Universidad de las Ciencias Informáticas

Facultad 2

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

# Herramienta para la evaluación centrada en el usuario de la usabilidad de aplicaciones web en la Universidad de las Ciencias Informáticas

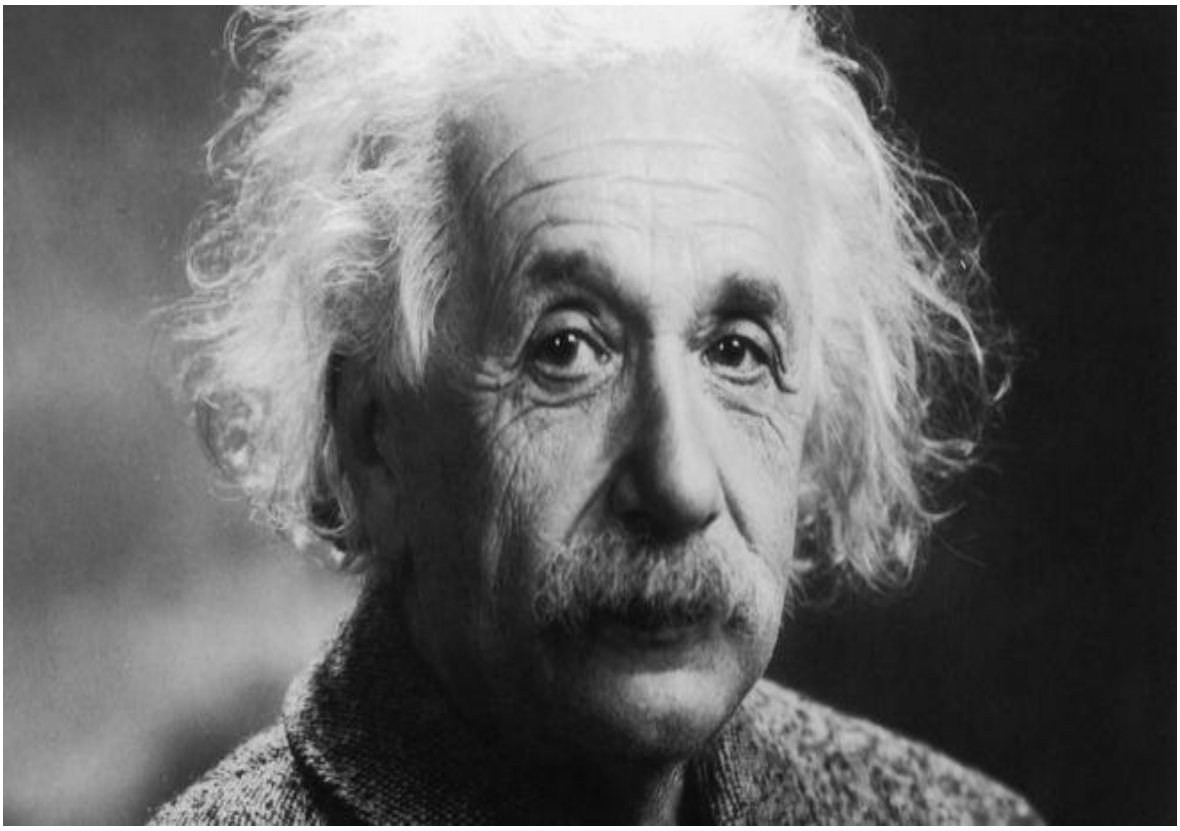

## Autora:

Jennifer Gispert Luaces.

## Tutores:

Ing. Lionel Rodolfo Baquero Hernández


**La Habana, 6 de junio del 2019**



# “Soy agradecido con todos los que me dijeron que no, por ellos lo hice yo mismo”

**Albert Einstein**

## Declaración de autoría

Declaro ser autora de la presente tesis que tiene por título: Herramienta para la evaluación centrada en el usuario de la usabilidad de aplicaciones web en la Universidad de las Ciencias Informáticas para optar por el título de Ingeniera en Ciencias Informáticas. Declaro el que asumo la responsabilidad moral y jurídica que se derive de este juramento profesional. Para que así conste firmo la presente a los 6 días del mes de junio del año 2019.

_____

Jennifer Gispert Luaces.

Autora

_____

Ing. Lionel Baquero Hernández

Tutor.

## Datos de contacto

**Síntesis de los Tutores:**

Ing. Lionel Rodolfo Baquero Hernández

Ingeniero en Ciencias Informáticas, graduado en el 2016, actualmente se desempeña como Profesor Instructor de Inteligencia Artificial del Departamento de Programación y Sistemas Digitales de la Facultad 2 de la Universidad de las Ciencias Informáticas.

Email: baquero@uci.cu

**Datos de la autora:**

Jennifer Gispert Luaces.

Universidad de las Ciencias Informáticas, La Habana, Cuba.
Email: jgispert@estudiantes.uci.cu

*A mis padres para darles otro motivo que los haga sentirse orgullosos de mí.*

## Resumen

Un atributo de calidad de software primordial es la usabilidad, por lo que esta influye directamente en el éxito de un producto en el mercado. Actualmente la Universidad de las Ciencias Informáticas (UCI), especializada en el proceso de desarrollo de software cuenta con el Laboratorio de Prueba de Software, el cual evalúa la usabilidad como característica deseable, realizando estas pruebas sobre el criterio de los expertos. Por ello surge la necesidad de desarrollar una herramienta que evalúe la usabilidad centrado en el usuario, con el fin de conocer si cumplen con las expectativas de los clientes. Existen herramientas que evalúan la usabilidad de un sistema, sin embargo, contienen un conjunto de limitaciones que impiden su aplicación en la universidad. El objetivo de la presente investigación es desarrollar una herramienta que realice la evaluación de la usabilidad centrada en el usuario de las aplicaciones web producidas en la UCI. La herramienta es capaz de devolver un reporte final donde se muestra un conjunto de elementos relacionados con los aspectos de usabilidad y el grado de usabilidad que alcanzan las aplicaciones.

**Palabras clave:** aplicaciones web, calidad de software, evaluación de la usabilidad, usabilidad

## Abstract

A primary software quality attribute is the usability, so it directly influences the success of a product in the market. Currently the University of Informatics Sciences (UCI), specialized in the process of software development has the Software Testing Laboratory, which evaluates usability as a desirable feature, performing these tests on the criteria of experts. Therefore, the need arises to develop a tool that evaluates usability centered on the user, in order to know if they meet customer expectations. There are tools that evaluate the usability of a system, however, contain a set of limitations that prevent its application in the university. The objective of this research is to develop a tool that performs user-centered usability assessment of web applications produced in the UCI. The tool is capable of returning a final report showing a set of elements related to the usability aspects and the degree of usability reached by the applications.

**Keywords:** software quality, usability evaluation, usability, web applications

## Índice

## Introducción

La Ingeniería de Software (ISW) abarca un proceso, una colección de métodos y una variedad de herramientas que permiten a los profesionales construir software de alta calidad [Pressman y Maxim, 2015]. Evaluar la calidad del software es costoso y complejo por la cantidad de recursos involucrados, resultando de vital importancia para la toma de decisiones [Fernández, 2018].

Los modelos de calidad de software existentes coinciden en la importancia de evaluar la usabilidad como característica deseable. La usabilidad determina, en gran medida, cómo los usuarios finales perciben la calidad de un software, de ahí la elevada importancia que en la actualidad le es conferida. Una baja usabilidad puede causar que los usuarios sientan que el software no cumple sus expectativas o que no se adapta a sus necesidades. La usabilidad determina el éxito de un producto en el mercado.

En la Universidad de las Ciencias Informáticas (UCI), se evalúa la usabilidad como aspecto importante que determina la calidad de las aplicaciones web que se desarrollan. Este aseguramiento se sustenta en realizar pruebas para la validación y verificación de la usabilidad en varios momentos del proceso de desarrollo de las aplicaciones web. En los Laboratorios de Pruebas de Software (LPS) de la Dirección de Calidad de Software de la universidad se evalúa la usabilidad utilizando evaluación heurística como método de inspección centrado en los expertos. No se realizan evaluaciones centradas en el usuario, aunque se han propuesto teóricamente formas de automatización para ello.

Las evaluaciones centradas en los expertos de los LPS se realizan manualmente con una lista de chequeo como artefacto de apoyo. Esta forma de evaluación genera una alta incertidumbre debido a que los resultados de la evaluación están sujetos al criterio de los expertos y puede variar con cada evaluador. Un aseguramiento de la calidad centrado solo en los expertos no es recomendable en el desarrollo de las aplicaciones web, debido a que esta forma de evaluación no sustituye la evaluación centrada en el usuario.

Al no realizarse evaluación centrada en los usuarios no se tiene una medida de cómo estos aprecian la usabilidad de las aplicaciones web. Esta es una de las principales limitaciones de la evaluación centrada en los expertos, que genera evaluaciones que pueden estar lejanas a la usabilidad real que el usuario percibe.

Entre las principales causas por las que en los LPS no se realiza evaluación centrada en los usuarios es su falta de automatización. Estos tipos de pruebas son muy costosas por la preparación del laboratorio y la interpretación de los datos que se obtienen. Los volúmenes de datos obtenidos son altos y son por su

naturaleza vagos e imprecisos, además pueden estar desordenados y provenir de diferentes fuentes. Esto trae consigo que sea necesario automatizarlas para ganar en certidumbre en los resultados.

Lo anteriormente planteado permite identificar las siguientes deficiencias en la evaluación de la usabilidad de las aplicaciones web desarrolladas en la UCI:

- Alta incertidumbre en la evaluación basándose solo en el criterio de los expertos.
- Falta de retroalimentación del criterio de los usuarios sobre la usabilidad de las aplicaciones web.
- Dificultades en el proceso de evaluación de la usabilidad de las aplicaciones web desarrolladas debido a la falta de automatización.

Considerándose lo analizado anteriormente, se identifica el siguiente **problema a resolver:** ¿Cómo realizar la evaluación centrada en el usuario de la usabilidad de las aplicaciones web desarrolladas en la Universidad de las Ciencias Informáticas? A partir del problema planteado se define como **objeto de estudio:** evaluación de la usabilidad de las aplicaciones web.

Se define como **objetivo general:** desarrollar una herramienta para la evaluación centrada en el usuario de la usabilidad de las aplicaciones web producidas en la Universidad de las Ciencias Informáticas. Concretando el desarrollo de la investigación en el **campo de acción:** evaluación de la usabilidad de las aplicaciones web desarrolladas en la UCI.

Para guiar el proceso investigativo se plantean las siguientes **preguntas de investigación:**

- ¿Cuáles son los fundamentos teóricos metodológicos relacionados con la evaluación de la usabilidad de las aplicaciones web y la realización de este proceso en la UCI?
- ¿Qué características tienen las herramientas existentes que automatizan la evaluación de la usabilidad de las aplicaciones web?
- ¿Qué técnicas, tecnologías y herramientas de desarrollo se pueden utilizar en la implementación de una herramienta para la evaluación centrada en el usuario de las aplicaciones web desarrolladas en la UCI?
- ¿Qué funcionalidades y características debe tener una herramienta para la evaluación centrada en el usuario de las aplicaciones web desarrolladas en la UCI?

- ¿Qué tipos de pruebas se pueden utilizar en la validación de la herramienta implementada para la evaluación centrada en el usuario de la usabilidad de las aplicaciones web y cómo se deben aplicar?

Para alcanzar el objetivo general propuesto se definen las siguientes **tareas de la investigación:**

1. Elaboración de un marco teórico-referencial sobre la evaluación de la usabilidad de las aplicaciones web y la realización de este proceso en la UCI para identificar limitaciones y deficiencias en su funcionamiento.
2. Análisis de herramientas que automatizan la evaluación de la usabilidad de las aplicaciones web para tomar características que apoyen el desarrollo de la aplicación.
3. Diseño de una herramienta que realiza la evaluación centrada en el usuario de la usabilidad de las aplicaciones web en la UCI para su posterior implementación.
4. Implementación de la herramienta diseñada para la evaluación centrada en el usuario de la usabilidad de las aplicaciones web en la UCI.
5. Realización de pruebas de software a la herramienta desarrollada para la evaluación centrada en el usuario de la usabilidad de las aplicaciones web.

Con el propósito de resolver el problema y lograr los objetivos de la investigación se utilizaron los siguientes **métodos científicos:**

**Métodos Teóricos**

- Analítico-Sintético: para sintetizar las ideas y conceptos empleados en el campo de la usabilidad de las aplicaciones web, expresando de manera resumida la posición del investigador.
- Histórico-Lógico: para realizar un estudio crítico de los sistemas existentes en el contexto de la evaluación de las aplicaciones web y utilizarlos como punto de referencia y comparación, además de constatar teóricamente cómo ha evolucionado el tema en el tiempo.
- Modelación: para la creación de abstracciones del proceso de evaluación de la usabilidad de las aplicaciones web que actuaron como eslabón intermedio entre el investigador y el campo de acción.

**Métodos Empíricos:**

- Observación: en el análisis de la evaluación de la usabilidad de las aplicaciones web en la UCI.
- Entrevista: realizada a los directivos y trabajadores de los LPS de la UCI para detectar insuficiencias en la evaluación de la usabilidad de las aplicaciones web.

**Estructura del Documento**

Para lograr la claridad y comprensión de los contenidos de la investigación realizada se ha estructurado el documento en 3 capítulos.

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

En el **CAPÍTULO 2 ANÀLISIS Y DISEÑO DE LA PROPUESTA DE SOLUCIÒN,** se hace una descripción de la arquitectura del sistema, los patrones de diseño utilizados, los diagramas de clases de diseño, el modelo de datos, así como la descripción de las tablas presentes en el mismo.

En el **CAPÍTULO 3 IMPLEMENTACIÒN Y PRUEBA DE LA PROPUESTA DE SOLUCIÓN,** en este capítulo se valida el sistema desarrollado aplicándole las pruebas necesarias como las de funcionalidad mostrándose los resultados que esta arroja para demostrar el correcto funcionamiento de la solución propuesta.

# Capítulo 1 Fundamentación teórica

En el presente capítulo se presentan las principales definiciones de usabilidad, además se tratan las consideraciones sobre la evaluación de la usabilidad de las aplicaciones web, se describe cómo se realiza el procedimiento de evaluación de la usabilidad de software en la Universidad de las Ciencias Informáticas. Se justifica del uso de lógica difusa para la automatización de la evaluación de la usabilidad. Se mencionan y describen las tecnologías, metodología y herramientas seleccionadas para desarrollar la solución. Se analizan las soluciones existentes según criterios de medida relevantes en la evaluación de la usabilidad y finalmente se realizan conclusiones parciales del capítulo.

## 1.1 La evaluación de la usabilidad del producto de software

La usabilidad es la facilidad con que un usuario puede utilizar una herramienta fabricada por otras personas con el fin de alcanzar un cierto objetivo [RAE, 2013]. Según [Sánchez, 2011] el término usabilidad es un atributo cualitativo definido comúnmente como la facilidad de uso, ya sea de una página web, una aplicación informática o cualquier otro sistema que interactúe con un usuario. El concepto generalmente se refiere a una aplicación informática o un aparato, aunque también puede aplicar a cualquier sistema hecho con algún objetivo particular.

Los estudios definen que la evaluación de la usabilidad es una de las tareas más importantes que deben emprenderse cuando se desarrolla una interfaz de usuario, por lo que evaluar la usabilidad de un software constituye solo una parte de la ingeniería de la usabilidad [Torrente, 2013].

Los modelos de calidad del producto de software han surgido como un esfuerzo para descomponer la calidad en características más sencillas que puedan ser medidas. Desde el primer modelo de calidad del producto de software propuesto en [McCall, 1977] se define la usabilidad como factor que influye en la evaluación de la calidad.

Basado en el modelo de McCall surge el estándar ISO 9126 [ISO, 1992], sustituyendo los atributos antes definidos por características. En este están definidas seis características que todo producto de software debe poseer y que determinan la calidad.

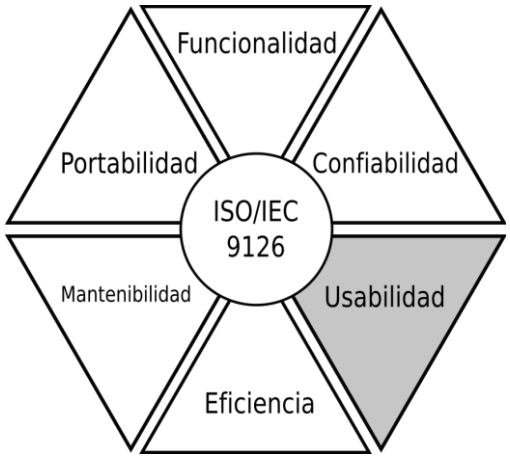



**Figura 1. Características definidas por las ISO 9126.**

**(Fuente: adaptado de [Obeso, 2005])**

Según [ISO/IEC 9126, 2005] la usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso. Para sustituir este concepto en [ISO/IEC 25000, 2016] se define la usabilidad como el grado en que el producto software satisface las necesidades expresadas o implícitas, cuando es usado bajo condiciones determinadas. En la presente investigación será asumida esta definición.

La serie de estándares ISO 25000 consta de cinco divisiones:

- División de Gestión de Calidad (ISO 2500n).
- División de Modelo de Calidad (ISO 2501n).
- División de Medición de la Calidad (ISO 2502n).
- División de Requisitos de Calidad (ISO 2503n).
- División de Evaluación de la Calidad (ISO 2504n)

En la ISO 25010 [ISO, 2011], correspondiente a la división del modelo de calidad se definen 8 características y 31 subcaracterísticas.

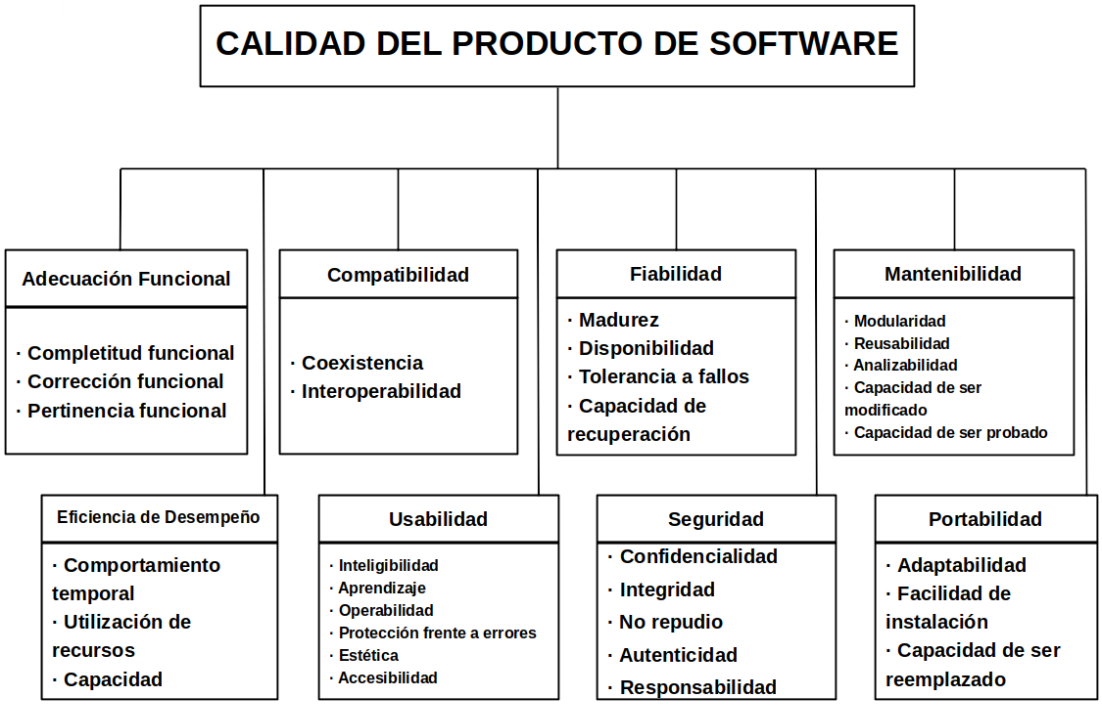



**Figura 2. Características y subcaracterísticas de la ISO 25010.**

**(Fuente: adaptado de [ISO, 2011])**

A los efectos de esta investigación se utilizará ISO 25000 como guía para la evaluación de la usabilidad del producto de software dada su actualidad, amplia aceptación y difusión, así como el excelente modelo de evaluación que ofrece. Actualmente ISO 25000 ha sustituido a la ISO 9126, que anteriormente surgió sobre la base del modelo de McCall y otros existentes.

Se puede definir la usabilidad de las aplicaciones web, según [Hassan, 2002] como la disciplina que estudia la forma de diseñar sitios web para que los usuarios puedan interactuar con ellos de la forma más fácil, cómoda e intuitiva posible. En la investigación mencionada se agrega además que la mejor forma de crear un sitio web usable es realizando un diseño centrado en el usuario, diseñando para y por el usuario, en contraposición a lo que podría ser un diseño centrado en la tecnología o uno centrado en la creatividad u originalidad.

La usabilidad es el atributo de calidad que mide lo fáciles que son de usar las interfaces web. Es decir, un sitio web usable es aquel en el que los usuarios pueden interactuar de la forma más fácil, cómoda, segura e inteligente posible [Nielsen, 2012].

## 1.2 La evaluación de la usabilidad del producto de software en la UCI

El software que se produce en la UCI puede llegar a ser muy complejo por sus funcionalidades y diversos campos de aplicación. Evaluar su usabilidad como producto no es tarea sencilla por la gran cantidad de elementos a tener en cuenta. En los Laboratorios de Pruebas de Software de la Dirección de Calidad, se establece una guía para las pruebas de usabilidad.

La institución ha certificado el nivel 2 de la Integración de Modelos de Madurez de Capacidades (CMMI). CMMI está orientado principalmente a garantizar la calidad del proceso de desarrollo de software y no tanto así a la evaluación de la calidad del producto de software. Regula el establecimiento de las actividades de aseguramiento de la calidad, aunque ofrece libertad en la decisión de las métricas y características de los modelos de la calidad del producto que se evaluarán.

En la UCI, la evaluación de la usabilidad se realiza de forma manual, limitando el proceso, elevando los costos del proyecto en cuanto a recursos y tiempo. La evaluación está estructurada en tres etapas: Planificación, Prueba y Conclusión. A continuación, en la Figura 3 se muestran cada una de las actividades realizadas para las etapas.

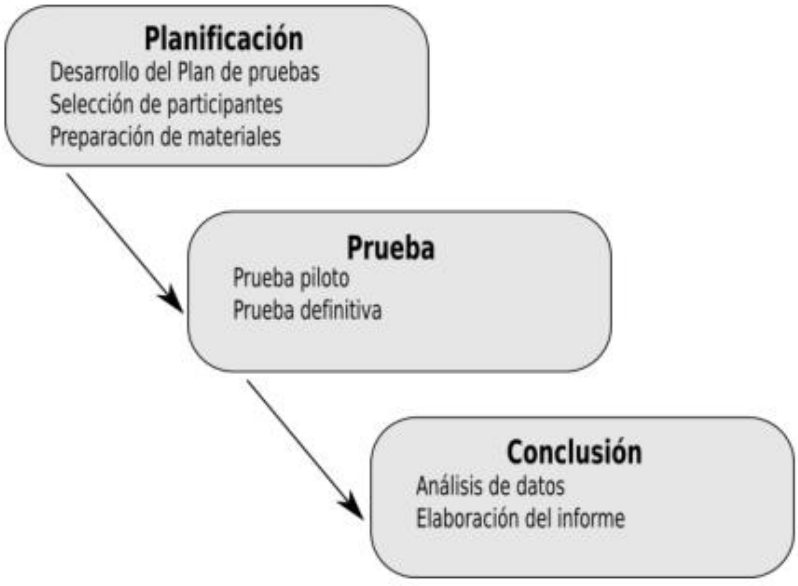



**Figura 3. Etapas de prueba de usabilidad en la UCI.**

**(Fuente: elaboración propia)**

En entrevista realizada a directivos y trabajadores de los LPS se identifican insuficiencias en las pruebas de usabilidad por falta de preparación del laboratorio (ver Anexo A). Estas limitaciones para la validación y verificación de la usabilidad están dadas por problemas tecnológicos, que entre otros aspectos están

relacionadas con la insuficiente automatización de mecanismos para la evaluación. En LPS solo se utiliza el método de evaluación heurística que se realiza sobre la técnica de la lista de comprobación o chequeo. Se reconoce por el personal entrevistado la insuficiencia del método de evaluación que se aplica en comparación con otros centrados en los usuarios. Además, se constató que no se ha establecido un modelo de evaluación de la calidad del producto de software.

Implementar una técnica de evaluación descriptiva basada en la opinión del usuario representaría la automatización de la evaluación centrada en el usuario de la usabilidad. Esta técnica permite obtener opiniones subjetivas del usuario. Con la automatización se evitaría un aumento exponencial de costo, esfuerzo y tiempo requerido para evaluar la usabilidad del producto. Por la riqueza de las técnicas basadas en la opinión del usuario sería recomendable utilizar entrevista, encuesta y/o cuestionario.

Las técnicas basadas en la opinión de usuario tienen como característica que generan muchos datos, ya sea de forma cuantitativa o cualitativa, dependiendo de la técnica utilizada. Los datos obtenidos son por naturaleza vagos e imprecisos a causa de las subjetividades implícitas en la evaluación. La opinión establecida por diferentes usuarios para un mismo software puede variar haciendo que los datos arrojados en la evaluación en ocasiones no sean del todo confiables y con frecuencia contradictorios. Dos usuarios pueden percibir de forma completamente diferente la usabilidad.

En la evaluación de la usabilidad del producto de software se aprecia la existencia de incertidumbre. El análisis de los resultados de la evaluación depende en gran medida del evaluador y de cómo interpreta los datos a su disposición. La lógica difusa es una técnica especialmente buena para minimizar la incertidumbre en un proceso. Un enfoque de este tipo puede servir para brindar solución al problema de la presente investigación.

## 1.3 Lógica difusa

La lógica difusa es una lógica multievaluada que permite representar matemáticamente la incertidumbre y la vaguedad, proporcionando herramientas formales para su tratamiento [González, 2017]. Permite establecer este mapeo de una forma adecuada, atendiendo a criterios de significado y no de precisión [Melin y Castillo, 2014].

En los conjuntos difusos un elemento x puede pertenecer simultáneamente a dos conjuntos, dependiendo de su grado de pertenencia, que se encontrará en el intervalo [0,1]. Mientras más cercano este el valor a 0 menos se puede asegurar la pertenencia de un elemento a un conjunto. Por el contrario, cuanto más

cercano este el valor a 1 mayor es la certeza de la pertenencia del elemento al conjunto [Melin y Castillo, 2014].

Las variables lingüísticas son conceptos que no se clasifican con valores de verdad absoluta, se clasifican mediante conjuntos difusos y son sustantivos que reflejan una propiedad de un elemento. A estas variables se les asignan etiquetas, que son las posibles clasificaciones de la variable. Las etiquetas se refieren a los posibles adjetivos con los que puedan ser clasificadas las variables lingüísticas. El intervalo en el que oscilarán los posibles valores de las variables lingüísticas será llamado universo discurso U [Zadeh, 1975].

El grado de pertenencia de un elemento μ(x) a un conjunto difuso será determinado por funciones de pertenencia. Estos conceptos se sustentan en la Definición 1.3.1 formalizada en [Atanassov, 2017].

**Definición 1.3.1** Se tiene un universo fijo U y su subconjunto A. Se denomina conjunto difuso al conjunto

$$A^* = \{ \langle x, \mu_A(x), \nu_A(x) \rangle \mid x \in U\}$$

donde

$$0 \leq \mu_A(x) + \nu_A(x) \leq 1$$

Las funciones $\mu_A$: $U \rightarrow [0,1]$ y $\nu_A$: $U \rightarrow [0,1]$ representan el grado de pertenencia y no pertenencia, respectivamente. También se puede definir la función $\pi_A$: $U \rightarrow [0,1]$ por

$$\pi_A(x) = 1 - \mu_A(x) - \nu_A(x)$$

que corresponde al grado de incertidumbre. Es obvio que para los sistemas de inferencia basados en lógica difusa π A (x) = 0 para cada x ∈ U, quedando que

$$A^* = \{\langle x, \mu_A(x), 1 - \mu_A(x)\rangle \mid x \in U\}$$

En la lógica difusa se usan reglas heurísticas, llamadas reglas de producción de la forma SI (antecedente), ENTONCES (consecuente), donde el antecedente y el consecuente son también conjuntos borrosos, ya sean puros o resultado de una operación [Maguiña, 2010].

A partir de esto se podrá inferir el factor de certeza del consecuente de la regla. Es importante acotar que en los casos en que no se tiene el factor de certeza para una regla se asume el máximo valor que este puede tomar, que es 1.

En un sistema de lógica difusa se tienen variables lingüísticas, sus etiquetas, las funciones de pertenencia de las etiquetas, las reglas de producción y los factores de certeza asociados a estas reglas. Como datos de entrada al sistema se tienen valores numéricos que toman las variables lingüísticas. Estos valores se convierten en valores de pertenencia a etiquetas difusas que son equivalentes a los factores de certeza. Este proceso se llama fusificación, dado que convierte valores numéricos a difusos.

A partir de estos valores obtenidos en el proceso de fusificación ocurre el proceso de propagación de certeza usando las reglas de producción definidas. Este es el proceso de Inferencia, obteniéndose como resultados valores de certeza que se refieren a las pertenencias a los conjuntos de salida. Por último, a partir de los valores de pertenencia a las variables lingüísticas de salida hay que obtener sus valores numéricos y a este proceso se le denomina desfusificación, el cual se le realiza a la variable procedimientos por ejemplo el método del centroide, el cual es el más utilizado.

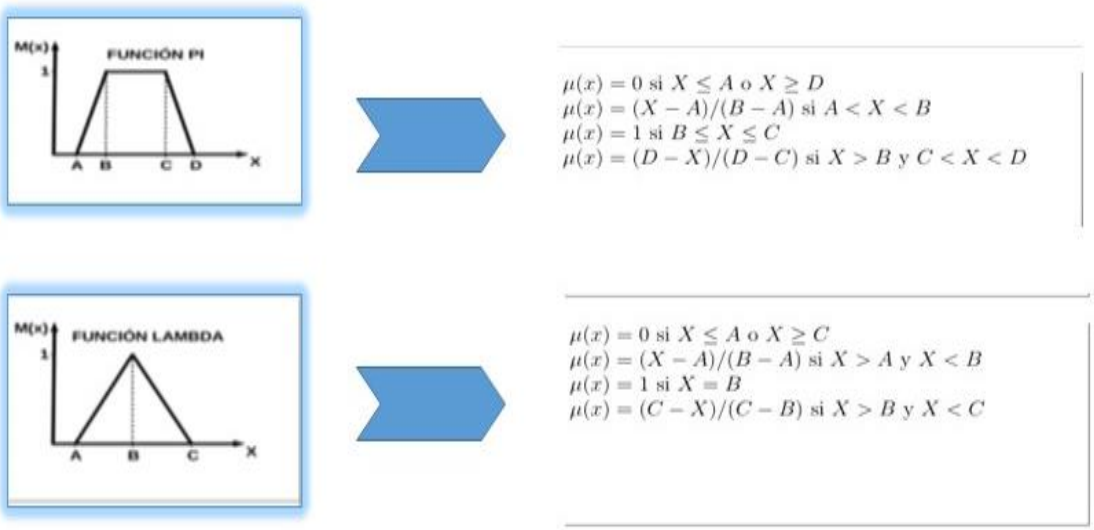



**Figura 4. Cálculo de las funciones de pertenencia.**

**(Fuente: elaboración propia)**

La lógica difusa ha sido empleada por numerosos autores para solucionar problemas relacionados con la evaluación de la usabilidad del software [Ahmad et al, 2003], [Chang y Dillon, 2006], [Hub y Zatloukal,

2008], [Hub y Zatloukal, 2009b], [Hub y Zatloukal, 2010], [Dubey et al, 2012], [Baquero et al, 2016], [McGlinn et al, 2017], [Zhou y Chan, 2017], [Ramanayaka et al, 2018].

## 1.4 Herramientas que evalúan la usabilidad de las aplicaciones web

Existen varias herramientas para realizar evaluaciones de usabilidad de aplicaciones web, las cuales se listan a continuación:

**Usabila:** es una de las herramientas más conocidas que ofrece mapas de calor y permite testear la usabilidad en un sitio web con hasta 25 participantes en línea que pueden navegar a través de un máximo de 5 páginas. Aunque existe posibilidad de testear más páginas con su versión de pago [Masherani, 2012].

**Crazy Egg:** esta herramienta proporciona información sobre el comportamiento del usuario en el sitio web como por ejemplo mapas de calor para identificar las zonas calientes de la página, obtener la proporción de clics por elemento o ver hasta dónde llega el usuario desplazándose en el sitio [Patel, 2005].

**Five Second Test:** herramienta que permite optimizar las páginas de destino y las llamadas de acción. Realiza test de cinco segundos entre los usuarios como pueden ser test de memoria, test para realizar alguna tarea o llegar a algún elemento en concreto [Rodriguez, 2019].

**UserPlus:** Esta aplicación identifica los elementos usables de una página ya que permite subir un pantallazo de la página que se desea testear para después, respondiendo a una serie de preguntas, obtener una puntuación, así como consejos e indicaciones de mejora [Rodriguez, 2009].

**Optimizely:** Es una herramienta para crear test A/B [1] de una manera sencilla ya que permite crear varias versiones de una manera rápida y después obtener resultados para poder analizarlos [Koomen, 2015].

**Navflow:** Herramienta para el análisis a través de *wireframes* [2]. Permite ver cómo los usuarios navegan por el sitio web a través de los *wireframes* que sube a esta aplicación y recibir retroalimentación de un grupo de probadores [Polo, 2010].

[1] Test A/B: para describir experimentos aleatorios con dos variantes.

[2] Wireframes: alambres

**Clicktale:** esta es una herramienta muy útil para saber cuánto tiempo pasan los usuarios en ciertas partes de la web. De esta manera, se puede saber las zonas donde el usuario está más interesado. Ofrece además otras métricas muy interesantes como mapas de desplazamiento del ratón, mapas de clics, e informes de visitas [Yavilevich, 2006].

**Click Heat:** es una aplicación que agrega al código y como resultado crea mapas de calor de acuerdo a los clicks que se generan en las visitas. Se pueden agregar filtros de revisión por fecha, navegador y página [Veugen, 2016].

En la Tabla 1 se puede observar la comparación realizada de forma resumida, teniendo en cuenta las subcaracterísticas de usabilidad asumidas en la presente investigación.

**Tabla 1. Comparación de las soluciones existentes.**

**(Fuente: elaboración propia)**

| Soluciones | Subcaracterísticas de usabilidad | | | | | | Propietario |
|---|---|---|---|---|---|---|---|
| | Inteligibilidad | Operabilidad | Aprendizaje | Accesibilidad | Protección | Estética | |
| **Usabila** | | x | | | | | x |
| **Crazy Egg** | | x | x | x | | | x |
| **Five Second Test** | | | x | | | | x |
| **UserPlus** | | | | x | | x | |
| **Optimizely** | | | x | | | | x |
| **Navflow** | | | x | x | | | |
| **Click Heat** | | | | x | | | x |

Luego del análisis y comparación de las soluciones existentes se concluye que ninguna evalúa todas las subcaracterísticas de usabilidad de algún modelo de calidad de marcada relevancia y vigencia. Se puede concluir que ninguna da solución al problema de la presente investigación por lo que se decide realizar una herramienta que permita realizar la evaluación centrada en el usuario de las aplicaciones web en la

UCI. Se hace necesario identificar las herramientas y tecnología a emplear en el desarrollo de una herramienta de este tipo.

## 1.5 Herramientas; Tecnologías y Metodología

### Lenguaje de programación

**Python (v 3.5)** es un lenguaje independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo. Tiene como propósito general la creación de todo tipo de programas.

No es un lenguaje creado específicamente para la web, aunque entre sus posibilidades sí se encuentra el desarrollo de páginas. Es multiplataforma y existen versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él. El lenguaje es interpretado lo cual quiere decir que no se debe compilar el código antes de su ejecución. En realidad, sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador.

En ciertos casos, cuando se ejecuta por primera vez un código, se produce código que se guarda en el sistema y que sirve para acelerar la compilación implícita que realiza el intérprete cada vez que se ejecuta el mismo código. Es interactivo y dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias.

Cada sentencia se ejecuta y produce un resultado visible, que puede ayudan a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente. Orientado a objetos y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables [Python,2001].

### Framework de desarrollo

Un *framework*[3] web es un conjunto de componentes que ayudan a desarrollar sitios web más fácil y rápidamente.

[3] Framework: marco de trabajo

**Django (v 2.1.5)** es un framework web de alto nivel que fomenta el desarrollo rápido y el diseño limpio y pragmático. Este entorno está escrito en Python, respeta el patrón de diseño Modelo-Vista-Controlador y es de código abierto, permite realizar aplicaciones web con menos líneas de código y completamente a la medida. el cual se ejecuta en muchas plataformas. Lo que significa que no está sujeto a ninguna plataforma en particular, y puede ejecutar sus aplicaciones en muchas distribuciones de Linux, Windows y Mac OS X. Además, Django cuenta con el respaldo de muchos proveedores de alojamiento web, y que a menudo proporcionan una infraestructura específica y documentación para el alojamiento de sitios de Django.

**Bootstrap (v3.3.6)**

Es un framework basado en HTML, CSS y JavaScript para crear webs *responsive* [4]

Sus principales características son:

- Open source: es un framework de código abierto y por tanto accesible para todos los usuarios sin que esto implique ningún coste o limitación.
- Grid: Su sistema de rejilla basado en un diseño de 12 columnas permite adaptar el contenido a los distintos tamaños de pantalla.
- HTML5 y CSS3: soporta las últimas versiones de HTML y CSS.
- Optimización: posee un código limpio y optimizado para que la web cargue lo más rápido posible.
- Uso: tiene una curva de aprendizaje moderada y cuenta con documentación suficiente para conseguir resultados desde el primer día.
- Componentes: posee elementos de fácil instalación como por ejemplo íconos, botones, menús desplegables, entre otros.
- Compatibilidad: es compatible con navegadores tan importantes como: *Google Chrome, Mozilla Firefox, Safari, Internet Explorer, Opera*.
- LESS: funciona a la perfección con *LESS*, un preprocesador que permite escribir CSS de forma mucho más rápida y con muchas más opciones.
- Garantías: ha sido creado por Twitter, una empresa de renombre mundial que asegura un correcto funcionamiento, así como continuas actualizaciones.

[4] Responsive: receptivo

**Entorno integrado de desarrollo**

**PyCharm (v 2.4)** es un IDE o entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, integración con VCS / DVCS y soporte para el desarrollo web con Django, entre otras bondades. PyCharm es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la Community que es gratuita y orientada a la educación y al desarrollo puro en Python y la Professional, que incluye más características como el soporte a desarrollo web.

**Gestor de Base de datos**

**PostgreSQL (v 9.4)** es una base de datos relacional de código abierto. Distribuida bajo licencia *BSD (del inglés, Berkeley Software Distribution),* lleva más de 15 años desarrollándose y su arquitectura goza de una excelente reputación por su fiabilidad e integridad de datos, dispone de versiones para prácticamente todos los sistemas operativos. Tiene soporte para claves extranjeras, *joins*, vistas, disparadores y procedimientos almacenados (en múltiples lenguajes de programación). Incluye la mayoría de los tipos de datos de SQL92 y SQL99 y, asimismo, soporta el almacenamiento de grandes objetos binarios, como imágenes, sonidos y vídeos. Tiene interfaces de programación nativas para C/C++, Java, .Net, Perl, PHP, Python, Ruby, Tcl y ODBC, entre otros, y una excepcional documentación [Martínez, 2012].

**Herramienta CASE**

Para el modelado con UML (Lenguaje unificado de modelado) se utilizará Visual Paradigm for UML 8.0 por ser una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite modelar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación. La herramienta agiliza la construcción de aplicaciones con calidad y a un menor coste. Posibilita la generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos, así como obtener ingeniería inversa de bases de datos [Rondón et al., 2011].

**Metodología AUP-UCI**

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello se usa en el Modelo CMMI-DEV v1.3. El cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad [Rodríguez, 2015].

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea

configurable. Se decide hacer una variación de la metodología AUP (Proceso Unificado Ágil), de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI [Rodríguez, 2015].

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega una fase de Cierre. Consta de 4 escenarios:

- Escenario No 1: Proyectos que modelen el negocio con CUN [5] solo pueden modelar el sistema con CUS [6].
- Escenario No 2: Proyectos que modelen el negocio con MC [7] solo pueden modelar el sistema con CUS.
- Escenario No 3: Proyectos que modelen el negocio con DPN [8] solo pueden modelar el sistema con DRP [9].
- Escenario No 4: Proyectos que no modelen negocio solo pueden modelar el sistema con HU [10].

Con la adaptación de AUP a AUP-UCI se propone tener un mayor desarrollo en la actividad de los proyectos. Específicamente para esta investigación se propone utilizar la metodología en su segundo escenario el cual se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio.

## 1.6 Conclusiones parciales

A partir del análisis realizado en el presente capítulo es posible llegar a las siguientes conclusiones parciales:

[5] CUN: Caso de uso del negocio

[6] CUS: Caso de uso del sistema

[7] MC: Modelo conceptual

[8] DPN: Diagrama del proceso del negocio

[9] DRP: Diagrama de relaciones de procesos

[10] HU: Historia de usuario

1. La mejor forma de realizar la evaluación de la usabilidad del producto de software es centrada en el usuario, aunque existen otras variantes ninguna sustituye a esta.
2. La lógica difusa ha demostrado ser efectiva en la resolución de problemas relacionados con la evaluación de la usabilidad del producto de software.
3. Las herramientas para la evaluación de la usabilidad de las aplicaciones web no resuelven el problema de la presente investigación. El análisis de estas soluciones demuestra la necesidad de implementación de una herramienta para la evaluación centrada en el usuario de la usabilidad del producto de software desarrollado en la UCI.
4. La metodología AUP–UCI utilizando su segundo escenario es la más adecuada para guiar el proceso de desarrollo de la herramienta y dar solución al problema de la presente investigación.

# Capítulo 2: Análisis y diseño de la propuesta de solución.

En este capítulo se realiza la descripción de las principales definiciones asociadas al dominio del problema. Se especifican los requisitos funcionales y no funcionales a tener en cuenta en la implementación de la solución. Se agrupan los requisitos funcionales en casos de uso y se describe textualmente cada uno de ellos. Finalmente se definen los patrones a seguir en el diseño de la herramienta.

## 2.1 Modelo de dominio

Para lograr una mayor comprensión del contexto en que se ubica la herramienta y la posterior representación de la solución, se recurre al empleo del modelo de dominio. Es un subconjunto del modelo de negocio y se realiza cuando no están claros los procesos o no se identifican claramente los actores y trabajadores del negocio. Además, el modelo de dominio captura los tipos más importantes de objetos que existen, se identifican conceptos, se definen y se relacionan en un diagrama de clases UML [Baquero y Mendoza, 2016].

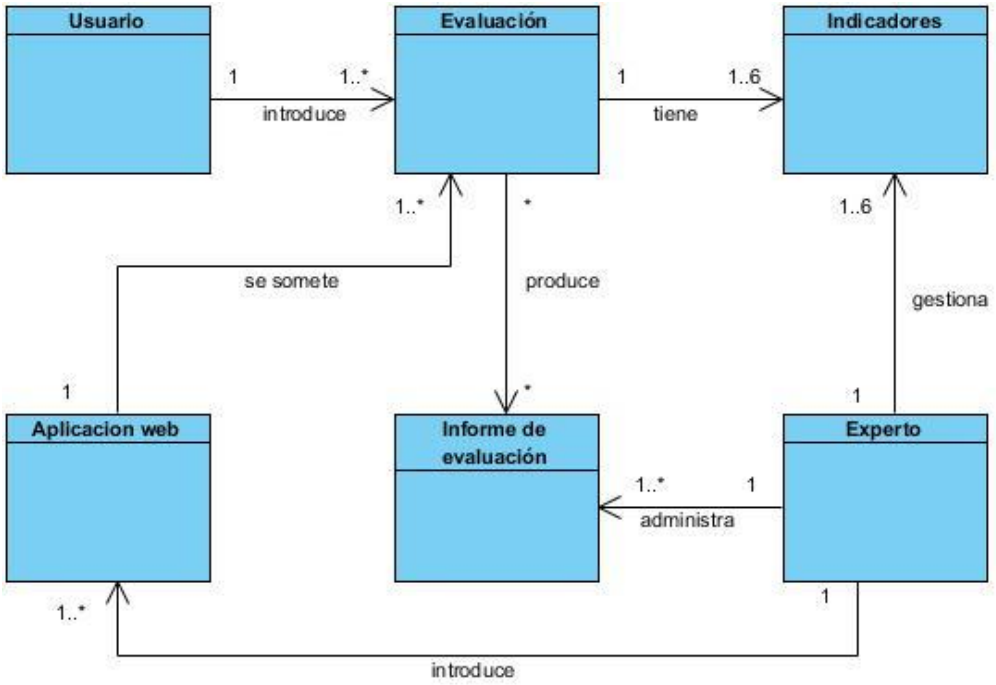



**Figura 5. Diagrama de clases del dominio.**

**(Fuente: elaboración propia)**

Tabla 2. Definición de los conceptos del modelo de dominio.

(Fuente: elaboración propia)

| Concepto | Definiciones |
|---|---|
| Usuario | Persona que utiliza la herramienta para evaluar la usabilidad de las aplicaciones |
| Aplicación web | Aplicación web que el usuario va a evaluar. |
| Evaluación | Es el proceso automatizado que se lleva a cabo para evaluar la usabilidad de la aplicación. |
| Indicadores | Indicadores por los cuales se rige el usuario para evaluar la aplicación. |
| Informe | Contiene los resultados de la evaluación |
| Experto | Es el que introduce la aplicación web y gestiona los indicadores. |

## 2.2 Propuesta de solución

De forma general la propuesta de solución funciona de forma tal que el usuario realiza una evaluación de la usabilidad de una aplicación web a través de un cuestionario. Los resultados de la evaluación son interpretados con un sistema de lógica difusa y finalmente se obtiene la evaluación de la usabilidad en un informe. La Figura 6 muestra con mayor detalle lo anteriormente explicado.

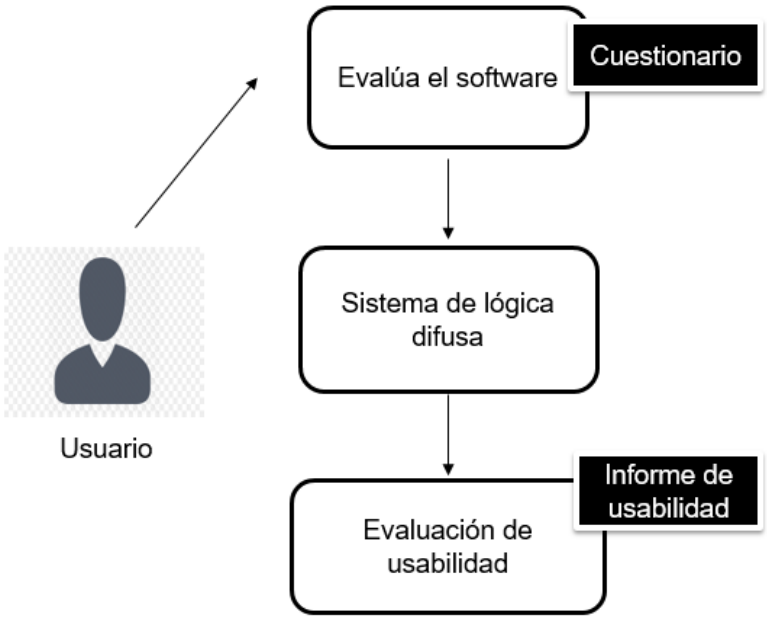



Figura 6. Propuesta de solución.

(Fuente: elaboración propia)

El sistema de lógica difusa que se usará es una variación de [Baquero et al, 2016]. En este sistema se evalúa la usabilidad del producto de software. Se tienen las variables lingüísticas de entrada y la variable de salida como se puede ver en la Figura 7.

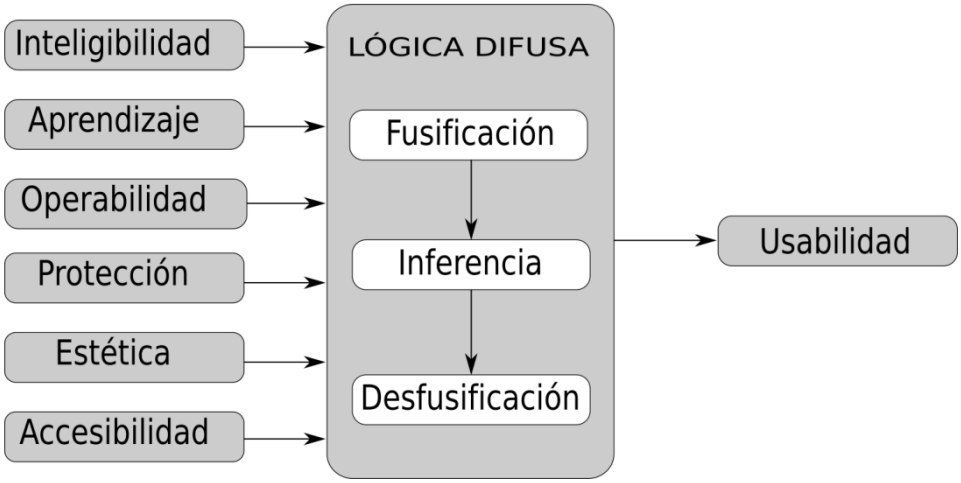



**Figura 7. Sistema de lógica difusa para evaluar la usabilidad del producto de software.**

**(Fuente: elaboración propia)**

El usuario evalúa cada una de estas variables de entrada respondiendo un cuestionario, según puede observarse en la Tabla 4. Este cuestionario es respondido por el usuario expresando con palabras su criterio en cada una de las preguntas.

**Tabla 4. Cuestionario para evaluar la usabilidad del producto de software.**

**(Fuente: elaboración propia)**

| No. | Pregunta | Evaluación |
|---|---|---|
| 1 | Exprese la medida en que el software satisface sus necesidades | |
| 2 | Diga en qué grado considera que es fácil operar y controlar el software | |
| 3 | Especifique la complejidad para operar y controlar el software | |
| 4 | Evalúe en qué medida el software le facilita ayuda y guía cuando comete un error | |
| 5 | Determine su criterio sobre la interfaz del software | |
| 6 | Cualifique la capacidad del software de adaptarse a sus particularidades y necesidades específicas | |

Los resultados del cuestionario son traducidos con un convertidor lingüístico que es capaz de traducir las evaluaciones entradas por el usuario en valores de las variables de entrada, además de detectar la intencionalidad con que ha sido evaluada la variable. Estos valores son fusificados, lo cual permite inferir la usabilidad a partir de la evaluación realizada por el usuario. La inferencia se realiza usando el método de Mandani y finalmente se desfusifican utilizando el método del centroide o centro de gravedad.

## 2.3 Especificación de requisitos

Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de este. En el otro extremo, es una definición detallada y formal de una función del sistema [Sommerville, 2005].

**Requisitos Funcionales**

La herramienta de evaluación centrada en el usuario de la usabilidad de aplicaciones web se basará fundamentalmente en la generación de informes con las evaluaciones de la usabilidad obtenida de los usuarios. A continuación, se listan los requisitos funcionales del sistema.

**RF 1** Insertar usuario al sistema

**RF 2** Listar los usuarios registrados en el sistema.

**RF 3** Eliminar los usuarios registrados en el sistema.

**RF4** Modificar los usuarios registrados en el sistema

**RF 5** Insertar los indicadores por los cuales se va a evaluar la aplicación web.

**RF 6** Eliminar los indicadores.

**RF 7** Modificar los indicadores.

**RF 8** Listar los indicadores.

**RF 9** Administrar evaluación

**RF 10** Generar los informes registrados en el sistema.

**RF 11** Permitir exportar un informe en formato PDF.

**RF 12** Permitir la impresión de los informes registrados en el sistema.

**RF 13** Permitir buscar los informes registrados en el sistema.

**RF 14** Administrar videos de las acciones del usuario en la aplicación web.

**Requisitos No Funcionales**

Los requisitos no funcionales representan características generales y restricciones de la aplicación o sistema que se esté desarrollando. Especifica los criterios que se deben usar para juzgar el funcionamiento de un sistema, en lugar de un comportamiento específico. En general, los requisitos funcionales definen lo que el sistema debería de hacer, mientras que los requisitos no funcionales verifican ¿cómo un sistema debería de ser? Son a menudo llamados las “cualidades de un sistema”.

**Usabilidad:** Debe ser operado por usuarios sin grandes conocimientos informáticos, por lo que todas sus funcionalidades deben ser accesibles de manera intuitiva.

**Seguridad:** Mostrar a cada usuario solo las funcionalidades del sistema sobre las cuales tiene permiso de acceso. La información será protegida contra el acceso de los usuarios no autorizados a través de la gestión de usuarios.

**Interfaz:** La gama de colores que se va a utilizar serán blanco, azul y gris. Todos los textos y mensajes en pantalla aparecerán en idioma español.

**Software:** En las PC´s de los clientes debe estar instalado un navegador web como el Chrome, Firefox, etc. En la PC del servidor debe estar instalado un sistema de gestor de base de datos Postgres

## 2.4 Diagrama de casos de uso del sistema

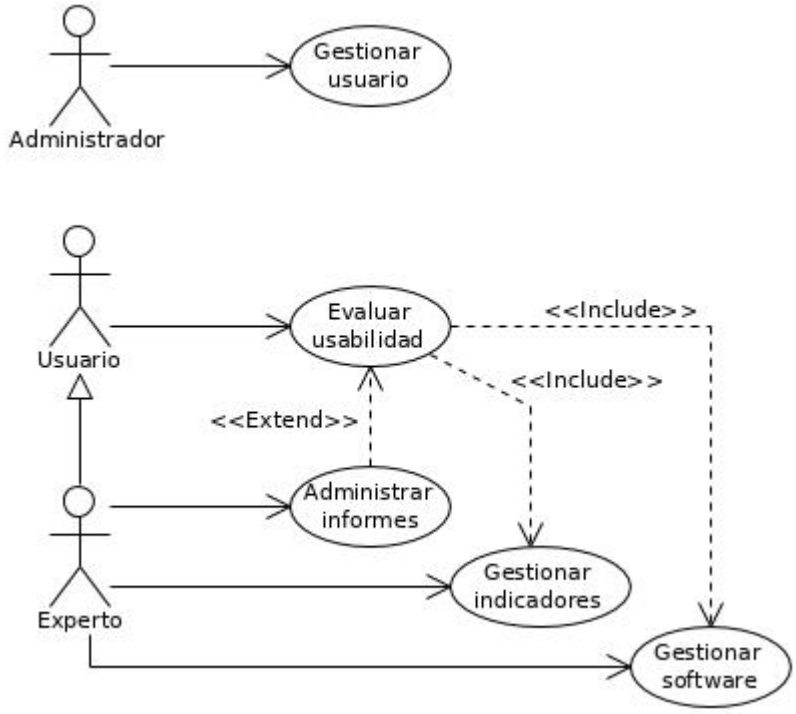



**Figura 1. Diagrama de caso de uso del sistema.**

**(Fuente: elaboración propia)**

A continuación, se realiza una descripción de los casos de uso Gestionar Usuario y Evaluación de aplicación web.

**Tabla 5. Caso de uso Gestionar usuario.**

**(Fuente: elaboración propia)**

| **Caso de Uso Gestionar usuarios.** | |
|---|---|
| Caso de uso | Gestionar usuario |
| Actores | Usuario |
| Resumen | El caso de uso inicia cuando el usuario accede al menú, donde puede insertar nuevos usuarios, modificar y eliminar los existentes. Finaliza cuando se han llevado a cabo las operaciones solicitadas. |
| Referencias | RF1, RF2, RF3, RF4 |

| | |
|---|---|
| Precondiciones | El gestor se ha autenticado y tiene los permisos necesarios para realizar la Gestión de Usuarios. |
| Prioridad | Alta |
| | |
| **Flujo Básico de Eventos** | |
| **“Gestionar Usuarios”** | |
| Acción del Actor Respuesta del Sistema<br>1. El caso de uso comienza cuando el usuario accede al menú Usuarios. | 2. Se muestra un listado con los usuarios existentes y la información de ellos. Además, se muestran botones para insertar nuevos, modificarlo o eliminarlos.<br>4. El sistema realiza la operación según la opción que escoja el usuario: |
| | |
| 3. El usuario selecciona una de las opciones que brinda el sistema. | |

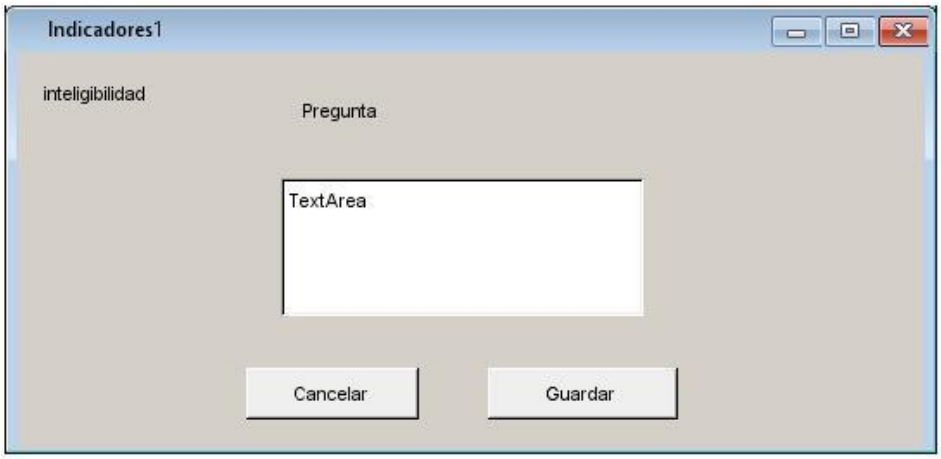
**Prototipo de interfaz**

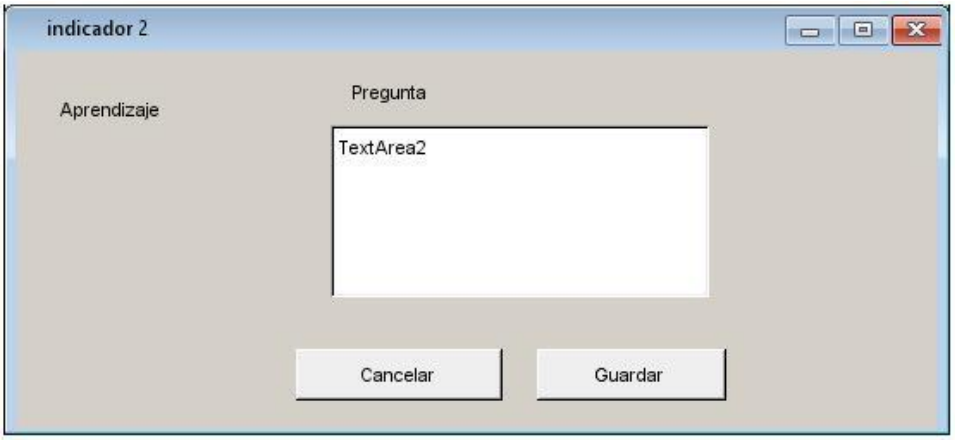
**Tabla 6. Caso de uso Evaluación de aplicación web.**

**(Fuente: elaboración propia)**

| **Caso de Evaluación de aplicación web** | |
|---|---|
| Caso de uso | Evaluación de aplicación web |
| Actores | Usuario |
| Resumen | El caso de uso inicia cuando el usuario accede al menú indicadores, done va a encontrarse con varios formularios a rellenar. Finaliza al guardar los datos insertados o puede cancelar la operación |
| Referencias | RF5 |
| Precondiciones | El usuario debe estar autenticado y debe tener una aplicación a evaluar. |
| Prioridad | Alta |
| | |
| **Flujo Básico de Eventos** | |
| **"Gestionar Usuarios"** | |
| **Acción del Actor** | **Respuesta del Sistema**<br>1.El sistema muestra un formulario para entrar los datos correspondientes.<br>3.El sistema guarda los datos vacía los campos del formulario. |
| **2.El usuario rellena los datos requeridos y presiona el botón Guardar** | |
| **Flujo Alterno** | |
| El usuario presiona el botón cancela | El sistema no guarda los cambios |

**Prototipo de interfaz**




## 2.5 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares; ayudan al éxito del proyecto, pues permiten la reutilización de código, garantizan la robustez y extensibilidad del software [Gamma et al, 2009].

### Patrones GRASP

Lo esencial de un diseño de objetos lo constituye el diseño de las interacciones de objetos y la asignación de responsabilidades. Las decisiones que se tomen pueden influir profundamente en la extensibilidad, claridad y mantenimiento del sistema de software de objetos, además en el grado y calidad de los

componentes reutilizables, por esta razón, durante el diseño se deben realizar los casos de usos con objetos basado en los patrones GRASP [Giraldo et al, 2011].
Los patrones GRASP codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Los patrones utilizados en el desarrollo de la investigación son:

**Patrón controlador:** Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. La clase controladora es admin.py, la cual controla todo el sistema.

**Patrón creador:** El patrón creador guía la asignación de responsabilidades relacionadas con la creación de objetos. Se emplea al momento de definir la creación de objetos, se garantiza que si una clase puede crear objetos de otra es porque guardan determinada relación, ya sea que una contiene a la otra, o que una contiene los datos que necesita la segunda.

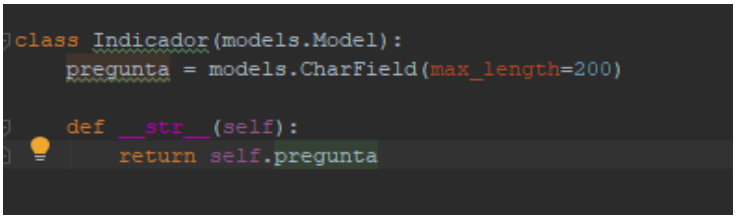



**Figura 2. Patrón Creador.**

**(Fuente: elaboración propia)**

**Patrón alta cohesión:** Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable, lo que facilita la organización del proyecto y que estas no estén sobrecargadas de funcionalidades ajenas.

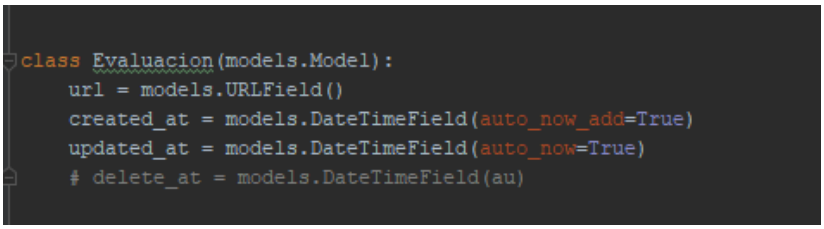



**Figura 3. Patrón Alta Cohesión.**

**(Fuente: elaboración propia)**

### Bajo acoplamiento

Este patrón responde a la pregunta: ¿cómo soportar baja dependencia e incrementar la reutilización? Propone tener las clases lo menos ligadas entre sí, de tal forma que, en caso de producirse una modificación en alguna de ellas, tenga la mínima repercusión en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre ellas.

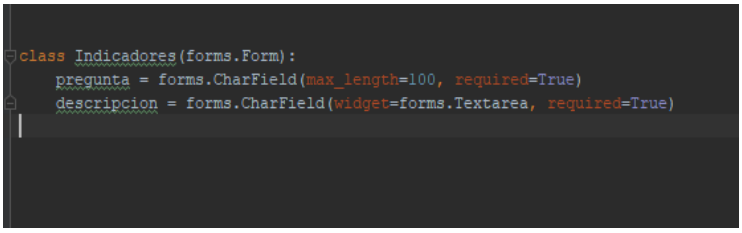



**Figura 4. Patrón Bajo Acoplamiento.**

**(Fuente: elaboración propia)**

**Patrón experto:** La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase “sencillas” y más cohesivas que son más fáciles de comprender y de mantener.

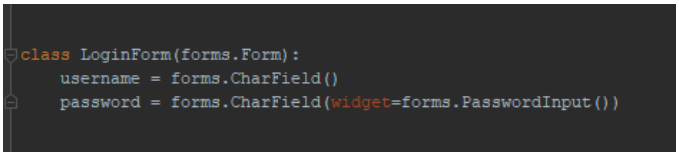



**Figura 5. Patrón experto.**

**(Fuente: elaboración propia)**

### Patrones GOF

Los patrones GOF se revelan como una forma indispensable de enfrentarse a la programación [Pressman, 2010]. Se clasifican en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento:

- **Creacionales:** conciernen al proceso de creación de objetos. Forma de crear instancias. Abstraer y ocultar la forma como son creados e inicializados los objetos.
- **Estructurales:** tratan la composición de clases u objetos. ¿Cómo se combinan clases y objetos para formar nuevas estructuras y proporcionar nuevas funciones?
- **Comportamiento:** caracterizan las formas en las que interactúan y reparten responsabilidades las distintas clases u objetos. Reducir acoplamiento.

Entre los patrones utilizados para el desarrollo de la extensión se encuentran el *Iterator y el Singleton* agrupados como patrones de comportamiento y de creación respectivamente según la clasificación de GOF. Estos se encuentran representados a través de una descripción detallada acerca del propósito que recoge cada uno y la aplicación que tienen durante el desarrollo de la extensión como se muestra a continuación.

**Patrón Singleton o Solitario:** Este patrón es de tipo creacional a nivel de objetos. Su principal objetivo es garantizar que una clase solo tenga una única instancia, para proporcionar un punto de acceso global a la misma. Permite realizar refinamientos en las operaciones y en la representación, mediante la especialización por herencia de “Solitario”. Es fácilmente modificable para permitir más de una instancia y para controlar el número de las mismas [Gamma, 1995].


**Figura 6. Patrón Singleton.**

**(Fuente: elaboración propia)**

**Decorator (Envoltorio):** Añade funcionalidad a una clase, dinámicamente. En cada archivo se define el código HTML.


**Figura 7. Patrón Decorator.**

**(Fuente: elaboración propia)**

## 2.6 Arquitectura de software

El patrón modelo-vista-plantilla(MTV) surge con el objetivo de reducir el esfuerzo de programación, necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, a partir de estandarizar el diseño de las aplicaciones. Es un paradigma que divide las partes que conforman una aplicación en el modelo, las vistas y los controladores, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo. A partir del uso de *frameworks* basados en el patrón MVC se puede lograr una mejor organización del. Según trabajo y mayor especialización de los desarrolladores y diseñadores [Sellarés 2011] MTV está compuesto por:

**El modelo:** contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.

**La vista:** o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos de interacción con este.

**La Plantilla:** actúa como intermediario entre el modelo y la vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

Este modelo de arquitectura presenta varias ventajas [Siegel, 2005]:

- Separación clara entre los componentes de un programa; lo cual permite su implementación por separado.
- Interfaz de programación de aplicaciones también definida como API *(Aplication programming interface)* la cual está muy bien definida. Cualquiera que use el API, podrá reemplazar el modelo, la vista o el controlador, sin aparente dificultad.

- Conexión entre el modelo y sus vistas dinámicas; se produce en tiempo de ejecución, no en tiempo de compilación.

Para el desarrollo de la propuesta de solución se utiliza el patrón arquitectónico MTV debido a que este separa la lógica de negocio de la interfaz de usuario en tres capas diferentes, cada una con funcionalidades bien definidas, reduciendo esfuerzo en la implementación de la aplicación y garantizando una mejor organización del trabajo.

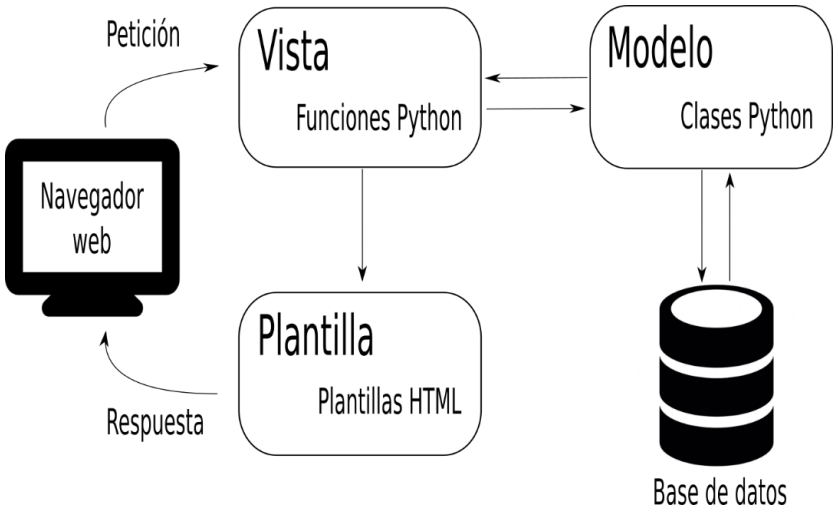



**Figura 8. Modelo Vista Plantilla.**

**(Fuente: adaptado de [Correia, 2018])**

## 2.7 Conclusiones parciales

Al finalizar el capítulo se puede concluir que:

1. La definición de los requisitos funcionales posibilitó obtener el diseño de una herramienta capaz de realizar la evaluación centrada en el usuario de la usabilidad de las aplicaciones web desarrolladas en la UCI.
2. La utilización de los patrones de diseño permitió que la aplicación sea escalable y se pueda realizar fácilmente un proceso de documentación de la implementación.
3. La utilización del patrón arquitectónico MTV, hizo posible obtener como resultado un diseño con alta tolerancia al cambio.

# Capítulo 3 Implementación y Prueba de la propuesta de solución

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

## 3.1 Modelo de implementación

La implementación constituye el núcleo fundamental durante la fase de construcción. Se inicia a partir del resultado obtenido durante el flujo de trabajo de análisis y diseño. El objetivo fundamental durante esta etapa es desarrollar la arquitectura y lograr estructurar el sistema final. Durante esta fase se organiza el código, se implementan los elementos del diseño, y se integran todos estos elementos para obtener un resultado satisfactorio que responda a lo definido durante las fases anteriores. Al llevar a cabo el modelo de implementación se establece la estructura de los elementos de implementación, basándose en las responsabilidades asignadas a los subsistemas de implementación y su contenido.

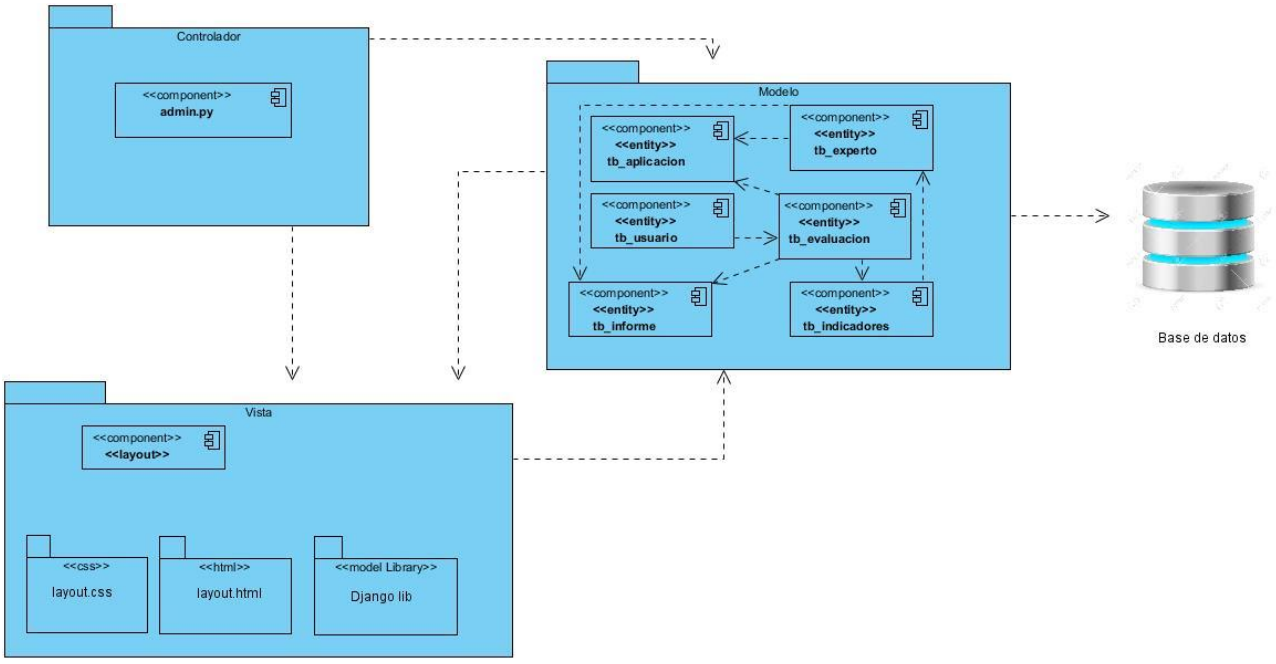



**Figura 9. Diagrama de Componente.**

**(Fuente: elaboración propia)**

## 3.2 Modelo de despliegue

El modelo de despliegue captura los elementos de configuración del procesamiento y las conexiones entre estos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos, así como la disposición de los dispositivos que carecen de capacidad de pre-procesado.


**Figura 10. Diagrama de despliegue.**

**(Fuente: elaboración propia)**

## 3.3 Estándares de codificación.

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a la estructura y apariencia física del código. Las normas de codificación definidas por las líneas de arquitectura están enfocadas a lograr una mejor comprensión y mantenimiento del código que responda a la complejidad dada por la cantidad de componentes y el alto nivel de integración que puede existir en el desarrollo del software (Quevedo et al., 2010).

Las realizaciones de revisiones frecuentes al código, la aplicación de forma continua de técnicas y estándares de codificación definidos, junto al empleo de buenas prácticas de programación, garantizan una alta calidad en el desarrollo de la extensión, además de proporcionar un código legible y reutilizable. (Baquero, 2016).

Entre los estándares de codificación se utilizaron:

- **Tamaño y organización de las líneas de código**

La longitud de línea es aproximadamente de 80 caracteres. En caso de que una expresión ocupe más de este rango, podrá romper o dividirse en una nueva línea y deberán estar todas alineadas con respecto a la anterior, para mantener la legibilidad del código.

- **Declaraciones**

Toda variable local tendrá que ser inicializada en el momento de su declaración, salvo que su valor inicial dependa de una llamada a otro método en determinado momento de ejecución de método. Las

declaraciones deben situarse al principio de cada bloque principal en el que se utilicen y nunca en el momento de su uso.

- **Llaves de apertura y cierre**

Se utiliza siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales. Esto aumenta la legibilidad y disminuye la probabilidad de errores lógicos.

**Ejemplo de código donde se encuentran estos estándares:**

```
def login_page(request):
    message = None
    if request.method == "POST":
        form = LoginForm(request.POST)
        if form.is_valid():
            username = request.POST['username']
            password = request.POST['password']
            user = authenticate(username=username, password=password)
            if user is not None:
                if user.is_active:
                    login(request, user)
                    message = "Te has identificado de modo correcto"
                else:
                    message = "Tu usuario esta inactivo"
            else:
                message = "Nombre de usuario y/o password incorrecto"
    else:
        form = LoginForm()
    return render_to_response('login.html', {'message': message, 'form': form},
                              context_instance=RequestContext(request))
```

**Figura 11. Estándares de codificación.**

**(Fuente: elaboración propia)**

## 3.4 Pruebas de software

La etapa de pruebas es una de las fases del ciclo de vida de los proyectos. Dichas pruebas no tienen el objeto de prevenir errores sino de detectarlos. Se efectúan sobre el trabajo realizado y se deben encarar con la intención de descubrir la mayor cantidad de errores posible, logrando así, desarrollar un software sin fallos y que satisfaga totalmente las necesidades de los clientes [Chiu, 2015].

### Estrategias de prueba

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requerirán. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la

prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados [Suarez y Fontela ,2003].

Existen cuatro niveles de prueba:

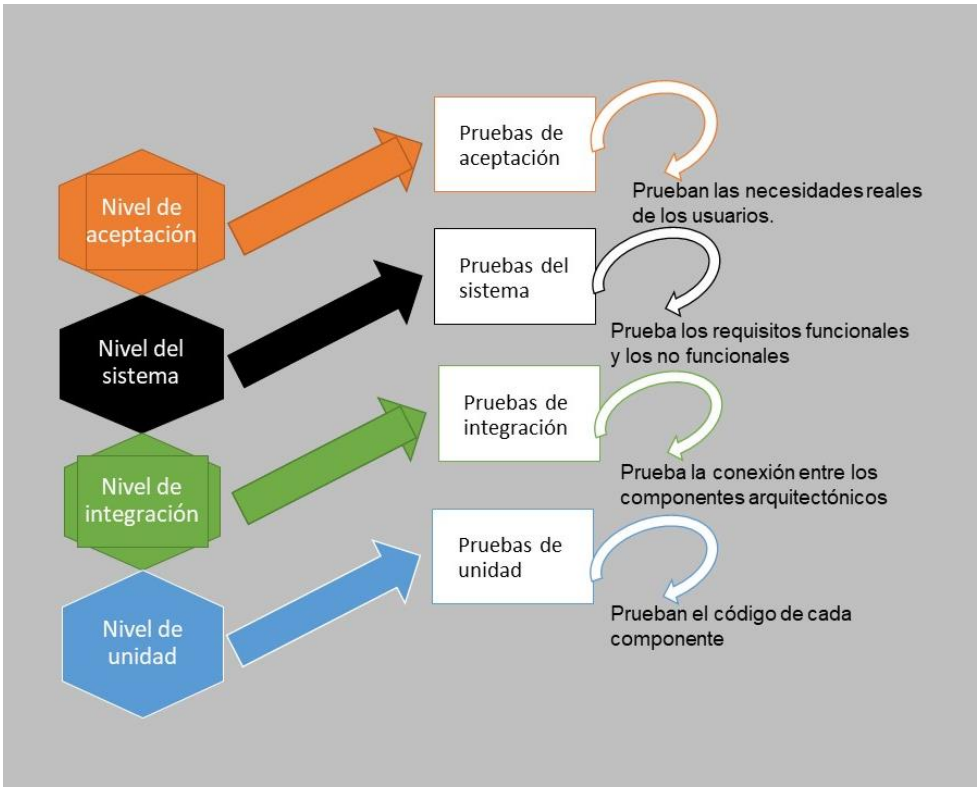



**Figura 12. Niveles de Pruebas.**

**(Fuente: elaboración propia)**

Las pruebas al sistema desarrollado serán realizadas en dos de los cuatro niveles anteriormente definidos, en el de pruebas de unidad y en el de pruebas del sistema. Para el nivel de pruebas unitarias se aplicará el método de caja blanca y para las del sistema el de caja negra.

**Pruebas de sistema**

Existen varios métodos de prueba, en este caso se escoge el de Caja Negra, el cual se centra principalmente en los requisitos funcionales del software. Permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Los casos de prueba del método de Caja Negra pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene [Pressman, 2010].

**Caso de uso Administrar Informes**

**Descripción General:** El caso de uso inicia cuando el experto accede a la función de Administrar reportes, donde puede listarlos, modificarlos y eliminarlos.

**Condiciones de Ejecución:** El actor se ha autenticado y tiene los permisos necesarios para realizar la Administración de Informes.

**Tabla 7. Secciones a probar en el caso de prueba Modificar Informe.**

**(Fuente: elaboración propia)**

| Nombre del requisito | Descripción general | Escenario de pruebas | Flujo del escenario |
|---|---|---|---|
| Modificar informe de evaluación. | Permite modificar un informe de evaluación de usabilidad con los campos requeridos. | EP 1: Modificar informe de forma correcta. | 1. Se muestra la interfaz Modificar informe con los campos requeridos.<br>2. Se introducen los siguientes datos: Nombre, descripción, nombre del solicitante, fecha de inicio, fecha de fin, ,<br>3. Se selecciona la opción Guardar.<br>4. El sistema muestra un mensaje de notificación "La solicitud ha sido creada satisfactoriamente". |
| | | EP 1: Modificar informe de forma incorrecta. | 1. Se muestra la interfaz modificar informe con los campos requeridos.<br>2. No se introduce todos los datos requeridos, dejando campos obligatorios vacíos.<br>3. Se señalan los campos en |

| | | | |
|---|---|---|---|
| | | | rojo y se muestra el mensaje: "Ha ocurrido un error.". |
| | | EP 1: Cancelar la operación. | 1. Se muestra la interfaz modificar informe con los campos requeridos.<br>2. Se introducen los siguientes datos: Nombre, descripción, nombre del solicitante, fecha de inicio, fecha de fin.<br>3. Se selecciona la opción Cancelar.<br>4. Se anulan todas las acciones efectuadas y se cierra la interfaz. |

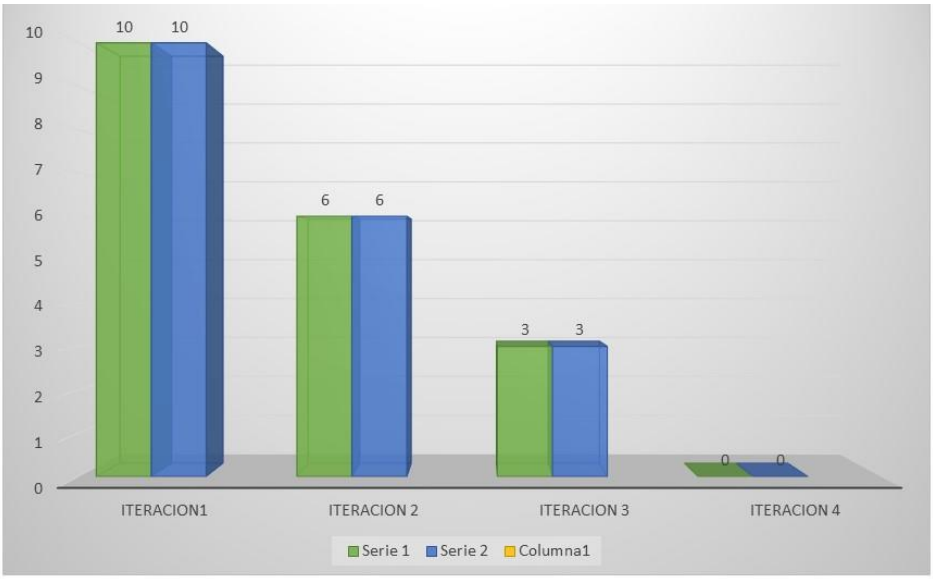



**Figura 13. Resultados de las pruebas por iteración.**

**(Fuente: elaboración propia)**

A continuación, se muestran las no conformidades detectadas en cada una de las iteraciones de las pruebas realizadas.

- Iteración 1: Fueron detectadas 10 NC y todas fueron corregidas.
- Iteración 2: Fueron detectadas 6 nuevas NC y todas fueron corregidas.
- Iteración 3: Fueron detectadas 3 nuevas NC y todas fueron corregidas.
- Iteración 4: No fueron detectadas nuevas NC.

### Pruebas Unitarias

Se concentran en ejecutar cada módulo individualmente para asegurar que funcione de manera apropiada como unidad, lo que provee un mejor modo de manejar la integración de las unidades en componentes mayores. Toma como guía la descripción del diseño a nivel de componente, y se prueban importantes caminos de control para describir errores dentro de los límites del módulo. Generalmente son realizadas por los propios programadores puesto que al conocer con mayor detalle el código, se les simplifica la tarea de elaborar conjuntos de datos de prueba para testearlo [Pressman, 2010].

Los métodos de cobertura de caja blanca tratan de recorrer todos los caminos posibles por lo menos una vez, lo que no garantiza que no haya errores, pero pretende encontrar la mayor parte.

Para una primera iteración se arrojaron 2 no conformidades

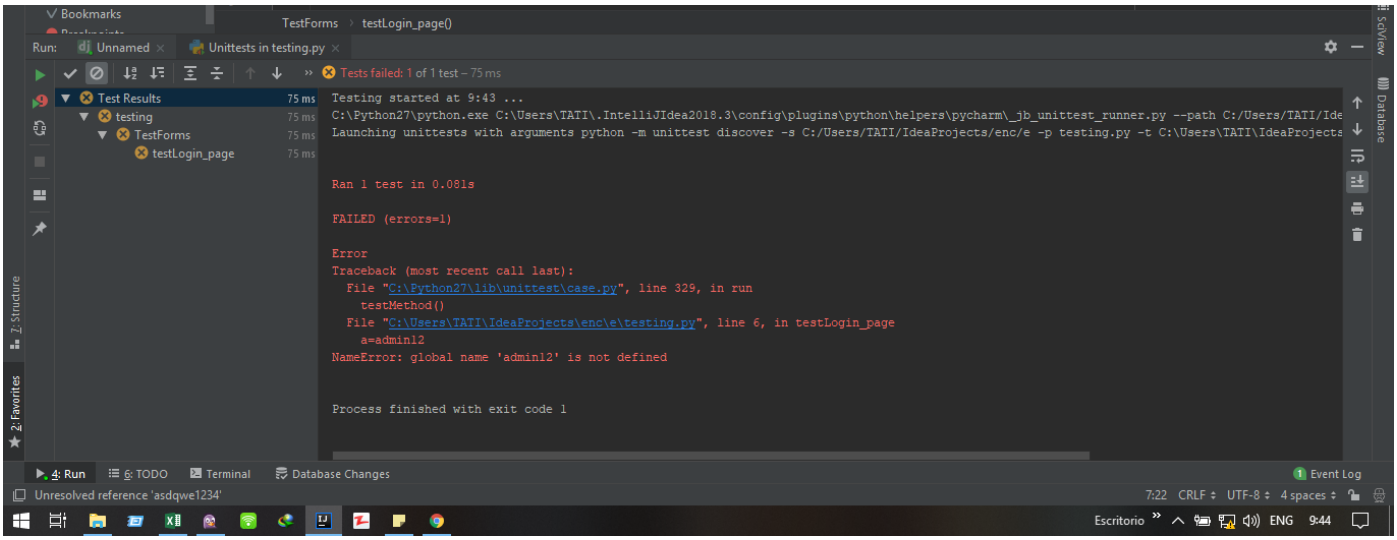



**Figura 14. Primera Iteración.**

**(Fuente: elaboración propia)**

Para una segunda iteración se arrojó 5 no conformidades

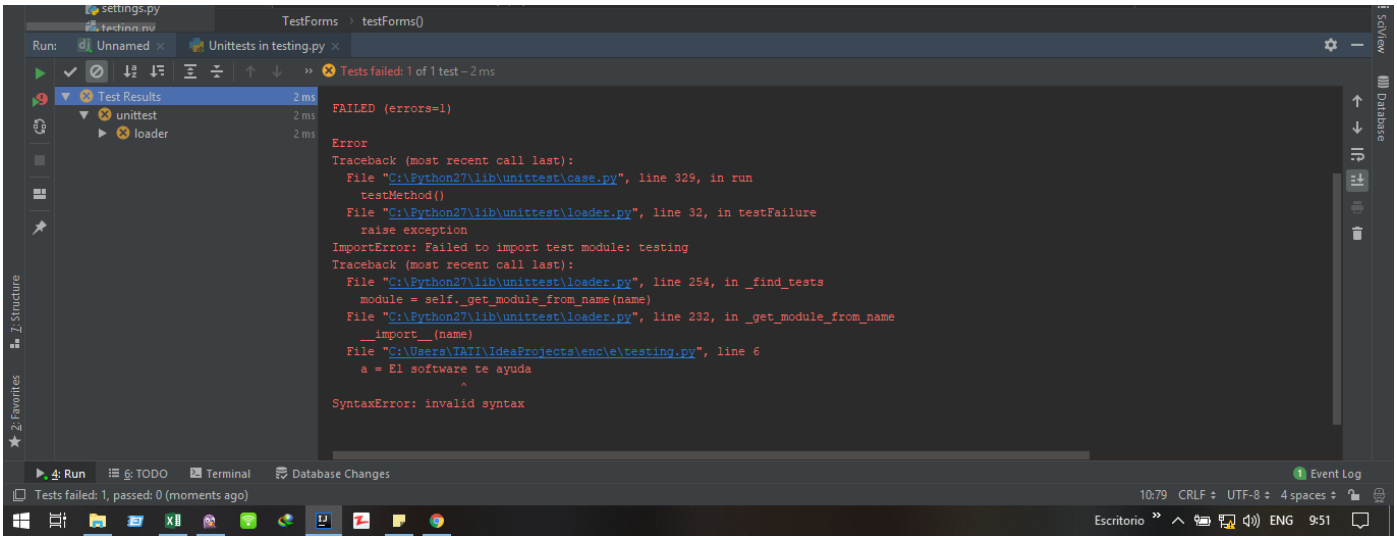



**Figura 15. Segunda Iteración.**

**(Fuente: elaboración propia)**

Para una tercera y última iteración no se encuentran no conformidades

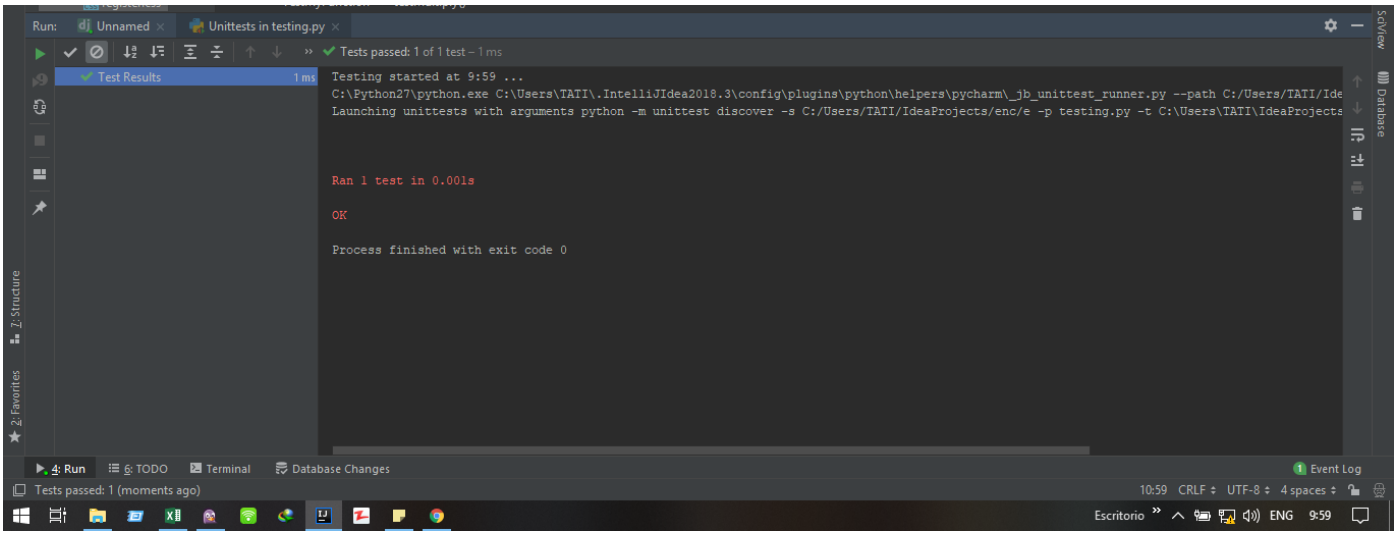



**Figura 16 Tercera Iteración.**

**(Fuente: elaboración propia)**

## 3.5 Conclusiones parciales

1. La implementación de la propuesta de solución permitió obtener una herramienta para evaluar la usabilidad centrada en el usuario de las aplicaciones web en la UCI.
2. La realización de las pruebas permitió obtener resultados satisfactorios, asegurando que el sistema implementado tiene una alta calidad y cumple con los requisitos funcionales de la propuesta de solución.

# Conclusiones

Al finalizar la presente investigación se puede concluir que:

1. La elaboración del marco teórico-referencial sobre la evaluación de la usabilidad de las aplicaciones web y la realización de este proceso en la UCI permitió identificar limitaciones y deficiencias en su funcionamiento.
2. El análisis de herramientas que automatizan la evaluación de la usabilidad de las aplicaciones web ha permitido comprobar que ninguna da solución al problema de la presente investigación, lo que fundamenta la necesidad de una herramienta con este fin para la UCI.
3. Se obtuvo el diseño de una herramienta que realiza la evaluación centrada en el usuario de la usabilidad de las aplicaciones web en la UCI para su posterior implementación.
4. Se ha obtenido una herramienta para la evaluación centrada en el usuario de la usabilidad de las aplicaciones web en la UCI con una alta adaptación a los cambios y escalable.
5. La realización de pruebas de software a la herramienta desarrollada para la evaluación centrada en el usuario de la usabilidad de las aplicaciones web en la UCI demostró que se ha implementado una solución de alta calidad.

# 1. Referencias bibliográficas


1. Definición de Usabilidad. [en línea], 2016. Disponible en: www.economiasimple.net/glosario/usabilidad(2016).
2. Herramientas para evaluar la usabilidad. [en línea], 2017. Disponible en: https://www.bravent.net/las-mejores-herramientas-para-evaluar-la-usabilidad-de-tu-portal-web.
3. ARREDONDO, T., 2014. Introducción a la Lógica Difusa.
4. SLIDER, W. y BUCKLEY, J.J., 2005. Fuzzy Expert System and Fuzzy Reasoning.
5. BAEZA-YATES, R., RIVERA LOAIZA, C. y VELASCO MARTIN, J., 2004. Arquitectura de la información y usabilidad en la web.
6. KOURO, S. y MUSALEM, R., 2015. Control Mediante Lógica Difusa.
7. MAGUIÑA, R.A., 2010. Sistemas de inferencia basados en lógica borrosa: Fundamentos y caso de estudio. Revista de investigación de Sistemas e Informática.
8. Python. Python [en línea], 2001. Disponible en: https://www.python.org/about/.
9. Qué es Python. Lenguaje de programación de propósito general, orientado a objetos, que también puede utilizarse para el desarrollo web [en línea], 2003. Disponible en: https://www.desarrolloweb.com/articulos/1325.php.
10. CALVO-FERNÁNDEZ, A., ORTEGA, S., VALLS, A. y ZAPATA, M., 2011. Evaluación de la usabilidad.
11. HASSAN, Y., MARTIN, F. y IAZZA, G., 2007. Diseño Web Centrado en el Usuario: Usabilidad y Arquitectura de la Información [en línea]. Disponible en: http://www.hipertext.net.
12. PRESSMAN, R., 2010. Ingeniería del software. Un enfoque práctico. 7ma Edición. México, D. F: s.n. ISBN 978-607-15-0314-5.
13. PERURENA, L. y MORÁGUEZ, M., 2013. Usability of Web sites, methods and evaluation techniques., pp. 19.
14. MENDOZA, D. y BAQUERO, L.R., 2016. Extensión de la herramienta Visual Paradigm for UML para la evaluación y corrección de Diagramas de Casos de Uso. S.l.: Universidad de Ciencias Informáticas.
15. GAMMA, E., HELM, R., JOHNSON, R. y VLISSIDES, J., 2009. Design Patterns_ Elements of Reusable Object-Oriented Software. S.l.: s.n.
16. GIRALDO, G.L., ACEVEDO, J.F. y MORENO, D.A., 2011. Una ontología para la representación de conceptos de diseño de software. Revista Avances en Sistemas e Informática, vol. 8, no. 3.
17. SELLARÈS, T., 2011. The Model View Controller: a Composed Pattern. Universitat de Girona [en línea], Disponible en: http://ima.udg.edu/~sellares/EINF-ES1/MVC-Toni.pdf.

18. SIEGEL, J., 2005. What is UML | Unified Modeling Language. [en línea]. [Consulta: 26 noviembre 2018]. Disponible en: http://www.uml.org/what-is-uml.htm.

19. Quevedo, V. D. et al. CEDRUX. Solución para sistemas de control logístico. 15 Convención Científica de Ingeniería y Arquitectura. Instituto Superior Politécnico José Antonio Hecheverría. 2010. 9p

20. RODRIGUEZ, J., 2008. Metodología de desarrollo de software. El Modelo en V o de Cuatro Niveles. [en línea], Disponible en: http://www.iiia.csic.es/udt/en/blog/jrodriguez/2008/metodologia-desarrollo-sotware-modelo en-v-o-cuatro-niveles.

21. OLIVERA. ANGEL Requerimientos funcionales y no funcionales. [En línea] SCRIBD, 2014. [Citado el: 4 de enero de 2018.] https://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales.

22. GONZALEZ ALVAR, FERNADEZ JAVIER. Requisitos no funcionales(RNF). *SoftQaNetwork.* [En línea] 2018. [Citado el: 1 de enero de 2018.] http://www.softqanetwork.com/requisitos-no-funcionales-nfr.

23. IBM Corp. Artefacto: Prototipo de interfaz de usuario. [En línea] 2012. [Citado el: 1 de enero de 2018.] http://cgrw01.cgr.go.cr/rup/RUP.es/LargeProjects/core.base_rup/workproducts/rup_user_interface_prototype_7237E5AA.html#.

24. Alegsa.com.ar. Definición de Arquitectura de sistemas (informática). [En línea] 2017. [Citado el: 1 de enero de 2018.] http://www.alegsa.com.ar/Dic/arquitectura_de_sistemas.php.

25. AdictosAlTrabajo.com. Patrones GRASP. [En línea] 2018. https://www.adictosaltrabajo.com/tutoriales/grasp/.

26. NanoPDF.com. Diagramas de despliegue. [En línea] 2018. https://nanopdf.com/download/figura-1-diagrama-de-despliegue_pdf.

27. Globe. Pruebas de caja negra. [En línea] 2017. https://www.globetesting.com/2012/08/pruebas-de-caja-negra/.

28. Hub, M. y Zatloukal, M. (2010). Model of usability evaluation of web portals based on the fuzzy logic. WSEAS Transactions on Information Science and Applications, 7(4):522–531.

29. [ISO, 1992] ISO (1992). ISO/IEC 9126: Information Technology - Software Product Evaluation Quality Characteristics and Guidelines for Their Use.

30. ISO (2005). ISO/IEC 25000 - System and Software Quality Requirements and Evaluation (SQuaRE).

31. Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning—i. Information sciences, 8(3):199–249.

32. Ahmad, A. R., Basir, O. A., y Hassanein, K. (2003). Fuzzy inferencing in the web page layout design. En WSMAI, pp. 33–41.
33. Alqurni, J., Alroobaea, R., y Alqahtani, M. (2018). Effect of user sessions on the heuristic usability method. International Journal of Open Source Software and Processes (IJOSSP), 9(1):62–81.
34. Correia, R. R. d. S. (2018). Detecc˜ ao de problemas de design em aplicac˜ oes model-template-view. Tesis de mer, Universidade Federal do Rio Grande do Norte.
35. Fernández Pérez, Y. (2018). Modelo computacional para la evaluación y selección de productos de software. Tesis doctoral, Universidad de Granada.
36. Gliwa, W., Zyluk, A., Grzesik, N., y Ku´ zma, K. (2018). The design and simulation of an automated de-icing control system in the aircraft diamond da42 Journal of KONES, 25.
37. Rodríguez, Tamara. “Metodología de Desarrollo Para La Actividad Productiva de La UCI.,” 2015.
38. Melin y Castillo, 2014] Melin, P. y Castillo, O. (2014). A review on type-2 fuzzy logic applications in clustering, classification and pattern recognition. Applied soft computing, 21:568–577
39. Mascheroni, Maximiliano. 2012. Herramienta para automatizar la evaluación de la usabilidad en productos software. Buenos Aires: s.n., 2012.
40. Obeso, María Elena Alva. 2005. Metodología de Medición y Evaluación de la Usabilidad en sitios web educativos. UNIVERSIDAD DE OVIEDO: s.n., 2005.
41. Rodríguez, Daniel. 2008. Las Claves de la Optimización de la Conversión. Trucos para vender más en Internet. 2008.
42. Koomen, Pete. “Optimizely,” 20015. optimizely.com.
43. Yavilevich, Arik. “ClickTale,” 2006. clictale.com.
44. Chang, E. y Dillon, T. S. (2006). A usability-evaluation metric based on a soft-computing approach. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 36(2):356–372.
45. Dubey, S. K., Rana, A., y Sharma, A. (2012). Usability evaluation of object oriented software system using fuzzy logic approach. International Journal of Computer Applications, 43(19):1–6.

46. Ramanayaka, K. H., Chen, X., y Shi, B. (2018). Unscale: A fuzzy-based multi-criteria usability evaluation framework for measuring and evaluating library websites. IETE Technical Review, pp. 1–20.
47. Baquero, L., Rodríguez, O., y Ciudad, F. (2016). Lógica difusa basada en la experiencia del usuario para medir la usabilidad. Revista Latinoamericana de Ingeniería de Software, 4(1):48–54.

# Anexos

## Anexo A. Entrevista realizada a directivos y trabajadores del Laboratorio de Pruebas de Software de la UCI

Objetivo: Obtener los criterios de los directivos y trabajadores del Laboratorio de Pruebas de Software relativos a:

• Modelos de calidad utilizados

• Estructuración de las pruebas de usabilidad

• Métodos y técnicas de evaluación de usabilidad

• Estado de la evaluación centrada en el usuario

Aspectos a abordar en la entrevista:

1. ¿Se han estandarizados modelos de calidad de software aplicable a la evaluación de la usabilidad del producto de software?

2. ¿Cómo está estructurado el proceso de evaluación de la usabilidad?

3. ¿Qué métodos y técnicas de evaluación de usabilidad son empleados?

4. ¿Considera necesario realizar evaluación centrada en el usuario? ¿? Cree que la evaluación centrada en los expertos pueda sustituirla?

5. ¿Qué considera necesario para la evaluación centrada en el usuario?

6. ¿Piensa que sea necesario automatizar las pruebas de usabilidad, en especial la evaluación centrada en el usuario?