larman, 2003).

Estas clases se encuentran dentro del paquete "/src/Aplicacion/PrincipalBundle/Controller" como se aprecia en la Figura 28 y constituyen la parte de la aplicación que se encarga de realizar una funcionalidad completa y específica, además en ellos se establecen las reglas que deben cumplirse.

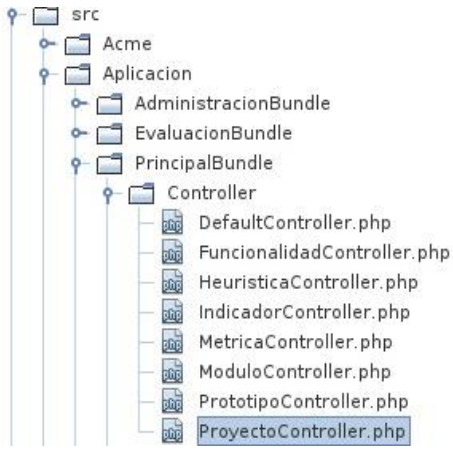

Figura 28. *Controlador en Datos Principales.*

2.6.4 Diagrama de clases del diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas (Larman, 2003).

En el siguiente diagrama de clases se muestra el escenario *"Gestionar Evaluación"*. Entre las principales clases que se representan se encuentran:

- La clase Realizar Evaluación.html.twig que representa la capa de presentación o vista del patrón MVC. Es la interfaz con la que el usuario interactúa, mostrándole los datos necesarios y las acciones q puede llevar a cabo.
- La clase controladora "EvaluacionController", encargada de las funcionalidades que debe realizar el escenario anterior.
- La clase entidad "CE_Evaluación" encargada de almacenar la información referente a una *Evaluación* la cual se utiliza para dar cumplimiento a las funcionalidades del escenario anterior.

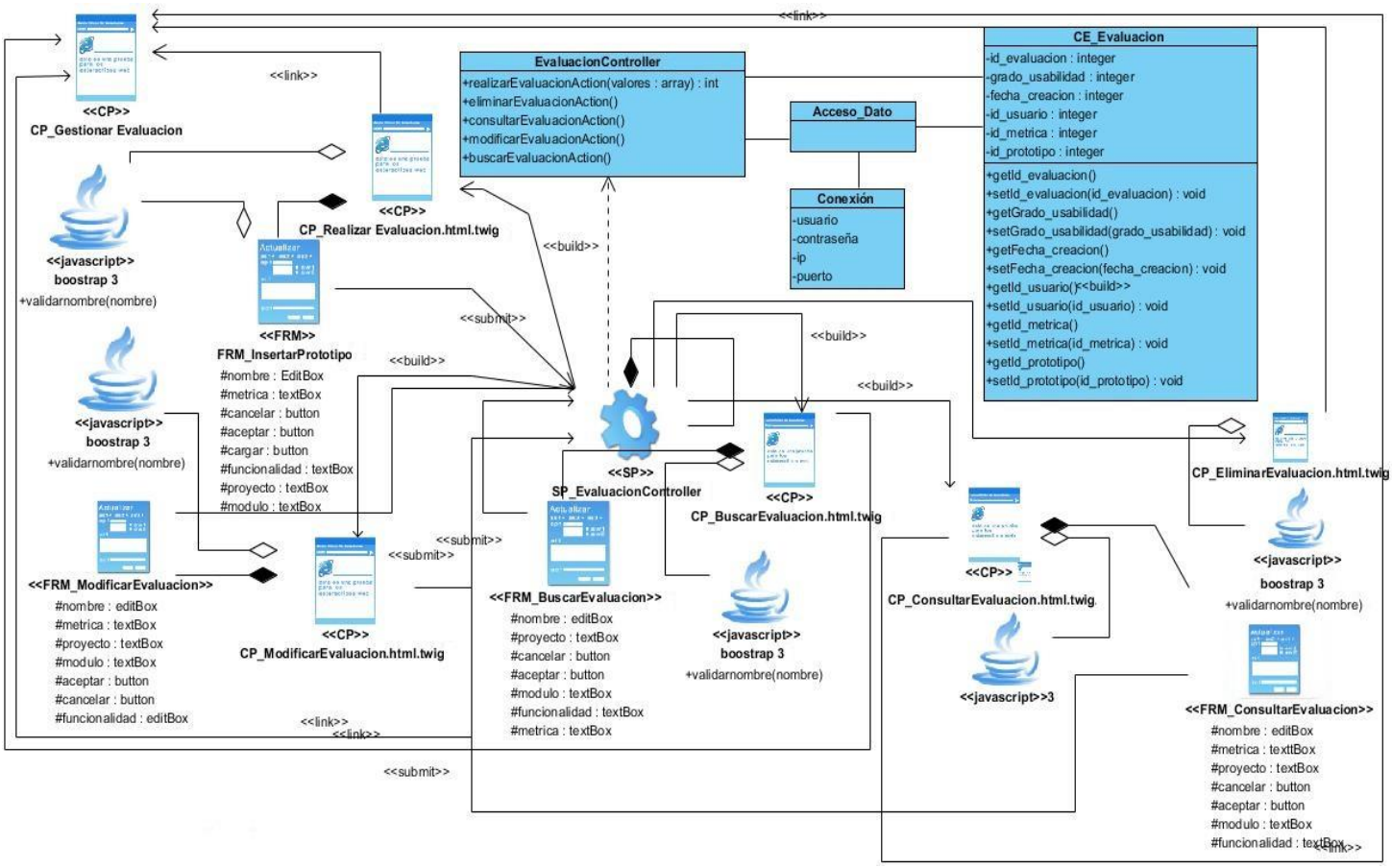

Figura 29. *Diagrama de clases de diseño. Escenario Gestionar Evaluación.*

Los restantes diagrama de clases pertenecientes a los escenarios del sistema se encuentra en el anexo 4.

2.6.5 Diagrama de interacción

Los diagramas de secuencia muestran las interacciones entre objetos, ordenadas en secuencia temporal y durante un escenario concreto (Larman, 2003). Estos diagramas son importantes para modelar los aspectos dinámicos de un sistema. A continuación se muestra el diagrama de secuencia para el escenario "*Gestionar Evaluación"* evidenciando la funcionalidad *"Realizar Evaluación".*

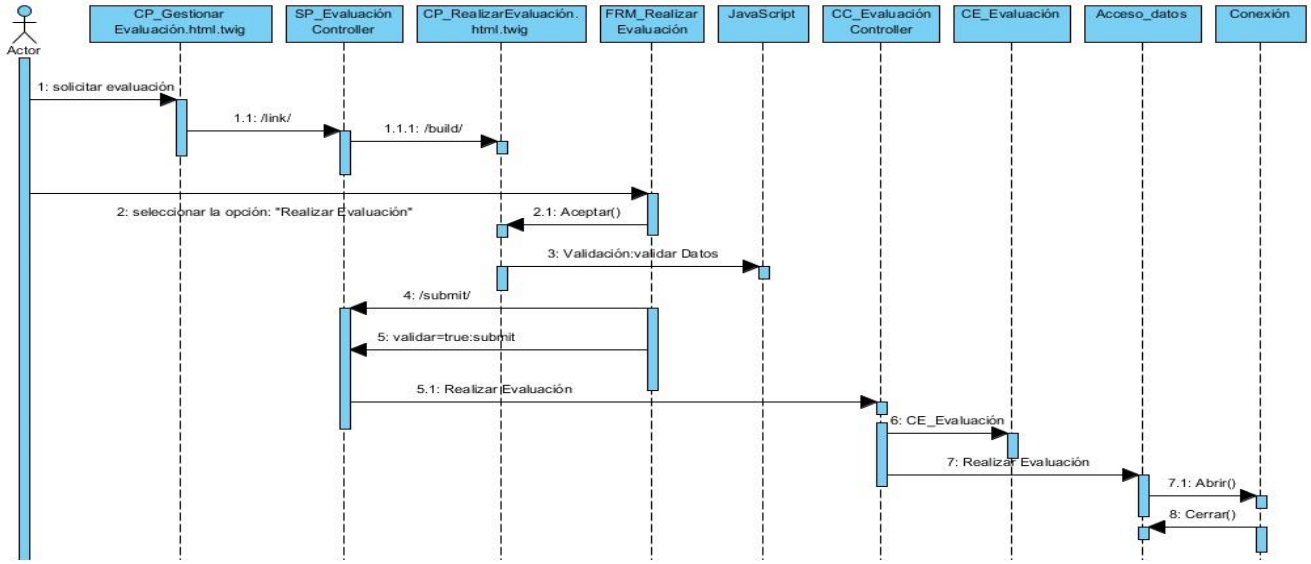

Figura 30. *Diagrama de interacción. Escenario Realizar Evaluación.*

Los restantes diagrama de interacciones pertenecientes a los escenarios del sistema se encuentra en el anexo 5.

2.6.6 Modelo de datos

Es un conjunto de conceptos que permiten describir los datos, las relaciones entre ellos, la semántica y las restricciones de consistencia(González, 2009).

El modelo de datos se encuentra en tercera Forma Normal (FN). Se describe el modelo de datos, cada una de las tablas y sus atributos.

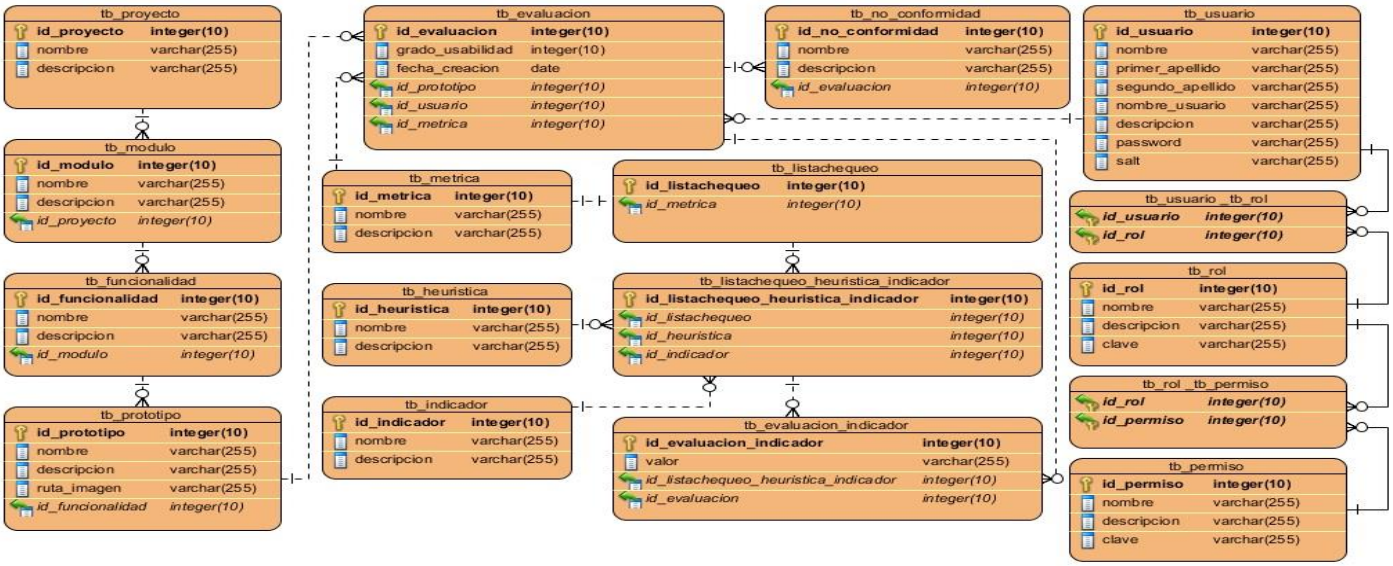

Figura 31. *Modelo de Datos.*

Descripción de las tablas

A continuación se muestran algunas de las descripciones de las entidades de la base de datos. En el anexo 6 se encuentran las restantes descripciones.

Tabla 5. *Descripción de las tablas de la Base datos.*

Nombre	Descripción
tb_proyecto	**Entidad que almacena la información de los proyectos que serán evaluados.**
tb_modulo	**Entidad que almacena la información de los módulos pertenecientes a los proyectos que serán evaluados.**
tb_funcionalidad	**Entidad que almacena la información de las funcionalidades pertenecientes a los módulos que serán evaluados.**
tb_prototipo	**Entidad que almacena la información de los prototipos pertenecientes a los proyectos que serán evaluados.**
tb_evaluacion	**Entidad que almacena la información de la evaluación realizada, donde se puede conocer el grado de usabilidad que se obtuvo.**
tb_metrica	**Entidad que almacena la información de la métrica seleccionada, para realizar la evaluación y obtener el grado de usabilidad.**

2.6.7 Validación del diseño del sistema

Para medir el diseño se utilizaron métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objeto, las cuales se clasifican en TOC (Tamaño Operacional de Clase) y RC (Relaciones entre Clases) (Pressman, 2010).

TOC: plantea que el tamaño general de una clase se puede determinar a partir de la suma del número de atributos que están encapsulados en la clase, más el número total de operaciones también encapsuladas (Pressman, 2010).

RC: está dado por el número de relaciones de uso entre clases y evalúan un conjunto de atributos de calidad (Pressman, 2010).

A continuación se muestra las clases del sistema evaluadas a través de las métricas:

Métrica TOC

Tabla 6. *Clases evaluadas según la métrica TOC.*

Proyecto	Prototipo	Indicador	Usuario
Módulo	Métrica	Evaluación	Rol
Funcionalidad	Heurística	No_Conformidad	Permiso
ProyectoController	PrototipoController	IndicadorController	RolController

MóduloController	MétricaController	EvaluaciónController	PermisoController
FuncionalidadController	HeurísticaController	UsuarioController	No_ConformidadController

Se trabajó con un total de 24 clases para un promedio de 10.02 de procedimientos para cada clase .En la figura se muestra una gráfica con agrupaciones por intervalos de la cantidad de clases según la cantidad de procedimientos.

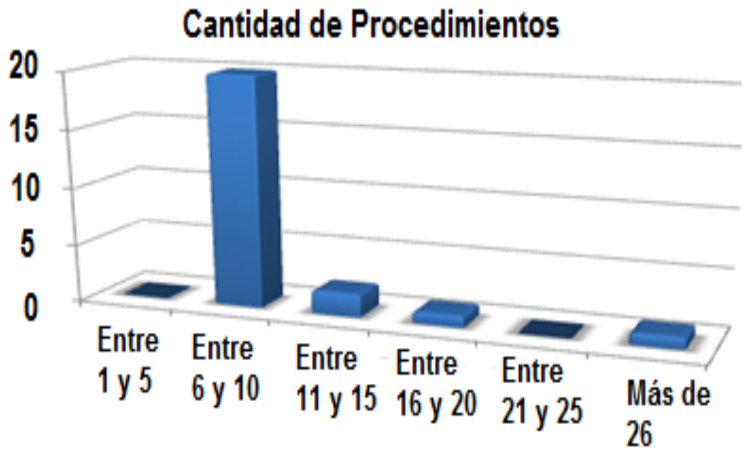

Figura 32. *Representación de la Cantidad de clases agrupadas en intervalos según la cantidad de procedimientos.*

El gráfico anterior muestra que existen entre 6 y 10 procedimientos asignados a cada clase, para un total de 20 clases que cumplen con dicho criterio, lo que implica que el diseño de las clases en cuanto a cantidad de funcionalidades por cada una es baja.

La métrica TOC evalúa atributos de calidad, el resultado se muestra a continuación:

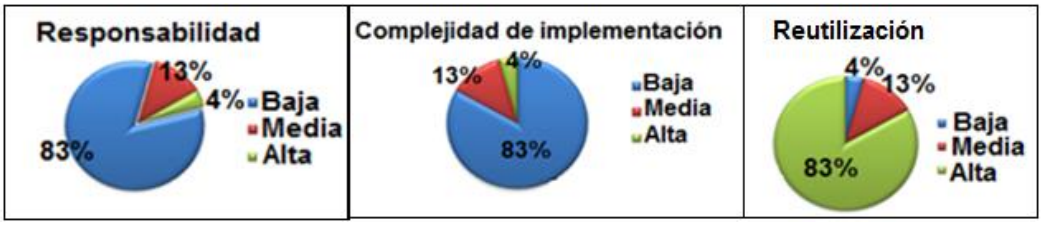

Figura 33. *Representación en porcientos (%) de los datos obtenidos de los atributos a medir.*

Se evidencia que las clases del sistema poseen:

- **Baja responsabilidad:** como existe una disminución del TOC, esto implica una baja responsabilidad asignadas a las clases.
- **Baja complejidad**: una disminución del TOC, implica una baja complejidad de implementación de las clases.

- **Alta reutilización**: una disminución del TOC, implica un alto grado de reutilización de la clase.

Tras aplicar dicha métrica se obtienen resultados positivos según los parámetros de calidad propuestos para esta métrica, arrojando como consecuencia que en el diseño realizado las clases tienen bajas responsabilidades, aumentando la reutilización y facilitando la implementación y las pruebas.

Métrica RC

Se muestra en la siguiente gráfica con agrupaciones por intervalos de la cantidad de clases según las dependencias entre ellas.

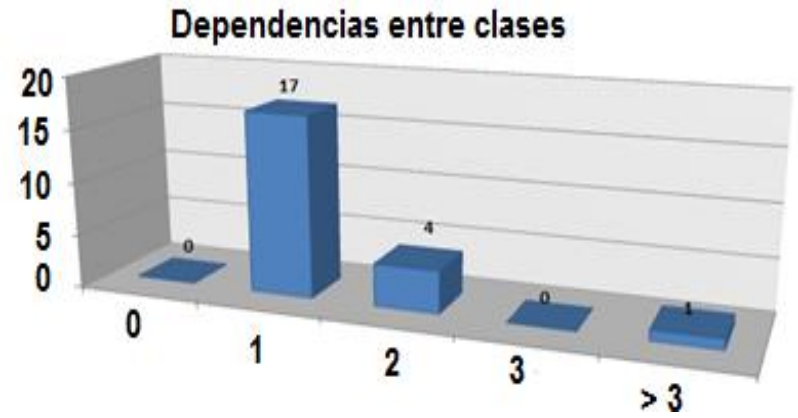

Figura 34. *Intervalos de las clases agrupadas según las dependencias entre ellas.*

El gráfico muestra que 17 clases poseen una sola dependencia, lo que implica que el número de relaciones de uso de una clase con otra es buena.

La métrica RC evalúa los siguientes atributos de calidad, el resultado se muestra a continuación:

Figura 35. *Representación en porcientos (%) de los datos obtenidos de los atributos a medir.*

Considerando que la media de relaciones entre clases es 1.83 y que un alto por ciento de las clases (77.27%) tienen un número de relaciones menor que este valor, se puede afirmar que el modelo presenta un **bajo nivel de colaboraciones** por clase. De igual forma muestra un **bajo nivel de complejidad de mantenimiento**, una **baja cantidad de pruebas** y **un alto nivel de reutilización**, evidenciando la realización de un mínimo de pruebas

sobre el modelo, la posibilidad de reutilización a gran escala y de un mantenimiento poco complejo.

Todo lo anteriormente descrito demuestra que las clases del sistema no tienen sobrecargas de responsabilidades por lo que permite la reutilización demostrando que el sistema tendrá una fácil implementación, pudiéndose afirmar que los resultados obtenidos por las métricas son positivos.

2.7 Conclusiones del Capítulo

Con la realización del capítulo, se logró llevar a la práctica lo estudiado en el capítulo anterior, mostrándose los elementos esenciales de la arquitectura definida para la presente investigación, en concordancia con la estructura organizativa propuesta por Symfony2.

Con el análisis y diseño de la solución se obtuvieron 56 requisitos funcionales para el sistema y se generaron los artefactos definidos por la metodología utilizada, obteniéndose entre ellos la especificación de requisitos de software, los diagramas de clases, secuencia, prototipos de interfaz gráfica de usuario, entre otros.

Además se evidencia el uso de los patrones de diseño utilizados como son: controlador, creador, experto, alta cohesión, bajo acoplamiento, fábrica abstracta, observador y decorador que permitieron obtener una propuesta de solución que respondió a las necesidades de desarrollo de la herramienta UsaProt.

Capítulo 3: Implementación y prueba

3.1 Introducción

El capítulo está orientado a la implementación y prueba de la solución propuesta. Se hará referencia al modelo de implementación, el cual está formado por los diagramas de componentes y el de despliegue. Además se verifica la calidad del resultado de la implementación, mediante los artefactos generados durante el flujo de trabajo de pruebas, exponiendo los resultados obtenidos por diferentes tipos de pruebas realizadas a la herramienta.

3.2 Modelo de implementación

3.2.1 Diagrama de componentes

Un diagrama de componentes contiene obviamente componentes, interfaces y relaciones. Los diagramas de componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura del sistema, es decir para describir la vista de implementación estática de un sistema (Schmuller, 2000).

La herramienta UsaProt en forma general incluye los siguientes componentes: **Ctr_Frontal** es el controlador frontal que recibe cada petición realizada al sistema, **Config** (para las configuraciones), **Routing** (para las rutas de acceso), **Security** (para la seguridad), **Doctrine** (para el manejo de las entidades de la base de datos) y **Bootstrap** (para la creación de las vistas).

Los módulos que conforman el sistema están compuestos por tres paquetes fundamentales, **Controlador** (encargado de dar respuesta a las peticiones de los usuarios), **Modelo** (contiene las clases entidades de la herramienta y lo relacionado con la lógica de negocio), **Vista** (encargado de la construcción y manejo de las interfaces de usuario). El paquete Controlador utiliza el componente Controladoras, para dar respuesta a las peticiones realizadas por los usuarios. El paquete Modelo utiliza los componentes Entidades, que a su vez es usado por el componente Repositorio el cual contiene las clases repositorios encargadas de realizar las consultas a la base de datos. El paquete Vista incluye los componentes Form (encargado del trabajo con los formularios), Twig (para las interfaces de usuarios y plantillas del módulo) y Extensiones (posee los JavaScript y CSS).

A continuación se muestra el diagrama de componentes de la herramienta UsaProt en específico para el módulo *Evaluaciones.*

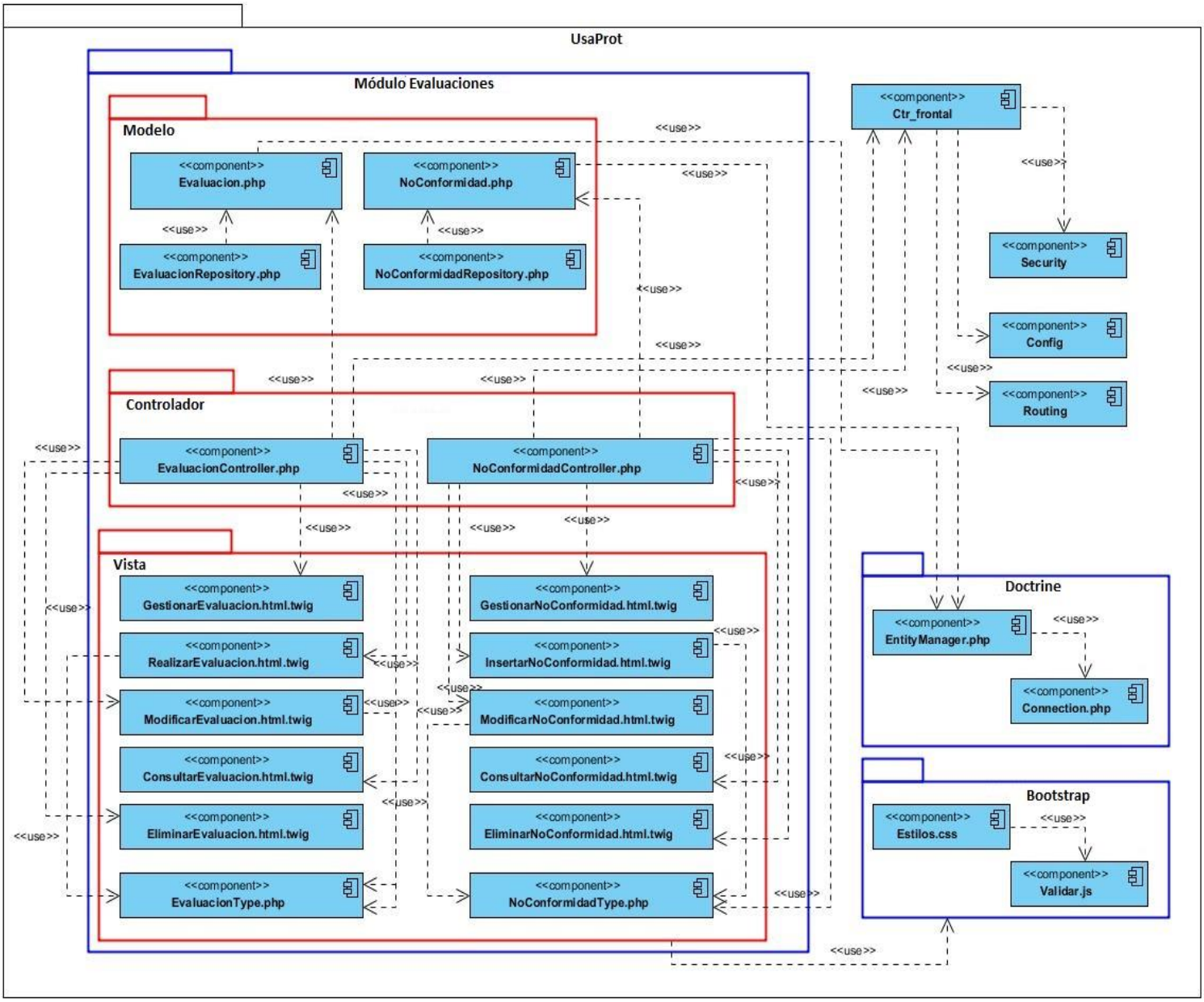

Figura 36. *Diagrama de componentes del módulo Evaluaciones.*

3.2.2 Diagrama de despliegue

Para comprender como se ejecutará a nivel de hardware un sistema desarrollado y tener una visión clara de la estructura del sistema en ejecución y las relaciones entre los componentes que interactúan en el mismo, se realiza el diagrama de despliegue (Schmuller, 2000). El diagrama tiene como objetivo representar la disposición de las instancias de los componentes de ejecución, en instancias de nodos conectados por enlaces de comunicación.

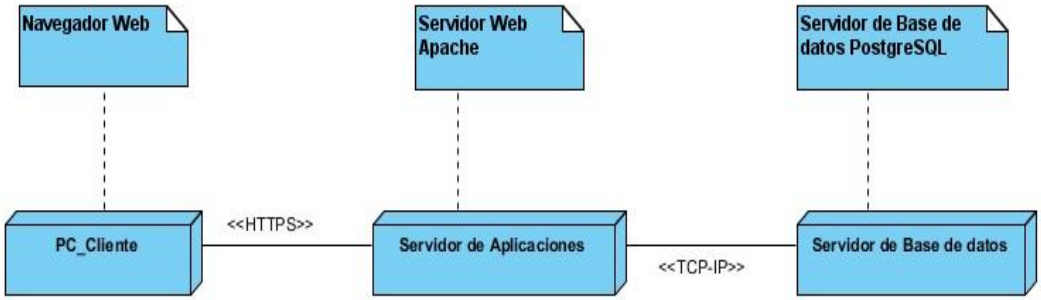

Figura 37. *Diagrama de Despliegue.*

Los componentes que integran el diagrama de despliegue se definen de la siguiente manera:

PC_Cliente: contendrá el sistema operativo Linux o Windows y el navegador web Mozilla Firefox mediante el cual los clientes tendrán acceso al sistema y harán uso del mismo.

Servidor de aplicaciones: se encontrará los archivos a través de los cuales el usuario logra acceder al sistema.

Servidor de base de datos: se encontrará la base de datos con la información del sistema.

3.3 Estándares de codificación

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Además si se aplica de forma continuada un estándar de codificación bien definido caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener.

- Los nombres deben ser descriptivos y concisos. No usar grandes frases o pequeñas abreviaciones.
- Los nombres de las páginas twig serán en minúscula siempre y underscore, por ejemplo: realizar _evaluación.
- Clase: en mayúscula. eje: Evaluación.
- Atributos: minúscula siempre y underscore eje: porciento_usabilidad.
- Funciones: empiezan con minúscula y continúan en mayúscula: realizarEvaluación().
- Cada atributo debe tener activo sus métodos getter y setter, además de los generados producto de relaciones.
- Se exceptúan el uso de las tildes.

3.4 Validación de la solución

La validación es el proceso de comprobar que lo que se ha especificado es lo que el usuario realmente desea. Como parte del proceso de desarrollo del software, la fase de pruebas añade valor al producto que se maneja. Las pruebas de software son los procesos que verifican y revelan la calidad de un producto de software (Bevan, 2009).

La estrategia de validación a seguir medirá dos variables: el grado de usabilidad y la herramienta UsaProt. Para la validación de la solución propuesta se utilizarán los métodos de prueba de Caja Negra y Caja Blanca. Para validar el **grado de usabilidad** se utilizará el método de Caja Negra donde se realizará las pruebas funcionales con la liberación de la herramienta usando la técnica partición equivalente con el objetivo de comprobar que la herramienta devuelva correctamente el grado de usabilidad. Para la validación de la **herramienta UsaProt** se tomará el método de Caja Blanca usando la técnica complejidad ciclomática. Además para validar la herramienta se usarán otros métodos como son las pruebas de usabilidad haciendo uso de la técnica evaluación heurística, y por último pero no menos importante la evaluación del placer de uso con la técnica EmoCard.

3.4.1 Método de Caja Negra. Pruebas funcionales

Las pruebas funcionales son aquellas que se llevan a cabo sobre la interfaz del software sin prestar atención al código, por lo que los casos de prueba son creados con el objetivo de demostrar que la entrada es aceptada de forma adecuada y que se produce una salida correcta. El diseño de estas pruebas se realiza con la intención de detectar funciones incorrectas o ausentes, errores en accesos a bases de datos externas, errores de interfaz, errores de rendimiento, y errores de inicialización y de terminación. Dentro de la prueba se incluyen la técnica de partición de equivalencia que será la empleada en la validación. Esta técnica divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba (Pressman, 2010).

Se realizaron los diseños de casos de prueba para validar la propuesta de solución implementada para comprobar que la herramienta devuelva correctamente el grado de usabilidad. A continuación se muestra parte de la especificación el caso de prueba para el requisito “*Realizar evaluación*”, los restantes casos de prueba se encuentran en el anexo 7.

Escenario	Descripción	Métrica	Proyecto	Módulo	Prototipo	Funcionalidad	0,10,50,75,100	Respuesta del sistema	Flujo central
EC 1.1 Realizar evaluación de acuerdo a la importancia de las heurísticas. Datos correctos.	*Se registran los datos de una evaluación*	*V*	*V*	*V*	*V*	*V*	*V*	*Muestra el grado de usabilidad del prototipo evaluado*	*1. Selecciona Gestionar evaluación 2. Clic en Aceptar.*
EC 1.2 Realizar evaluación de acuerdo a la importancia de las heurísticas. Datos incompletos.	*Existen campos obligatorios vacíos.*	*V*	*V*	*V*	*V*		*I*	*Muestra el mensaje "Existen campo obligatorios vacíos."*	
EC 1.3 La operación es cancelada.	*La operación es cancelada.*	*V*	*V*				*V*	*Cierra la interfaz y no guarda los cambios.*	*1. Selecciona Gestionar evaluación 2. Clic en Cancelar.*

Figura 38. *Caso de prueba: Escenario Realizar Evaluación.*

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Métrica	Campo de selección	No	Contiene letras
2	Proyecto	Campo de selección	No	Contiene letras y números.
3	Módulo	Campo de selección	No	Contiene letras y números.
4	Prototipo	Campo de selección	No	imagen del prototipo
5	Funcionalidad	Campo de selección	No	Contiene letras y números.
6	0	Campo de selección	SI	toma lo valores:-1,0 o 1
7	25	Campo de selección	SI	toma lo valores:-1,0 o 1
8	50	Campo de selección	SI	toma lo valores:-1,0 o 1
9	75	Campo de selección	SI	toma lo valores:-1,0 o 1
10	100	Campo de selección	SI	toma lo valores:-1,0 o 1

Figura 39. *Descripción de las variables.*

Descripción del escenario de prueba y resultados

Para la liberación de la herramienta se realizó con un total de tres revisores pertenecientes al departamento de calidad del Centro de Gobierno Electrónico de la facultad (CEGEL). Se probaron un total de 56 requisitos, siendo crítica la funcionalidad "*Realizar evaluación*" para las diferentes métricas aplicadas.

- En la primera Iteración se detectaron 28 No Conformidades (NC), de las cuales 11 fueron errores de validación, los restantes de redacción en lo mensaje y funcionamiento de la herramienta.
- En la segunda Iteración se detectaron 12 NC, entre ellas de redacción y ortografía, así como algunos problemas de validación que aún quedaban presente.
- En la tercera Iteración se corrigieron las no conformidades detectadas, para un 100% de satisfacción.

Los resultados generales de las pruebas se encuentran en el siguiente gráfico:

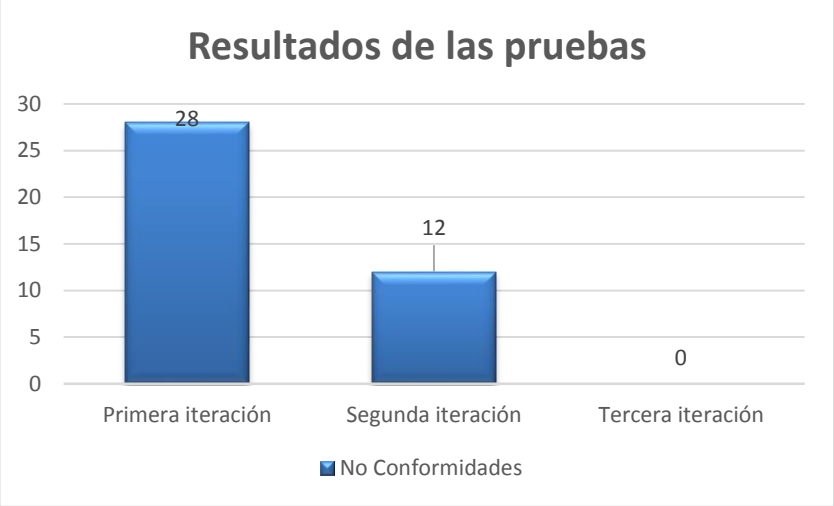

Figura 40. *Resultados generales de las pruebas.*

Luego de haber realizado las pruebas es válido señalar que los resultados obtenidos fueron satisfactorios. Cada una de las no conformidades detectadas fue debidamente atendida logrando un correcto comportamiento de las funcionalidades ante diferentes situaciones (entradas válidas y no válidas), los resultados de las pruebas se muestran en el anexo 8. Las pruebas realizadas permiten afirmar que los requisitos funcionales han sido correctamente implementados y que la herramienta devuelve correctamente el grado de usabilidad para cada prototipo evaluado. El acta liberación de la herramienta UsaProt emitida por el grupo de calidad del centro se encuentra en el anexo 9.

3.4.2 Método de Caja Blanca

Técnica complejidad ciclomática

La técnica “Complejidad Ciclomática” permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución (Pressman, 2010).

Para aplicar dicha técnica se debe introducir la notación para la representación del flujo de control, puede representarse por un grafo de flujo con los siguientes componentes:

- **Nodo**: son los círculos representados en el grafo, cada uno puede comprender una o más secuencias de código.
- **Aristas**: son constituidas por las flechas del grafo y terminan en un nodo; constituyen el flujo de control del procedimiento.

- **Regiones**: son las áreas delimitadas por las aristas y nodos. Las regiones se enumeran, obteniendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

Con el fin de realizar la prueba del camino básico, es necesario calcular la complejidad ciclomática del algoritmo analizado. A continuación se muestra el código del método Insertar Prototipo.

```
public function InsertarPrototipo(Request $request) {
    $foto = $request->files->get('aplicacion_principalbundle_prototipo');    1
    if (!$foto) {  2
        $this->get('session')->getFlashBag()->add('errors', 'El archivo es demasiado grande.'); 3

    }
    $em = $this->getDoctrine()->getManager(); 4
    $entity = new Prototipo(); 4
    $proyecto = $request->get('proyectos'); 4
    $proy = $em->getRepository('PrincipalBundle:Proyecto')->find($proyecto);  4
    $modulo = $request->get('modulos'); 4
    $mod = $em->getRepository('PrincipalBundle:Modulo')->find($modulo); 4
    $funcionalidad = $request->get('funcionalidades'); 4
    $func = $em->getRepository('PrincipalBundle:Funcionalidad')->find($funcionalidad); 4
    $form = $this->createCreateForm($entity);   4
    $form->handleRequest($request);  4
    if ($form->isValid()) {  5
        $entity->setProyecto($proy); 6
        $entity->setModulo($mod); 6
        $entity->setFuncionalidad($func); 6
        $entity->subirFoto($this->container->getParameter('aplicacion.directorio.imagenes')); 6
        $em = $this->getDoctrine()->getManager(); 6
        $em->persist($entity); 6
        $em->flush(); 6

    }
    $proyectos = $em->getRepository('PrincipalBundle:Proyecto')->findAll(); 7
    $modulos = $em->getRepository('PrincipalBundle:Modulo')->findAll(); 7
    $cont = 0;  7
    $contm = 0;  7
    foreach ($modulos as $key => $val) { 8
        $contm = $cont + 1; 9
    }
    $funcionalidades = $em->getRepository('PrincipalBundle:Funcionalidad')->findAll();   10
    return $this->render('PrincipalBundle:Prototipo:new.html.twig', array(
                'entity' => $entity,
                'proyectos' => $proyectos,
                'modulos' => $modulos,
                'contm' => $contm,
                'funcionalidades' => $funcionalidades,
                'form' => $form->createView(),
    ));  10
}
```

Figura 41. *Método insertarPrototipo().*

La siguiente figura muestra el grafo de flujo asociado al código antes presentado, a través de nodos, aristas y regiones, el cual es necesario para el cálculo de la complejidad ciclomática:

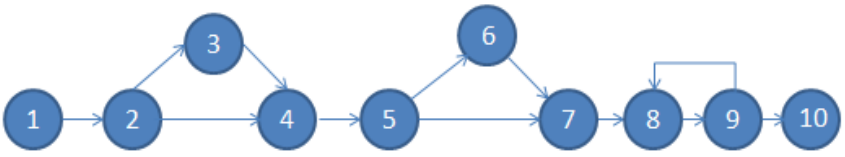

Figura 42.*Grafo del flujo correspondiente al método.*

Una vez construido el grafo de flujo asociado al procedimiento anterior, se calcula la complejidad ciclomática; dicho cálculo es necesario efectuarlo mediante tres fórmulas diferentes.

Fórmula 1: V (G) = (A – N) + 2. Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos. **Resultado:** V (G) = (12-10)+2= 4

Fórmula 2: V (G) = P + 1. Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas). **Resultado:** V (G) = 3+1

Fórmula 3: V (G) = R. Siendo “R” la cantidad total de regiones. **Resultado:** V (G) =4

A continuación se especifican los caminos básicos que puede tomar el algoritmo durante su ejecución. Los elementos de cada camino que los hacen independientes a los demás se encuentran subrayados.

Camino básico#1: 1-2-3-4-5-6-7-8-9-10 **Camino básico#2:** 1-2-3-4-5-7-8-9-10

Camino básico#3: 1-2-4-5-6-7-8-9-10 **Camino básico#4:** 1-2-4-5-7-8-9-10

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para el procedimiento, realizando al menos un caso de prueba por cada camino. Para realizarlos es necesario cumplir con las siguientes exigencias:

- **Descripción:** se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- **Condición de ejecución:** se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- **Entrada:** se muestran los parámetros que entran al procedimiento.
- **Resultados esperados:** se expone el resultado que se espera devuelva el procedimiento.
- **Salida:** se presenta el valor final.

La tabla 7 muestra el diseño de uno de los casos de prueba realizado, para consultar los restantes ver anexo 10.

Tabla 7. *Caso de Prueba para el camino básico #1*.

Diseño de caso de prueba para el camino básico # 1	
Descripción	Los datos de entrada son correctos y cumplen con el formato indicado.
Condición de ejecución	**Campos válidos:** El nombre del proyecto será: Fiscalía 1 El nombre del módulo será: Administración El nombre de la funcionalidad será: Gestionar Usuario El nombre del prototipo será: Insertar Usuario La descripción del prototpo será el prototipo contiene imágenes de la Fiscalia general, además de los datos asociados a esta. La imagen del prototipo debe estar en formato: jpg ,jpeg ó png.

Entrada	**Nombre del proyecto:** Fiscalía 1 **Nombre del Módulo:** Administración **Nombre de la Funcionalidad:** Gestionar Usuario **Nombre del Prototipo:** Insertar Usuario **Descripción:** El prototipo contiene imágenes de la Fiscalia general, además de los datos asociados a esta. **Imagen del prototipo:** debe estar en formato jpg ,jpeg ó png.
Resultados esperados	Se espera que se inserte correctamente el prototipo.

Con la aplicación de los casos de prueba expuestos anteriormente se comprobó que el flujo de trabajo de las funcionalidades es correcto, ya que se probó que cada sentencia es ejecutada al menos una vez, cumpliéndose así las condiciones de la prueba

3.4.3 Evaluación heurística

Técnica evaluación heurística

La evaluación de la usabilidad es la etapa más importante en el proceso de Diseño Centrado en el Usuario (Fadraga Artiles, et al., 2012) .Se puede realizar a través de varios métodos o técnicas y sobre diferentes representaciones del sistema (prototipos en papel, prototipos software, sitio web implementado y otros).

La Evaluación Heurística es una técnica que pertenece al método de inspección, tiene como ventaja la facilidad y rapidez con la que se puede llevar a cabo. Este tipo de evaluación normalmente la desarrolla un grupo reducido de evaluadores que, en base a su propia experiencia, fundamentándose en reconocidos principios de usabilidad (heurísticos), y apoyándose en guías elaboradas para tal fin, evalúan de forma independiente la aplicación, contrastando finalmente los resultados con el resto de evaluadores (Nielsen, 1993).

En la actual investigación se aplicará el modelo de evaluación heurística donde se utilizará la lista de chequeo de usabilidad propuesta en la tesis de Maestría “Procedimiento para el aseguramiento y evaluación de la usabilidad basado en patrones arquitectónicamente sensibles para los sistemas de gestión del Centro de Informatización de la Gestión de Entidades” (Leyva, 2012).

Para la aplicación de la lista de chequeo se realizaron dos iteraciones con el objetivo de corregir elementos erróneos de la aplicación y evidenciar los posibles avances obtenidos a partir de la primera aplicación de la lista de chequeo.

Ejecución de la evaluación de usabilidad

Se estableció un plan de dos iteraciones en las cuales se aplicó la lista de chequeo a la herramienta en dos etapas diferentes de su implementación. Luego de concluida una primera versión completamente funcional, se realizó la primera prueba para la corrección temprana de errores y posibilitar una retroalimentación en aras de obtener una herramienta con mejores características. A continuación se exponen los resultados obtenidos de la primera iteración de la lista de chequeo (Leyva, 2012).

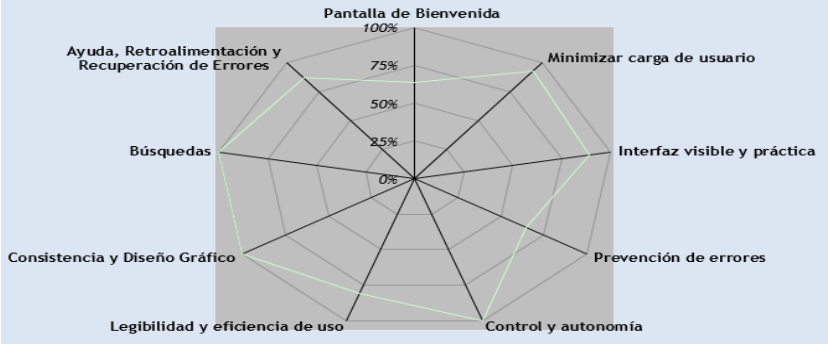

Figura 43. *Gráfico de resultados finales de la primera iteración de la prueba* **(Leyva, 2012).**

La calificación total que obtuvo la herramienta luego de la evaluación fue de 86%. Los aspectos menos favorables fueron la interfaz visible y práctica, pues el sistema no posee las opciones de navegación ordenadas en la manera más lógica u orientada a las tareas, otros de los aspectos que tuvo repercusión negativa fue la prevención de errores pues algunos campos no estaban validados, entre otros elementos desfavorables. Estos aspectos tiene efectos en el cómo los usuarios interactúan con la aplicación por lo que se tomaron en cuenta para su corrección.

Principios de Usabilidad	Calificación Neta	# Preguntas	# Respuesta	Calificación
Pantalla de Bienvenida	3	11	11	64%
Minimizar carga de usuario	23	27	27	93%
Interfaz visible y práctica	26	33	33	89%
Prevención de errores	6	20	20	65%
Control y autonomía	9	9	9	100%
Legibilidad y eficiencia de uso	13	21	21	81%
Consistencia y Diseño Gráfico	20	20	20	100%
Búsquedas	15	15	15	100%
Ayuda, Retroalimentación y Recuperación de Errores	22	30	30	87%
Calificación Final		186	186	86%

Figura 44. *Tabla de resultados finales de la primera iteración de la prueba.*

Luego de corregir los aspectos negativos que se encontraron en la primera iteración de las pruebas se procedió a realizar una segunda iteración .Los resultados obtenidos fueron los siguientes.

Principios de Usabilidad	Calificación Neta	# Preguntas	# Respuesta	Calificación
Pantalla de Bienvenida	9	11	11	91%
Minimizar carga de usuario	27	27	27	100%
Interfaz visible y práctica	27	33	33	91%
Prevención de errores	20	20	20	100%
Control y autonomía	9	9	9	100%
Legibilidad y eficiencia de uso	21	21	21	100%
Consistencia y Diseño Gráfico	20	20	20	100%
Búsquedas	15	15	15	100%
Ayuda, Retroalimentación y Recuperación de Errores	26	30	30	93%
Calificación Final		**186**	**186**	**97%**

Figura 45 .*Tabla de resultados finales de la segunda iteración de la prueba.*

En la segunda iteración de la prueba se obtuvieron resultados positivos, alcanzando un 97% de nivel de usabilidad. Se evidencia un nivel de usabilidad satisfactorio en el sistema.

3.4.4 Evaluación del placer de uso

Estudios demuestran que los seres humanos son más eficientes y creativos en la solución de problemas cuando son felices. Algunas investigaciones también indican que la emoción está estrechamente vinculada al placer de uso (Bevan, 2009).

Las pruebas que se realizarán están encaminadas a determinar la respuesta emocional, el sentimiento que se obtiene de la interacción con la IU para medir la aceptación y el placer que producen en los usuarios de la aplicación.

Existen diversos medios que permiten la medición de la respuesta emocional tanto de forma verbal como no verbal como las técnicas EmoCard, PreEmo y la técnica pensando en voz alta.

Técnica EmoCard

Es una técnica efectiva para la mediación de la respuesta emocional de forma no verbal. Utiliza tarjetas que les permita a los usuarios expresar lo que sienten en el instante de forma sencilla, eliminando las limitaciones en el lenguaje. Para los usuarios supone una dificultad expresar con palabras sus emociones, y es común que los evaluadores no logren encontrar realmente la verdadera sensación que provoca la aplicación. Esta técnica permite al usuario identificar (a través de rostros que emulan estados de ánimos) como se siente al interactuar con el sistema (Valdivia, 2012).

Figura 46. *Conjunto de tarjetas EmoCard* **(Valdivia, 2012) .**

Aplicación de la técnica

Se evaluó una muestra de 10 usuarios pertenecientes al grupo de calidad del CEGEL se dividió en dos grupos de enfoque donde los cinco usuarios del grupo B se enfrentaron por primera vez con la aplicación, mientras que los 5 restantes pertenecientes al grupo A ya poseían una experiencia previa. A los usuarios se les entregó según su género un juego de 8 tarjetas con rostros de hombre o mujer en donde estaban registrados diversos estados de ánimo, desde «excitación» hasta «calma» y con inclinaciones positivas o negativas.

Para el completamiento de esta prueba a los usuarios se les pidió reflejar su estado de ánimo en dos momentos diferentes, al entrar y al salir de la aplicación.

Primera etapa: respuesta emocional del usuario al comenzar a utilizar la aplicación.

Una vez que los usuarios ya se habían registrado en la aplicación se les pidió que reflejaran sus emociones a través de las tarjetas, resultando de la siguiente forma.

Usuario	Elección	Usuario	Elección
a.	4	f.	3
b.	5	g.	3
c.	1	h.	4
d.	4	i.	2
e.	2	j.	3

Figura 47. *Resultados de la aplicación de EmoCard en la primera etapa.*

Los resultados que se muestran a continuación en la gráfica evidencia que los usuarios en su mayoría seleccionaron la tarjeta «calma agradable», otro grupo considerable seleccionó «excitado agradable» y otro número de usuarios seleccionó «promedio agradable». En esta primera etapa ningún usuario seleccionó tarjetas negativas, por lo tanto se puede afirmar que en el primer instante en que los usuarios interactúan con la aplicación existe una reacción agradable.

Figura 48.*Resultados de la aplicación de EmoCard en la primera etapa.*

Segunda etapa: respuesta emocional del usuario una vez que termina de utilizar la aplicación.

Se repitió la prueba pero esta vez luego de que los usuarios habían terminado las tareas que se les había pedido realizar dentro de la aplicación. Se obtuvieron los siguientes resultados.

Usuario	Elección	Usuario	Elección
a.	2	f.	2
b.	4	g.	3
c.	3	h.	4
d.	4	i.	3
e.	2	j.	2

Figura 49. *Resultados de la aplicación de EmoCard en la segunda etapa.*

Figura 50. *Resultados de la aplicación de EmoCard en la segunda etapa.*

Los resultados de la gráfica muestran que las tres tarjetas con mayor índice de selección fueron «excitado agradable», «promedio agradable» y «calma agradable. Por lo que la media de las selecciones fue positiva denotando que luego de concluir el trabajo con la herramienta los usuarios tuvieron una experiencia de uso positiva y se sintieron satisfechos.

3.5 Conclusiones del capítulo

Para lograr una vista estática del sistema se realizaron los diagramas de componentes y despliegue, describiendo cómo los elementos del Modelo de Diseño se implementan en términos de componentes y cómo se organizan de acuerdo a los nodos físicos en el Modelo de Despliegue.

Con el objetivo de evaluar la solución desarrollada se aplicaron los métodos y tipos de pruebas expuestos anteriormente, arrojando resultados satisfactorios que validan que la herramienta devuelve satisfactoriamente el grado de usabilidad y la calidad de la misma. Se evaluó la usabilidad de la herramienta teniendo en cuenta la lista de chequeo heurística, lo cual propició elevar los niveles de confianza en los resultados obtenidos que indicaron un 97% de correlación exitosa. La aplicación de la técnica de «*EmoCard*» para medir el placer de uso del usuario con la herramienta desarrollada indicó como promedio que los usuarios tuvieron experiencias agradables al interactuar con la aplicación. Los resultados anteriores permitieron evaluar la solución dando cumplimiento los objetivos de la investigación.

Conclusiones

Una vez realizadas todas las tareas de la investigación se puede afirmar que se cumplieron de forma satisfactoria arribando a las siguientes conclusiones:

- El estudio de homólogos permitió identificar elementos significativos del estado y las características de las herramientas de evaluación de usabilidad analizadas. Se evidenció la necesidad de la creación de una nueva herramienta para la evaluación de la usabilidad en los prototipos de interfaz de usuarios que pueda ser empleada en la universidad.

- Se desarrolló un sistema que satisface adecuadamente las limitaciones detectadas en el estudio de homólogos, dando cumplimiento a las metas propuestas en la investigación. La implementación de un conjunto de métricas que evalúan la usabilidad en los prototipos y la generación de reportes que ayuden a la toma de decisión en los proyectos del CEIGE.

- Las pruebas de software, la evaluación de la usabilidad y el placer de uso realizadas mostraron la calidad del producto construido, así como el cumplimiento de los objetivos trazados en la investigación.

Recomendaciones

- Se recomienda poner en práctica el uso de la herramienta UsaProt para mejorar la calidad de los productos informáticos del Centro CEIGE.
- Extender el uso de la herramienta para todos los centros de la Universidad de las Ciencias Informáticas.
- Para una nueva versión que la herramienta cumpla con:

Evaluar la usabilidad en sistemas ya concluidos.

Debe contener la incorporación de otras métricas que evalúen la usabilidad.

Ser adaptable para cualquier tipo de sistema.

Referencias bibliográficas

Visual Paradigm. Visual paradigm. [En línea] [Citado el: 7 de diciembre de 2014.] http://www.visualparadigm.com/aboutus/10reasons.jsp.

A.Dix, y otros. 1998. *Human-computer interaction.* 1998.

ALCALA, MARIO LORENZO. 2007. *MEDIDA DE LA USABILIDAD EN APLICACIONES DE ESCRITORIO.* 2007.

Alfredo, Rolando. 2011. *Proceso de investigación científica.* Ciudad de la Habana : Universitaria, 2011.

Barros, Diana Rodriguez. 2008. *DISEÑO DE INTERFACES Y CONDICIONES DE USABILIDAD.Definición de pruebas heurísticas para evaluar la usabilidad .* 2008.

Bevan, Nigel. 2009. [En línea] 2009. [Citado el: 7 de diciembre de 2014.] http://uxpajournal.org/international-standards-for-usability-should-be-more-widely-used/.

Boehm, Barry. 1978. *Características de la calidad del software.* Holanda del Norte : s.n., 1978.

C. Newman, C.Pikenton-Taylor. 2002. *Do Web usability questionnaires measure Web site usability?* Park City : s.n., 2002.

Casanovas, Josep. 2004. Usabilidad y arquitectura del software. 31 de 7 de 2004, pág. 7.

Clements, P. 1996. *"A Survey of Architecture Description Languages" in Proceedings of the International Workshop on Software and Design.* 1996.

Corporation, Netbeans. 2014. Netbeans IDE . [En línea] 7 de diciembre de 2014. [Citado el: 7 de diciembre de 2014.] https://netbeans.org/community/releases/73/.

Css3. 2014. Características del Css3. [En línea] 2014. [Citado el: 7 de Diciembre de 2014.] http://www.css3.com/.

D.Mayhew. 1999. *The Usability Engineering Lifecycle.* San Francisco, California : s.n., 1999.

Denzer, Patricio. 2002. *PostgreSQL.* 2002.

Doctrine . 2013. Doctrine. [En línea] 2013. [Citado el: 7 de diciembre de 2014.] http://www.doctrineproject.org/.

Eguíluz, Javier. 2012. *Desarrollo web ágil con synfony 2.* 2012.

Ferré, X. 2000. *Principios básicos de usabilidad para ingenieros de software.* Valladolid : s.n., 2000.

Folmer, Eelke. 2005. *Software architecture analysis of usability.* 2005.

Glass. 2002. *Facts and Fallacies of Software Engineering.* 2002.

Gómez, Javier M. Martínez. 2011. *Métodos de diseño industrial en el ciclo de vida de software. El papel de la usabilidad en el diseño de interfaces de usuario.* Madrid : Editorial académica española, 2011.

González, Imelda Pena. 2009. *Evaluación del Sistema Automatizado para la evaluación y seguimiento.* Colima : s.n., 2009.

—. 2009. *Evaluacion del Sistema Automatizado para la evaluacion y seguimiento .* Colima : s.n., 2009.

Hakiel. 1997. *Delivering ease of use.* 1997.

Larman, Craig. 1999. *UML y Patrones.* México : PRENTICE HALL, 1999. 970-17-0261-1.

—. 2003. *UML y Patrones. 2da Edición.* 2003.

Leyva, Iliannis Pupo. 2012. *Procedimiento para el aseguramiento y evaluación de la usabilidad.* La Habana : s.n., 2012.

Mascheroni, Maximiliano. 2012. *Herramienta para automatizar la evaluación de la usabilidad en productos software.* Buenos Aires : s.n., 2012.

Mohammed, Kabir. 2003. *La Biblia del Servidor Apache 2.* 2003.

Moreno, Ana Maria. 2001. *Patrones de Usabilidad.* 2001.

Nielsen, Jackob. 1993. *Proceso iterativo para mejorar la usabilidad de un sistema.* 1993.

Nielsen, Jakob. 1993. *Usability engineering .* 1993. (ISBN 0-12-518406-9).

Obeso, María Elena Alva. 2005. *Metodología de Medición y Evaluación de la Usabilidad en sitios web educativos.* UNIVERSIDAD DE OVIEDO : s.n., 2005.

Pacheco, Nacho. 2011. *Doctrine 2 ORM Documentation.* 2011.

—. 2013. *Manual de Twig.* 2013.

Pérez, javier Eguíluz. 2009. *Introducción a javaScript.* 2009.

Pressman, Roger S. 2010. *Software Engineering. A Practitiones's Approach. Seventh Edition.* Nueva York : McGraw-Hill, 2010. 978-0-07-337397-7.

Project. 2010. 2010.

R.Fitzpatrick. 2001. *Strategies for evaluating software usability.* Dublin Institute of Technology Ireland. : s.n., 2001.

Rodríguez, Daniel. 2008. *Las Claves de la Optimización de la Conversión. Trucos para vender mas en Internet.* 2008.

Rubin, J. 2008. *Handbook of usability testing: how to plan, design, and conduct effective tests.* Indianapolis : Wiley Technical Communications, 2008.

Schmuller, Joseph. 2000. *Aprendiendo UML en 24 horas.* s.l. : Pearson Mexico, 2000.

—. 2001. *Aprendiendo UML en 24 horas.* s.l. : Prentice Hall, 2001. ISBN: 968444463X.

Sommerville, Ian. 2007. *Ingeniería de Software. 8va Edición.* Boston : Addison-Wesley, 2007. ISBN 9780321313799.