Facultad 3

Sistema para la Gestión de Evaluaciones Heurísticas

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

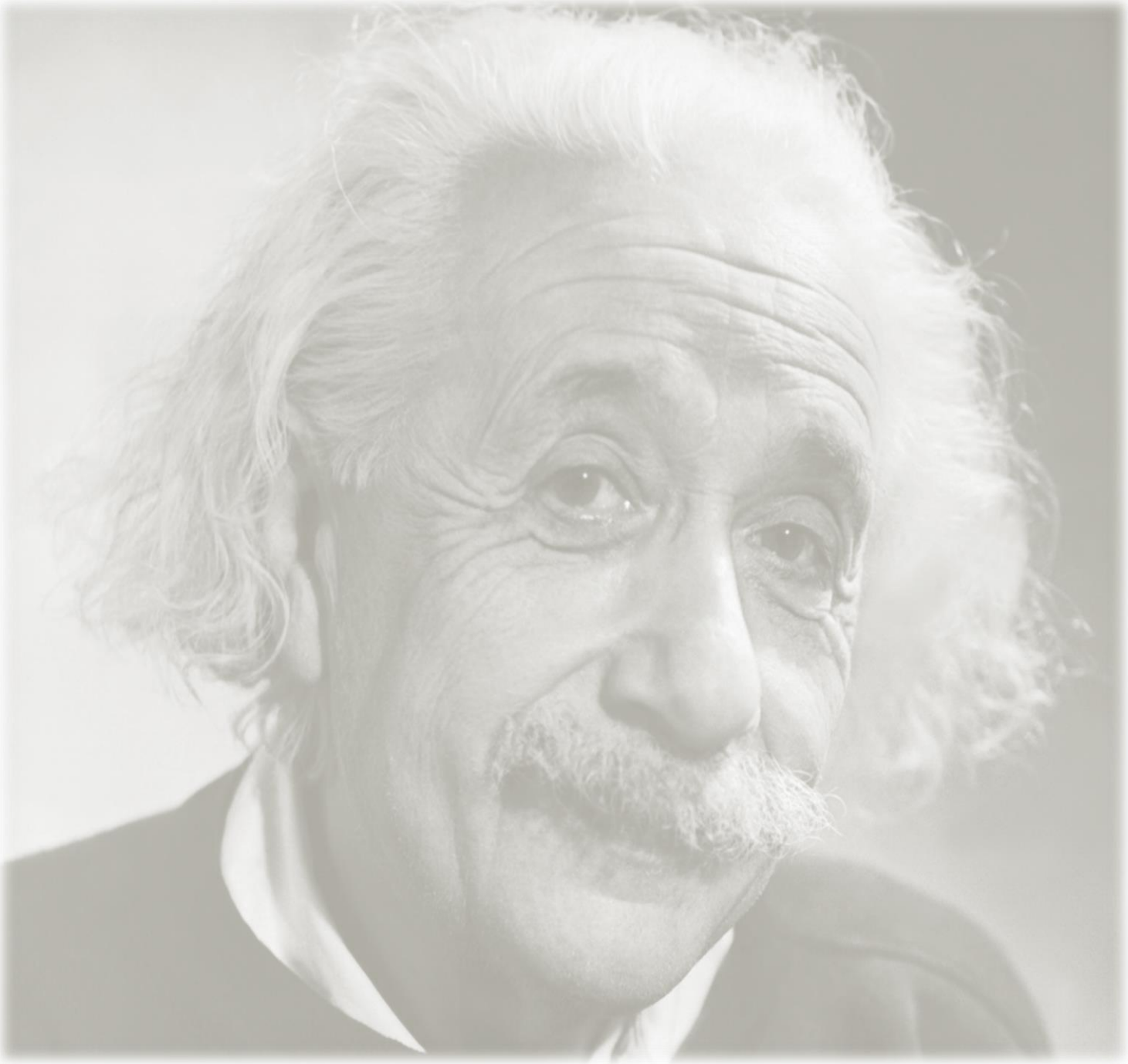
Autores: Joel Rego Pérez

Gendry Eduardo Portela Rodríguez

Tutora: Adisleydis Rodríguez Álvarez

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

La Habana, 25 de junio de 2015. "Año del 57 Aniversario de la Revolución"



"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad".

Albert Einstein

Declaración de Autoría

Declaramos ser autores de la presente tesis y concedemos a la Universidad de las Ciencias Informáticas y al Centro Nacional de Calidad de Software los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los 25 días del mes de Junio del año.

Joel Rego Pérez

Firma del Autor

Gendry E. Portela Rodríguez

Firma del Autor

Ing. Adisleydis Rodríguez Álvarez

Firma de la Tutora

Datos de contacto

Autor:

Nombre y apellidos: Joel Rego Pérez

Correo electrónico: jrego@estudiantes.uci.cu

Institución: Universidad de las Ciencias Informáticas.

Dirección: Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.

Autor:

Nombre y apellidos: Gendry Eduardo Portela Rodríguez

Correo electrónico: geportela@estudiantes.uci.cu

Institución: Universidad de las Ciencias Informáticas.

Dirección: Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.

Tutora:

Nombre y apellidos: Ing. Adisleydis Rodríguez Álvarez

Correo electrónico: aralvarez@uci.cu

Situación Laboral: Especialista B en Ciencias Informáticas.

Institución: Centro Nacional de Calidad de Software.

Años de Experiencia: 4 años.

Dirección: Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.

Dedicatoria y Agradecimientos

A mis padres por siempre haber estado a mi lado, apoyándome y guiándome por el buen camino y que gracias a ellos hoy he podido llegar hasta aquí.

A mi familia en general que siempre estuvo presente y que depositaron toda su confianza en mí, en especial a mi tía Inés y mi tío Castillo que durante esta etapa de mi vida fueron para mí como mis segundos padres.

A mi otra familia, que me ha apoyado mucho durante estos 5 años, a mis compañeros de cuarto Gendry, Ube, Orlando, el Robe, al resto de mis compañeros de aula y a mis antiguos compañeros de grupo.

A mi tutora Ady que durante todo este tiempo fue como una hermana mayor para mí.

A todos los profesores que contribuyeron en mi formación durante estos años.

Joel

Dedicatoria y Agradecimientos

Quiero dedicar este logro personal principalmente a mi familia, de ella a mis padres Belkis y Eduardo, a mi hermana Elaine y a mis dos abuelas Aida y a mis abuelos Elio y Eduardo. Agradecerles su confianza incondicional, el apoyo en esos momentos en que realmente no tenía donde encontrarlo y sobre todo por su sacrificio para que yo llegara hasta aquí.

Al resto de mi familia, no puedo mencionarlos a todos, pero gracias por siempre estar ahí.

Dedicar y recordar a todas esas personas que en algún momento de la vida me vieron comenzar este reto, pero que no les fue posible verme culminarlo: tío Oreste, Lucio, Elio Sanchez, Eriberto, tío Romelio, abuelo Andres y abuela Rosa.

A mis compañeros de trayectoria estudiantil, especialmente a Dargel, Miguel, Jesus, Jeandy, Angel Luis, Joel, Ube, Orlando y Robe. Muchachos sin ustedes muy poco hubiese avanzado, así que esta va para ustedes.

A mi tutora Ady, millones de gracias por ser como eres y por soportarme por estos meses. Sin tu guía no estaríamos acá así que de más está decir cuan agradecidos te estamos.

Gendry

Resumen

La usabilidad en los productos de software se ha convertido paulatinamente en una necesidad y casi una obligación para las empresas dedicadas al diseño y desarrollo de aplicaciones informáticas. No contar con una evaluación o prueba de este tipo afecta la calidad del producto final y conlleva a incrementar los gastos en el proceso de fabricación. El presente trabajo investigativo ha tenido como objetivo desarrollar una herramienta web para el Centro Nacional de Calidad de Software (CALISOFT) que permita gestionar el proceso de evaluación de usabilidad mediante las evaluaciones heurísticas realizadas en el Departamento de Evaluación de Productos de Software (DEPS). Para ello se estudiaron conceptos relacionados con la usabilidad y las evaluaciones heurísticas, también se analizaron las diferentes herramientas existentes que dan soporte a las mismas y se estudió el proceso descrito por el centro para realizar las pruebas. Para el desarrollo del sistema se utilizó XP como metodología de desarrollo, PHP 5.4 como lenguaje de programación, Symfony 2.5 como marco de trabajo, Apache 2.0 como servidor web y PostgreSQL 9.1 como gestor de bases de datos. Asimismo se describieron los artefactos generados en el desarrollo de la solución. Todo esto posibilitó la creación de una herramienta que gestiona el proceso de evaluaciones heurísticas en el DEPS. El correcto funcionamiento de dicha herramienta fue validado mediante el uso de pruebas de aceptación, de caja blanca y de caja negra. Finalmente se comprobó que mediante la utilización de la herramienta implementada se logra una reducción considerable del tiempo de duración de las evaluaciones en el DEPS.

Palabras Claves: Evaluación, usabilidad, heurística.

Índice de contenidos

Índice de figuras

Índice de tablas

Introducción

El auge de las TICS en las distintas organizaciones ha condicionado que las personas sean usuarios de algún sistema informático. Por ello, es fundamental que usuarios inexpertos puedan manejar con mayor facilidad la interfaz gráfica de cualquier sistema informático y así satisfacer sus necesidades y expectativas. La usabilidad es el atributo que debe tener toda aplicación para lograr que los usuarios interactúen con ella con mayor desenvolvimiento.

La usabilidad es considerada como un atributo determinante para el éxito de un proyecto, generando un interés creciente en el mundo del desarrollo de software. Del mismo modo, es el grado en el cual un producto puede ser utilizado por sus usuarios para lograr metas con efectividad, eficiencia y satisfacción en un determinado contexto de uso. Está íntimamente relacionada a la percepción del usuario respecto de la calidad del sistema (1).

Los algoritmos internos o la definición de la arquitectura pueden ser excelentes, pero el usuario no tiene visibilidad de eso, sino de la interfaz con la que interactúa. De ahí que el equipo encargado de desarrollar una aplicación debe tener en cuenta una buena técnica de especificación y validación de los requisitos de usabilidad, así como realizar pruebas para evaluar el resultado final.

Esto constituye un aliciente que ha despertado gran interés en torno a la integración y evaluación de la usabilidad en el proceso de diseño e implementación, así como en la aplicación en uso. Muchos autores como Nielsen (2) o Molich (3) coinciden en definir un conjunto de métodos basados en tener evaluadores que inspeccionen o examinen los principios relacionados con la usabilidad de un software, confiando en la experiencia y conocimiento del evaluador. Estos métodos no requieren extensa preparación o experiencia del evaluador y pueden ser aplicados e integrados en el proceso de desarrollo.

Una de las pruebas que se llevan a cabo para dar tratamiento a la usabilidad es la Evaluación Heurística, que no es más que una variante de la inspección de usabilidad donde los especialistas juzgan si cada elemento de la interfaz de usuario sigue los principios establecidos en los requisitos. Es significativo el empleo de esta técnica para corroborar la capacidad de un producto de software de ser operado y controlado por los usuarios, contribuyendo a la calidad del mismo.

La Evaluación Heurística es un método de evaluación discontinuo, ampliamente aceptado para diagnosticar problemas potenciales de usabilidad en la interfaz de usuario. Define un proceso de inspección de una interfaz particular donde algunos evaluadores la examinan para juzgar el grado de acercamiento con reconocidos principios de usabilidad llamados heurísticas. Puede ser aplicado en las diferentes etapas del ciclo de desarrollo, detectando un buen porcentaje de problemas.

Existen entidades como la española Perception (4) que se dedican a la realización de pruebas de usabilidad. Demostrando cómo un laboratorio de usabilidad ayuda a focalizar al equipo de desarrollo en la experiencia del usuario. Además centran su trabajo en cómo la usabilidad contribuirá a mejorar los resultados esperados de sus clientes.

En nuestro país, existen empresas dedicadas al desarrollo de software que cuentan con departamentos para la realización de pruebas. De ellas, la rectora en este campo es el Centro Nacional de Calidad de Software (CALISOFT). Centro que está adscrito al Ministerio de Comunicaciones (MINCOM), el cual surge con el objetivo de contribuir al desarrollo de aplicaciones informáticas de la industria cubana.

CALISOFT es una unidad presupuestada que entre los servicios que ofrece se encuentran: auditorías, consultorías, diagnóstico, evaluación de productos, pruebas de aceptación con el cliente y en adquisiciones de productos. Este centro cuenta con un Departamento de Evaluación de Productos (DEPS), donde se realiza la evaluación de la calidad de los productos de software.

El trabajo del DEPS está enfocado a la ejecución de pruebas dinámicas y estáticas que evalúan las características planteadas en la Norma NC-ISO 9126 para asegurar que las aplicaciones cuentan con la calidad requerida para ser puestas en explotación. Como parte fundamental de uno de sus procesos, se realizan pruebas de usabilidad donde se destaca el análisis heurístico de las aplicaciones. A pesar de contar con un Sistema de Gestión de la Calidad, la eficiencia del servicio de pruebas de software se ve notablemente afectada por la escasez de herramientas que automaticen parte de los procesos de evaluación que allí tienen lugar.

Actualmente la Evaluación Heurística a productos de software se realiza de forma manual, lo que trae como consecuencia que: no existe un control automático de los recursos humanos que se encuentren disponibles, ni del trabajo realizado. La generación de la lista de chequeo a utilizar por los evaluadores es una tarea compleja debido a que hay que seleccionar de una gran lista de indicadores los que se correspondan con las características del artefacto a evaluar. La asignación Evaluadores–Proyecto no se realiza de forma eficiente. No existen reportes que muestren el seguimiento del trabajo realizado por los evaluadores. No existen mecanismos para transmitir los conocimientos de los expertos en al área a los evaluadores. Igualmente la pérdida de tiempo y derroche de los recursos trae consigo que no se pueda hacer una buena gestión en la conciliación de las No Conformidades y de los reportes de las mismas. Estas deficiencias aumentan los riesgos a fallas del proceso de evaluación de usabilidad y provoca que se extienda significativamente el tiempo de duración de las evaluaciones.

El análisis de la **situación problemática** descrita anteriormente permitió definir como **problema a resolver**: ¿Cómo contribuir a la disminución del tiempo de duración del Proceso de Evaluaciones Heurísticas en el DEPS?

Se plantea como **objeto de estudio**: Gestión del Proceso de Evaluaciones Heurísticas. Asociado a este objeto, se propone como **campo de acción**: los sistemas informáticos para la gestión del Proceso de Evaluaciones Heurísticas.

Para solucionar el problema planteado se define como **objetivo general** de la presente investigación: Desarrollar un sistema informático de gestión del proceso de evaluaciones heurísticas que permita la realización automática de las tareas de evaluación para disminuir el tiempo de duración.

Para cumplir con el objetivo trazado y guiar la presente investigación se proponen los siguientes **objetivos específicos**:

1. Establecer el marco teórico-conceptual para fundamentar la investigación.
2. Generar los artefactos necesarios para el desarrollo del Sistema para la Gestión de las Evaluaciones de Heurísticas en el DEPS.
3. Validar el sistema propuesto para garantizar que cumple con los requisitos definidos.

Se plantea como **idea a defender** de la investigación: El desarrollo de un Sistema para la Gestión del Proceso de Evaluaciones Heurísticas en el Departamento de Evaluación de Productos de CALISOFT, contribuirá a la reducción considerable del tiempo de duración de las evaluaciones.

Con el propósito de dar cumplimiento a los objetivos planteados se hace necesario realizar las siguientes **tareas de investigación**:

OE1
Búsqueda de bibliografía asociada al objeto de estudio.
Selección de bibliografía relevante.
Análisis de la bibliografía seleccionada.
OE2
Definición de las herramientas y tecnologías a utilizar para el desarrollo de la propuesta de solución.

Definición del modelo de desarrollo de software a seguir.
Construcción de los artefactos asociados al modelo de desarrollo de software seleccionado.
Implementación de las funcionalidades definidas.
Validación de las funcionalidades del software implementadas mediante evaluaciones dinámicas.
OE3
Diseño de los Casos de Prueba de Aceptación.
Validación de la propuesta de solución.
Análisis de los resultados obtenidos y conclusiones de la investigación.

Esta investigación está sustentada sobre la base de la utilización de **métodos científicos** para su realización. Los métodos utilizados son:

Métodos teóricos:

Histórico-lógico (5): Para realizar un análisis del devenir histórico del tema en cuestión. Posibilitando contextualizar el problema de investigación, sus antecedentes y desarrollo.

Analítico-sintético (5): Para profundizar en la esencia de las evaluaciones heurísticas en el proceso de evaluación de usabilidad. Se estudia sus particularidades, sus fundamentos y utilización.

Método empírico:

Análisis documental (6)**:** Análisis de documentos relacionados con la temática para constatar información relativa al problema de investigación.

Las **técnicas utilizadas para la recolección de información** son:

La entrevista (7)**:** Esta técnica facilitó recopilar información valiosa para el desarrollo de esta investigación, proporcionada por especialistas que trabajan en el grupo que realiza las pruebas de usabilidad en el DEPS en el centro CALISOFT.

El presente trabajo estará conformado por la siguiente estructura: introducción, tres capítulos, conclusiones, recomendaciones y referencias bibliográficas. Los capítulos abordarán los siguientes temas:

Capítulo I Fundamentación Teórica: Este capítulo hace referencia a los principales conceptos claves para la investigación. En él se realiza un análisis de sistemas similares y un estudio del estado del arte. Asimismo se fundamentan las herramientas, metodología y tecnologías a utilizar en el desarrollo de la solución que se propone.

Capítulo II Propuesta de Solución: En este capítulo se describen los requisitos funcionales y no funcionales identificados, así como los patrones de diseño utilizados. Se muestran los diferentes artefactos generados, como parte de la metodología seleccionada, durante las etapas de Análisis, Diseño e Implementación. Asimismo aborda aspectos relacionados con el desarrollo de la solución explicando los aspectos principales de la implementación.

Capítulo III Pruebas y Validación de la Solución Propuesta: Este capítulo está dedicado a la prueba y validación de la investigación desarrollada. A partir de la ejecución de pruebas de software, se comprobará que las funcionalidades se correspondan con los objetivos del cliente. Además se verificará la validez y correspondencia de la solución propuesta con los objetivos y la idea a defender planteada.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

El campo de desarrollo de software ha reflejado un creciente interés en pruebas de usabilidad que son generalmente ejecutadas en los estados de desarrollo de nuevas aplicaciones. La evaluación de usabilidad es usada para hacer tan útil como sea posible el software y así aumentar la calidad del sistema y la satisfacción del cliente del producto potencial.

En este capítulo se abordan los aspectos fundamentales que sirven de soporte teórico para el desarrollo de toda la investigación. Al mismo tiempo se analizan los diferentes sistemas existentes en el ámbito internacional que gestionan los procesos correspondientes a la realización de evaluaciones heurísticas. Por último, se realiza una breve caracterización de las tecnologías, herramientas y metodología de desarrollo utilizada para la solución del problema planteado.

1.2 Conceptos básicos relacionados con el dominio del problema

Con el fin de facilitar la comprensión de la presente investigación se deben exponer primeramente los términos que se emplean durante su desarrollo. Varios son los conceptos relacionados con el tema que fueron estudiados y se relacionan a continuación:

1.2.1 Usabilidad

En la elaboración de cualquier producto es la calidad uno de los aspectos más importantes para determinar si este es mejor o peor que otro, y el desarrollo de software no puede prescindir de este factor, sino más bien todo lo contrario. Uno de los parámetros de calidad más determinantes para el éxito de una aplicación es la usabilidad, hasta el punto que las más prestigiosas estandarizaciones (ISO 91, IEEE 98) vienen considerando este factor desde hace mucho tiempo.

La definición de usabilidad dada por el estándar ISO 9241-10, establece que los métodos que permiten evaluar si un software es usable, son los métodos que requieren entradas de datos del usuario. Sin embargo, con el fin de incrementar la usabilidad del mismo, estos pueden ser combinados con los métodos de inspección, realizados por expertos y que evalúan la conformidad de la aplicación con un conjunto de principios y aspectos de la interfaz generalizados.

Definición ISO 9241-11 (8)**:** Este estándar internacional define cómo especificar y medir la usabilidad de productos y aquellos factores que tienen un efecto en la usabilidad. La usabilidad es conceptualizada en este documento como: “La extensión para la que un producto puede ser usado por usuarios específicos, para lograr metas específicas con efectividad, eficacia y satisfacción en un contexto de uso específico”.

Para especificar o medir la usabilidad es necesario identificar las metas y descomponer la efectividad, eficiencia y satisfacción, así como los componentes del contexto de uso en subcomponentes con atributos medibles y verificables:

- **Eficacia**: definido en términos de la exactitud y completitud con que usuarios específicos pueden lograr metas específicas en ambientes particulares.
- **Eficiencia**: referido a los recursos gastados en relación con la precisión y completitud de la meta lograda, es decir recursos de tiempo, financieros y humanos.
- **Satisfacción**: que evalúa el confort o comodidad y la aceptabilidad del trabajo del sistema para sus usuarios y otras personas afectadas por su uso.

Definición ISO/IEC 9126 (9)**:** De acuerdo con el estándar (Software Product Evaluation - Quality Characteristics and Guidelines for the User)**,** usabilidad es un atributo de la calidad del software. El término es utilizado para referirse a la capacidad de un producto para ser usado fácilmente. Esto corresponde a la definición de usabilidad como parte de la calidad del software, siendo la calidad del software definida por el estándar como: "Un conjunto de atributos de software que se sostienen en el esfuerzo necesitado para el uso y en la valoración individual de tal uso por un conjunto de usuarios declarados o implicados". Esto está relacionado con la capacidad del producto de software para ser entendido, aprendido, usado y atractivo para el usuario, cuando es utilizado bajo condiciones específicas.

En la parte 9126-1 de este estándar, la usabilidad es analizada en términos de su comprensibilidad, aprendizaje, operabilidad, atractividad y complacencia, tal como se describe a continuación**:**

- **Comprensibilidad**, define la capacidad del producto software para permitir al usuario entender si el software es adecuado, y cómo puede ser usado para tareas y condiciones de uso particulares.
- **Aprendizaje**, referido a la capacidad del producto software para permitir a los usuarios aprender a usar sus aplicaciones.
- **Operabilidad**, es la capacidad del producto software para permitir al usuario operarlo y controlarlo. Aspectos de conformidad, mutabilidad, adaptabilidad e instalación pueden afectar a la operabilidad. Este atributo corresponde a la tolerancia de error, y conformidad con las expectativas del usuario. En un sistema, sobre el que opera un usuario, la combinación de funcionalidad, confiabilidad, usabilidad y eficiencia pueden ser medidas externamente por la calidad de uso.

- **Atractivo**, está referido a los atributos del software pensados para hacer que el mismo sea más llamativo al usuario, tal como el uso de color y la naturaleza del diseño gráfico.
- **Conformidad a estándares y pautas**, referido a la capacidad del producto de software para adherirse a estándares, convenciones, guías de estilo o regulaciones relacionadas con la usabilidad.

Definición de la ISO/IEC 25010 (10)**:** Este estándar adopta como definición de la característica de usabilidad la propuesta por la norma ISO 9241-11. Solo que le incorpora una serie de subcaracterísticas, las cuales se especifican a continuación:

- **Conveniencia reconocible:** es el grado en el que los usuarios pueden reconocer si el producto es apropiado para sus necesidades.
- **Capacidad de aprendizaje:** es el grado en que un producto puede ser utilizado por determinados usuarios para conseguir objetivos de aprendizaje especificados con eficacia, eficiencia, seguridad y satisfacción en un contexto de uso específico.
- **Operabilidad:** es el grado en que el producto tiene atributos que lo hacen fácil de operar y de controlar.
- **Protección contra errores de usuario:** es el grado en que el sistema protege a los usuarios de cometer algún error.
- **Estética de la interfaz de usuario:** es el grado en que la interfaz permite la interacción agradable y satisfactoria para el usuario.

Definición de Nielsen (11)**:** Desde la visión de Nielsen, la usabilidad se define en términos de cinco atributos de usabilidad: aprendizaje, eficiencia, memorización, prevención de error y satisfacción subjetiva.

- **Aprendizaje**, significa que nuevos usuarios deberían aprender fácilmente a usar el sistema.
- **Eficiencia**, el sistema debería ser eficiente para su uso cuando el usuario ha aprendido a usarlo.
- **Memorización**, el sistema deberá ser fácil de recordar incluso después de algún periodo sin uso.
- **Prevención de error**, el sistema deberá tener un bajo porcentaje de error y el usuario deberá fácilmente recuperarse de posibles errores.
- **Satisfacción**, significa que el sistema debe ser agradable de usar.

En el modelo de Nielsen, la usabilidad es una porción de la utilidad del sistema, la cual es parte de la aceptabilidad práctica y, finalmente parte de la aceptabilidad del sistema.

Luego de estudiar los conceptos planteados por diferentes autores sobre la usabilidad y habiendo analizado sus puntos en común y diferencias, se llega a la conclusión de que el más completo y el que se tomará como basamento para el presente trabajo es el definido por el estándar ISO/IEC 25010, dado que se considera que agrupa el mayor número de puntos coincidentes entre todos. Además son las subcaracterísticas de usabilidad que plantea, las que se toman en cuenta en la conceptualización de la evaluación que se realiza en el DEPS.

1.2.2 El Proceso de evaluación

La selección del método y técnica de evaluación dependerá de los requisitos generales relacionados con el propósito de la evaluación. Estos requisitos pueden ser derivados de la demanda de los usuarios, las tareas realizadas con el producto software y el estado de la técnica.

En CALISOFT se establece como proceso, el descrito en el Procedimiento para realizar la Evaluación de Usabilidad de Productos de Software mediante Evaluaciones Heurísticas (12). Este procedimiento, el cual forma parte del Proceso de Evaluación de Productos de CALISOFT, especifica los elementos a tener en cuenta para la ejecución de pruebas a artefactos desarrollados por entidades de la industria cubana de software o desarrolladores independientes del país que así lo soliciten; especificando cómo llevar a cabo las actividades correspondientes y orientando sobre los principios, reglas, normativas y disposiciones generales a tener en cuenta para desarrollar las mismas. Es la investigación realizada por la Ing. Delmys Pozo Zulueta para su maestría en la cual al mismo tiempo se propone el desarrollo de una herramienta de gestión para el procedimiento. A continuación se muestra en la Ilustración 1 la descripción gráfica de cómo se realiza este proceso en el centro (12).

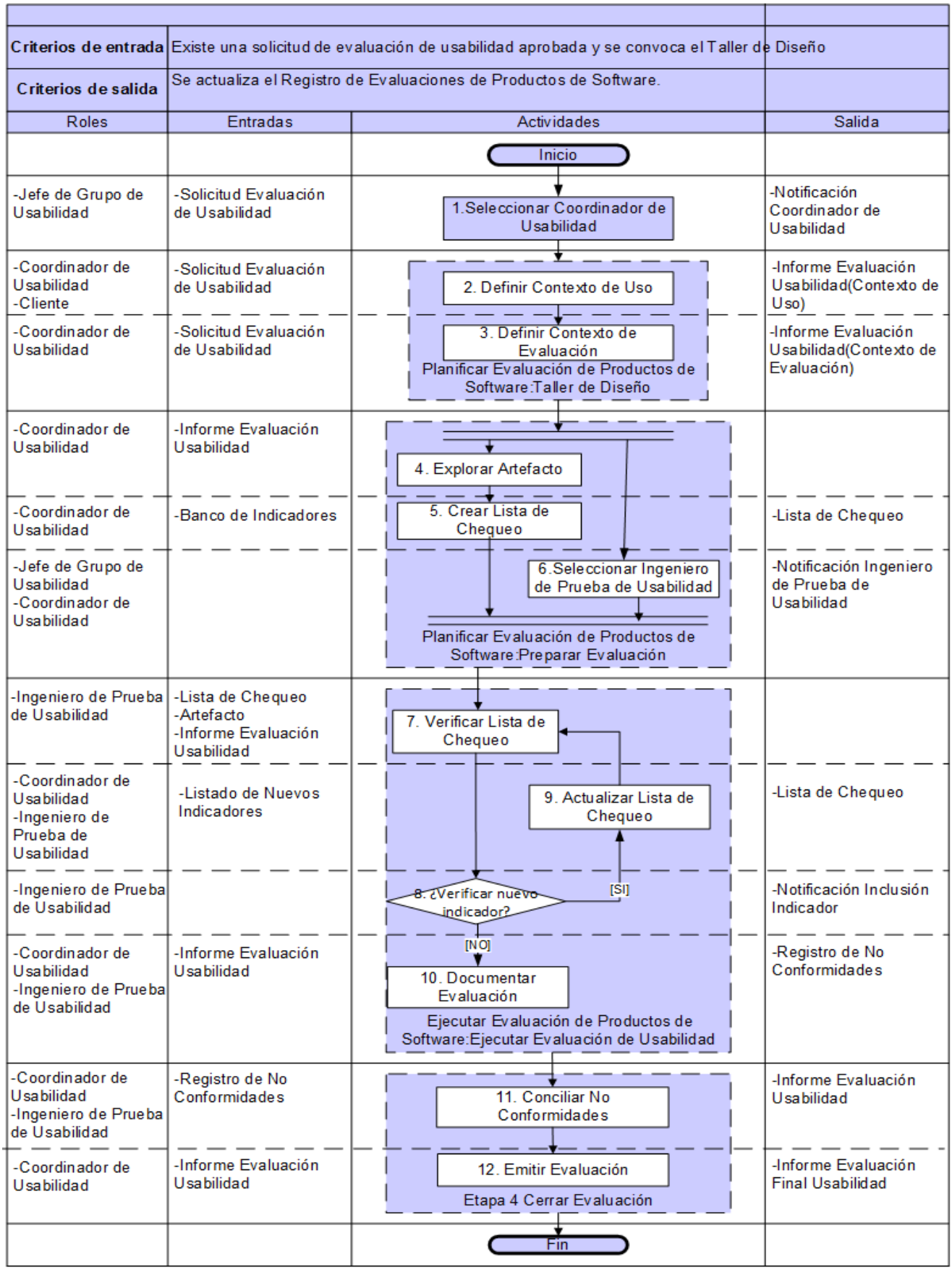


Ilustración 1: Procedimiento para evaluar la Usabilidad de Productos Software mediante Evaluaciones Heurísticas en el Centro Nacional de Calidad de Software.

1.2.3 Evaluación de la usabilidad

La evaluación de la usabilidad es un proceso para proporcionar una medida de la facilidad de uso. En la evaluación hay un objeto que está siendo evaluado y un proceso a través del cual uno o más atributos son juzgados o se les da un valor (13)**.** La evaluación de usabilidad para algunos autores como Mayhew**,** "es un estudio empírico con usuarios reales del sistema propuesto, con el propósito de proporcionar retroalimentación en el desarrollo de software durante el ciclo de vida de desarrollo iterativo" (14).

La realización de este tipo de evaluaciones, es una de las tareas más importantes que debe emprenderse cuando se desarrolla una interfaz de usuario**.** Las interfaces pobres pueden, en el ambiente comercial, ahuyentar a clientes potenciales o en el ambiente educativo llevar al fracaso a un aprendiz. En el mundo competitivo de ingeniería de software, una interfaz pobre puede empujar a los usuarios a las manos de la competencia.

Algunos métodos de evaluación pueden requerir un completo laboratorio de usabilidad y otros pueden lograrse con poco más que una interacción semiformal entre el grupo de desarrollo y los usuarios. Incluso con una inversión relativamente pequeña en métodos de usabilidad puede obtenerse una mejora significativa de la usabilidad de un sistema de software (15)**.**

En los años 80, se había reconocido que las pruebas de usabilidad fallaban para encontrar los prerrequisitos, como sugiere Whiteside (16), debido principalmente a que:

a) Algunos estudios de laboratorios se realizaron en condiciones tan distantes del uso real del sistema que la relación de los datos obtenidos respecto a la vida real obtenidos son irrelevantes y distorsionadas.

b) El uso de usuarios no representativos (vistos como sujetos de experimento), condujo a que en los años noventa ya se hubiera determinado que los dos prerrequisitos principales para tener resultados válidos y útiles en cualquier evaluación de usabilidad eran:

- Los datos, los cuales debían deducirse de circunstancias que tuvieran validez aceptable.
- La aplicación de la técnica de análisis de datos más apropiada para los datos obtenidos (17)**.**

Analizando lo anterior expuesto se puede afirmar que los propósitos fundamentales de la evaluación de usabilidad son:

1. Proporcionar retroalimentación para mejorar el diseño.
2. Valorar qué objetivos de usuarios y organizaciones están siendo logrados.
3. Monitorizar el uso de productos o sistemas a largo plazo.

Por otro lado, para efectos de la evaluación de la usabilidad no solo es de interés el escenario físico y de organización de la evaluación, sino también las características de los usuarios y de las tareas. Ya que una evaluación basada en el usuario puede estudiar sólo un subconjunto de todas las posibles tareas que un sistema puede soportar, la evaluación debe estar basada en el estudio de las tareas más representativas, escogidas por su frecuencia o criticidad. Las características del usuario son importantes en la determinación de la usabilidad, de forma que es fundamental que esta pueda ser evaluada por un grupo representativo de usuarios y no por los propios desarrolladores que poco pueden aportar del uso real.

1.3 Clasificación de los métodos de evaluación de usabilidad

Los métodos de evaluación de usabilidad pueden ser clasificados por numerosos criterios como son: el grado de implicación del usuario, escenarios de tarea, el empleo de reglas o por el objetivo de la evaluación. Diversos han sido los criterios planteados por autores que han abordado el tema y a continuación se describen los métodos definidos por los más destacados en el campo de la usabilidad.

Nielsen y Molich (3)

Nielsen & Molich dividen los métodos de evaluación en cuatro categorías:

- **Evaluación formal**, realiza la evaluación de la interfaz de usuarios mediante algunos análisis técnicos. Los modelos de análisis formal son actualmente objeto de extensa investigación para poder ser aplicados en proyectos de desarrollo de software real.
- **Evaluación automática**, aquella que utiliza procedimientos computarizados para la evaluación de usabilidad.
- **Evaluación empírica**, realizada mediante experimentos con pruebas de usuario, con el objetivo de lograr una completa evaluación de usuario. Actualmente la mayoría de situaciones prácticas no conducen a evaluaciones empíricas por falta el tiempo, especialización, inclinación, o simplemente tradición para hacerlo.
- **Evaluación Heurística**, realizada revisando la interfaz del usuario y generando un informe de acuerdo a la propia opinión.

Wixon y Wilson (18)

Estos investigadores ofrecen una visión amplia y sugieren que la ingeniería de usabilidad coloca al usuario en el centro del proceso. Proponen la siguiente clasificación:

- **Evaluación formativa vs. Sumativa**, los métodos de evaluación formativa son usados para generar nuevas ideas durante el desarrollo, en tanto que los métodos de evaluación sumativos son usados para evaluar sistemas existentes.
- **Método de evaluación de descubrimiento vs. Método de decisión**, los métodos de descubrimiento o métodos cualitativos son usados para descubrir cómo trabajan, se comportan o piensan los usuarios y qué problemas tienen. Por otro lado los métodos de decisión también llamados cuantitativos, son usados en la selección de un diseño determinado entre algunas alternativas o para escoger elementos de diseño de interfaz.
- **Evaluación formalizada vs. Evaluación informal**, los primeros utilizan análisis técnico mientras los segundos son más bien de juicio.
- **Evaluación con usuarios comprometidos vs. Evaluación con usuarios no comprometidos**, estos métodos se diferencian en el grado de compromiso del usuario en la evaluación, análisis y diseño.
- **Evaluación completa vs. Evaluación de componente**, los primeros llamados así porque cubren todos los pasos necesarios para completar los esfuerzos de diseño de usabilidad, mientras que los segundos representan solo una parte de un proceso completo de usabilidad.

Preece (19)

- Por su parte Preece considera cuatro métodos para la evaluación de usabilidad:
- **Evaluación de expertos**, conocido como Evaluación Heurística, normalmente es llevada a cabo por personas experimentadas en diseño de interfaces y/o en la investigación de factores humanos a quienes se solicita describir los problemas potenciales que ellos consideran para usuarios menos experimentados, sugiriendo soluciones a los problemas que ellos identifican. Estos expertos no deberían haber estado involucrados con versiones previas del prototipo bajo evaluación y su rol necesita estar definido previamente para asegurar que ellos adopten la apropiada perspectiva cuando usamos el prototipo.
- **Evaluación observacional**, permite la colección de datos que proporcionan información acerca de qué están haciendo los usuarios cuando interactúan con el software. Pueden ser usadas algunas colecciones de datos técnicos. De acuerdo a Preece pueden obtenerse dos categorías de datos: cómo capturan los usuarios las tareas dadas donde están las mayores dificultades y qué puede hacerse y medidas de desempeño tales como frecuencia de completar la tarea correcta, cronometrado de la tarea, frecuencia de errores de los participantes, etc.

- **Evaluación por investigación**, empleada para conocer las opiniones de los usuarios o para entender sus preferencias sobre un producto potencial o uno existente a través de cuestionarios y entrevistas.
- **Evaluación experimental**, en esta evaluación, un evaluador puede manipular un número de factores asociados con la interfaz de usuario y estudiar sus efectos en el desempeño del usuario. Es necesario planear muy cuidadosamente el nivel de experiencia requerido del usuario, la hipótesis a ser probada, estructura de las tareas y tiempo necesario para completar el experimento, entre otros.

Los estudios realizados permiten afirmar que aún en la actualidad no existe un acuerdo unificado para clasificar los métodos de evaluación de usabilidad. Además los diferentes autores e investigadores del campo, han definido sus propias clasificaciones de métodos para la evaluación, aunque existe coincidencia en algunas categorías y solapamiento entre otras.

1.3.1 Métodos de evaluación de inspección

Autores como Nielsen (2) y Molich (3) coinciden en definir la evaluación de inspección como un conjunto de métodos basados en tener evaluadores que inspeccionen o examinen los principios relacionados con la usabilidad de un software o sitio Web, confiando en la experiencia y conocimiento del evaluador. Diferentes evaluadores encuentran disímiles problemas. Así, aumentando el número de evaluadores aumenta la capacidad para encontrar problemas, aunque la gran mayoría de problemas pueden ser encontrados con los primeros cinco evaluadores (2) **.**

Estos métodos no requieren extensa preparación o experiencia del evaluador y pueden ser aplicados e integrados en el proceso de desarrollo. Los evaluadores pueden ser especialistas en usabilidad, asesores del desarrollo del software con experiencia en determinados estilos de interfaces persona-ordenador, usuarios finales con conocimientos sobre las tareas, etc.

Las inspecciones son propuestas para complementar otros métodos de evaluación proporcionando cambios para encontrar problemas potenciales de usabilidad en el proceso de desarrollo de manera fácil, rápida y muy tempranamente, incluso antes de que sea preparado cualquier prototipo (20)**.** Igualmente permiten encontrar problemas de usabilidad en el diseño o ser dirigidos a la severidad de los problemas y la usabilidad global de un diseño completo. Por lo demás, según señala Nielsen (21), permiten la inspección de especificaciones de la interfaz de usuario (aunque esta no sea implementada inmediatamente).

Dentro de estos métodos se encuentran: Evaluación Heurística, Seguimiento Cognitivo, Seguimiento Pluralista, Inspección Formal, Inspección de Características, Inspecciones de Consistencia, Inspección de Estándares y las Listas de guías de comprobación.

1.4 Evaluación heurística

Este método de inspección fue desarrollado por Nielsen (21), (2) y (22) como una manera para probar interfaces de una manera rápida y económica. Puede definirse como la inspección sistemática de usabilidad de un diseño de la interfaz de usuario. Un especialista en usabilidad juzga si cada elemento de una interfaz de usuario sigue los principios de usabilidad establecidos (23). La meta de la Evaluación Heurística es encontrar los problemas de usabilidad en el diseño de la interfaz de usuario para que estos puedan ser atendidos como parte de un proceso de diseño iterativo.

La Evaluación Heurística es un método de inspección discontinuo, ampliamente aceptado para diagnosticar problemas potenciales de usabilidad en la interfaz de usuario. Define un proceso de inspección de una interfaz particular donde algunos evaluadores la examinan para juzgar el grado de acercamiento con reconocidos principios de usabilidad llamados heurísticas. Puede ser aplicado en las diferentes etapas del ciclo de desarrollo, detectando un buen porcentaje de problemas de usabilidad.

La Evaluación Heurística está basada en un conjunto de reglas que describen propiedades de interfaces usables, llamadas heurísticas, las cuales pueden ser reconocidas en el campo de la investigación o bien consideradas por el grupo de evaluación que puedan ser relevantes para cualquier elemento específico de la interfaz. Cada heurística es presentada de una manera estructurada, con uno o más de los siguientes elementos:

1. **Preguntas de conformidad**, qué debe hacer el sistema/usuario para satisfacer las heurísticas.
2. **Evidencia de conformidad**, qué aspectos de diseño deben considerarse, que indiquen satisfacción o infracción de la heurística.
3. **Motivación**, captura aspectos no conformes a las heurísticas (defectos) en un informe donde los evaluadores describen el problema, su severidad, y sugieren como arreglarlo.

Respecto al número de evaluadores existen diferentes alternativas. Algunos autores como Nielsen señalan que el número de evaluadores debe estar entre tres y cinco (22), fundamentando que una mayor cantidad de evaluadores reduce el beneficio drásticamente y que la proporción de rentabilidad es más alta cuando se emplean tres o cuatro evaluadores. Sin embargo, otros autores como Jacobsen (24) cuestionan la propuesta de Nielsen, señalando que esto funciona cuando se aplica la evaluación en condiciones ideales, pero en casos como evaluaciones a aplicaciones web no responde a las necesidades de evaluación.

1.5 Herramientas para la evaluación heurística

Este tipo de herramientas es el que agrupa aquellas aplicaciones que facilitan o dan soporte durante la realización de la Evaluación Heurística propiamente, permiten elegir o no las heurísticas a evaluar, dan soporte durante la puntuación de cada uno de los criterios heurísticos y en ocasiones, proporcionan algún tipo de resultado tanto cualitativo como cuantitativo. Posteriormente se presentará un breve estudio de cuatro de las herramientas estudiadas.

1.5.1 Herramientas para realizar una evaluación heurística

RIDE: En el año 2006 Kemp y Setungamudalige publicaron la creación de R-IDE: herramienta web que daba soporte en la realización de una evaluación de usabilidad (25). Los orígenes de esta herramienta se remontan en la aparición años antes del framework DECIDE y se inspiran en las últimas tres fases de este framework (IDE) de ahí el nombre de R-IDE.

El proceso de funcionamiento de la herramienta consiste en seleccionar atributos específicos para determinar cuáles son las heurísticas más adecuadas según el sistema que se esté analizando. Estos atributos son: tipo de sistema, categoría general, categoría específica y grupo de usuarios.

La herramienta web permite, dado un conjunto de heurísticas, la elección de las más adecuadas (o elección del conjunto general) para la posterior evaluación individual por parte de cada usuario evaluador del sistema. Es decir, el sistema proporciona, de acuerdo con los atributos seleccionados, un listado de recomendaciones de heurísticas que el evaluador puede modificar y adaptar a su gusto. Otra opción configurable por el evaluador es la escala de valores que se utilizará para puntuar cada una de las heurísticas.

UsabAIPO-GestorHeurística: En el año 2006 el grupo GRIHO diseñó un gestor de heurísticas que permitía gestionar el proceso de realización de una Evaluación Heurística (26). La herramienta desarrollada se denomina UsabAIPO-GestorHeurística y estaba destinada a almacenar la información pertinente para la experimentación, los datos de las webs de las universidades que se evaluaban en el proyecto para el cual se diseñó la herramienta, los criterios heurísticos a contestar por cada evaluador así como las puntuaciones y observaciones que pudieran realizar cada uno de ellos.

ACCUSA (27)**:** El grupo Squac (Software Quality Usability and Certification), que forma parte del Instituto Tecnológico de Informática de la Universidad Politécnica de Valencia, diseñó un primer prototipo de una herramienta denominada Accusa (Squac) que es capaz de gestionar los datos relacionados con todo el proceso de la Evaluación Heurística.

El prototipo de herramienta permite, siempre que se tengan conocimientos mínimos del software de Microsoft Office Access, realizar el proceso de Evaluación Heurística completo.

Desde la definición de las heurísticas, creación de los perfiles de los evaluadores, realización de la evaluación puntuando cada una de las heurísticas y teniendo la posibilidad de añadir observaciones. Finalmente, Accusa es capaz de generar un informe o listado con los problemas de usabilidad que ha detectado cada evaluador, basado en una plantilla de informe por defecto.

SIRIUS: Es un sistema de evaluación basado en una revisión heurística, que integra un conjunto de elementos que lo distinguen de otras propuestas. Es útil tanto para cuantificar el nivel de usabilidad de un sitio web a través de una métrica cuantitativa, como para considerarlo como un conjunto de pautas que sirvan de orientación durante el ciclo de vida de un sitio web. Las principales características del sistema son (28):

- Se aplica a cualquier tipo de sitio web.
- Es aplicable durante todo el ciclo de vida del sitio.
- Da como resultado un valor porcentual del nivel de usabilidad del sitio evaluado, un dato por tanto cuantitativo, lo cual permite:

✓ Cuantificar y comparar la mejora de usabilidad de un sitio en el tiempo.
✓ Comparar la usabilidad de diferentes portales de un mismo sector o que por ejemplo compitan por un premio de usabilidad.
✓ Establecer clasificaciones y rankings en base a la usabilidad.
✓ Comparar el nivel de usabilidad obtenido por un sitio en el tiempo con los resultados de las ventas obtenidas por dicho sitio en ese periodo de tiempo.
✓ Determinar la relación entre usabilidad y accesibilidad (si a mayor nivel de accesibilidad el portal consigue un mejor resultado en su valor de usabilidad).

- Se tiene en cuenta el tipo de sitio evaluado, de manera que la relevancia de los errores está relacionado con el tipo de sitio.
- Permite inferir los elementos a subsanar en el sitio atendiendo a su prioridad, pudiéndose ordenar por este criterio aquellas mejoras que son críticas.
- Permite desarrollar una herramienta de evaluación que dé soporte al sistema de evaluación planteado y facilite la validación empírica de la propuesta de evaluación.

Las herramientas antes descritas dan explícitamente algún tipo de soporte en alguna de las fases de realización de una Evaluación Heurística. En la tabla 1 se muestra una comparación entre dichas herramientas mencionadas teniendo en cuenta algunos indicadores:

Herramienta	Plataforma de desarrollo	Herramientas que usa	Forma en que realiza la evaluación	Hacia dónde va dirigida su utilización	Cantidad de Evaluadores	Reportes
R-IDE	Plataforma Web	Servidor PHP, MYSQL	Dado un grupo de heurísticas se selecciona la más adecuada para realizar la evaluación	Solamente ofrece unas categorías específicas de heurísticas para sitios web en general	No se especifica en la bibliografía consultada.	No se especifica en la bibliografía consultada.
UsabAIPO-GestorHeurística	Microsoft Office Access	Microsoft Office Access	Se presenta una interfaz usable para poder puntuar cada una de las heurísticas sin necesidad de conocer el funcionamiento de Access.	Destinada a almacenar la información pertinente para la experimentación, los datos de las webs de las universidades que se evaluaban en el proyecto para el cual se diseñó.	No se especifica en la bibliografía consultada.	No se especifica en la bibliografía consultada.
ACCUSA	Microsoft Office Access	Microsoft Office Access	Realiza la evaluación puntuando cada una de las heurísticas y añadiendo observaciones.	Destinada para realizar evaluaciones a cualquier sistema	No se especifica en la bibliografía consultada.	Genera un informe con los problemas de usabilidad encontrados
SIRIUS	No se especifica en la bibliografía consultada.	No se especifica en la bibliografía consultada.	Útil tanto para cuantificar el nivel de usabilidad de un sitio web a través de una métrica cuantitativa, como para considerarlo como un conjunto de pautas que sirvan de orientación durante el ciclo de vida de un sitio web	Destinada para realizar evaluaciones a cualquier sistema web.	Tres	No se especifica en la bibliografía consultada.

Tabla 1: Comparación por indicadores entre las herramientas que se utilizan para realizar una Evaluación Heurística.

Luego de realizar una investigación y un posterior estudio de las herramientas creadas para realizar evaluaciones heurísticas, concluimos que a pesar de que estas tengan como funcionalidad: crear y mantener directrices de usabilidad, chequear si los sitios web son usables y accesibles de acuerdo con unos criterios preestablecidos, gestionar los datos relacionados con todo el proceso de la Evaluación Heurística o dar soporte en la evaluación de usabilidad de sistemas interactivos, no existe ninguna herramienta cubana o extranjera que combine más de una de las funcionalidades mencionadas. Además de que la mayoría de ellas son creadas con software que no son multiplataforma, afectando su uso en sistemas operativos de software libre. Tampoco se encontró en la bibliografía datos sobre los reportes que cada una de ellas pueda generar. Por lo tanto se decide implementar una herramienta que se adapte al proceso descrito por el Departamento de Evaluaciones de Productos de Software del centro CALISOFT, que integre todas las funcionalidades requeridas en dicho proceso y que al mismo tiempo ofrezca resultados cualitativos de usabilidad así como proporcione reportes que contribuyan a la obtención de estadísticas del proceso así como a la evaluación de los involucrados durante la ejecución.

1.6 Ambiente de desarrollo

Se describen a continuación la metodología y las herramientas utilizadas para el desarrollo del sistema propuesto. Fueron seleccionadas las más adecuadas de cada una de ellas, de acuerdo a las características del proyecto y el equipo de desarrollo, en aras de implementar un producto con calidad.

1.6.1 Metodología de desarrollo

La metodología hace referencia al conjunto de procedimientos racionales utilizados para alcanzar una gama de objetivos que rigen en una investigación científica, una exposición doctrinal o tareas que requieran habilidades, conocimientos o cuidados específicos. Alternativamente puede definirse la metodología como el estudio o elección de un método pertinente para un determinado objetivo (29).

Extreme Programming, más conocida por XP es una metodología de desarrollo ágil creada por Kent Beck. Está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, así como en una comunicación fluida entre todos los participantes (30)**.**

XP consta de 4 fases:

- ✓ **Planificación**: En esta primera fase se debe hacer primero una recopilación de todos los requisitos del proyecto, también debe haber una interacción con el usuario, y se debe planificar bien entre los desarrolladores del proyecto qué es lo que se quiere para el proyecto

para así lograr los objetivos finales. En esta fase se generan artefactos donde se especifican los usuarios relacionados con el sistema, las historias de usuario, así como su estimación de esfuerzo, se realiza del mismo modo el plan de duración de las iteraciones y se elabora un plan de entrega.

- ✓ **Diseño:** Como artefactos a desarrollar en esta fase se encuentran las tarjetas CRC y el diagrama Entidad-Relación, el cual no está comprendido entre los artefactos a generar por la metodología, pero dada su marcada importancia dentro del desarrollo de aplicaciones de gestión web, se ha decidido incluir su diseño.
- ✓ **Implementación:** En esta fase de codificación los clientes y los desarrolladores del proyecto deben estar en comunicación para que los programadores puedan codificar todo lo necesario para el proyecto que se requiere. En esta fase está incluido todo lo referente a la codificación o programación del proyecto. Es importante en esta fase identificar y especificar las tareas de ingeniería a desarrollar por iteraciones.
- ✓ **Pruebas:** Las pruebas son una de las prácticas fundamentales en las cuales se basa XP. Esta actividad se realiza en forma continua a lo largo del proyecto. Existen dos tipos de pruebas, las unitarias y las de aceptación. Siendo los Casos de Prueba de Aceptación los artefactos principales a desarrollar.

Una de las principales ventajas de esta metodología es el aprovechamiento del tiempo, ya que permite agilizar todo el proceso. Entre sus principios básicos se encuentran trabajo de alta calidad, asumir la simplicidad, realimentación rápida y cambio incremental. Esta metodología está dirigida para equipos de trabajo, pequeños o medianos permitiendo que todos trabajen juntos, en la misma dirección y con un objetivo claro, aprobando seguir el avance de las tareas a realizar. Los requisitos y el diseño se realizan de forma simple, a través de las Historias de Usuario, Tareas de Ingeniería y las Tarjetas Clases-Responsabilidad-Colaborador (30)**.**

Se determina utilizar como metodología de desarrollo de software XP, debido a que es una metodología ágil diseñada para equipos de trabajo pequeños. Está centrada en vincular al cliente al ciclo de desarrollo, incrementando la posibilidad del éxito y disminuyendo las no conformidades, con el objetivo de cumplir los requisitos y especificaciones trazadas. Realiza las pruebas al final de cada iteración comprobando si la herramienta cumple con los requisitos propuestos.

1.6.2 Herramienta CASE para el modelado.

Las herramientas de Ingeniería de Software Asistida por Computadoras (CASE por sus siglas en inglés), son aquellas que permiten la automatización de metodologías paso a paso para el desarrollo de software y sistemas. Entre sus objetivos están la mejora de la planificación de un proyecto, y la

gestión global de todas las fases de desarrollo de software con una misma herramienta, de igual forma facilitan la creación de documentación estructurada y la coordinación de los esfuerzos de desarrollo del equipo de trabajo.

Una de las herramientas CASE más utilizadas es el Visual Paradigm, la cual permite el modelado del sistema ayudando a una rápida construcción del mismo con calidad. Posee un soporte multiplataforma y proporciona excelente facilidades de interoperabilidad con otras aplicaciones. Ha sido diseñada para automatizar y acelerar el ciclo de desarrollo de software, permitiendo la captura de requisitos, análisis, diseño e implementación (31).

Como herramienta de modelado se elige Visual Paradigm, pues permite hacer el trabajo más organizado y ágil. Propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Aporta gran facilidad y ayuda en la generación de bases de datos, posibilitando la transformación de diagramas de Entidad-Relación en tablas de base de datos.

1.6.3 Lenguaje de Modelado Unificado (UML 5.0)

Un lenguaje de modelado es una manera artificial diseñada para expresar modelos. Como los modelos habitualmente se muestran en forma de diagramas por comodidad, UML es un lenguaje para especificar, construir, visualizar y documentar los artefactos (información que es utilizada o producida mediante un proceso de desarrollo de software) de un sistema de software orientado a objetos, que por su potencialidad se ha convertido en un estándar. Proporciona un vocabulario y una regla para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema (32).

1.6.4 Marco de trabajo

Symfony2: Se escoge como marco de trabajo para el desarrollo de la herramienta el Symfony2. El cual ha sido ideado para aprovechar al límite todas las nuevas características de PHP 5.3 y por eso es uno de los frameworks PHP con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en el proyecto. Symfony2 también es el framework que más ideas incorpora del resto de los frameworks, incluso de aquellos que no están programados con PHP (33).

Este marco de trabajo basa su funcionamiento interno en el estilo arquitectónico Modelo – Vista – Controlador (34) y optimiza el desarrollo de las aplicaciones web. Ha sido probado en numerosos proyectos reales y utiliza conceptos exitosos de terceros, entre los cuales se encuentra plenamente un framework ORM de los más importantes que existen en PHP llamado Doctrine; que se encarga de la comunicación con la base datos, independientemente del Gestor de Base Datos utilizado. Incluye

otro framework bastante conocido llamado Twig que constituye un motor de plantillas que permite separar el código PHP del código HTML.

Doctrine 2: Se utiliza Doctrine por ser un completo sistema ORM para PHP 5.2+ que incorpora una capa de abstracción a base de datos DBL. Una de sus características es el bajo nivel de configuración que necesita para empezar un proyecto. Doctrine puede generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas.

Funcionalidades que facilitan la implementación:

- Exporta una base de datos existente a sus clases correspondientes.
- Convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de dato (35).

Twitter Bootstrap 3.0: Se decide utilizar Bootstrap dado que es un framework de aplicaciones para usuario elegante e intuitivo, de gran alcance para el desarrollo web más rápido y sencillo. Asimismo Bootstrap hace uso de determinados elementos HTML y propiedades CSS 3 que requieren el uso del HTML5 (36).

HTML5 es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 es considerado el producto de la combinación de HTML, CSS y JavaScript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML5 (37).

1.6.5 Lenguaje de programación a utilizar.

PHP es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Es una tecnología de código abierto que resulta muy útil para diseñar de forma rápida. PHP funciona sobre prácticamente todas las plataformas y garantiza una excelente estabilidad (38). Tiene la capacidad de conexión con la mayoría de los Sistemas de gestión de bases de datos de bases de datos, MySQL, PostgreSQL, Oracle entre otras.

Para la implementación del sistema, teniendo en cuenta los requisitos no funcionales de la misma, y habiendo seleccionado la arquitectura a utilizar se elige como lenguaje de programación PHP en su versión 5.4, dado que es un lenguaje gratuito e independiente de plataforma, rápido, con una amplia gama de funciones y documentación asociada a él.

1.7 Herramientas de desarrollo

1.7.1 Sistema gestor de bases de datos

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacional (ORDBMS, por sus siglas en inglés), basado en el proyecto POSTGRES, de la universidad de Berkeley. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos (39).

Principales características de este gestor de bases de datos (40)**:**

1. Implementación del estándar SQL92/SQL99.
2. Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
3. Permite la declaración de funciones propias, así como la definición de disparadores.
4. Soporta el uso de índices, reglas y vistas.
5. Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
6. Permite la gestión de diferentes usuarios, y los permisos asignados a cada uno de ellos.

Se decidió utilizar como gestor de bases de datos el PostgreSQL 9.1 dado que es ideal para la tecnología Web. Administrarlo es relativamente fácil, su sintaxis SQL es estándar y fácil de aprender. Conjuntamente es un sistema multiplataforma con una gran capacidad de replicación de datos.

1.7.2 Entorno integrado de desarrollo

Netbeans es un Entorno de Desarrollo Integrado (IDE), está disponible libre de costo y de código abierto, escrito en lenguaje Java lo cual lo convierte en multiplataforma. Tiene todas las herramientas necesarias para crear aplicaciones de escritorio y web profesionales, mediante el uso de diferentes lenguajes de programación como Java, C/C++, PHP, Ruby, JavaScript, HTML y Groovy. Entre sus características fundamentales se encuentra un editor de código muy potente, permite depurar y ejecutar programas, tiene integración con marcos de trabajo de desarrollo de PHP, sus funcionalidades se pueden extender con el uso de complementos o extensiones que se le incorporan según las necesidades del programador, permite integración con sistemas de control de versiones (41).

Para la implementación de la aplicación se optó por utilizar el IDE Netbeans en su versión 8.0 ya que permite el uso de un amplio rango de tecnologías de desarrollo de aplicaciones web. Tiene gran interacción con Symfony2, que es el marco de trabajo seleccionado. La plataforma NetBeans permite

que las aplicaciones sean desarrolladas a partir de un conjunto de módulos. Igualmente permite el acceso al gestor de base de datos utilizado y desde el propio IDE realizar consultas, modificaciones y la visualización de las tablas. Permite la integración con diferentes servidores web como el Apache.

1.7.3 Servidor de aplicaciones web

Es propicio utilizar como servidor de aplicaciones web el Apache 2.0. Este servidor no representa gastos por concepto de licencia y es de código abierto, puede ser usado en varios sistemas operativos. Es un servidor fácilmente configurable, es decir, se pueden elegir qué características van a ser incluidas en el servidor seleccionando qué módulos se van a cargar, ya sea al compilar o al ejecutar el servidor (42).

1.8 Conclusiones del capítulo

El estudio de los elementos conceptuales relacionados a la investigación permitió afianzar el conocimiento teórico necesario para el desarrollo de la solución propuesta.

El estudio de los sistemas informáticos presentados que gestionan y realizan las evaluaciones heurísticas demostró la necesidad de implementar un sistema que gestione este proceso en CALISOFT ya que no existe ninguna que se adecue a las características del proceso de evaluación que tienen descrito los clientes para la evaluación de la usabilidad.

La necesidad de aprovechar el tiempo mediante la agilización del proceso de desarrollo y de utilizar un equipo de trabajo pequeño, conllevó a adoptar como metodología de desarrollo la XP.

Para el desarrollo de la propuesta se utilizará como marco de trabajo Symfony2.5, para el diseño de las interfaces será utilizando Twitter Bootstrap 3.0, el cual incluye HTML5 y CSS3. Como lenguaje de programación se utilizará PHP 5.4, el gestor de bases de datos será PostgreSQL 9.1 y como servidor web se utilizará el Apache 2.0.

Capítulo 2: Propuesta de Solución

2.1 Introducción

En este capítulo se describirán los artefactos de las tres primeras fases de la metodología propuesta: la planificación, el diseño y la implementación. Para ello se desarrollarán los artefactos importantes de las mismas como la Definición de la audiencia, Historia de Usuarios, Plan de Iteraciones, Plan de Duración de Iteraciones, Plan de Entregas, Tarjetas CRC y Tareas de Ingeniería.

2.2 Requisitos de software

Según Sommerville (43), “un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema…” Es importante destacar que la calidad con que se realiza la captura de los requisitos afecta todo el proceso de desarrollo del software repercutiendo en el resto de las fases del mismo.

2.2.1 Requisitos funcionales y no funcionales

Sommerville plantea que los requisitos funcionales “son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares” (43).

Los requisitos no funcionales son definidos por Sommerville como “restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares” (43).

Seguidamente son especificados los requisitos del sistema en la lista de reservas del producto.

Número	Descripción	Estimación	Estimado por
Requisitos funcionales			
Prioridad: Alta			
1	Registrar usuario	2	Equipo de Desarrollo
2	Cambiar contraseña	1	Equipo de Desarrollo
3	Editar perfil	1	Equipo de Desarrollo
4	Gestionar fase de desarrollo	2	Equipo de Desarrollo
5	Gestionar subcaracterística de usabilidad	2	Equipo de Desarrollo
6	Gestionar tipo de no conformidad	2	Equipo de Desarrollo
7	Gestionar tipo de artefacto	2	Equipo de Desarrollo

8	Gestionar indicador	2	Equipo de Desarrollo
9	Gestionar proyecto	2	Equipo de Desarrollo
10	Gestionar Plantilla de Solicitud	2	Equipo de Desarrollo
11	Gestionar Plantilla de Evaluación	2	Equipo de Desarrollo
12	Gestionar Informe de Evaluación	2	Equipo de Desarrollo
13	Asignar ingenieros	2	Equipo de Desarrollo
14	Gestionar lista de chequeo	2	Equipo de Desarrollo
15	Gestionar artefacto	2	Equipo de Desarrollo
16	Gestionar no conformidad	2	Equipo de Desarrollo
17	Gestionar tema de debate	2	Equipo de Desarrollo
18	Gestionar comentario	2	Equipo de Desarrollo
Prioridad: Media			
19	Mostrar reporte de evaluadores asignados a un proyecto.	2	Equipo de Desarrollo
20	Generar reporte de proyectos por evaluador.	2	Equipo de Desarrollo
21	Mostrar reporte de NC por Evaluador en una evaluación.	2	Equipo de Desarrollo
22	Mostrar reporte de indicadores evaluados por subcaracterística.	2	Equipo de Desarrollo
23	Mostrar reporte de cantidad de NC por tipo en un rango de fechas.	2	Equipo de Desarrollo
24	Generar reporte de cantidad de NC por evaluación en un rango de fechas.	2	Equipo de Desarrollo
25	Mostrar reporte de NC por tipo por evaluación en un rango de fechas.	2	Equipo de Desarrollo
26	Mostrar el comportamiento de la evaluación de subcaracterística por evaluador.	2	Equipo de Desarrollo
27	Proporcionar porciento de subcaracterística por evaluación.	2	Equipo de Desarrollo
28	Generar reporte de NC significativas y No significativas por evaluación en rango de fechas.	2	Equipo de Desarrollo

Prioridad: Baja			
29	Notificar indicador	1	Equipo de Desarrollo
Requisitos no funcionales			
Eficiencia de desempeño			
30	Proporcionar un sistema que sea capaz de ejecutar todas las peticiones y operaciones en un máximo de 5 segundos.		
Compatibilidad			
31	Asegurar la compatibilidad con diferentes navegadores (Mozilla, Chrome, Opera), diferentes versiones de los mismos y su correcto funcionamiento en sistemas operativos propietarios y de software libre.		
Usabilidad			
32	Cumplir con las pautas de diseño de las interfaces descritas en el documento: Metodología para el diseño y desarrollo de interfaces de usuario web (44).		
33	Agrupar vínculos y botones por grupos funcionales.		
34	Utilizar diálogos o formularios similares para tareas equivalentes.		
35	Disponer de mensajes precisos, claros e insistentes durante una operación que oriente al usuario en cada procedimiento.		
36	Permitir al usuario la autenticación de manera sencilla.		
37	Mantener una tipografía y estética coherente en toda la aplicación.		
Seguridad			
38	Proteger la aplicación mediante una correcta utilización de los roles, de tal manera que no permita el acceso no autorizado a la misma, para contribuir a la integridad y confiabilidad de los datos procesados.		
Mantenibilidad			
39	Garantizar las modificaciones de forma efectiva y eficiente en el sistema, dado las necesidades correctivas o perfectivas del cliente.		
Software			
40	Usar como servidor el Web Apache Server 2.0, compatible con GPL.		
41	Utilizar para el almacenamiento de la información el servidor de Base de Datos PostgreSQL 9.1.		
Hardware			
42	Utilizar una estación de trabajo con al menos 1GB de memoria RAM, al menos 40 GB de disco duro.		
43	Usar un servidor que tenga como mínimo 1GB de memoria RAM y 500 MB de espacio libre.		

Tabla 2: Lista de reserva del producto.

2.3 Fase de planificación

2.3.1 Definición de la audiencia

La audiencia constituye uno de los elementos más importantes de esta fase de desarrollo del sistema, ya que consiste en definir al público a quien va dirigida la solución propuesta, en este caso la solución va dirigida a los especialistas de usabilidad del DEPS en el centro CALISOFT, a los colaboradores que puedan sumarse a la evaluación y de igual forma puede ser utilizada por expertos en usabilidad para planificar y ejecutar evaluaciones heurísticas.

2.3.2 Usuarios relacionados con el sistema

La aplicación web que se desarrolla muestra un grupo de funcionalidades y servicios para cumplir con los objetivos trazados. En la implementación se establecen roles para asignar los diferentes permisos para su acceso. Los roles y permisos son utilizados para restringir el nivel de acceso a las funciones del sistema de cada usuario que interactúe con el sistema, es decir que en dependencia de los roles de cada usuario y los permisos que tengan cada uno de los roles será su acceso a las funciones del sistema. Se denomina usuario a cualquier persona relacionada con el sistema, ya sea vinculada al desarrollo del mismo o que de una forma u otra interactúa con la aplicación, incluyendo a los que mantendrán el sistema funcionando y actualizado (45).

Usuario	Descripción
Jefe de Grupo de Usabilidad	Distribuye el trabajo de forma equitativa y tiene en cuenta las competencias de los Ingenieros de Pruebas de Usabilidad y Coordinadores de Usabilidad al asignarlos a una evaluación. Es también el responsable de revisar y actualizar el procedimiento.
Coordinador de Usabilidad	Es quien dirige el proceso de evaluación durante su aplicación. Realiza la conciliación de las no conformidades.
Ingeniero de Pruebas de Usabilidad	Su trabajo consiste en ejecutar la evaluación mediante la verificación del cumplimiento de los indicadores. Participa en la conciliación de las no conformidades.
Colaborador	Su trabajo consiste en ejecutar la evaluación mediante la verificación del cumplimiento de los indicadores.

Tabla 3: Usuarios relacionados con el sistema.

2.3.3 Historias de usuarios

La metodología XP utiliza la técnica de las Historias de Usuario (HU) para sustituir a los documentos de especificación funcional y a los casos de uso. Estas HU son escritas por el cliente en su propio lenguaje como descripciones cortas de lo que el sistema debe realizar. El tratamiento de las HU es muy dinámico y flexible, permite que en cualquier momento se puedan romper, reemplazar por otras

más específicas o generales, añadirse nuevas o ser modificadas. Para ser implementadas las HU, el cliente y los desarrolladores se reúnen para detallar las funcionalidades de cada una. El tiempo de desarrollo ideal para una HU varía entre 1 y 3 semanas (46).

A continuación se muestran las HU de una de las funcionalidades del sistema.

Historia de Usuario	
Código: HU# 1	**Nombre historias de usuario:** Registrar Usuario
Modificación de historia de usuario (número): Ninguno	
Referencia: RF#1	
Programador: Joel Rego Pérez	**Iteración asignada:** Primera
Prioridad: Alta	**Puntos estimados:** 2 días
Riesgo en desarrollo: Medio	**Puntos reales:** 2 días
Descripción: La funcionalidad registrar usuario, debe permitir adicionar, los datos correspondientes a un usuario del sistema. Inicialmente se debe mostrar una interfaz donde aparezca un formulario donde se introducirán los datos del usuario.	
Observaciones: Las acciones sobre los usuarios del sistema solo pueden ser realizadas por quien posea privilegios de Jefe de grupo en el sistema.	

Tabla 4: HU Registrar Usuario.

2.3.4 Estimación de esfuerzos por HU

Las HU deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogan directamente con el cliente para obtener todos los detalles necesarios. Inicialmente, el equipo de desarrolladores estima el esfuerzo necesario para implementar las HU y los clientes aprueban los objetivos y tiempos de entrega, la estimación temporal se basa en un cálculo estimado por parte de los desarrolladores de cada una de las HU.

Historias de Usuario	Puntos de Estimación (por días)
Registrar usuario	2
Editar perfil	1
Gestionar fase desarrollo	2

Gestionar subcaracterística de usabilidad	2
Gestionar tipo de no conformidad	2
Gestionar tipo de artefacto	2
Gestionar indicador	2
Gestionar proyecto	2
Gestionar plantilla de solicitud	2
Gestionar plantilla de evaluación	2
Gestionar informe de evaluación	2
Asignar ingenieros	1
Gestionar lista de chequeo	2
Gestionar artefacto	2
Notificar indicador	1
Gestionar no conformidad	2
Reporte de evaluadores asignados a un proyecto.	1
Reporte de proyectos por evaluador.	1
Reporte de NC por Evaluador en una evaluación.	1
Reporte de indicadores evaluados por subcaracterística.	1
Reporte de cantidad de NC por tipo en un rango de fechas.	1
Reporte de cantidad de NC por evaluación en un rango de fechas.	1
Reporte de NC por tipo por evaluación en un rango de fechas.	1
Comportamiento de la evaluación de subcaracterísca por evaluador.	1
Porciento de subcaracterísca por evaluación.	1
Reporte de NC significativas y No significativas por evaluación en rango de fechas.	1
Gestionar tema de debate	2
Gestionar comentario	2

Tabla 5: Estimación de esfuerzo por HU.

2.3.5 Plan de iteraciones

Luego de identificar y definir las historias de usuario y estimar el esfuerzo necesario para la realización de cada una de las HU, se pasa a la planificación de la etapa de implementación del sistema. Se seleccionan las historias de usuario que se implementan en cada iteración, de acuerdo al nivel de prioridad de las mismas, así como las posibles fechas de sus entregas.

2.3.5.1 Plan de duración de las iteraciones

Este plan tiene como objetivo mostrar la duración de cada iteración, así como el orden en que serán implementadas las historias de usuario en cada una de las mismas.

Iteraciones	Orden de las historias usuario a implementar	Duración de las iteraciones
Iteración 1	Registrar usuario	2
	Editar perfil	
	Gestionar fase desarrollo	
	Gestionar subcaracterística de usabilidad	
Iteración 2	Gestionar tipo de no conformidad	3
	Gestionar tipo de artefacto	
	Gestionar indicador	
	Gestionar proyecto	
	Gestionar plantilla de solicitud	
	Gestionar plantilla de evaluación	
	Gestionar informe de evaluación	
	Asignar ingenieros	
	Gestionar lista de chequeo	
	Gestionar artefacto	
	Notificar indicador	
	Gestionar no conformidad	
	Reporte de evaluadores asignados a un proyecto.	
	Reporte de proyectos por evaluador.	
	Reporte de NC por Evaluador en una evaluación.	

Iteración 3	Reporte de indicadores evaluados por subcaracterística.	4
	Reporte de cantidad de NC por tipo en un rango de fechas.	
	Reporte de cantidad de NC por evaluación en un rango de fechas.	
	Reporte de NC por tipo por evaluación en un rango de fechas.	
	Comportamiento de la evaluación de subcaracterísca por evaluador.	
	Porciento de subcaracterísca por evaluación.	
	Reporte de NC significativas y No significativas por evaluación en rango de fechas.	
	Gestionar tema de debate	
	Gestionar comentario	

Tabla 6: Plan de duración de las iteraciones.

2.3.6 Plan de entrega

En este plan se concentran las funcionalidades referentes a un mismo tema en módulos, esto permite un mayor entendimiento en la fase de implementación. Tiene como objetivo definir el número de liberaciones que se realizarán en el transcurso del proyecto y las iteraciones que se requieren para desarrollar cada una.

Módulos	Historia de Usuario que Abarca
Administración	Gestionar fase desarrollo
	Gestionar subcaracterísticas usabilidad
	Gestionar tipo de no conformidad
	Gestionar tipo de artefacto
	Gestionar indicador
Evaluación	Gestionar proyecto
	Gestionar plantilla de solicitud
	Gestionar plantilla de evaluación
	Gestionar informe de evaluación
	Asignar ingenieros

	Gestionar lista de chequeo
	Gestionar artefacto
	Notificar indicador
	Gestionar no conformidad
Reportes	Reporte de evaluadores asignados a un proyecto.
	Reporte de proyectos por evaluador.
	Reporte de NC por Evaluador en una evaluación.
	Reporte de indicadores evaluados por subcaracterística.
	Reporte de cantidad de NC por tipo en un rango de fechas.
	Reporte de cantidad de NC por evaluación en un rango de fechas.
	Reporte de NC por tipo por evaluación en un rango de fechas.
	Comportamiento de la evaluación de subcaracterísca por evaluador.
	Porciento de subcaracterísca por evaluación.
	Reporte de NC significativas y No significativas por evaluación en rango de fechas.
Foro	Gestionar tema de debate
	Gestionar comentario
Seguridad	Registrar usuario
	Editar perfil

Tabla 7: Funcionalidades por módulos.

	Iteración 1	Iteración 2	Iteración 3
Módulo	**3ra semana de Marzo**	**3ra semana de Abril**	**2da semana de Mayo**
Administración	V1.0	V1.2 Final	Finalizado
Evaluación		V1.3	V1.6 Final
Reportes		V1.4	V1.7 Final
Foro		V1.5	V1.8 Final
Seguridad	V1.1	V1.9	V2.0 Final

Tabla 8: Plan de duración de entrega.

2.4 Fase de diseño

La metodología de desarrollo XP plantea prácticas especializadas que accionan directamente en la realización del diseño para lograr un sistema robusto y reutilizable. Asimismo hace especial énfasis en los diseños simples y claros. Los conceptos más importantes de diseño en esta metodología son: simplicidad, soluciones, recodificación y metáforas (46).

2.4.1 Tarjeta Clase Responsabilidad Colaborador (CRC)

Las tarjetas CRC son en la práctica pequeñas tarjetas que se elaboran para ser mostradas al cliente, de manera que se pueda llegar a un acuerdo sobre la validez de las abstracciones propuestas, lo que ayuda al equipo durante el diseño e implementación del sistema. Estas constituyen documentación adicional que será adjuntada a las HU (46).

Las tarjetas CRC trabajan con la técnica de modelado basada en objetos, representando cada tarjeta CRC a un objeto, identificando las clases y sus responsabilidades. Las tarjetas están compuestas por el nombre de la clase colocado como título, en la parte izquierda se colocan las responsabilidades (funcionalidades) y en la parte derecha las clases que se implican en cada funcionalidad (46).

Clase: FosUser	
Responsabilidad	**Colaborador**
Se encarga de controlar toda la información referente a un usuario.	Proyecto, Notificación

Tabla 9: Tarjeta CRC para la clase FosUser.

Clase: Notificación	
Responsabilidad	**Colaborador**
Se encarga de controlar toda la información referente a las notificaciones.	FosUser

Tabla 10: Tarjeta CRC para la clase Notificación.

Clase: FosUserController	
Responsabilidad	**Colaborador**
Se encarga de la interacción entre la vista y el modelo de la clase FosUser.	FosUserType, Controller, FosUser

Tabla 11: Tarjeta CRC para la clase FosUserController.

2.4.2 Estándares de codificación

XP promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo y que facilite la recodificación (46). En el caso de la herramienta del sistema que se desarrolla, los estándares que se utilizan en la implementación son:

- Todas las clases terminarán con una etiqueta ";", no usando la etiqueta de cierre ">".
- Las clases y métodos comenzarán con letra inicial minúscula, en el caso de ser un nombre compuesto por más de una palabra no serán separadas, y el resto de las palabras comenzarán con letra inicial mayúscula.
- Todas las interfaces utilizarán el sufijo interface.
- Todas las clases empezarán con el sufijo namespace.
- Se utilizará Symfony como el namespace de primer nivel.
- Las propiedades de las clases serán declaradas antes de los métodos.
- Se agregará un único espacio alrededor de operadores.
- No se utilizará espacios después de la apertura de un paréntesis y antes del cierre del mismo.

2.4.3 Patrones de diseño

2.4.3.1 Patrones GRASP

GRASP es el acrónimo de General Responsibility Assignment Software Patterns. Los GRASP son patrones para la asignación de responsabilidades, que ayudan a entender el diseño de objetos.

Incluso cuando se utilizan metodologías ágiles como XP, es necesario elegir cuidadosamente las clases adecuadas y decidir cómo estas deben interactuar (47).

A continuación se exponen los diferentes patrones manejados durante el proceso de desarrollo de la solución. Los mismos son implementados por Symfony 2, marco de trabajo utilizado. Seguidamente se ilustrarán cuáles son y cómo fueron utilizados:

- **Creador**: En las clases controladoras se encuentran todas las acciones definidas para el módulo AdministraciónBundle. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que las clases controladoras son creadoras de dichas entidades.
- **Experto**: Symfony utiliza el Doctrine para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades. Las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que simbolizan.
- **Bajo acoplamiento**: Las clases controladoras heredan solamente de Controller.php para lograr un bajo acoplamiento de las clases. Aparte de que el sistema posee pocas dependencias entre clases.
- **Alta cohesión**: Symfony 2 permite asignar responsabilidades con una alta cohesión ya que los controladores definen las acciones para las plantillas y colaboran con otras clases para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades, es decir está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas proporcionando que el software sea flexible frente a grandes cambios.
- **Controlador**: La responsabilidad de controlar el flujo de eventos del sistema se debe asignar a clases específicas. En la aplicación es utilizado a través del controlador frontal app.php, siendo el encargado de manejar todas las peticiones web. El controlador app.php es el punto de entrada único de toda la aplicación.

2.4.3.2 Patrones GoF

Según (48), estos se clasifican en tres categorías: de creación, estructurales y de comportamiento.

1. **Patrones creacionales:** Abstraen el proceso de creación de instancias. Se observan en la inicialización y configuración de objetos.
2. **Patrones estructurales:** Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.

3. **Patrones de comportamiento:** Más que describir objetos o clases, describen la comunicación entre ellos y la asignación de responsabilidades.

Algunos de los patrones de diseño GoF utilizados son los que se muestran a continuación:

- **Patrón command:** Este patrón se observa en la clase WebFrontController, en el método dispatch(). Esta clase está por defecto y es la encargada de establecer el módulo y la acción que se va a usar según la petición del usuario. Este patrón se aplica en la clase Routing, que está desactivada por defecto y procede según las necesidades del administrador del sistema donde se aplique el framework, la cual se puede activar o desactivar. En este método es analizada la URL con el objetivo de precisar los parámetros de la misma y de esta forma saber el método que debe responder a la petición.

- **Patrón decorator:** Este método pertenece a la clase abstracta View, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo nombrado layout.php es el que contiene el layout de la página. Este archivo conocido como plantilla global, guarda el código HTML que es usual en todas las páginas del sistema para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde otro punto de vista, el layout decora la plantilla. Este procedimiento es una implementación del patrón Decorador.

- **Patrón registry:** Este patrón es muy útil para los desarrolladores en la Programación Orientada a Objetos. Es un medio sencillo y eficiente de compartir datos y objetos en la aplicación sin la necesidad de preocuparse por conservar numerosos parámetros o hacer uso de variables globales. Este patrón se aplica en la clase Config, que es la encargada de acumular todas las variables de uso global en el sistema. Symfony aplica también el patrón “Front Controller” (Controlador frontal), por lo que posee una estructura bien organizada de controladores que comienza desde el “index.php” del ambiente y termina en los actions. Cada clase tiene su responsabilidad y es única, hay controladores que se encargan de la seguridad del sistema trabajando con ficheros YML y otros que se encargan de identificar mediante algunos datos las clases que deben realizar determinadas tareas (Patrón GoF Command, clase Routing) y las clases relacionadas con la configuración del sistema (Config, y Handler).

2.4.4 Diseño arquitectónico

Para la implementación de la solución se adopta un estilo en capas (capa de presentación, capa de negocio, capa de acceso a datos y capa de datos) y el patrón Modelo-Vista-Controlador. El mismo separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Es el encargado de aislar el modelo y la vista de los

detalles del protocolo utilizado para las peticiones. El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes del tipo de gestor de bases de datos utilizado por la aplicación (48). En la Ilustración 2 se ejemplifica el uso de este patrón según Symfony2 (34).

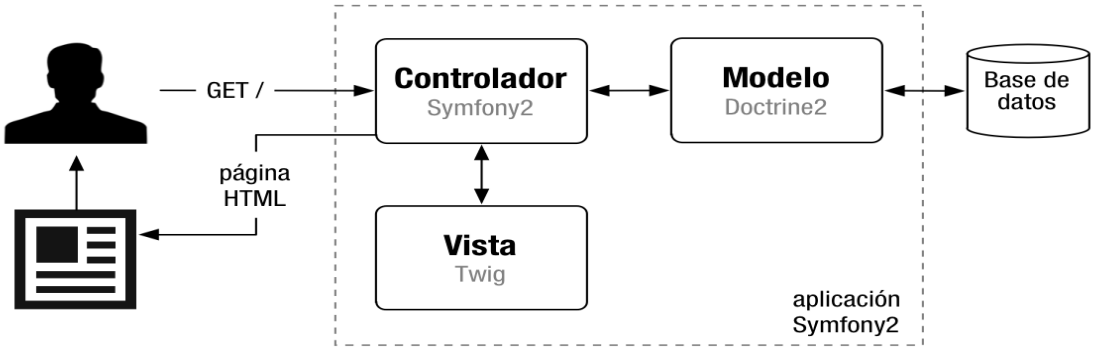


Ilustración 2: Modelo-Vista-Controlador en Symfony2.

2.4.5 Diseño de la base de datos

Una de las tareas más importantes para la elaboración de un sistema web es la construcción de la base de datos porque son estas las que permiten acceder a la información almacenada de manera fácil, tanto para consultarla como para añadir nuevos datos. Para el correcto diseño de la misma fue utilizado el patrón de diseño llave subrogada, que plantea se genere una llave primara única para cada entidad, en vez de usar un atributo identificador en el contexto dado. A continuación, en la Ilustración 3 se muestra el modelo de datos:

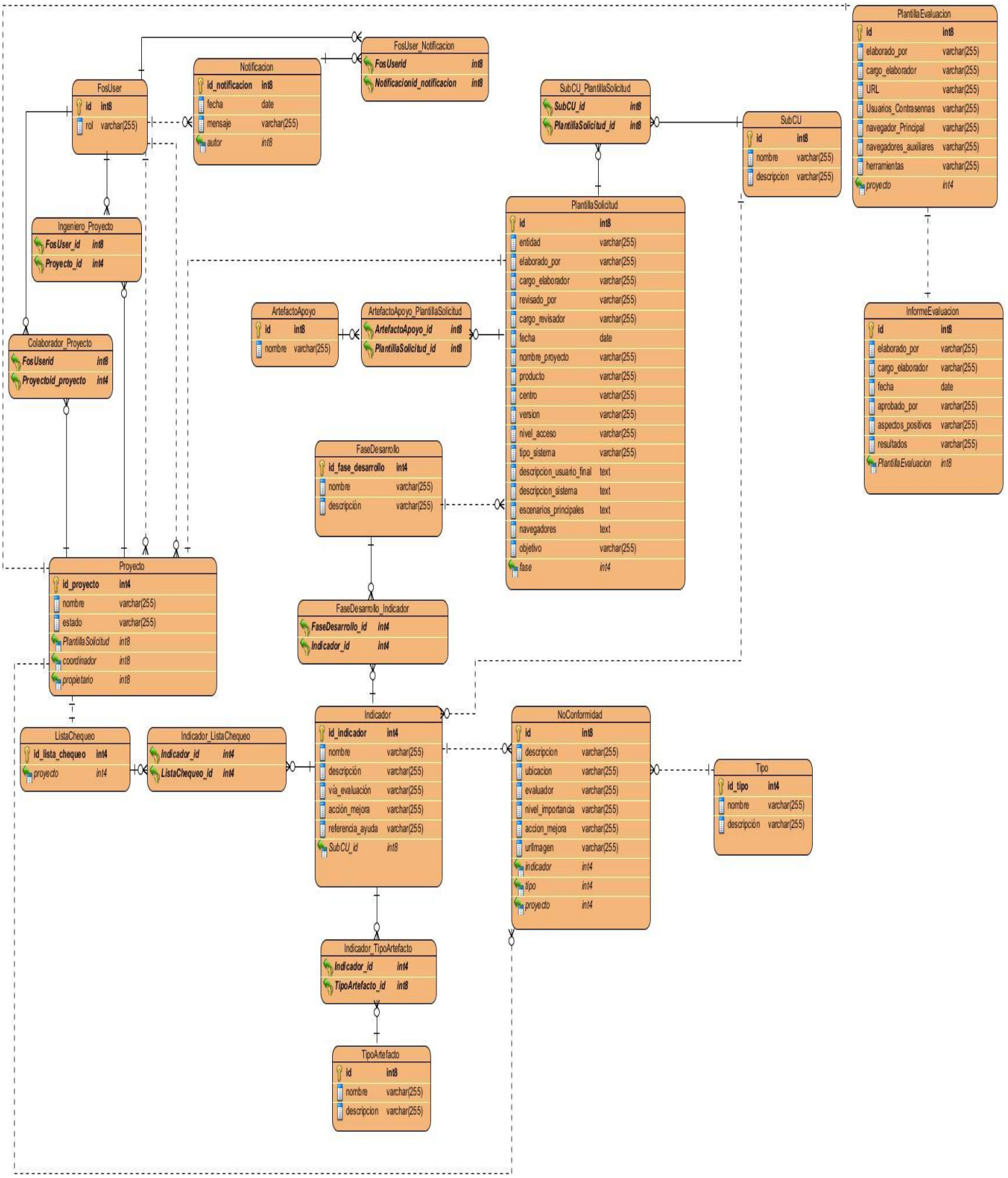


Ilustración 3: Modelo Entidad-Relación.

2.5 Implementación

En esta fase se realiza la implementación de las HU que fueron seleccionadas por cada iteración. Al inicio se lleva a cabo un chequeo del plan de iteraciones por si es necesario realizar modificaciones. Como parte de este plan se crean tareas de ingeniería para ayudar a organizar la implementación exitosa de las HU.

2.5.1 Tareas de ingeniería por iteraciones

Cada HU está compuesta por una o varias tareas de ingeniería, éstas no son más que pasos lógicos a seguir por el programador para realizar la implementación de una HU. A continuación se detallan para la iteración número uno, las tareas a desarrollar por cada HU.

Nro. de HU	Historias de Usuario	Nro. TI	Tarea de ingeniería por historias de usuario
1	Registrar usuario	1	Adicionar usuario
		2	Registrar y confirmar contraseña
		3	Adicionar correo
		4	Escoger Rol
2	Editar perfil	1	Editar datos
3	Gestionar fase desarrollo	1	Adicionar fase desarrollo
		2	Eliminar fase desarrollo
		3	Modificar fase desarrollo
		4	Listar fase desarrollo
4	Gestionar subcaracterística de usabilidad	1	Adicionar subcaracterística de usabilidad
		2	Eliminar subcaracterística de usabilidad
		3	Modificar subcaracterística de usabilidad
		4	Listar subcaracterística de usabilidad

Tabla 12: Tareas de ingeniería por iteración (Iteración 1).

2.5.2 Desarrollo de las tareas de ingeniería

A continuación se detallan el desarrollo de algunas de las tareas de la ingeniería divididas en las iteraciones correspondientes a cada historia de usuario.

Número de Tarea: 1	**HU (Nro.1):** Registrar usuario
Nombre de la tarea: Adicionar usuario	
Tipo de Tarea: Desarrollo	**Puntos estimados: 0.1**
Fecha Inicio: 9 de Marzo 2015	**Fecha Fin: 10 de Marzo 2015**
Programador Responsable: Joel Rego Pérez	
Descripción: Se muestra una interfaz donde se adiciona un nuevo usuario a la Base de Datos.	

Tabla 13: Descripción de la Tarea de Ingeniería 1. Adicionar usuario.

Número de Tarea: 2	**HU (Nro.1):** Registrar usuario
Nombre de la tarea: Registrar y confirmar contraseña	
Tipo de Tarea: Desarrollo	**Puntos estimados: 0.1**
Fecha Inicio: 9 de Marzo 2015	**Fecha Fin: 10 de Marzo 2015**
Programador Responsable: Joel Rego Pérez	
Descripción: Se muestra una interfaz donde se registra y se confirma la contraseña del usuario.	

Tabla 14: Descripción de la Tarea de Ingeniería 2. Registrar y confirmar contraseña.

Número de Tarea: 3	**HU (Nro.1):** Registrar usuario
Nombre de la tarea: Adicionar correo	
Tipo de Tarea: Desarrollo	**Puntos estimados: 0.1**
Fecha Inicio: 10 de Marzo 2015	**Fecha Fin: 11 de Marzo 2015**
Programador Responsable: Joel Rego Pérez	
Descripción: Se muestra una interfaz donde se añadirá una dirección de correo electrónico al usuario.	

Tabla 15: Descripción de la Tarea de Ingeniería 3. Adicionar correo.

Número de Tarea: 4	**HU (Nro.1):** Registrar usuario
Nombre de la tarea: Escoger rol	
Tipo de Tarea: Desarrollo	**Puntos estimados: 0.1**
Fecha Inicio: 11 de Marzo 2015	**Fecha Fin: 11 de Marzo 2015**
Programador Responsable: Joel Rego Pérez	
Descripción: Se muestra una interfaz donde se listan todos los roles seleccionables para que sean asignados al usuario.	

Tabla 16: Descripción de la Tarea de Ingeniería 4. Escoger Rol.

2.6 Conclusiones del capítulo

La elaboración de las HU de acuerdo a las pautas establecidas por la metodología de desarrollo XP permitió determinar tiempos de entrega, estimar esfuerzos y tiempo de desarrollo de la aplicación, quedando evidenciadas un total de 28 historias de usuarios distribuidas en cuatro iteraciones.

La aplicación de forma continua de estándares de codificación bien definidos y de patrones de diseño incrementó las posibilidades de que la aplicación se convirtiera en un sistema de software fácil de comprender y de mantener. Asimismo, la utilización de patrones de diseño en la creación de la base de datos resultó ser un proceder imprescindible para el cumplimiento de los requerimientos de calidad en la realización del diseño de la misma.

El desarrollo del sistema mediante el patrón arquitectónico MVC, garantizó la flexibilidad y extensibilidad del mismo.

La correcta distribución y desarrollo de las tareas de ingeniería por iteraciones permitió la implementación del Sistema Informático para la Gestión de Evaluaciones Heurísticas cumpliendo con los tiempos de entrega y las especificaciones descritas en las historias de usuario identificadas.

Capítulo 3: Pruebas y validación

3.1 Introducción

En el presente capítulo se hará énfasis en dos aspectos fundamentales durante el desarrollo de un software: la verificación del sistema que incluye las métricas para validar el diseño, técnicas de validación de requisitos, pruebas de caja blanca y caja negra, así como la validación de la investigación.

3.2 Validación de requisitos

Con la validación de los requisitos se comprueba la veracidad, consistencia y completitud de los mismos. Las técnicas existentes para desarrollar esta actividad son (49):

- **Revisiones:** consiste en la lectura y corrección de la documentación o modelado de la definición de requisitos.
- **Prototipos:** permite que el usuario visualice cómo será el diseño de las interfaces de la aplicación.

Para la validación de los requisitos se utilizaron dos técnicas, las revisiones de verificabilidad y los prototipos de interfaz. Se realizaron varias revisiones por el cliente en busca de anomalías. Los prototipos de interfaz fueron realizados por el equipo de desarrollo y presentados al cliente para ver si cumplían con las necesidades de los usuarios finales. A continuación se muestra un prototipo de interfaz en la Ilustración 4.

Ilustración 4: Interfaz de Usuario (Registrar Usuario).

3.3 Métricas para validar el diseño

Existen varias métricas para validar el diseño realizado. Es tarea del equipo de desarrollo determinar cuál o cuáles son más factibles a utilizar. Se emplean las métricas Tamaño Operacional de la Clase (TOC) y Relaciones entre Clases (RC), diseñadas para evaluar los siguientes atributos de calidad (50):

- **Responsabilidad:** Consiste en la responsabilidad asignada a una clase en un marco de modelado de dominio o concepto de la problemática propuesta.
- **Complejidad de implementación:** Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización:** Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento:** Consiste en el grado de dependencia o interconexión de una clase o estructura de clase con otras, está muy ligada a la característica reutilización.
- **Complejidad del mantenimiento:** Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta o directamente en los costos y la planificación del proyecto.
- **Cantidad de pruebas:** Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

3.3.1 Métrica Tamaño Operacional de Clases (TOC)

Está dado por el número de métodos asignados a una clase. A continuación se muestran las tablas 18 y 19 enfocadas a un mejor entendimiento de la utilización de esta métrica (51):

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

Tabla 17: Métrica Tamaño Operacional de Clase.

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

Tabla 18: Rango de valores para la métrica TOC.

En la Ilustración 5 se grafica detalladamente la cantidad de clases que contienen el número de métodos contenido en el rango establecido.

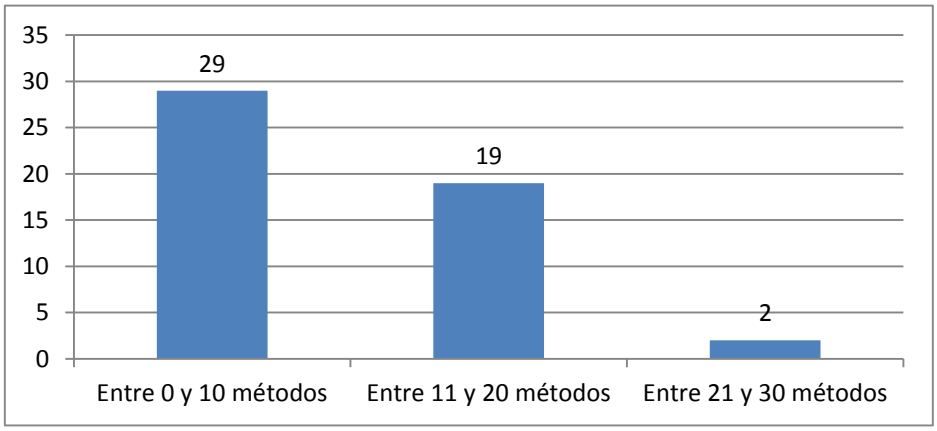


Ilustración 5: Representación de la métrica TOC.

En la Ilustración 6 se muestran los resultados obtenidos en porciento agrupados en los intervalos definidos.

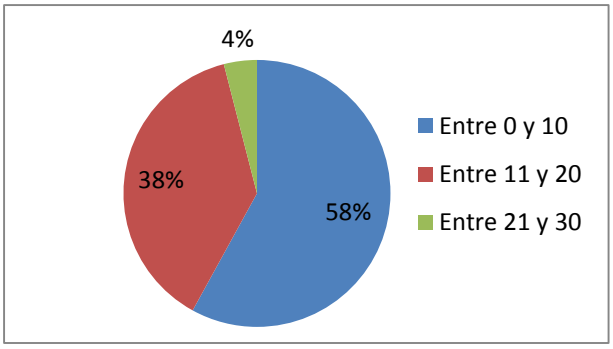


Ilustración 6: Representación en porciento obtenidos en la evaluación de la métrica TOC.

Es en la Ilustración 7 donde se observa la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo responsabilidad. Este gráfico muestra un resultado satisfactorio pues el 94% de las clases poseen una responsabilidad que está entre las categorías baja y media. Esta característica permite que en caso de fallos, como la responsabilidad está distribuida de forma equilibrada ninguna clase es demasiado crítica para dejar al sistema fuera de servicio.

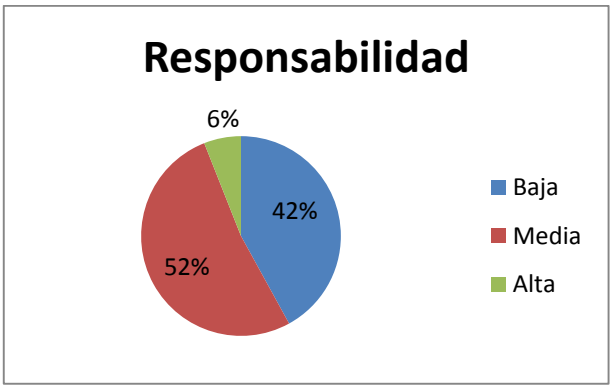


Ilustración 7: Resultado de la métrica TOC para el atributo responsabilidad.

En la siguiente figura, Ilustración 8, se observa la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo complejidad de implementación. Este gráfico muestra un resultado satisfactorio pues el 94% de las clases poseen una baja y media complejidad de implementación. Esta característica permite mejorar el mantenimiento y soporte de estas clases.

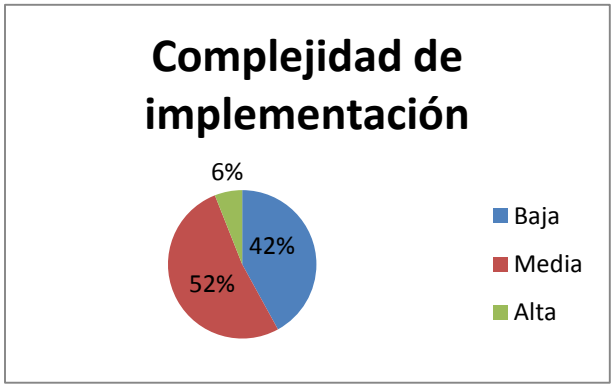


Ilustración 8: Resultado de la métrica TOC para el atributo complejidad de implementación.

En la Ilustración 9 se observa la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo reutilización. El diseño del sistema tiene un grado de eficiencia aceptable pues solamente el 6% del total de las clases poseen una baja reutilización.

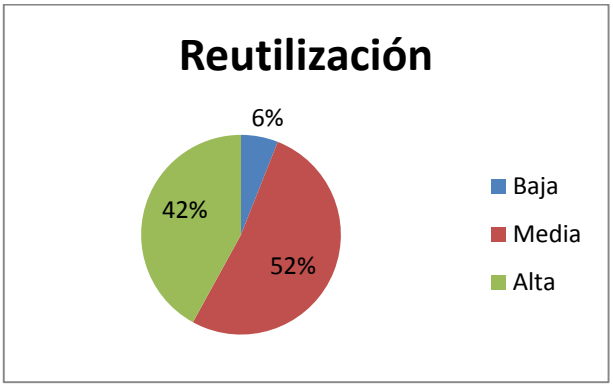


Ilustración 9: Resultados de la métrica TOC para el atributo reutilización.

Analizando los resultados obtenidos de la métrica TOC, se puede concluir que el sistema tiene una calidad aceptable teniendo en cuenta los resultados arrojados por los atributos analizados. Solo el 6% de las clases presentan una alta responsabilidad, alta complejidad de implementación y una baja reutilización, lo cual demuestra que el resultado es satisfactorio.

3.3.2 Métrica Relaciones entre Clases (RC)

Está dada por el número de relaciones de uso de una clase con otra. A continuación se muestran las tablas 20 y 21 para un mejor entendimiento de la utilización de esta métrica (51):

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

Tabla 19: Atributos de calidad evaluados por la métrica RC.

Atributo	Categoría	Criterio
Acoplamiento	Baja	1
	Media	2
	Alta	> 2
Complejidad de	Baja	<= Promedio

mantenimiento	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Reutilización	Baja	> 2 * Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	<= Promedio
Cantidad de pruebas	Baja	<= Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio

Tabla 20: Rango de valores para la métrica RC.

El promedio utilizado para evaluar el criterio es el resultado del cálculo del promedio de la columna cantidad de relaciones entre clases, en este caso el promedio es 2,16 que se aproximó a 2.

El gráfico de la Ilustración 10 refleja que el 58% tienen 1 dependencia, el 10% 2 dependencias y el 32% más de 2 de dependencias de otra clase. Este resultado es positivo, pues demuestra que el 68% de las clases se encuentran dentro de los niveles aceptables de calidad.

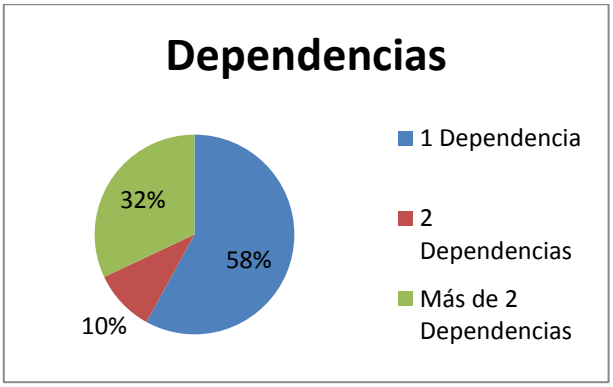


Ilustración 10: Representación de dependencias.

En la siguiente figura se muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo acoplamiento. Se evidencia un bajo acoplamiento entre las clases pues el 52% de las clases tienen dependencia con otra. Este resultado, mostrado en la Ilustración 11, es muy favorable para el diseño del sistema pues al existir poca dependencia entre las clases aumenta el grado de reutilización del sistema.

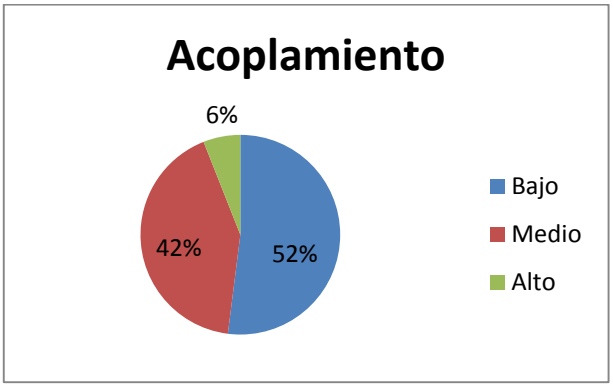


Ilustración 11: Resultados de la métrica RC para el atributo acoplamiento.

La Ilustración 12 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo complejidad de mantenimiento. El gráfico refleja el resultado aceptable del atributo pues el 67% de las clases presentan una baja complejidad de mantenimiento.

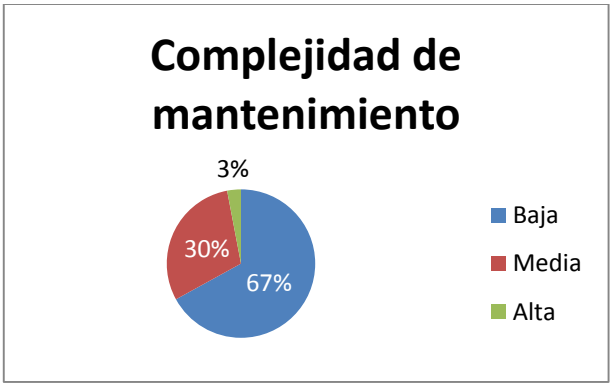


Ilustración 12: Resultados de la métrica RC para el atributo complejidad de mantenimiento.

En la Ilustración 13 se muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo reutilización. Esto evidencia que el 67% de las clases poseen una alta reutilización lo que es un factor fundamental que debe ser tenido en cuenta en el desarrollo de software.

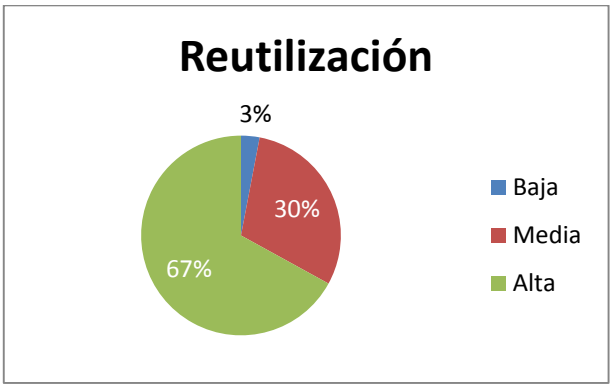


Ilustración 13: Resultados de la métrica RC para el atributo reutilización.

En la siguiente figura, Ilustración 14, se muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo cantidad de pruebas.

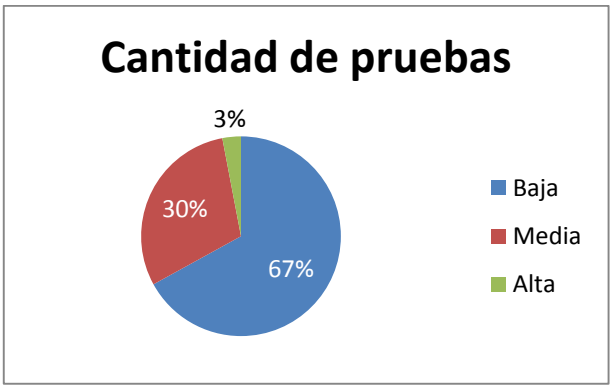


Ilustración 14: Resultados de la métrica RC para el atributo cantidad de pruebas.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del Sistema para la Gestión de Evaluaciones Heurísticas tiene una calidad aceptable.

La complejidad de mantenimiento y la cantidad de pruebas son bajas a un 67%, lo que representa valores favorables para el diseño realizado. Así mismo, existe un alto grado de reutilización al 67%, comportamiento también favorable para este atributo de calidad.

El 68% de las clases que conforman el sistema poseen menos de dos dependencias con otras clases. Los atributos de calidad se encuentran en un nivel satisfactorio; en el 58% de las clases el nivel de acoplamiento es mínimo.

3.4 Pruebas

3.4.1 Pruebas de Caja Blanca

Las pruebas unitarias son la forma de probar el correcto funcionamiento en código de un sistema informático, lo que es de vital importancia a la hora de obtener un sistema de calidad. Seguidamente se describirá la realización de una de las pruebas unitarias realizadas mediante la técnica del camino básico, desglosándola en sus elementos fundamentales: confección del grafo de flujo, cálculo de la complejidad ciclomática, extracción de los caminos independientes y realización de los casos de prueba.

```php
private function getOpciones()
{
    $usuario = $this->getUser();//1
    $opciones = array();//1

    if (in_array('ROLE_JEFE_GRUPO', $usuario->getRoles())) { //2
        $opciones['Taller de diseño'] = 'plantillasolicitud'; // 3
        $opciones['Listado de proyectos'] = 'proyecto'; //3


    } else if (in_array('ROLE_COORDINADOR', $usuario->getRoles())) { //4
        $opciones['Taller de diseño'] = 'plantillasolicitud'; //5
        $opciones['Listado de proyectos'] = 'proyecto'; //5

    } else if (in_array('ROLE_INGENIERO', $usuario->getRoles()))// ROLE_INGENIERO 6
    {
        $opciones['Listado de proyectos'] = 'proyecto';//7

    } else {
        $opciones['Listado de proyectos'] = 'proyecto';//8

    }

    return $opciones; //9
}

```

Ilustración 15: Código fuente de la funcionalidad *getOpciones().*

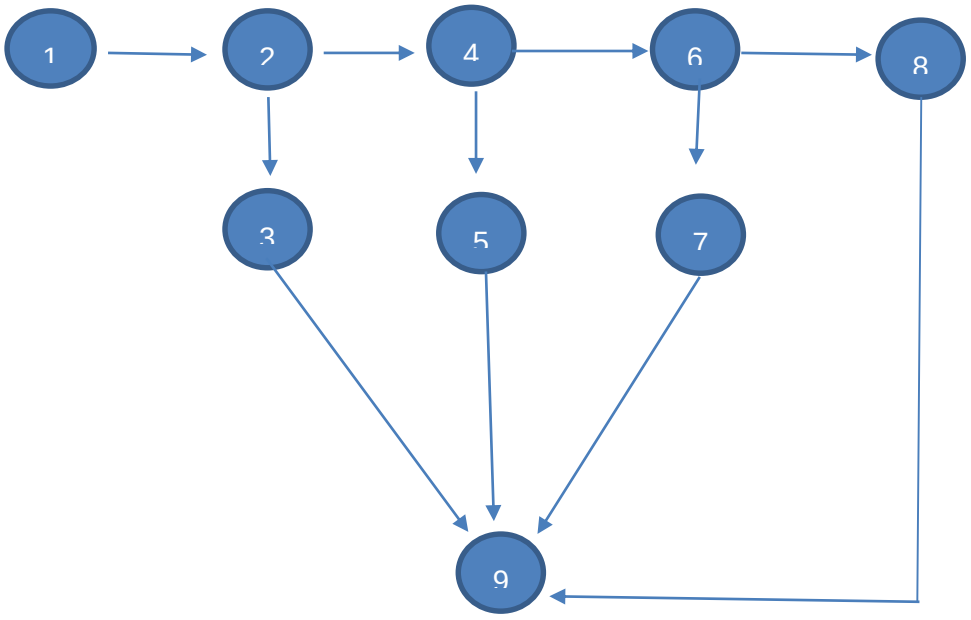


Ilustración 16: Grafo de flujo asociado a la funcionalidad *getOpciones().*

Posteriormente de haber construido el grafo se realiza el cálculo de la complejidad ciclomática mediante las tres fórmulas descritas a continuación. Dichas métricas deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad sea el correcto.

1. V (G) = R donde R representa la cantidad total de regiones**.**

V (G) = 4

2. V (G) = A – N + 2 donde **A** es el número de aristas del grafo de flujo y **N** es el número de nodos del mismo.

V (G) = 11– 9 + 2

V (G) = 4

3. V (G) = P + 1 donde **P** es el número de nodos predicado contenidos en el grafo de flujo (se denomina nodo predicado a los nodos de los cuales parten dos o más aristas).

V (G) = 3 + 1

V (G) = 4

Dado que el cálculo de la complejidad ciclomática arrojó el mismo resultado en las tres fórmulas se puede afirmar que esta tiene un valor de cuatro. Esto significa que existen cuatro posibles caminos por donde el flujo puede circular. Dicho valor representa el número mínimo de casos de pruebas para el procedimiento tratado.

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

Para cada camino básico definido se realiza un diseño de caso de prueba.

Caso de prueba para el camino básico #1 (1-2-3-9)	
Descripción	Debe determinar las opciones para los usuarios con el rol Jefe de Grupo en el Módulo Evaluación.
Condición de ejecución	Debe haber un usuario con este rol en el sistema
Entrada	usuario
Resultado esperado	Se le brindan las opciones al usuario
Resultado de la prueba	Satisfactorio

Tabla 21: Caso de prueba para el camino básico #1.

Caso de prueba para el camino básico #1 (1-2-4-5-9)	
Descripción	Debe determinar las opciones para los usuarios con el rol Coordinador en el Módulo Evaluación.
Condición de ejecución	Debe haber un usuario con este rol en el sistema
Entrada	usuario
Resultado esperado	Se le brindan las opciones al usuario
Resultado de la prueba	Satisfactorio

Tabla 22: Caso de prueba para el camino básico #2.

Caso de prueba para el camino básico #1 (1-2-4-6-7-9)	
Descripción	Debe determinar las opciones para los usuarios con el rol Ingeniero en el Módulo Evaluación.
Condición de ejecución	Debe haber un usuario con este rol en el sistema
Entrada	usuario
Resultado esperado	Se le brindan las opciones al usuario
Resultado de la prueba	Satisfactorio

Tabla 23: Caso de prueba para el camino básico #3.

Caso de prueba para el camino básico #1 (1-2-4-6-8-9)	
Descripción	Debe determinar las opciones para los usuarios sin rol en el Módulo Evaluación.
Condición de ejecución	Debe estar el usuario en el sistema
Entrada	usuario
Resultado esperado	Se le brindan las opciones al usuario
Resultado de la prueba	Satisfactorio

Tabla 24: Caso de prueba para el camino básico #4.

3.4.2 Pruebas de Caja Negra

Para la aplicación de las pruebas de caja negra se confeccionaron los diseños de caso de prueba asociados a cada una de las funcionalidades del sistema. La evaluación fue ejecutada por especialistas de CALISOFT. Fueron realizadas al sistema durante el proceso de evaluación, cuatro tipos de pruebas para comprobar el correcto funcionamiento del mismo: funcionales, usabilidad, seguridad y carga y estrés. Las no conformidades que fueron detectadas durante el proceso en cada

una de ellas están distribuidas en la tabla 25, con los resultados obtenidos luego del análisis de los errores.

Tipo de prueba	Total de no conformidades	Resueltas	No proceden
Funcionalidad	104	100	4
Usabilidad	50	50	0
Seguridad	11	5	6
Carga y estrés	1	1	0
Total	166	156	10

Tabla 25: Tabla de no conformidades detectadas.

3.4.3 Pruebas de aceptación del cliente

Las pruebas de aceptación son una herramienta para validar que el sistema cumple con el funcionamiento requerido y esperado por el cliente.

A continuación se muestra un ejemplo de los casos de prueba de aceptación aplicados y los resultados arrojados luego de haber sido presentados al cliente.

Caso de prueba de aceptación		
Código: 1	**Historia de Usuario (Nro. y nombre):** HU #1 Registrar usuario.	
Funcionalidad que se prueba: Registrar usuario.		
Condiciones de ejecución: El usuario que está autenticado debe tener privilegios de especialista.		
Acción	**Datos de entrada**	**Resultado esperado**
Se escoge la opción Usuarios		El sistema debe mostrar un listado de los usuarios registrados en el sistema, con sus respectivos roles.
Se elige la opción Registrar		El sistema muestra un formulario con todos los datos necesarios para la inserción de un nuevo usuario.
Se guardan los datos introducidos en cada uno de los campos del formulario.	Nuevo usuario	El sistema debe mostrar un mensaje indicando que la acción se ha realizado satisfactoriamente. Mostrar el listado de usuarios incluyendo el adicionado.
Resultado de la prueba: Satisfactorio		

Tabla 26: Caso de Prueba de Aceptación. HU#1 Registrar Usuario.

Fueron presentados 28 casos de prueba de aceptación al cliente. El proceso de revisión de los mismos tardó tres días en completarse y fueron solicitados un total de 17 pedidos de cambio, de ellos 15 fueron aceptados y solucionados por el equipo de desarrollo y dos relegados para próximas versiones.

3.5 Validación de la Investigación

El proceso de Evaluación de Usabilidad llevado a cabo en el Departamento de Evaluaciones de Productos de Software está compuesto por un total de 13 actividades. De ellas las relacionadas con la generación de la lista de chequeo, plantillas e informes y la realización de la evaluación son las más críticas en función del tiempo que toma en ser cumplidas por los especialistas, teniendo en cuenta que participa en la prueba un grupo comprendido entre tres y cinco evaluadores. Este proceso como se ejecuta actualmente tiene una duración aproximada de ocho días de trabajo.

Para realizar un análisis del comportamiento del tiempo de duración de una evaluación, se realizaron dos procesos, uno realizando la prueba de forma manual y otro utilizando, para la gestión del mismo, la propuesta de solución que fue implementada. Dado que los dos sistemas que fueron evaluados corresponden a la misma categoría, la cantidad de indicadores a evaluar en ambos fue la misma, por lo que el hecho de que sean sistemas diferentes no afectó el tiempo de ejecución de la prueba.

Para la ejecución del proceso de forma manual fue seleccionado como producto de muestra el portal del Partido Comunista de Cuba. Participaron en la prueba un total de cinco especialistas del DEPS, los cuales realizaron la evaluación utilizando la lista de chequeo que fue creada de forma manual con un total de 130 indicadores que previamente fueron seleccionados de un documento donde se encuentran descritos. Además identificaron las no conformidades las cuales debieron registrar llenando una plantilla, para su posterior análisis durante la conciliación. En la reunión de conciliación se creó con las no conformidades seleccionadas durante el análisis el Registro de la Evaluación y luego el Informe Final que se entregó al cliente. La realización de las actividades definidas por el proceso tomó un total de siete días.

Se seleccionó el Portal Web de la Fiscalía General de la República como producto de muestra a evaluar utilizando el sistema SIGEH. Participaron en la prueba un total de cinco especialistas del DEPS, los cuales realizaron la evaluación utilizando la lista de chequeo generada automáticamente en el sistema con un total de 130 indicadores. Igualmente identificaron las no conformidades y una vez que terminaron, quedaron registradas en la aplicación para su posterior análisis durante la conciliación. Al finalizar el proceso los documentos fueron generados automáticamente para ser enviados al cliente con la incorporación de nuevos datos que aportan información sobre la medida de usabilidad del producto. La realización de las actividades definidas por el proceso tomó un total de cuatro días.

Se puede comprobar que la reducción del tiempo es significativa. Como se evidencia en la Tabla 27, las actividades se realizaron de manera más eficiente utilizando el sistema SIGEH que de forma manual. Se obtuvo como resultado para la duración total de la prueba al Portal Web de la Fiscalía General de la República solo la mitad del tiempo que fue empleado en la evaluación del portal del Partido Comunista de Cuba.

Etapas del Proceso	Pruebas Manuales	Pruebas utilizando SIGEH
Seleccionar Coordinador de Usabilidad	1/2 Hora/hombre	1/4 Hora/hombre
Elaborar solicitud con el cliente	1/2 Hora/hombre	1/4 Hora/hombre
Definir contexto de Uso	1/2 Hora/hombre	1/4 Hora/hombre
Definir contexto de Evaluación	1/2 Hora/hombre	1/4 Hora/hombre
Explorar el artefacto	3 Hora/hombre	3 Hora/hombre
Crear Lista de Chequeo	8 Hora/hombre	Automático
Seleccionar Ingenieros de Pruebas de Usabilidad	1/4 Hora/hombre	Automático
Ejecutar Evaluación	24 Hora/hombre	24 Hora/hombre
Documentar Evaluación	16 Hora/hombre	Automático
Conciliar No Conformidades	8 Hora/hombre	8 Hora/hombre
Emitir Evaluación	1 Hora/hombre	Automático
Tiempo total de duración	7 Días	4 Días

Tabla 27: Comparación del tiempo empleado en la realización de actividades del proceso.

3.6 Conclusiones del capítulo

Mediante las revisiones de verificabilidad y los prototipos de interfaz de usuario, se logró constatar por parte de los usuarios, que los requisitos especificados son válidos, consistentes, y completos.

La utilización de técnicas como el Tamaño Operacional de Clases y la de Relaciones entre Clases permitieron validar y evidenciar que el diseño propuesto para la aplicación es satisfactorio.

Con la realización de pruebas de Caja Blanca, utilizando la técnica camino básico, fue posible conocer y establecer el mínimo de casos de pruebas a realizar, garantizando que todas las rutas independientes dentro de los métodos se ejecuten al menos una vez.

Con la realización de las pruebas de Caja Negra por parte de CALISOFT al sistema se detectaron las no conformidades asociadas al mismo, siendo analizadas y solucionadas por el equipo de desarrollo. Permitiendo certificar la calidad de la herramienta implementada.

Mediante el diseño y aplicación de los casos de prueba de aceptación se lograron valorar los resultados y verificar que las funcionalidades implementadas cumplieran con las restricciones definidas por el DEPS, evidenciándose la satisfacción de los requisitos funcionales acordados con el cliente**.**

Posteriormente con la validación de la idea a defender se demostró que con la utilización de la solución planteada se reduce sustancialmente el tiempo de realización de las evaluaciones.

Conclusiones generales

Para el desarrollo del presente trabajo de diploma se realizó un análisis y una valoración de las herramientas que gestionan y realizan el proceso de evaluaciones heurísticas, tanto a nivel internacional como nacional, posibilitando determinar que en Cuba no hay ninguna que se adapte al procedimiento establecido en el Departamento de Evaluaciones de Software en el Centro Nacional de Calidad de Software. El uso conjunto de las metodologías, herramientas y tecnologías facilitó la implementación de un sistema capaz de gestionar todos los procesos que se describen en el procedimiento definido por el Departamento de Evaluaciones de Productos de Software.

Se logró la implementación del Sistema Informático para la Gestión de Evaluaciones Heurísticas, siguiendo las pautas establecidas por la metodología de desarrollo XP. Dicho sistema está caracterizado por la utilización del patrón Modelo-Vista-Controlador propuesto por el marco de trabajo Symfony2.

Se verificó la elaboración correcta del sistema desde el punto de vista ingenieril con la validación de los requisitos mediante las pruebas de verificabilidad y los prototipos no funcionales, la validación del diseño propuesto mediante la utilización de las métricas Tamaño Operacional de Clases y Relaciones entre Clases, las pruebas de Caja Blanca mediante la utilización de la técnica de Camino Básico, las pruebas de Caja Negra y la utilización de Casos de Pruebas de Aceptación del Cliente.

Finalmente se validó la idea a defender planteada mediante la comprobación del sistema por parte de un conjunto de especialistas del centro. Realizando la evaluación a un software usando la solución implementada como soporte, se dejó patente como resultado una notable reducción del mismo.

Recomendaciones

A lo largo de la presente investigación fueron identificados elementos que no se contemplaron en el desarrollo de la solución al no ser parte del alcance inicial, pero se considera que son relevantes e incrementarían la utilidad del sistema y la calidad de uso del mismo:

1. Implementar la versión para dispositivos móviles del sistema.
2. Implementar de una forma más eficiente la interfaz de la lista de chequeo.
3. Exportar los documentos con el formato de las plantillas utilizadas por el centro para los documentos oficiales (actualmente en rediseño).

Bibliografía

1. BURGOS, J and FERNÁNDEZ de ARRIBA, M. Development of adaptive Web sites with usability and accessibility features. Oviedo : s.n., 2002. pp. 501-504.

2. NIELSEN, Jakob. Heuristic evaluation. [Autor del libro] MACK R. L. NIELSEN Jakob. Usability Inspection Methods. s.l. : John Wiley & Sons, 1994.

3. Heuristic evaluation of user interfaces. NIELSEN, Jakob and MOLICH, Rolf. New York : s.n., 1990. CHI '90 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems . pp. 256-349.

4. PERCEPTION TECHNOLOGIES, S.L. Perception. [En línea] 2015. [Citado: Junio 14, 2015.] http://www.perception.es/servicios/para-mejorar-la-pagina-que-tienes/realizar-pruebas-de-usabilidad.

5. HERNÁNDEZ LEÓN, Rolando y COELLO Sayda. El proceso de la investigación científica. 2011.

6. GARCÍA GUTIERREZ, Antonio. Lingüística Documental. Barcelona : Mitre, 1984.

7. gestiopolis. Métodos y técnicas de investigación. [En línea] 2015. [Citado: Enero 12, 2015.] http://www.gestiopolis.com/metodos-y-tecnicas-de-investigacion/.

8. 9241-11, ISO CD. Guidelines for specifying and measuring usability. 1993.

9. 9126, ISO. Software product evaluation - Quality characteristics and guidelines for their use. 1991.

10. 25010, ISO/IEC FCD. Systems and software engineering - System and software product Quality Requirem and Evaluation (SQuaRE) - System and software quality models. Montréal : Department of Industrial and Management Systems Eng, 2010.

11. Usability engineering. NIELSEN, Jakob. California : Academic Press, 1993.

12. ZULUETA POZO, Delmys. Procedimiento para evaluar la Usabilidad de Productos Software mediante Evaluaciones Heurísticas. Centro Nacional de Calidad de Software. La Habana : s.n., 2015. p. 3.

13. Karat, J. User-centered software evaluation methodologies. [Autor del libro] Landauer T.K., Prabhu P. Helander M. Handbook of human-computer interaction. Segunda. Nueva York : Elsevier Science, 1997.

14. Mayhew, D. The Usability Engineering Lifecycle. California. : s.n., 1999.

15. Nielsen, Jakob. Big paybacks from discount. Usability engineering. s.l. : Addison Wesley, 1990. pp. 107-108.

16. WHITESIDE, John, BENNETT, John and HOLZBLATT, Karen. Usability engineering: Our experience and evolution. s.l. : Elsevier Science Publishers, 1998.

17. WHITEFIELD, Andy, WILSON, Frank and DOWELL, John. A framework for human factors evaluation, behaviour and Information Technology. s.l. : Taylor & Francis, 1991. pp. 65-79.

18. WIXON, Dennis y WILSON, C. The usability-engineering framework for product design and evaluation. Segunda Edición. California : Elsevier Science, 1997. pp. 653-688.

19. PREECE, Jenny. A Guide to Usability: Human factors in computing. Boston : Addison Wesley, 1993.

20. Empiricism versus judgement: Comparing user interface evaluation methods on a new telephone-based interface. DESURVIRE, Heather, LAWRENCE, Debbie and ATWOOD, Michael. 1991, ACM SIGCHI Bulletin, Vol. 4, pp. 58-59.

21. Paper versus computer implementations as mockup scenarios for heuristic evaluation. NIELSEN, Jakob. Cambridge : s.n., 1990. Human-Computer Interaction. pp. 315-320.

22. NIELSEN, J and MACK, R. Usability inspection methods. New York : John Wiley & Sons, 1994.

23. Usability inspection methods. MACK R, NIELSEN J. Junio 1993, ACM SIGCHI Bulletin, pp. 23-33.

24. JACOBSEN, N, HERTZUM, M and JOHN, B. The evaluator effect in usability tests. 1998. pp. 255-256.

25. A resource support toolkit (R-IDE): supporting the DECIDE. KEMP ELIZABETH, SETUNGAMUDALIGE D T. 2006. CHINZ '06 Proceedings of the 7th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction: design centered HCI. pp. 61-66.

26. Evaluación heurística de sitios web academicos Latinoamericanos dentro de la iniciativa UsabAIPO. GONZALEZ, Paula, et al. Leéida : Universidad de Lleida, 2006, pp. 143-153.

27. SANCHEZ J, GIL ROSA M, OLIVA M. Más allá del cuchillo de palo: hacia una herramienta integrada para un verdadero diseño centrado en el usuario. Valencia : s.n., 2010.

28. CARRERAS, Olga. Usable Accesible. [En línea] Julio 20, 2011. [Citado: Enero 23, 2015.] http://olgacarreras.blogspot.com.es/2011/07/sirius-nueva-sistema-para-la-evaluacion.html.

29. BOOCH, Jacobson. El proceso unificado de desarrollo de software. s.l. : Addison Wesley, 2005.

30. ProcesosdeSoftware. [En línea] 2015. [Citado: Marzo 3, 2015.] http://procesosdesoftware.wikispaces.com/METODOLOGIA+XP.

31. Visual-Paradigm. [En línea] 2015. [Citado: Marzo 3, 2015.] http://www.visual-paradigm.com/aboutus/.

32. OMG. Unified Modeling Language. [En línea] 2015. [Citado: Marzo 3, 2015.] http://www.uml.org/.

33. EGUILUZ, Javier. Desarrollo web ágil con Symfony2. Primera. 2013. p. 20.

34. —. Desarrollo web ágil con Symfony2. Primera. 2013. p. 117.

35. Doctrine, Team. Doctrine. [En línea] 2006. [Citado: Junio 12, 2015.] http://www.doctrine-project.org/about.html.

36. COCHRAN, David. Twitter Boostrap Web Development. s.l. : Packt Publishing, 2012. p. 100.

37. GAUCHAT, J. El gran libro de HTML5, CSS3 y Javascript. Primera. s.l. : Marcombo, 2012.

38. PHP. [En línea] PHP Group, 2015. [Citado: Febrero 12, 2015.] http://php.net/releases/5_4_0.php..

39. PostgreSQL-es. [En línea] PosgreSQL-es, 10 2, 2010. [Citado: 2 6, 2015.] http://www.postgresql.org.es/sobre_postgresql.

40. MOMJIAM, Bruce. PostrgreSQL Introduction and Concepts. Boston : Addison Wesley, 2001.

41. NetBeans IDE. [En línea] Oracle, 2015. [Citado: Marzo 3, 2015.] https://netbeans.org/community/releases/80/.

42. Apache HTTP Server Project. [En línea] Apache Software Foundation, 2015. [Citado: Enero 20, 2015.] http://httpd.apache.org/ABOUT_APACHE.html.

43. SOMMERVILLE, Ian. Ingeniería del Software. Séptima. Massachusetts : Addison Wesley, 2005.

44. Metodología para el diseño y desarrollo de interfaces de usuario web. GARCÍA BALMASEDA, Grilian and VELAZCO SOCARRÁS, Yudenia. 9, La Habana : Grupo Editorial Ediciones Futuro, 2010, Serie Científica de la Universidad de las Ciencias Informáticas (SC-UCI), Vol. 3.

45. Glosario de Informática e Internet. [En línea] [Citado: Marzo 2, 2015.] http://www.internetglosario.com/.

46. JOSKOWICZ, José. Reglas y Prácticas en eXtreme Programming. España : s.n., 2008.

47. LARMAN, Craig. UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Segunda. México : Pearson, 1999. p. 162.

48. PAVÓN MESTRAS, Juan. Estructura de las Aplicaciones Orientadas a Objetos. El patrón Modelo-Vista-Controlador (MVC). Madrid : Departamento de Informática. Universidad Complutense de Madrid., 2011.

49. RAMÍREZ FERNÁNDEZ, Adelaida. Clasificación de tipos de requisitos para la mejora del proceso de desarrollo de software. Madrid : Universidad Carlos III de Madrid. Departamento de Informática, 2012.

50. VÁZQUEZ P, MORENO M, GARCÍA F. MÉTRICAS ORIENTADAS A OBJETOS. Salamanca : Departamento de Informática y Automática, 2001.

51. Desarrollo de Software. [En línea] IngSoftwareII-CUFM , Noviembre 2012. [Citado: Marzo 26, 2015.] https://cufmingsoftware.wordpress.com/estandares-de-diseno/.