Facultad 4

INFORMATIZACIÓN DEL PROCEDIMIENTO DE LAS PRUEBAS DE USABILIDAD EN EL CENTRO FORTES

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor:

Carlos Ernesto Deniz Peña

Tutores:

Ing. Lizardo Ramírez Taboada

Ing. Leannys Rodríguez Moreno

La Habana, junio 2015

"Año 57 de la Revolución"

Declaración de autoría

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas y a la facultad 4 los derechos patrimoniales de la misma, con carácter exclusivo y la autorizo para que haga el uso que estime pertinente.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2015.

Autor

_____
Carlos Ernesto Deniz Peña

Tutor

_____
Ing. Lizardo Ramírez Taboada

Tutora

_____
Ing. Leannys Rodriguez Moreno

Dedicatoria

A mi abuela que tanto anhelaba verme graduado, a mis padres que nunca dejaron de soñar con este momento, a mi hermana que siempre quiso un hermano ingeniero y a mis tíos que nunca dejaron de apoyarme.

Agradecimientos

En especial a Yamile que sin ella nunca hubiera sido posible llegar hasta aquí.

A mis padres que siempre me apoyaron y exigieron.

A mis tíos Ernesto y Tania que siempre me dieron fuerzas para seguir adelante convirtiéndose en padres para mí, a Susy que siempre estuvo pendiente de mí, a mi hermana que siempre me exigió al máximo y al resto de la familia.

A mis viejos amigos y compañeros por siempre apoyarme.

A mis compañeros de ahora por hacerme uno más de ellos. De forma especial a Wilfredo por toda la ayuda prestada y a mis tutores por siempre estar para guiarme y orientarme. A los profesores que a lo largo de estos años me apoyaron y soportaron.

A todos GRACIAS.

Resumen

En la actualidad, la usabilidad es una exigencia del *software*, los cuales están dirigidos a un público cada vez más amplio y a usuarios cada vez menos expertos en el manejo de sistemas informáticos. En el centro FORTES de la Universidad de las Ciencias Informáticas (UCI) se realizan pruebas de usabilidad, estas se llevan a cabo mediante una lista de chequeo, la cual es un documento Excel, dividido por las categorías a evaluar. Para agilizar el proceso de estas pruebas se realizó el trabajo de diploma Informatización del procedimiento de las pruebas de usabilidad en el Centro FORTES que tiene como objetivo informatizar el procedimiento de las pruebas de usabilidad en el Centro FORTES. La presente investigación analiza aspectos teóricos y conceptuales relacionados con la evaluación de usabilidad. Para guiar el proceso de desarrollo de la propuesta de solución se utilizó la metodología *Extreme Programming* (XP) generándose los artefactos fundamentales que propone cada fase. Las herramientas y tecnologías utilizadas fueron: como lenguaje del lado del cliente JavaScript, HTML y CSS, del lado del servidor PHP, como *framework Symfony* y el entorno de desarrollo integrado PHPStorm. También, se usó MySQL como gestor de base de datos y Apache como servidor web. Finalmente, se realizaron las pruebas de aceptación y unitarias a la solución, verificándose el cumplimiento de los objetivos propuestos. Esta herramienta permite agilizar, controlar y realizar un seguimiento de las pruebas de usabilidad a los proyectos del centro.

Palabras clave: lista de chequeo, pruebas de usabilidad, software, usabilidad.

Índice

Índice de figuras

Índice de tablas

INTRODUCCIÓN

Con el avance acelerado de la ciencia y la tecnología se han desarrollado numerosos recursos informáticos que permiten la difusión de la información, el comercio y demás cuestiones de gran importancia para el usuario, pero la dificultad de uso que presentan algunos, desmotiva al cliente en cuanto a la exploración y navegación en los mismos. A todo esto se ha incrementado el interés por perfeccionar las Tecnologías de la Información y las Comunicaciones (TIC) con una serie de buenas prácticas dentro del proceso de desarrollo de *software*.

Los usuarios rechazan aquel *software* que implementan interfaces complejas, difíciles de usar y poco amigables, donde pasan trabajo buscando la información, pues esta no está clara, aquellos en los cuales tienen que navegar con profundidad para encontrar lo que buscan. La interacción con el usuario constituye uno de los aspectos más importante de cualquier sistema interactivo y es precisamente la interfaz la parte del sistema que facilita dicha interacción.

Dada esta situación se impone desarrollar sistemas en los que se cumplan las exigencias y necesidades de los usuarios, con un buen funcionamiento de la interfaz y adecuada organización de la información, para obtener una correcta comprensión de los contenidos que permita a los usuarios interactuar de manera sencilla, fácil y cumpliendo sus objetivos. En este contexto surge la e*xperiencia de usuario* como disciplina, la cual engloba la usabilidad, accesibilidad, arquitectura de información y diseño de interfaz e interacción que intentan proporcionar estas ventajas (Carreras, 2007).

Desde las fases tempranas del proceso de desarrollo de *software* se debe gestionar y aplicar disciplinas como la usabilidad, elemento que adquiere un significado específico en el campo de la informática e Internet, definiéndose la usabilidad por la ISO/IEC 9241-11 como: "*el grado en el que un producto puede ser utilizado por usuarios específicos para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un determinado contexto de uso*" (ISO, 1998). De esta manera atraer y fidelizar a los usuarios, para que estos no sientan la necesidad de buscar otros productos para satisfacer sus necesidades.

En la actualidad la usabilidad es una exigencia del *software*, los cuales están dirigidos a un público cada vez más amplia y a usuarios cada vez menos expertos en el manejo de sistemas informáticos, donde esta se destaca como atributo fundamental para el éxito de un producto informático. Es tanto así, que para incorporar estas prácticas dentro del proceso de desarrollo de *software* se han creado laboratorios especializados en empresas y universidades con el objetivo de aplicar metodologías, técnicas y

herramientas enfocadas a gestionar y medir la usabilidad de los productos. Grandes empresas internacionales que operan en Internet como Microsoft, Sun Microsystems, IBM y Oracle, destinan considerables sumas de sus presupuestos, incorporando estas iniciativas para elevar la calidad de sus productos (HERRERA, y otros, 2010).

La usabilidad se ha convertido paulatinamente en una necesidad y casi obligación para las empresas dedicadas al diseño y desarrollo de sitios web y de *software*. No contar con una evaluación o prueba de este tipo afecta la calidad del producto final y conlleva a incrementar los gastos del proceso de fabricación. Estas son de gran importancia porque permiten conocer de forma más exacta como se siente el usuario respecto al producto.

La Universidad de las ciencias informáticas (UCI), es una institución que entre sus misiones asume la de impulsar el desarrollo de la industria del *software* en Cuba. Uno de los factores que influye en la gestión de la usabilidad en la UCI es que los usuarios finales son mayoritariamente excluidos del desarrollo del producto y en ocasiones su participación es nula. Debe existir una interacción usuario-grupo de desarrollo-cliente desde las etapas iniciales del proyecto para así lograr mayor calidad en las aplicaciones producidas en la universidad.

La UCI cuenta con una infraestructura productiva que incluye varios centros de desarrollo entre ellos el Centro de Tecnologías para la Formación (FORTES). Este tiene entre sus misiones el aseguramiento de la calidad de los diferentes productos que desarrolla, para esto brinda una serie de servicios como son las auditorias, revisiones y pruebas, entre las que se encuentran las de usabilidad.

En el Centro FORTES se realiza este tipo de pruebas mediante una lista de chequeo que tiene todos los aspectos que se deben medir a un *software* en una prueba de usabilidad. Esto hace el trabajo más engorroso y conlleva a un empleo considerable de tiempo. No se cuenta con un historial de las pruebas que se han llevado a cabo, lo que imposibilita la no realización de un seguimiento de las deficiencias detectadas y que no se controle el estado de solución. Esto puede traer consigo que en otras iteraciones se repitan algunas de estas deficiencias ya encontradas. Además, no se almacenan criterios de otros revisores impidiendo que los especialistas obtengan conocimientos de pruebas realizados con anterioridad.

Para darle solución a la problemática planteada anteriormente se propone el siguiente **problema a resolver:** ¿Cómo informatizar el procedimiento de las pruebas de usabilidad en el Centro FORTES?

Para un entendimiento global del tema, se trabaja sobre el **objeto de estudio:** Las pruebas de usabilidad.

Para darle solución al problema se plantea el siguiente **objetivo general:** Informatizar el procedimiento de las pruebas de usabilidad en el Centro FORTES.

En correspondencia con el objetivo general de la investigación que se presenta se propone realizar los siguientes **objetivos específicos:**

- ✓ Determinar conceptos y elementos teóricos para el análisis de las pruebas de usabilidad.
- ✓ Diseñar la herramienta para la gestión de las pruebas de usabilidad en el Centro FORTES.
- ✓ Realizar la implementación y pruebas de la herramienta para el procedimiento de las pruebas de usabilidad en el Centro FORTES.

De lo que se deriva el siguiente **campo de acción:** La informatización del procedimiento de las pruebas de usabilidad para el Centro FORTES.

Con vista a resolver el problema planteado se propone la siguiente **hipótesis:** La informatización del proceso de pruebas de usabilidad permitirá agilizar, controlar y realizar un seguimiento de estas pruebas realizadas a los proyectos del Centro FORTES.

Las tareas de investigación definidas para dar cumplimiento al objetivo de la investigación fueron las siguientes:

- ✓ Identificación y análisis de las tendencias actuales sobre las pruebas de usabilidad.
- ✓ Selección de la metodología que guiará el proceso de desarrollo del *software*.
- ✓ Selección de las tecnologías y herramientas informáticas a utilizar en el desarrollo de la propuesta de solución.
- ✓ Descripción de las funcionalidades y características de la herramienta.
- ✓ Diseño de la herramienta utilizando las metodologías tradicionales.
- ✓ Implementación y pruebas de la herramienta.

Para la realización de esta investigación se utilizan la combinación dialéctica de los métodos teóricos y empíricos, los que permitieron develar la parte de la ciencia que está siendo objeto de estudio.

Métodos teóricos

Histórico-lógico: utilizado al analizar la caracterización de la evolución histórica de la usabilidad como base para concebir la herramienta actual

Analítico-sintético: se empleó para extraer y diferenciar técnicas, metodologías y conceptos básicos sobre la usabilidad en los productos del *software*. Estos facilitaron identificar características, según el grado de aplicación y tendencias en el mundo. Además, permitió analizar y sintetizar el objeto de estudio pues la síntesis se realiza sobre la base de los resultados previos del análisis.

Métodos empíricos

Observación: se identificaron las principales dificultades de los usuarios al interactuar con un sistema, así como la necesidad de los mismos que se diseñen interfaces adaptadas a su modelo mental. Esto permitió arribar a conclusiones y ofrecer recomendaciones para mejorar la calidad de los productos.

Estructura capitular

La presente investigación está conformada por la siguiente estructura: introducción, tres capítulos, conclusiones, recomendaciones, trabajos citados, glosario de términos y anexos. Los capítulos abordan los siguientes temas:

Capítulo 1: Fundamentación teórica.

En este capítulo se realiza un análisis de los principales conceptos relacionados con el objeto de estudio, así como el estado del arte de las herramientas en apoyo a las pruebas de usabilidad y en sentido general. Se describe además la metodología a utilizar en el desarrollo del *software*. También, se realiza un estudio de las distintas herramientas y tecnologías a utilizar.

Capítulo 2: Descripción de la propuesta de solución.

En este capítulo se determinan los servicios que brindará la herramienta, definiéndose las funcionalidades que debe cumplir, así como el diseño y se generan los artefactos que propone la metodología.

Capítulo 3: Implementación y pruebas.

Este capítulo abarca todo lo relacionado con la implementación de la herramienta y el proceso de pruebas utilizado. Se implementan todas las funcionalidades identificadas, logrando una herramienta que permita cumplir el objetivo general de los estudios que se presentan. Se detallan también, las pruebas que se le realizaron a la herramienta ya finalizada, con el objetivo de asegurar la eficiencia de la solución.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se exponen aspectos y conceptos relacionados con el objeto de estudio y se realiza un análisis sobre la evolución de la usabilidad, así como un estudio de sistemas que realizan estas pruebas. Posteriormente, se presenta un análisis de las características fundamentales de las metodologías de desarrollo de *software*, herramientas, lenguajes y tecnologías. Esto permitió seleccionar las más adecuadas para el dar cumplimiento al objetivo general de la investigación.

1.1. Principales conceptos de la investigación

Para una mejor comprensión del concepto de usabilidad a continuación se explican algunos términos generales que están relacionados.

Se puede decir que la interacción persona-ordenador (IPO) es la disciplina que estudia el intercambio entre las personas y los ordenadores. Se encarga del diseño, evaluación e implementación de las interacciones. La comunicación entre el usuario y la máquina debe ser especial para poder comprender cada información. Los sistemas que se desarrollan deben ser cada vez más fáciles de manejar y tratar de motivar a los usuarios a los cuales van dirigidos. La idea es que los usuarios sean capaces de aprender de una forma fácil y sencilla las principales cuestiones para manejar el *software* que está brindándole la información. El intercambio debe ser agradable para que el usuario se sienta identificado con la herramienta y de una forma u otra se motive a seguir visitando, trabajando e incluso recomendando dicho *software* (Prieto, y otros, 2010).

Relacionada de la IPO se encuentra la experiencia de usuario (Carreras, 2007), esta disciplina estudia la sensación, sentimiento, respuesta emocional, valoración y satisfacción del usuario respecto a un producto, resultado del fenómeno de interacción con el producto y la interacción con su proveedor.

La experiencia del usuario representa un cambio emergente del propio concepto de usabilidad, donde el objetivo no se limita a mejorar el rendimiento del usuario en la interacción, eficacia, eficiencia y facilidad de aprendizaje, sino que se intenta resolver el problema estratégico de la utilidad del producto y el problema psicológico del placer y diversión de su uso. Para esta investigación resulta de gran importancia la valoración y satisfacción del usuario respecto al producto, resultado del fenómeno de la interacción con él y su proveedor. Por lo tanto se asume que la usabilidad mide la calidad de la experiencia del usuario cuando interactúa con un producto.

1.2. Usabilidad

La usabilidad, anglicismo que significa "facilidad de uso", como indican (Bevan, y otros, 1991) parece tener su origen en la expresión *"user friendly"* o amigable al usuario, que es reemplazada por sus connotaciones vagas y subjetivas.

A partir de la bibliografía consultada y considerando los aspectos establecidos anteriormente, en la actualidad existen numerosas definiciones de usabilidad reconocidas por los principales autores en el tema y son las que a continuación se exponen:

1.2.1. Definición ISO/IEC 9241-11

La Organización Internacional para la Estandarización1 (ISO por sus siglas en inglés) en el borrador internacional del estándar ISO/DIS 9241-11 (*Guidance on Usability*) define la usabilidad 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*" (DIS, 1998).

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 (Cabezón, 2014):

- ✓ **Eficacia:** definido en términos de que el usuario logra lo que quiere.
- ✓ **Eficiencia:** referido a que el usuario logra lo que quiere rápidamente.
- ✓ **Satisfacción:** ausencia de incomodidad y actitud positiva en el uso del producto.

[1] Organismo encargado de promover el desarrollo de normas internacionales de fabricación

Componentes y la relación entre ellos ilustrado en la gráfica siguiente:

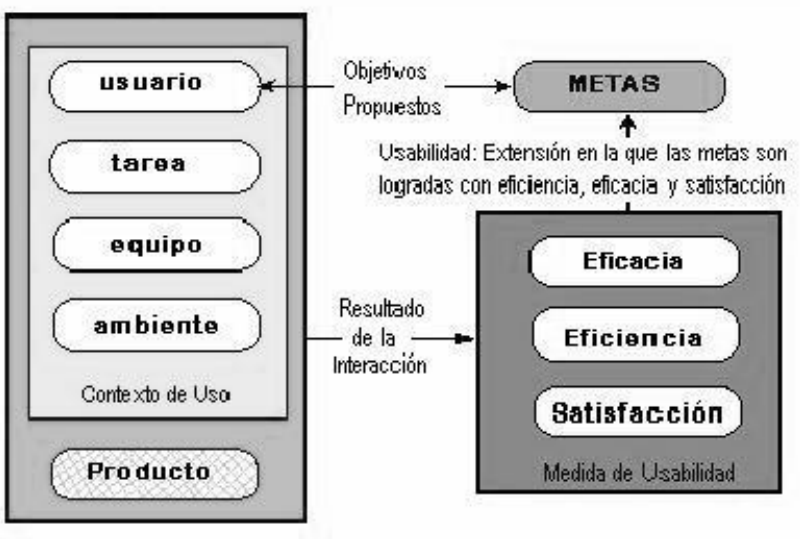

Figura 1. Marco de definición de usabilidad de acuerdo a (ISO, 1998).

ISO 9241 define la usabilidad en términos de calidad del trabajo, un sistema en uso, la cual depende de todos los factores que pueden influenciar el uso de un producto en el mundo real: factores organizacionales (prácticas de trabajo, ubicación o apariencia de un producto), diferencias individuales entre usuarios (factores culturales y preferencias), experiencia (ISO, 1998).

1.2.2. Definición ISO/IEC 9126

De acuerdo al estándar ISO/IEC 9126 (*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 definida por el estándar (ISO/IEC, 1991) 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*" (ISO/IEC, 1991).

En la parte ISO/IEC 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 (Bevan, 1995).

- ✓ **Comprensibilidad:** el *software* debe permitir al usuario entender si 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:** el producto debe permitir al usuario operarlo y controlarlo. Aspectos de conformidad, mutabilidad, adaptabilidad e instalación pueden afectar a la operabilidad. También, 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 el *software* más atractivo al usuario, tal como su uso de color y la naturaleza del diseño gráfico.

1.2.3. Definición de Nielsen

Desde la visión de Nielsen (Nielsen, 1993), la usabilidad se define en términos de cinco atributos:

- ✓ **Aprendizaje:** significa que nuevos usuarios deberían aprender fácilmente a usar el sistema.
- ✓ **Eficiencia:** el sistema debería ser eficiente cuando el usuario ha aprendido a usarlo.
- ✓ **Memorización:** fácil de recordar incluso después de algún período sin uso.
- ✓ **Prevención de error:** un producto de *software* deberá tener un bajo porcentaje de error y en presencia de estos el usuario pueda recuperarse fácilmente.
- ✓ **Satisfacción:** significa que el sistema debe ser agradable y amigable cuando sea usado o utilizado.

En el modelo de Nielsen, la usabilidad es:

"*Parte de la utilidad del sistema, la cual es parte de la aceptabilidad práctica y finalmente parte de la aceptabilidad del sistema*" (Nielsen, 1993), tal como muestra la figura 2.

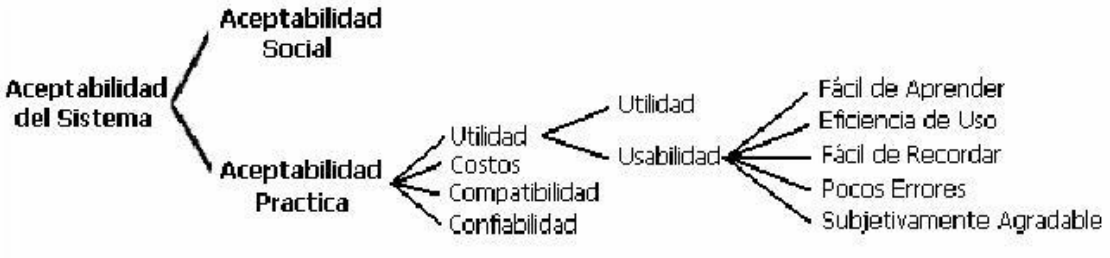

Figura 2. Marco de definición de usabilidad de acuerdo a Nielsen.

Una vez analizados estas definiciones se concluye que la usabilidad estudia la manera de diseñar productos para que los usuarios puedan interactuar con ellos de la forma más fácil, cómoda e intuitiva posible, para que consigan sus objetivos de una manera rápida y sencilla. El autor de la presente

investigación coincide que el grado de usabilidad es un requisito valioso para una aplicación web, constituye un concepto importante dentro de la interacción usuario–aplicación, puesto que es una vía fundamental para crear de una forma sencilla y eficaz un entorno gráfico relacionado con la web, además de ser un instrumento para comprobar la efectividad desde el punto de vista del usuario final.

1.3. Lista de chequeo

La lista de chequeo es un documento que apoya la comprobación de usabilidad de un sitio web o aplicación, a través de preguntas directas acerca de las características de su interfaz.

Para evitar las observaciones técnicas que se alejan un poco de los criterios de interacción humano – computador (IHC) se utilizan las listas de chequeo que dan las pautas para que los desarrolladores tengan la posibilidad de observar más de cerca la interacción.

Las listas de chequeo están creadas a partir de un conjunto de heurísticas (en el caso de esta investigación se denominan características) de usabilidad recopiladas de las propuestas de (Nielsen, 1994). Estas características sirven como material de referencia para realizar una evaluación consistente y objetiva. La utilización de la lista de chequeo es una herramienta importante a la hora de evaluar, pues ayuda a centralizar las observaciones en aspectos que tienen que ver en forma directa con la interfaz y el desempeño del usuario eficaz y eficientemente.

En el Centro FORTES hasta el momento se realizan las pruebas de usabilidad mediante una lista de chequeo. Un documento Excel que está dividido en categorías como por ejemplo aspectos generales, formularios, mensajes (tratamiento de errores). Las cuales a su vez contienen una serie de preguntas asociadas que permiten realizar una evaluación minuciosa de los productos a analizar.

Como la investigación, busca enfrentar el desarrollo del producto de *software* desde una perspectiva centrada en los principios de usabilidad para lograr un producto de calidad en cuanto a usabilidad se refiere, se hace necesario automatizar esta lista de chequeo para que el trabajo sea más eficiente y eficaz.

1.3 Pruebas de usabilidad de software

Con vista a determinar la calidad del *software* se realizan pruebas de liberación, entre ellas las de usabilidad. El grado de esta, en una aplicación es una medida empírica y relativamente. Empírica porque no se basa en opiniones o sensaciones, sino en pruebas de usabilidad realizadas en un laboratorio u observadas mediante trabajo de campo. Relativa porque el resultado depende de las metas planteadas o de una comparación con otras aplicaciones similares (Picasso, 2012).

La esencia de las pruebas de usabilidad radica en la observación objetiva del usuario, mientras utiliza el sistema, para llevar a cabo acciones reales. En estas los observadores no le enseñan a los participantes

cómo utilizar el sistema y tampoco contestan preguntas; debe ser como si los participantes estuvieran solos.

Hay tres factores claves para realizar las pruebas de usabilidad (Maurer, 2004):

- ✓ Los participantes deben ser usuarios actuales o futuros del sistema. Un error común es dejar que los gerentes prueben el sistema, una vez que esté en liberación.
- ✓ Los participantes deben intentar realizar las tareas que normalmente realizarían con el sistema. Es crucial que estas tareas sean realistas.
- ✓ El contexto bajo el que se realiza la prueba, debe ser lo más cercano posible al contexto real en el que el sistema se utilizará.

El proceso de estas pruebas en el Centro FORTES no involucra a los usuarios finales de sus productos. Son realizadas por especialistas que han estudiado el tema y conocen los criterios que se deben tener en cuenta. Además, hacen uso de la lista de chequeo para guiarse y llegar a conclusiones sobre el estado de la usabilidad del producto que evalúan.

1.4 Descripción del proceso de pruebas de usabilidad de software del Centro FORTES

Las pruebas de usabilidad es un servicio que ofrece el Centro FORTES a sus productos, el cual consta de los siguientes pasos:

Paso 1: Solicitud de las pruebas.

El líder de proyecto realiza la solicitud del servicio al Asesor de calidad mediante la Planilla de solicitud de las pruebas. Este es el encargado de realizar la planificación y asignar el o los especialistas que ejecutarán las pruebas.

Paso 2: Planificación.

Se realiza el cronograma de las pruebas que contiene las fechas en la que se llevará cada una de las tareas que se deben ejecutar.

Paso 3: Capacitación del equipo de pruebas.

En este aspecto un responsable del proyecto al que se le realizará las pruebas se reúne con el o los especialistas que llevarán a cabo el procedimiento y explica el negocio del sistema.

Paso 4: Ejecución de las pruebas.

En este paso se ejecutan las pruebas mediante una lista de chequeo que contiene los aspectos que deben evaluarse en el sistema. La primera información que se desea obtener es el grado de entendimiento, para ello se observa únicamente la interfaz.

Paso 5: Evaluación de las pruebas.

Se analizan los resultados obtenidos y se discuten sus posibles implicaciones con el responsable del proyecto al cual se le realizaron las pruebas.

Paso 6: Culminación de las pruebas.

Con las observaciones efectuadas y el análisis de los datos se elaborará un informe que contendrá los resultados y las recomendaciones con el objetivo de mejorar el producto. Este informe no solo cumple con la formalidad de terminar la prueba, sino que es el respaldo para explicar a otras personas que no estuvieron involucradas los motivos de las acciones y decisiones.

La propuesta del modelo de proceso para la realización de las pruebas de usabilidad de *software* a nivel de sistema en el Centro FORTES se muestra en la siguiente figura:

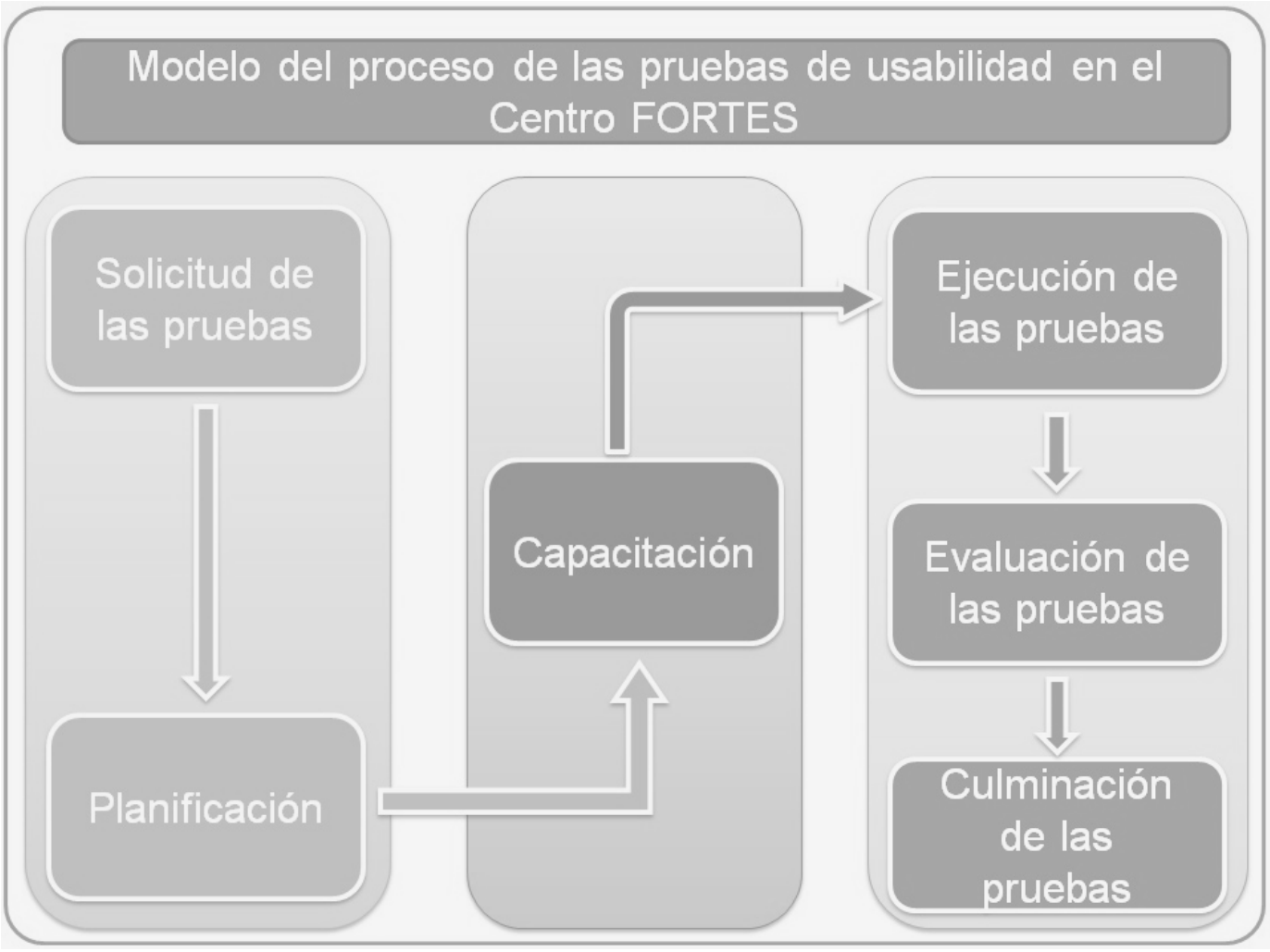

Figura 3. Modelo del proceso de las pruebas de usabilidad en el Centro FORTES.

1.5 Técnicas de evaluación de usabilidad

La evaluación de la usabilidad abarca una serie de técnicas que ayudan a medir la forma en que los usuarios son capaces de utilizar un producto, al mismo tiempo que determinan la manera en que lo hacen. El llevarla a cabo derivará en la creación de mejores productos, por lo que conseguirá que los usuarios realicen sus actividades fácilmente. De hecho, sin evaluación será imposible saber si un producto cumple las expectativas de sus creadores o si se adapta a su contexto social, físico y organizativo (Perurena Cancio, y otros, 2013).

1.5.1 Inspección

Es una técnica que emplea expertos; que son personas que conocen del tema en cuestión, se basan en el recorrido y análisis de la aplicación identificando errores y problemas de diseño.

La evaluación heurística es un tipo de método de inspección, que tiene como ventaja la facilidad y rapidez con la que se puede llevar a cabo.

Este tipo de evaluación normalmente la lleva a cabo un grupo reducido de evaluadores que, en base a su propia experiencia, fundamentándose en reconocidos principios de usabilidad (heurísticos) y apoyándose en guías elaboradas para tal fin, evalúan de forma independiente la aplicación, contrastando finalmente los resultados con el resto de evaluadores (Granollers, y otros, 2004).

1.5.2 Indagación

Se realiza mediante la recogida de información; ya sea mediante observación, preguntas o interacción con los usuarios. Se utiliza a menudo en las primeras etapas del proceso de desarrollo de *software* que es donde se necesita una mayor información con respecto al trabajo que se va a realizar (Granollers, y otros, 2004).

Observación de campo

La técnica de evaluación conocida como observación de campo tiene como principal objetivo entender como los usuarios de los sistemas interactivos ejecutan sus tareas y concretamente conocer todas las acciones que estos realizan.

Grupo de discusión dirigido

El grupo de discusión dirigido es una técnica de recogida de datos donde se reúnen de seis a nueve personas guiados por el evaluador experto en usabilidad. Consiste en indagar cuanto sea posible respecto a una interfaz de usuario.

Entrevistas

Una entrevista consiste básicamente en una conversación donde uno o varios usuarios reales del sistema que se va a desarrollar o a rediseñar, responden a una serie de preguntas, relacionadas con el sistema que el entrevistador les va formulando.

Cuestionarios

Los cuestionarios son unas listas de preguntas que el evaluador distribuye entre usuarios y/o implicados para que estos devuelvan las respuestas y así poder extraer conclusiones. Normalmente se distribuye en formato escrito y las preguntas plantean aspectos relacionados con el sistema o aplicación concreta. La base del cuestionario es la recolección de información a partir de respuestas contestadas por los usuarios y/o los implicados.

Grabación de uso

La técnica grabación de uso se basa en la utilización de una aplicación *software* que captura todas las acciones de un usuario.

1.5.3 *Test* de usuarios

El *test* con usuarios es una prueba de usabilidad que se basa en la observación y análisis de cómo un grupo de usuarios reales utiliza la aplicación, anotando los problemas de uso con los que se encuentran para poder solucionarlos posteriormente.

Como toda evaluación de usabilidad, cuanto más se espera para su realización, más costosa resultará la reparación de los errores de diseño descubiertos. Esto quiere decir que no solo debemos realizar este tipo de pruebas sobre la aplicación una vez implementado, sino también, sobre los prototipos de la misma.

Es una prueba complementaria a la evaluación heurística, pero un *test* con usuarios es más costoso, por lo que es recomendable realizarlo siempre después de una evaluación heurística, ya que sería desperdiciar tiempo y dinero utilizarlo para descubrir errores de diseño motivados por el no cumplimiento en el desarrollo de principios generales de usabilidad (heurísticos).

Las ventaja que ofrecen los *test* de usuarios frente a otro tipo de evaluaciones es que por un lado es una demostración con hechos, por lo que sus resultados son más fiables y por otro porque posibilitan el descubrimiento de errores de diseño imposibles o difíciles de descubrir mediante la evaluación heurística.

Llevar a cabo un *test* de usuarios formal obligaría a alquilar un local, laboratorio adecuado, contratar a evaluadores especializados, así como a delegar en alguna empresa la selección y reclutamiento de los

participantes de la prueba. Realmente sería bastante costoso y poco viable para la gran mayoría de casos (Hassan Montero, y otros, 2003).

Estas técnicas de evaluación de la usabilidad como la indagación y el *test* de usuario involucran a usuarios finales o a personas que no son especialista en el tema. Así se recopila información y se arriban a conclusiones, además posibilita conocer si el producto cumple las expectativas de sus creadores o si se adapta al contexto social, físico y organizativo para el que se desarrolló. Se recomienda al Centro FORTES que un futuro emplee algunas de estas técnicas para que sus productos cumplan estrictamente con los principios de usabilidad y de esta forma lograr calidad en cuanto a usabilidad.

1.6 Herramientas homólogas

La mayoría de las empresas que desarrollan *software* utilizan iniciativas para contar con productos usables. Ejemplo de esto son los laboratorios de usabilidad, espacios especialmente adaptados para gestionarla y medirla. Son varias las instalaciones que se dedican a este fin, pero también existen consultorías que se encargan de gestionar y medir la usabilidad, ejemplo de esto es *Xperience Consulting* que se dedica a brindar servicios a compañías como Google, Microsoft, Dell y Motorola (Baquía, 2007). En estas iniciativas se utilizan algunas de las herramientas que a continuación se mencionan para la evaluación de la usabilidad.

1.6.1 UserZoom

UserZoom es un *software* de testeo remoto, desarrollado por un equipo de expertos de *Xperience Consulting.* Este analiza la información sobre las tareas más importantes que los usuarios realizan en una web, como el lugar exacto donde se han realizado los clics, el tiempo empleado, por ciento de éxito y fracaso de las tareas, tiempo y número de clics medio, máximo y mínimo (Nuez, 2009).

Realiza las técnicas de:

- ✓ Seguidor de ojo.
- ✓ *Test* de usuarios.
- ✓ Agrupamiento de tarjetas.
- ✓ Pensando en voz alta.
- ✓ Grabación de uso.
- ✓ Evaluación heurística.

1.6.2 Morae

Es un paquete completo para realizar los exámenes, se encuentra disponible solo para el sistema operativo Windows. Permite conectar una grabadora de vídeo para luego ver las reacciones de las personas al mismo tiempo que interactúan con la evaluación. La suite consta de tres productos:

- ✓ ***Recorder***: es el programa principal y permite grabar la sesión (pantalla, teclado, mouse) y al usuario al mismo tiempo.
- ✓ ***Observer****:* se utiliza para ver en tiempo real y de forma remota el *test* de usabilidad y permite realizar notas sobre el mismo.
- ✓ ***Manager****:* sirve para tomar decisiones en base a los resultados obtenidos en cada examen, permite analizar las grabaciones y generar reportes (Carvajal, y otros, 2010).

1.6.3 Obremus

Una herramienta para la observación remota de usuario, se ha creado como un módulo de la herramienta integrada WebA (Web Análisis, una herramienta de ayuda para el análisis de sitios web), está formada por módulos que permiten la evaluación semiautomática de la usabilidad mediante *test* de satisfacción de usuario y basados en normas ISO. Además, define el *test* de usuario remoto, la recopilación de información y el posterior análisis de los mismos.

Es un sistema compuesto por cuatro módulos independientes. Los módulos de grabación y moderación, suministrados en formato de aplicación multilenguaje para entornos Windows, permiten a usuarios y moderadores la participación en pruebas de usabilidad remotas. Por otra parte, el módulo de evaluación facilita la definición y la gestión de los estudios o proyectos de usabilidad, así como de sus sesiones asociadas (Eguizabal Alonso, y otros, 2009).

1.6.4 Usertesting

Usertesting es una herramienta con un costo de 39 dólares americanos. Para la configuración de un *test* se especifica la dirección url y se describe el escenario que se le plantea al participante. El análisis de los resultados cuenta con un video de grabación del rastro del ratón durante la sesión de *test* acompañado por la grabación de la voz de los usuarios. También, un resumen de sugerencias que los usuarios proponen para cada sitio web y tarea (Rodríguez, y otros, 2009).

Es una herramienta que permite realizar:

- ✓ *Test* de usuarios.
- ✓ Hablando en voz alta.
- ✓ Mapa de clics.

- ✓ Grabación de uso.

1.6.5 Loop11

Es una herramienta web, se configura fácil y rápidamentel. Permite la realización de *test* de usuarios con un máximo de cinco tareas a desarrollar y dos preguntas. Cuenta con varios idiomas para la interfaz, estos son inglés, español, alemán y chino. Al concluir el *test* de usuario proporciona datos que incluyen la tasa media de éxito o fracaso y el abandono del *test* completo. Sobre cada una de las tareas ofrece indicadores como: promedio de páginas vistas y tiempo medio por tarea, página más frecuente de éxito, fracaso y abandono, el primer clic y la ruta de navegación más frecuente. Además, de los indicadores anteriores, recoge una gran cantidad de datos sobre cada participante, tales como la dirección IP, navegador, fecha del *test*, tiempo total y por tarea del *test*, número total y por tarea de las páginas vistas (Díaz, 2010).

1.6.6 ItalC

ItalC es una herramienta didáctica para los profesores, que permite ver y controlar otras computadoras en su red de varias maneras. Es compatible con los sistemas operativos Linux y Windows XP, Vista y 7. Es gratis y el código fuente está disponible. Permite a los profesores ver lo que está pasando en los laboratorios de informática mediante el modo de visión general. En la pantalla del profesor se muestra las computadoras de todos los alumnos en tiempo real (Morales, 2006).

Es importante dejar claro que esta herramienta no se enfoca al objeto de estudio, pero por las posibilidades que brinda de hacer observaciones de usuarios se considera útil para la concepción del trabajo de diploma.

Se evidencia que las herramientas existentes a las que se tiene acceso solo ejecutan algunas de las técnicas, por ejemplo ItalC y Loop11 tienen la ventaja de ser no privativas, por otra parte solo utilizan grabación de uso y *test* de usuario. La herramienta más completa *UserZoom* es de una consultoría dedicada a prestar servicio. La evaluación heurística solo se hace de forma semiautomática. Por lo general las técnicas que se ejecutan son el *test* de usuario y grabación de uso. La técnica de seguidor de ojo se considera interesante, pero por cuestiones de tecnología se obliga a la no utilización. Sin embargo, el estudio de estas herramientas fue provechoso, ya que se identificaron características comunes que sirven como base para el desarrollo de la herramienta, se realizó una recopilación de información con respecto a la aplicación, identificando además errores y problemas de diseño.

1.7 Metodologías de desarrollo de *software*

Las metodologías de desarrollo de *software* son un conjunto de procedimientos, técnicas, herramientas y soporte documental que ayuda a los desarrolladores a realizar un *software* (Piattini Velthuis, y otros, 2013). Una metodología está formada por fases, cada una se puede dividir en subfases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto. También, a planificarlo, gestionarlo, controlarlo y evaluarlo (Avison, y otros, 1995).

Entre las metodologías tradicionales se encuentran: MSF (por sus siglas en inglés, *Microsoft Solution Framework)*, RUP (por sus siglas en inglés, *Rational Unified Process*) y sus híbridos.

Generalmente las metodologías tradicionales se caracterizan por: requisitos fijados a lo largo de todo el proyecto, basadas en procesos bien documentados y gestión predictiva de los proyectos (Carvajal Riola, 2008). Además, suelen aplicarse a proyectos fundamentados con un desarrollo orientado a objetos y un equipo grande. Con tales características, a pesar de que han demostrado ser efectivas y necesarias en proyectos de gran tamaño (respecto a tiempo y recursos) donde por lo general se obtienen resultados satisfactorios cuando el equipo de trabajo tiene amplia experiencia en su aplicación, este enfoque no resulta ser el más adecuado para proyectos donde el entorno del sistema es muy cambiante y se requiere reducir drásticamente el tiempo de desarrollo manteniendo alta calidad.

Además, del enfoque tradicional y ágil, algunos investigadores proponen metodologías de desarrollo de *software* centradas específicamente en el desarrollo web, entre las que se encuentran: *Navigational Development Techniques* (NDT), *Ubiquituos Web Applications* (UWA), *Object Oriented Hypermedia Design Model* (OOHDM). Estas metodologías presentan algunas deficiencias, por ejemplo: la mayoría de estas propuestas conceden una menor importancia al tratamiento de requisitos, centran su trabajo en las etapas de diseño e implementación, haciendo mucho énfasis en la navegabilidad del usuario y la apariencia visual de la aplicación, no incluyen demasiada documentación y algunas de ellas están concebidas para el uso de paradigma orientado a objetos (Suárez Jorge, 2009).

Por los planteamientos anteriores, las características propias de la solución y el ambiente de desarrollo, dígase cliente, equipo de desarrollo y tiempo de desarrollo, se decide desechar la aplicación de las metodologías anteriores en el proceso de desarrollo y se opta por el uso de una metodología ágil.

1.7.1 Metodologías ágiles

Las metodologías ágiles están especialmente definidas para el desarrollo de proyectos con requisitos poco definidos o cambiantes y son aplicables a equipos pequeños de trabajo que resuelven problemas concretos minimizando los fallos y los costes (Carvajal Riola, 2008). Su objetivo principal es crear un

producto que funcione y cumpla las exigencias del cliente, contando con la colaboración cliente en todo momento, antes que escribir mucha documentación.

Dentro de las metodologías ágiles se incluyen: *Extreme Programming* (XP), SCRUM, *Crystal Methodologies*, *XBreed*, *Adaptive Software Development* (ASD) y *Dynamic Systems Development Method* (DSDM) (Letelier, y otros, 2008).

Para el desarrollo de la presente investigación se analizaron las metodologías ágiles SCRUM y XP debido a que ambas promueven el trabajo en equipo, fomentan la interacción sistemática entre el cliente y el equipo de desarrollo y están suficientemente documentadas. Vale destacar además, que estas permiten la realización de reuniones y entregas al cliente continuamente en cortos períodos de tiempo.

SCRUM

La metodología ágil SCRUM se basa fundamentalmente en entregas iterativas con ciclos de corta duración de veinticuatro a treinta días. A estos ciclos se denomina sprint en los cuales se desarrollan las mejoras de los productos que se hayan planificado (Arellano Moncayo, 2011).

Las fases de SCRUM son: planificación del sprint, seguimiento del sprint y revisión del sprint (ver Figura 3).

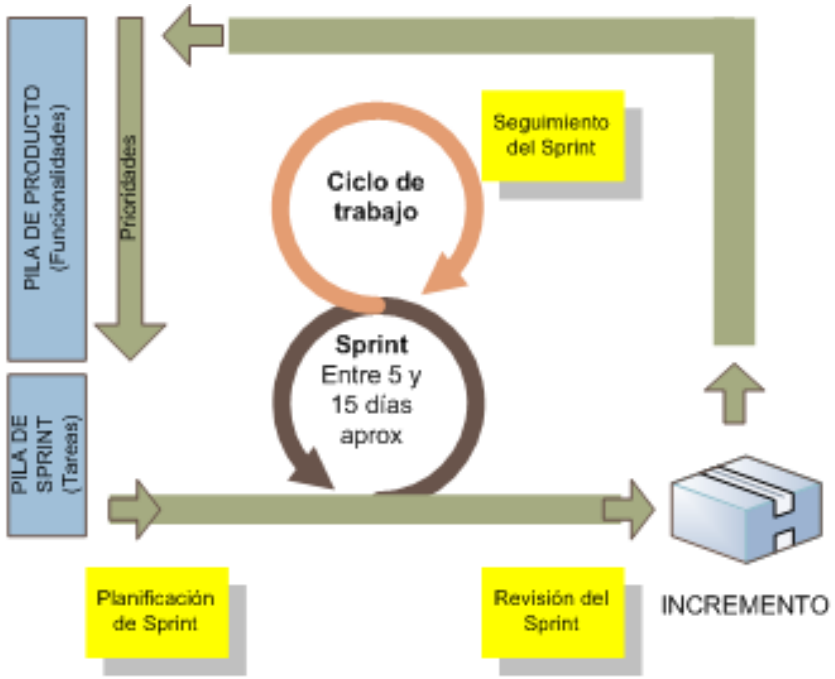

Figura 4. Ciclo de vida de SCRUM.

Es una metodología ágil principalmente indicada para proyectos con un rápido cambio de requisitos (Palacio, 2007), dispone de herramientas para la gestión de cada una de sus fases y es ideal para pequeños equipos de diez o menos miembros; sin embargo no indica y/o provee de ninguna práctica concreta para el desarrollo de *software* (Carvajal Riola, 2008). Una comparativa de las metodologías ágiles SCRUM y XP indica que esta última está más enfocada a presentar diferentes prácticas y la primera a la gestión de proyectos (ver Figura 4). Además, a pesar de las ventajas que pudiera suponer la aplicación de la metodología ágil SCRUM, es necesario señalar que los roles que esta propone, propietario del producto, *Scrum Master*, equipo de desarrollo, cliente y gestor no están visiblemente representados en el presente proyecto (Carvajal Riola, 2008).

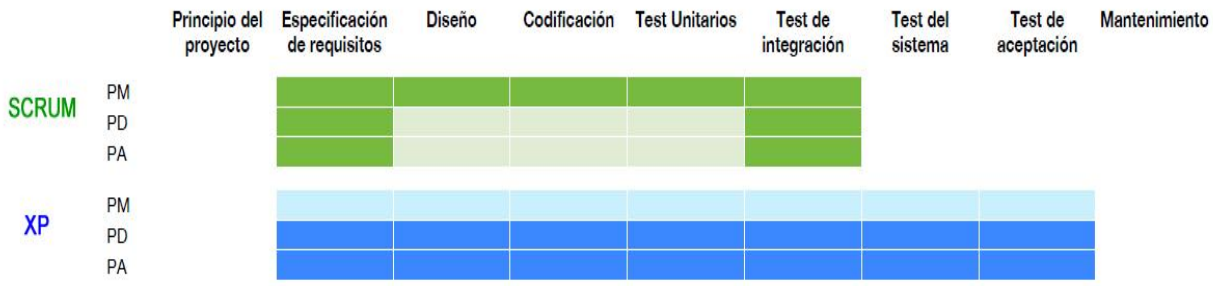

Figura 5. Comparativa entre las metodologías SCRUM y XP.

XP

Es una metodología ágil formulada por Kent Beck en 1996, a partir de un conjunto de principios, prácticas y técnicas que facilitan de manera exitosa la finalización de proyectos, que surgen como respuesta y posible solución a los problemas derivados del constante cambio en los requerimientos (Beck, 1999). Centrada en potenciar las relaciones interpersonales como clave imprescindible para el éxito en el desarrollo de *software*, la metodología XP promueve el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo (Letelier, y otros, 2008).

Una de las prácticas más significativas que posee la metodología XP es la programación en pares, que proclama que dos personas escriban código en el mismo ordenador, lo que posibilita que parcialmente se puedan evitar y/o reducir errores y malos diseños en la creación del código, pues de esta forma se controla cada línea de código y decisión del diseño instantáneamente. Además, la buena conexión entre ambos desarrolladores generará discusiones que sin duda conducirá a mejores estructuras y algoritmos, aumentando así la calidad del producto.

XP no es un conjunto de reglas a seguir (ver Figura 5), con suficientemente eficiencia probada, que definen una forma de trabajar en armonía con los valores personales y organizacionales, se tiene su punto de partida en cinco valores fundamentales que debe cumplir el equipo de desarrollo (Shore, y otros, 2008):

- ✓ **Simplicidad**: se simplifica el diseño para agilizar el desarrollo, facilitar el mantenimiento y descartar las ideas que realmente no necesitemos.
- ✓ **Comunicación**: la comunicación con el cliente es fluida ya que forma parte del equipo de desarrollo.
- ✓ **Retroalimentación (*feedback*)**: ejecutar las pruebas unitarias frecuentemente permite descubrir fallos debido a cambios recientes en el código.
- ✓ **Coraje**: el equipo de desarrollo debe estar preparado para enfrentarse a los continuos cambios que se presentarán en el transcurso de la vida del proyecto.
- ✓ **Respeto**: los miembros del equipo de desarrollo deben tratarse entre ellos y los demás con dignidad, reconociendo la experiencia de todos y su deseo de éxito.

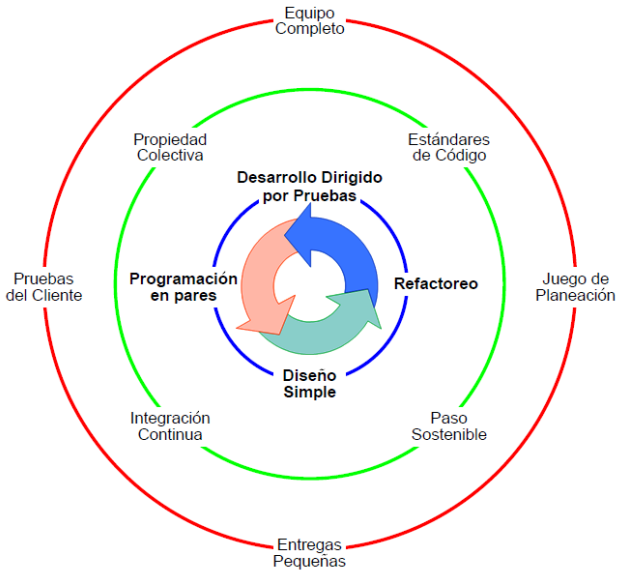

Figura 6. Prácticas comunes de la metodología de desarrollo XP.

El ciclo de vida de la metodología XP está compuesto de cinco fases:

- ✓ **Exploración:** donde los clientes escriben las historias de usuario (HU) que quieren que sean incluidas en la primera versión, describen las funcionalidades que serán añadidas al sistema.
- ✓ **Planificación:** donde se establece la prioridad de las HU y se acuerda el contenido de la primera entrega del proyecto, así como su estimación temporal.

- ✓ **Diseño:** en esta fase se logrará crear la parte física del proyecto, la interfaz, donde se llevan a cabo un conjunto de pruebas extras, de rendimiento y funcionamiento que son necesarias antes de entregar el producto.
- ✓ **Implementación (Codificación):** en esta fase está incluido todo lo de codificación o programación por parte de los desarrolladores del proyecto.
- ✓ **Pruebas:** se realizan pruebas de aceptación y unitarias para verificar que las necesidades del cliente han sido satisfechas.

XP es difícil de implementar de una sola vez y es recomendable ir escogiendo con cuidado que prácticas aplica a un determinado proyecto (Beck, 1999). Las iteraciones son relativamente cortas, entre más rápido se entregue un resultado al cliente más retroalimentación se obtiene y esto va a representar mejor calidad del producto a largo plazo. De manera general, esta última característica es relevante y resulta conveniente para el desarrollo de la presente solución.

Metodología seleccionada

Luego del análisis anterior de las metodologías ágiles para el desarrollo del *software* y se tiene en cuenta que la metodología ágil SCRUM propone los roles de propietario del producto, Scrum Master, equipo de desarrollo, cliente y gestor no están visiblemente representados en el presente proyecto, además el equipo de desarrollo es pequeño y existe disponibilidad del cliente, se decide seleccionar XP pues esta metodología propone:

- ✓ Una estructura de roles adaptable al equipo de desarrollo.
- ✓ Frecuente comunicación entre equipo de desarrollo y cliente.
- ✓ Ambiente de desarrollo basado en un único local y ordenador utilizando espacios abiertos
- ✓ Programación por pares.
- ✓ Producción de pequeñas entregas funcionales para mostrar un resultado rápido.
- ✓ Integración continua.
- ✓ Empleo de estándares de codificación.
- ✓ Propiedad colectiva del código.
- ✓ Horario de trabajo de un máximo de 40 horas por semana sin trabajar horas extras.

Estas características se ajustan a las condiciones reales del presente proyecto. Además, es necesario señalar que durante el desarrollo de la herramienta se trabajó directamente con el cliente; por tanto, en aras de continuar una única línea para generar la documentación asociada se decide aplicar la misma

metodología porque el uso de otras podría traer inconveniencias en el cronograma definido por el proyecto.

1.8 Lenguajes de programación

Un lenguaje de programación "*es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes*" (Eguíluz Pérez, 2008).

En el desarrollo de aplicaciones web, los lenguajes de programación se suelen clasificar como: lenguajes del lado del cliente y del lado del servidor.

1.8.1 Lenguajes y tecnologías del lado del cliente

Las tecnologías del lado del cliente son las que se ejecutan en el navegador del usuario, son las páginas dinámicas que se procesan en el cliente. En estas páginas, toda la carga del procesamiento de los efectos y funcionalidades la soporta el navegador (Padilla Matos, 2008).

JavaScript

El lenguaje de programación JavaScript es utilizado para crear programas encargados de realizar acciones dentro del ámbito de una página web. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas, así como las interactividades con el usuario.

Técnicamente es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Pérez Eguíluz, 2009). Actualmente es utilizado por millones de páginas web para validar formularios, crear *cookies*, detectar navegadores y mejorar el diseño, su fácil aprendizaje lo hace un lenguaje muy demandado.

Se usará el lenguaje JavaScript en su versión v1.8.5.

ExtJs

El *framework* de JavaScript ExtJs permite desarrollar potentes aplicaciones, está basado en JavaScript orientado a objetos. Entre las funcionalidades que más destacan se pueden mencionar, la creación de formularios, validaciones de forma sencilla y rápida, creación de gráficos estadísticos con buena presentación para el usuario, listas llamadas grillas con paginación, ordenamiento y filtro (Soto, 2012).

Esta librería de Javascript permite construir aplicaciones complejas en Internet (Walker, 2015) e incluye lo siguiente:

- ✓ Componentes interfaz de usuario de alto performance y personalizables.
- ✓ Modelo de componentes extensibles.
- ✓ Un API (por sus siglas del inglés: *Application Programming Interface*) fácil de usar.
- ✓ Licencias Open *source* y comerciales.

Algunas de las principales ventajas que propone la utilización de ExtJs son:

- ✓ Permite crear aplicaciones complejas utilizando componentes predefinidos.
- ✓ La ventana flotante que provee es excelente por la forma en la que funciona. Su funcionamiento lo pone por encima de cualquier otro.
- ✓ Existe un balance entre cliente–servidor la carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- ✓ Comunicación asíncrona, en este tipo de aplicación el motor de *render* puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente pueda notarlo.
- ✓ Eficiencia de la red, disminuye el tráfico en la red pues las aplicaciones cuentan con las posibilidades de elegir que datos desea trasmitir al servidor y viceversa.

Por todas las posibilidades anteriormente expuestas que brinda la librería ExtJs se decide emplear en su última versión 4 para el desarrollo y diseño de la propuesta.

CSS

Las hojas de estilo en cascada (CSS) se define según Rivera como "…*un lenguaje de hojas de estilo creado para controlar la presentación de los documentos*..." (Rivera, y otros, 2012), se utilizan para darle forma y aplicar diseño o presentación al contenido o estructura HTML y XML que forma una página o aplicación web. Es la mejor forma de separar contenidos de presentación y es imprescindible para la creación de páginas web complejas.

Permite realizar cambios a múltiples elementos dentro del código, agilizando considerablemente así el proceso de cambios. Este describe como se muestra un documento en pantalla, por impresora, por voz (cuando la información es pronunciada a través de un dispositivo de lectura) o en dispositivos táctiles basados en Braille. Este lenguaje se basa en una serie de reglas que rigen el estilo de los elementos en los documentos estructurados y que forman la sintaxis de las hojas de estilo (De Luca, y otros, 2012).

En los últimos años el código CSS ha evolucionado hasta su tercera revisión, la cual incluye propiedades que permiten alcanzar efectos complejos, facilitando la aplicación, edición y actualización de formato en todo un sitio web.

Se usará el lenguaje CSS en su versión v3.0.

1.8.2 Lenguajes del lado del servidor

Los lenguajes del lado del servidor son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él.

PHP

PHP (*Hipertext Preprocessor*) es un lenguaje del lado del servidor para la generación de HTML. Es un lenguaje interpretado de alto nivel embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl, con solamente un par de características PHP específicas. La meta de este lenguaje es permitir escribir a los creadores de páginas web, paginas dinámicas de una manera rápida y fácil.

Lo que distingue a PHP de JavaScript, es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá los resultados de ejecutar el script, sin ninguna posibilidad de determinar el código que ha producido el resultado recibido. El servidor web puede ser incluso configurado para que procese todos los archivos HTML con PHP.

Actualmente es utilizado como un poderoso lenguaje para extraer información almacenada en base de datos, aplicar reglas del negocio, instrucciones para temas y eventualmente crear HTML que será enviado al servidor (Travis, 2011).

Este lenguaje es gratuito e independiente de plataforma, rápido, con una gran librería de funciones y abundante documentación. El tiempo de aprendizaje es corto gracias a su potencial y simplicidad (Converse, y otros, 2004). Sus principales características son: rapidez, puede ser utilizado en diversos sistemas operativos, servidores web y la característica más potente y destacable es su soporte para una gran cantidad de bases de datos (KadaSoftware, 2013). Ha evolucionado como lenguaje orientado a objetos permitiendo incluir nuevas funcionalidades, mejoras y reutilización de código en sitios web (Travis, 2011).

Se usará el lenguaje PHP en su versión v5.5.

1.9 Framework

Los *frameworks* son un conjunto de librerías o funciones que se pueden emplear en el desarrollo de las aplicaciones. Se puede decir que son de fácil uso puesto hace abstracciones de complejas y básicas

implementaciones para emplearlas de una forma más sencilla y organizada. Entre los más usado actualmente se encuentran

Symfony

Ha sido ideado para exprimir al límite 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 (López, 2008). Es uno de los *frameworks* más flexibles que existen. Se han adaptado todas esas técnicas al mundo PHP y los resultados han sido increíbles. Cuenta con un micro-*kernel* altamente optimizado. El núcleo es la pieza central del *framework* y es el responsable de inicializar la configuración de la aplicación y arrancar (*bootear*) los *bundles* (paquetes, *plugins*) (López, 2008).

Symfony2 está construido utilizando un contenedor de inyección de dependencias, inspirado en el *framework* Spring de Java. En un proyecto, el desarrollador no interactúa directamente con el contenedor. Todos los detalles de implementación están ocultos detrás de un buen sistema de configuración que permite personalizar todo a través de archivos YAML o XML, o incluso a través de código PHP (etnassoft.com, 2012).

CodeIgniter

Es un *framework* para el desarrollo de aplicaciones que posee un conjunto de herramientas para construir sitios web usando PHP. Su objetivo es permitir al desarrollar proyectos rápidos, proporcionándole un conjunto de bibliotecas para tareas comunes, así como una interfaz sencilla y una estructura lógica para acceder a estas bibliotecas. CodeIgniter posibilita enfocarse creativamente en un proyecto al minimizar la cantidad de códigos necesarios para una tarea dada (autores, 2013). Tiene como características principales que:

- ✓ Es un framework con una huella pequeña.
- ✓ Rendimiento excepcional.
- ✓ Necesita una amplia compatibilidad con las cuentas estándar de hospedaje que usan una gran variedad de versiones y configuraciones de PHP.
- ✓ No necesita casi configuración.
- ✓ No requiere del uso de línea de comandos.
- ✓ No se necesita aprender un lenguaje de plantillas.
- ✓ Evita la complejidad, favoreciendo soluciones simples.
- ✓ Necesita una clara y completa documentación.

Framework seleccionado

Después de realizar un estudio de estos dos framework se selecciona Symfony para llevar a cabo el desarrollo de la aplicación. Este presenta características que influyen en su selección por encima de CodeIgniter como la facilidad de aprendizaje, el trabajo con los formularios y las validaciones. Aporta sub-*framework* para el trabajo con el enrutamiento y las pruebas, además agrega un alto nivel de seguridad a las aplicaciones. Es fácil de instalar y configurar en cualquier plataforma. Libera a los desarrolladores de la tarea de crear funcionalidades menores y en ocasiones aburridas de implementar. Las aplicaciones desarrolladas con Symfony son compatibles con la mayoría de las plataformas, bibliotecas e infraestructuras que existen. Se adaptan a entornos de negocio en cambio permanente, requiere menos esfuerzo para su mantenimiento. Es fácil de extender por tratarse de proyectos *open source*, se le pueden agregar nuevas funciones desarrolladas por programadores externos.

Promueve el uso de buenas prácticas de programación y genera código fácilmente comprensible por el desarrollador. Si alguna vez se encuentran dificultades, se podrá contar con la colaboración de una comunidad de cientos de miles de programadores y con la seguridad de que cualquier posible defecto será corregido en versiones posteriores.

Por lo antes mencionado se escoge el empleo de Symfony en su versión 2.6.3 para la solución de la aplicación.

1.10 Sistema gestor de base de datos

El estudio de la bibliografía consultada indica que un SGBD (*Data Base Managment System* en inglés, abreviado como DBMS) es un sistema de *software* que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos (Asenjo Sánchez, 2009). En estos sistemas se proporciona un conjunto de funcionalidades, procedimientos y lenguajes que permiten a los distintos usuarios realizar tareas habituales con los datos, garantizando la seguridad de los mismos.

Symfony soporta la integración con diferentes SGBD como: *MySQL*, *SQLite*, *PostgreSQL* y *Oracle*. Esto se consigue mediante una capa de abstracción de la base de datos que convierte las instrucciones genéricas proporcionadas por Symfony en instrucciones particulares de cada base de datos. Es por esta razón que de forma convencional, se decide analizar aquí las posibles ventajas que brinda el SGBD MySQL 5.5, que actualmente soporta el almacenamiento de datos en la herramienta.

MySQL

MySQL es un SGBD relacional y multiusuario con más de seis millones de instalaciones. Actualmente es uno de los mejores sistemas de administración de base de datos y cuenta con mejoras constantes de parte de desarrolladores a nivel mundial. Es un programa capaz de almacenar una enorme cantidad de datos de gran variedad y de distribuirlos para cubrir las necesidades de cualquier tipo de organización, convirtiéndose en una solución altamente viable para la administración de datos (Gilfillan, 2003).

El SGBD MySQL como solución para la administración de datos de la herramienta involucra:

- ✓ **Velocidad**: es mucho más rápido que la mayor parte de sus rivales.
- ✓ **Funcionalidad**: dispone de muchas de las funciones que exigen la mayor parte de los desarrolladores profesionales. Así mismo, se desarrolla y actualiza de forma mucho más rápida.
- ✓ **Portabilidad**: se ejecuta en la inmensa mayoría de sistemas operativos y en la mayor parte de los casos los datos se pueden transferir de un sistema a otro sin dificultad.
- ✓ **Facilidad de uso**: resulta fácil de utilizar y de administrar. Las herramientas de MySQL son potentes y flexibles, sin sacrificar su capacidad de uso.

1.11 Entorno de desarrollo integrado

En el análisis de la bibliografía consultada se coincide en que un Entorno de Desarrollo Integrado o IDE (*Integrated Development Environment* en inglés) es un programa informático que agrupa diversas herramientas de programación para facilitar las tareas al programador y obtener mayor rapidez en el desarrollo. En otras palabras, es un sistema que facilita el trabajo del desarrollador de *software*, integrando sólidamente la edición orientada al lenguaje, la compilación o interpretación, la depuración, las medidas de rendimiento, la incorporación de las fuentes a un sistema de control de fuentes, normalmente de forma modular (González Barahona, y otros, 2003). Un buen IDE debe incluir las siguientes características:

- ✓ Multiplataforma.
- ✓ Soporte para diversos lenguajes de programación.
- ✓ Integración con sistemas de control de versiones.
- ✓ Reconocimiento de sintaxis.
- ✓ Extensiones y componentes para el IDE.
- ✓ Integración con *framework* populares.
- ✓ Depurador.
- ✓ Importar y exportar proyectos.

- ✓ Múltiples idiomas.
- ✓ Manual de usuarios y ayuda.

Entre los ejemplos de IDE más utilizados se encuentran: NetBeans, Zend Studio y PHPStorm, los cuales se analizan a continuación:

1.11.1 Zend Studio

Zend Studio es uno de los IDE que agrupa todos los componentes de desarrollo necesarios para el ciclo de desarrollo de aplicaciones PHP, disponible para desarrolladores profesionales.

Incluye un conjunto de herramientas de edición, depurado, análisis y optimización, lo que permite simplificar los proyectos complejos. Proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código. Consta de dos partes: las funcionalidades de parte del cliente (contiene interfaz de edición y ayuda) y las del servidor (instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración). A pesar de todas estas ventajas se distribuye a través de una licencia comercial.

1.11.2 NetBeans 7.3

NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrita en Java, aunque permite programar en diferentes lenguajes (NETBEANS.ORG, 2011). Es una base sólida para desarrollar aplicaciones complejas con un enfoque modular y posee características como la extensibilidad y la escalabilidad. Posee un número importante de módulos para extenderlo permitiendo ampliar y facilitar su utilización. Es un producto libre y gratuito sin restricciones de uso.

Principales características:

- ✓ Creación de proyectos PHP.
- ✓ Integración con Symfony y ZenFramework.
- ✓ Editor de código fuente.
- ✓ Depuración de PHP.
- ✓ Integración con sistemas de control de versiones.

1.11.3 PHPStorm

PHPStorm en su versión 8 es el IDE preferido por la mayoría de programadores Symfony. Al margen de todas las características habituales en los entornos de desarrollo. Es el único que ofrece una integración casi perfecta con Symfony gracias a su plugin para Symfony2 (symfony.es, 2014).

A continuación se resaltan otras de las características de esta herramienta (Código, 2014):

- ✓ Compatibilidad.
- ✓ Permite la gestión de proyectos fácilmente.
- ✓ Proporciona un fácil autocompletado de código.
- ✓ Soporta el trabajo con PHP 5.5.
- ✓ Sintaxis abreviada.

Este IDE es uno de los entornos de programación más completos de la actualidad, permite editar código no solo del lenguaje de programación PHP como lo indica su nombre. Actualmente es compatible con Sistemas Operativos Windows, Linux y Mac OS X. Algo que destaca en PhpStorm es la ejecución del código en la misma interfaz del IDE. Así como también, la interpretación y visualización inmediata de código .php hasta en cinco de los navegadores web más populares. Con esto no habrá que cargar manualmente el navegador e ingresar la dirección url de la página.

1.11.4 IDE seleccionado

Para el desarrollo de la propuesta de solución se selecciona PHPStorm 8, además de las características anteriormente mencionadas ofrece un formato de código acorde a los estándares de código de Symfony. Por otra parte permite la integración con *frameworks* como Symfony de PHP y JQuery de JavaScript. También, es un IDE totalmente libre y presenta una comunidad de desarrollo que ayuda al trabajo de los desarrolladores de aplicaciones.

1.12 Servidor web

Un servidor web es un programa que se ejecuta continuamente en un ordenador, manteniéndose a la espera de peticiones por parte de un cliente y responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error (Duque González, 2011).

Para su correcta ejecución, una aplicación en Symfony necesita alojarse en un servidor web. Según Travis, los servidores web recomendados para Symfony son IIS (*Internet Information Server*) y Apache (Travis, 2011). Se decide que esta última sea el servidor usado para el desarrollo de la herramienta porque presenta licencia de *software* libre y de código abierto siendo IIS propietario. Apache es

multiplataforma a diferencia de IIS, que funciona solo sobre la plataforma Windows, lo que es una limitante. Además, soporta PHP a diferencia de IIS (NETCRAFT, 2011).

1.12.1 Apache

El servidor Apache es un *software* de código abierto para plataformas Unix (BSD, GNU/Linux), Microsoft Windows y Macintosh. Es un potente servidor web de *software* libre cuyo objetivo es servir o suministrar páginas web a los clientes o navegadores que las solicitan.

Entre las principales características de este potente servidor web, se destacan las siguientes:

- ✓ Apache se ha concentrado en la escalabilidad, en la seguridad y en el rendimiento.
- ✓ Es un servidor web altamente configurable.
- ✓ Plataforma de servidores web de código fuente abierto.
- ✓ Trabaja con diversas tecnologías como PHP, *mod_perl* y *servlets* de Java.

Se seleccionó como servidor web Apache en su versión 2.2.22 debido a que es una tecnología gratuita, de código abierto, coincidiendo con Kabir cuando refiere que "…*es la plataforma de servidores web de código fuente abierto más poderosa en el mundo*…" (Kabir, 1999). Además, es compatible con el Symfony, pues se aplica este *framework* para la administración del sistema. Se considera que es la solución perfecta para la mayor parte de las aplicaciones por esto se utilizará en la herramienta a desarrollar.

Conclusiones

Luego del estudio y análisis realizado del objeto de investigación del presente trabajo de diploma, apoyado en los métodos de la investigación científicos definidos, se concluye lo siguiente:

- ✓ Se construyó el marco conceptual que soporta la investigación que permitió adquirir el conocimiento necesario sobre las metodologías, herramientas y lenguajes, seleccionándose las más adecuadas para el cumplimiento del objetivo general propuesto.
- ✓ Se identificó la metodología XP como guía del proceso de desarrollo, pues provee un marco teórico integrado por principios, prácticas y técnicas concretas que se ajustan al entorno y/o condiciones del proyecto y tienen el potencial de acelerar el tiempo de desarrollo.
- ✓ La integración de Symfony 2.6.3 soportado por los lenguajes del lado del cliente XHTML y CSS 3 y del lado del servidor PHP 5.3.5, así como del SGBD MySQL 5.5, el IDE PHPStorm 8 y el servidor web Apache 2.2.22; guiados por la metodología XP, permite continuar parcialmente con la misma línea inicial de desarrollo del proyecto y al mismo tiempo constituye una base sólida para el desarrollo de la solución.

CAPÍTULO 2: DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN

Introducción

Al terminar el estado del arte se hace necesario tener claridad del sistema que se desarrollará y para esto se mostrará una propuesta de solución a la problemática planteada en el presente trabajo de diploma, es necesario analizar sus funcionalidades y elementos principales; basado en la uso de la metodología ágil XP, de la que se incluyen algunos de sus artefactos: la metáfora (espacio donde los usuarios de la aplicación puedan de forma sencilla, acceder y evaluar la usabilidad de los productos creados en el centro), las historias de usuario, tarjetas Clases Responsabilidad Colaborador (CRC). Además, en este capítulo se determinan los servicios que debe brindar la solución y se describen las funcionalidades a través de la exposición de los principales artefactos generados por la metodología en las fases de exploración, planificación y diseño.

2.1 Propuesta de solución

Como solución a la problemática descrita en la introducción del presente trabajo se propone la implementación de un sistema que permitirá informatizar el procedimiento de las pruebas de usabilidad en el centro FORTES.

Esta herramienta constará de dos tipos de usuarios, uno administrador y otro usuario del sistema. El administrador tiene acceso a todos los servicios que prestará la aplicación. Es el que gestionará los usuarios que harán uso de la herramienta, para ello tendrá que insertar sus datos y asignarle el rol deseado dentro de la misma.

También, podrá insertar la lista de chequeo con la cual se realizarán las pruebas, para esto tendrá que incluir las categorías por las que está compuesta y a su vez las preguntas asociadas a cada categoría. Además, tendrá la opción de adicionar, eliminar o modificar alguna categoría o respuesta cuando lo desee. Otras de las acciones que podrá realizar son adicionar, eliminar o modificar los proyectos a los cuales se le realizarán las pruebas, teniendo la posibilidad de asignarle a cada proyecto los usuarios deseados, así como las categorías que estime convenientes con sus respectivas preguntas.

El usuario del sistema podrá realizar las pruebas al proyecto que le fue asignado respondiendo a las preguntas de las categorías que fueron asociadas, las preguntas se podrán responder de forma literal, cuantitativa o cualitativamente y se podrá dar una breve descripción del motivo de la respuesta, teniendo la posibilidad de agregar todas no conformidades que se encuentren.

Una vez terminadas las pruebas el administrador tendrá la posibilidad de ver un informe de las respuestas por preguntas de cada usuario y generará un archivo en formato .pdf con esta información, además podrá ver todas las no conformidades que fueron encontradas por los usuarios.

2.2 Usuarios del sistema

Los usuarios del sistema son personas que tienen permiso para acceder e interactuar con la aplicación y hacer uso de sus servicios. En estos se implementan diferentes roles o niveles y de ellos dependen los privilegios que se tendrán en la aplicación. En la presente investigación se definieron los siguientes usuarios:

Tabla 1. Usuarios relacionados con el sistema y su descripción.

Usuarios	Justificación
Usuario autenticado	Es la persona que se encuentra autenticada en la aplicación cumpliendo este rol, además tendrá acceso a diferentes módulos y componentes debido al rol que cumple dentro de la aplicación.
Administrador	Es la persona autorizada para la gestión del sistema, encargado de actualizar, modificar, eliminar e insertar toda la información de la aplicación. Dispone de posibilidades ilimitadas para ejecutar todas las funciones administrativas del sistema.

2.3 Planificación

En esta fase es donde los clientes escriben las HU que quieren que sean incluidas en la primera versión, que describen las funcionalidades que serán añadidas al sistema; se establece la prioridad de las HU y se acuerda el contenido de la primera entrega del proyecto, así como su estimación temporal.

2.3.1 Historias de usuario

Las historias de usuario (HU) es la técnica utilizada para especificar los requisitos y características del sistema. Las escribe el cliente en su propio lenguaje, son cortas descripciones de lo que el sistema debe realizar (Beck, y otros, 2005). El tratamiento de las HU es flexible, siendo lo suficientemente comprensibles y delimitadas para que los programadores puedan implementarlas en pocas semanas.

La información de una HU puede variar y ajustarse a las características específicas del proyecto. Por ejemplo, existen varias plantillas sugeridas para representarlas, pero no un consenso al respecto. A continuación se describen aspectos esenciales que debe incluir cada una de ellas en cualquier proyecto de desarrollo (Beck, 1999):

- ✓ **Número**: posee el número asignado a la HU.
- ✓ **Nombre de HU**: atributo que contiene el nombre.
- ✓ **Usuario**: es el usuario del sistema que utiliza o protagoniza el requisito que se describe.
- ✓ **Prioridad en el negocio**: evidencia el nivel de prioridad de la HU en el negocio.
- ✓ **Riesgo de desarrollo**: evidencia el nivel de riesgo en caso de no realizarse.
- ✓ **Puntos estimados**: atributo que contiene la estimación hecha por el equipo de desarrollo del tiempo de duración de la HU. Cuando el valor es uno equivale a una semana ideal de trabajo. En la metodología XP está definida una semana ideal como cinco días hábiles, trabajando cuarenta horas semanales.
- ✓ **Iteración asignada**: precisa la iteración en la que será desarrollada la HU.
- ✓ **Descripción**: posee una breve descripción de lo que realizará la HU.
- ✓ **Observaciones**: Brinda información extra que se estime agregar para hacer más comprensible la HU.

A continuación se muestran ejemplos de HU generados, que representan requisitos que serán implementados (Ver anexo I).

Tabla 2 HU Gestionar pregunta.

Historia de Usuario	
No: 2	Nombre: Gestionar pregunta
Usuario: Administrador	
Prioridad en el negocio: Alta	Puntos Estimados: 3
Riesgo de desarrollo: Alta	Iteración asignada: 1
Descripción: Se podrá adicionar, modificar y eliminar las preguntas.	
Observaciones: El usuario debe estar autenticado para realizar la acción.	

Tabla 3 HU Gestionar proyecto.

Historia de Usuario	
No: 3	Nombre: Gestionar proyecto
Usuario: Usuario autenticado	
Prioridad en el negocio: Media	Puntos Estimados: 2
Riesgo de desarrollo: Alta	Iteración asignada: 2
Descripción: Se podrá adicionar, modificar y eliminar el proyecto.	
Observaciones: El usuario debe estar autenticado para realizar la acción.	

Tabla 4. HU Gestionar categoría.

Historia de Usuario	
No: 1	Nombre: Gestionar categoría
Usuario: Usuario autenticado	
Prioridad en el negocio: Alta	Puntos Estimados: 2
Riesgo de desarrollo: Alta	Iteración asignada: 1
Descripción: brinda la posibilidad al usuario de adicionar, modificar y eliminar una categoría a la aplicación.	
Observaciones:	

2.3.2 Estimación de esfuerzos por HU

Las estimaciones de esfuerzo asociado a la implementación de las HU las establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las HU generalmente valen de uno a tres puntos. Por otra parte, los clientes deciden sobre el ámbito, tiempo de las entregas y de cada iteración (Letelier, y otros, 2008).

Tabla 5. Estimación de esfuerzos por HU.

No	Historia de Usuario	Estimación (semanas)
1	Gestionar categoría	1

2	Gestionar pregunta	1
3	Gestionar proyecto	2
4	Gestionar usuario	2
5	Gestionar respuesta	1
6	Adicionar no conformidad	1
7	Eliminar no conformidad	1
8	Autenticar usuario	1
9	Responder pregunta	1
10	Mostrar informe	1
11	Asignar proyecto a usuario	1

La estimación total del esfuerzo fue de 13 semanas.

2.3.3 Plan de iteraciones

En la metodología XP, la duración ideal de una iteración está entre una y tres semanas. Para estas se establecen un conjunto de HU definidas que serán implementadas en cada iteración, superando las pruebas de aceptación las cuales son realizadas al final de cada ciclo (Bajo de Luque, y otros, 2011).

Iteración 1: en la primera iteración se entregarán las funcionalidades que tienen prioridad alta para el cliente correspondiendo a las HU 1, 2 y 3, las cuales son:

Gestionar categoría

Gestionar pregunta

Gestionar proyecto

Iteración 2: en la segunda iteración se realizarán las restantes HU que se encargan de gestionar los usuarios y las no conformidades en el sistema siendo estas: las 4, 5, 6, 7 y 8, las cuales tendrán como funcionalidad:

Gestionar usuario

Gestionar respuesta

Adicionar no conformidad

Eliminar no conformidad

Iteración 3: en esta iteración se implementan las HU de baja prioridad para el cliente pero no menos importante que las anteriores para los desarrolladores. Las HU son:

Autenticar usuario

Responder pregunta

Mostrar informe

Asignar proyecto a usuario

2.3.4 Plan de duración de las iteraciones

El plan de duración de las iteraciones se realiza luego de tener el estimado en días que demora implementar cada HU. Se tendrá en cuenta además, la prioridad que el cliente le asigna a cada historia y el nivel de complejidad que estas poseen.

Tabla 6. Plan de duración de las iteraciones.

Iteración	Orden de las HU	Duración total
1	Gestionar categoría Gestionar pregunta Gestionar proyecto	4 semanas
2	Gestionar usuario Gestionar respuesta Adicionar no conformidad Eliminar no conformidad	5 semanas
3	Autenticar usuario Responder pregunta Mostrar informe Asignar proyecto a usuario	4 semana

2.3.5 Plan de entregas

El cronograma de entregas establece las HU que serán agrupadas para conformar una entrega y el orden de las mismas. El cliente es quien las agrupa según su prioridad. El cronograma de entregas se analiza sobre la base de las estimaciones de tiempos de desarrollo (Bajo de Luque, y otros, 2011).

Tabla 7. Plan de entregas.

Historia de usuario	Primera iteración	Segunda iteración	Tercera iteración
Gestionar categoría	V 1.0	Finalizado	Finalizado
Gestionar pregunta	V 1.0	Finalizado	Finalizado
Gestionar proyecto	V 1.0	Finalizado	Finalizado
Modificar categoría	-	V 1.0	Finalizado
Gestionar usuario	-	V 1.0	Finalizado
Gestionar respuesta	-	V 1.0	Finalizado
Adicionar no conformidad	-	V 1.0	Finalizado
Eliminar no conformidad	-	V 1.0	Finalizado
Autenticar usuario	-	-	V 1.0
Responder pregunta	-	-	V 1.0
Mostrar informe	-	-	V 1.0
Asignar proyecto a usuario	-	-	V 1.0

2.4 Modelo de diseño

El patrón arquitectónico Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de *software* encargado de separar la lógica de negocio de la interfaz del usuario y es el más utilizado en aplicaciones web. Facilita la funcionalidad, estabilidad y escalabilidad del sistema, de forma simple y sencilla, a la vez que permite no mezclar lenguajes de programación en el mismo código. MVC divide las aplicaciones en tres niveles de abstracción:

- ✓ **Modelo:** representa la lógica de negocios. Es el encargado de acceder de forma directa a los datos actuando como intermediario con la base de datos.
- ✓ **Vista:** es la encargada de mostrar la información al usuario de forma gráfica.

- ✓ **Controlador:** es el intermediario entre la vista y el modelo. Controla las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista para que esta lo presente al usuario.

A consecuencia de la aplicación del patrón arquitectónico Modelo-Vista-Controlador (MVC), Symfony propone un modelo de diseño que por lo general se rigen las aplicaciones que se desarrolla bajo su uso (Ver figura 6).

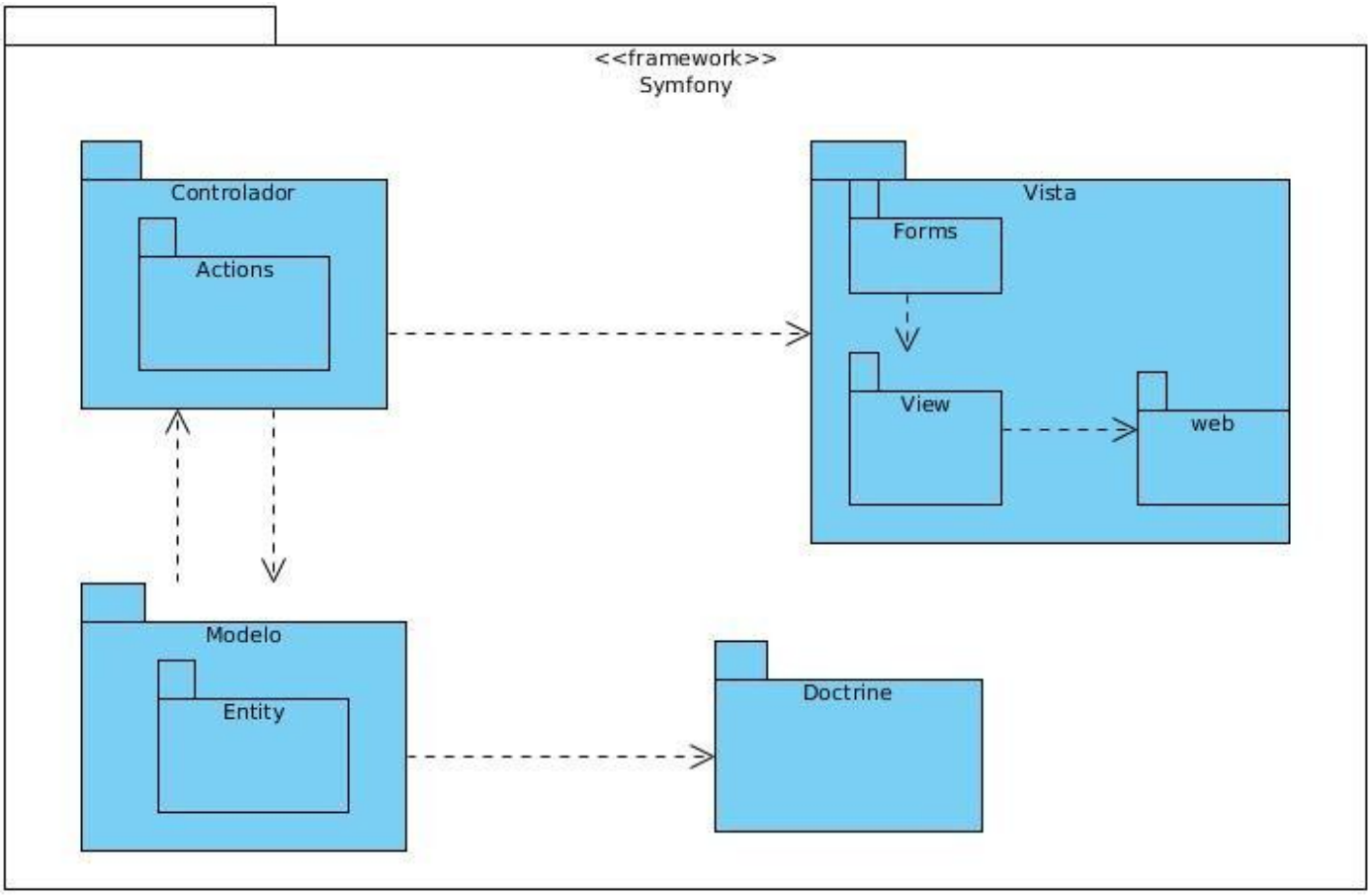

Figura 7. Modelo de Diseño basado en el framework Symfony.

2.5 Prototipo de interfaz de usuario

Los prototipos de interfaz de usuario se pueden utilizar para explorar un diseño de interfaz de usuario alcanzable y adecuado que cumpla los requisitos, ayudando a reducir las distancias entre lo que es necesario (expresado a través de la adquisición de requisitos) y lo que es factible. El objetivo principal de la creación de un prototipo de interfaz de usuario es poder probar el diseño de interfaz de usuario, incluyendo la capacidad de utilización antes de que empiece el desarrollo real. De este modo, se puede

garantizar que se está construyendo el sistema correcto, antes de dedicar demasiado tiempo y recursos al desarrollo (autores, 2006).

Para mostrar una vista preliminar de la solución se crearon los prototipos de interfaz (Ver Figura 7) que básicamente incluye las características definidas de la propuesta de solución, es de fácil modificación a partir de la retroalimentación del cliente. Se inicia desde la interfaz inicial que presenta la herramienta.

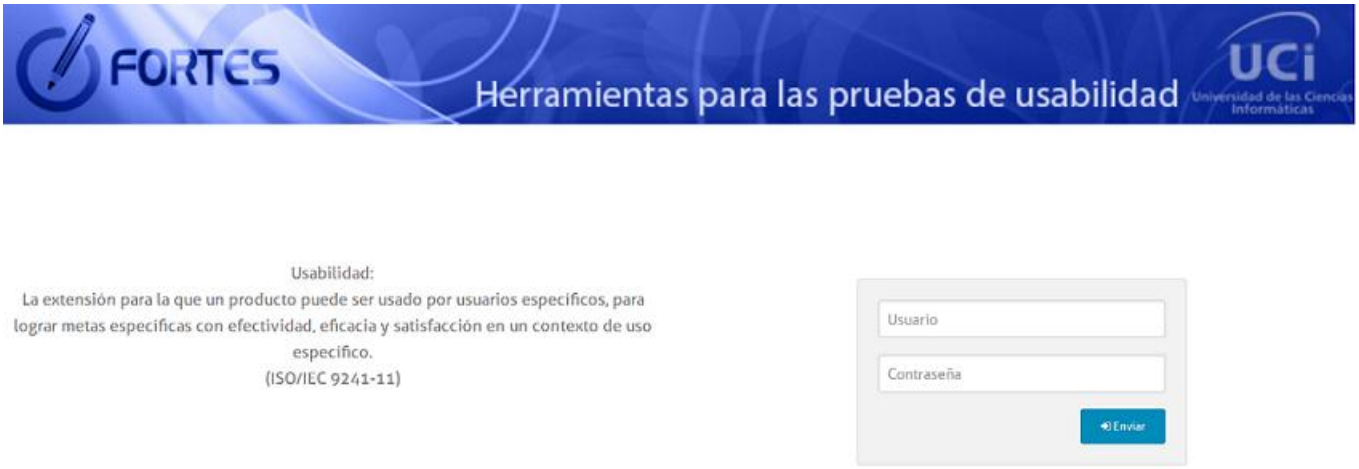

Figura 8. Prototipo de interfaz principal.

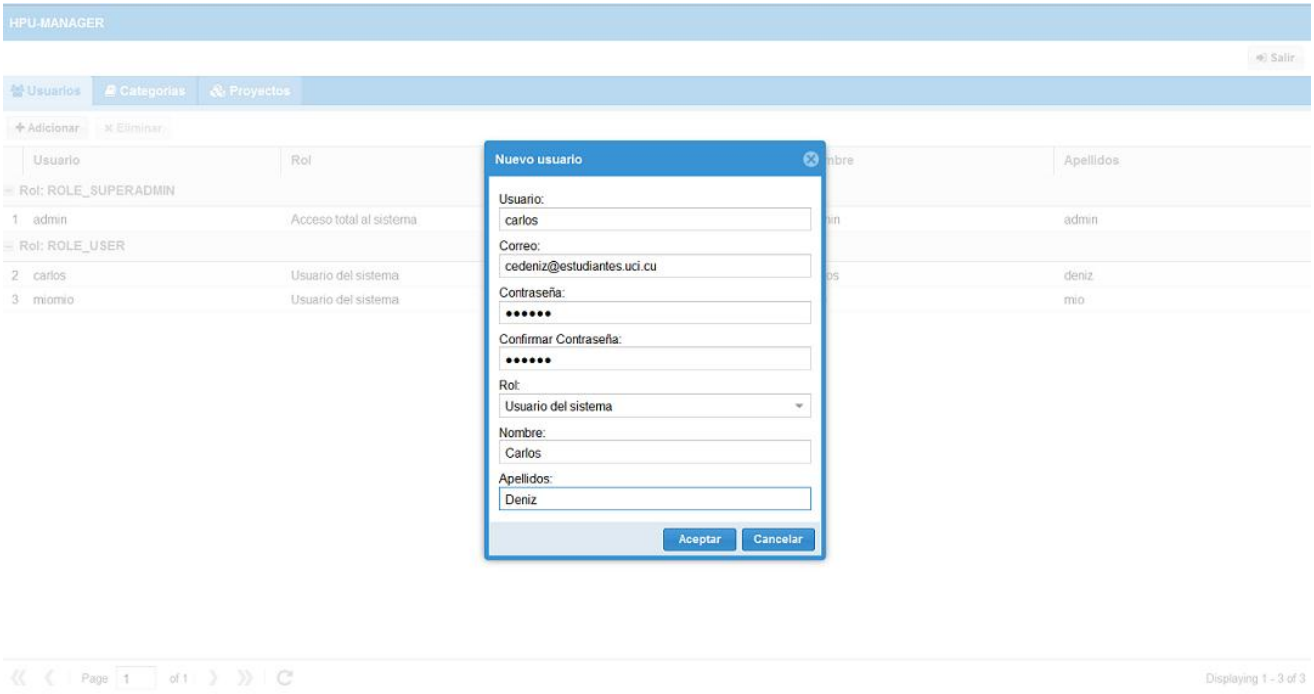

Figura 9. Prototipo de interfaz adicionar usuario.

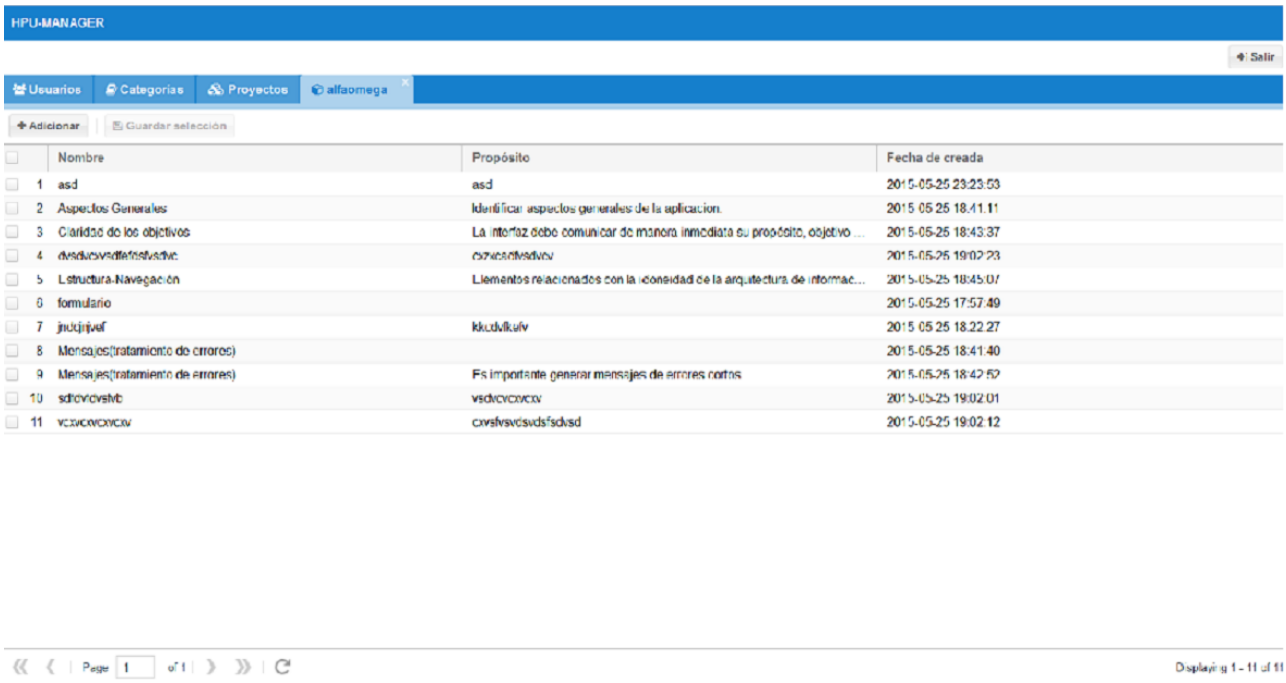

Figura 10. Prototipo de interfaz realizar evaluación.

2.6 Tarjetas CRC

La metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración). Cada una representa una clase, objeto, módulo o paquete, con su nombre en la parte superior. En la parte inferior izquierda se describen las responsabilidades y a la derecha las clases que le sirven de soporte (Bajo de Luque, y otros, 2011).

El *framework* Symfony, hace un uso continuo de los mecanismos orientados a objetos disponibles en PHP 5, utiliza en su diseño varios conceptos o técnicas del paradigma orientado a objeto (“objetos”, “abstracción”, “polimorfismo”, “encapsulación” y “herencia”) (Potencier, y otros, 2015), adaptando la técnica de las tarjetas CRC a estas características, representando cada tarjeta CRC a un módulo, quedando las responsabilidades definidas como las funciones que realiza y los colaboradores serían los métodos del módulo. A continuación se muestran la tarjeta CRC de la propuesta de solución:

Tabla 8 Tarjeta CRC Herramienta para pruebas de usabilidad.

Herramienta para pruebas de usabilidad	
Responsabilidades	**Colaboraciones**

Gestionar categoría	nuevaCategoria
	editarCategoria
	eliminarCategoria
	listarPreguntas
Gestionar pregunta	nuevaPregunta
	modificarPregunta
	eliminarPregunta
Adicionar no conformidad	nuevaNC
Eliminar no conformidad	eliminarNC
Asignar proyecto a usuario	relacionarProyectoUsuario
Autenticar usuario	securityCheckAction
Responder pregunta	responderPreguntaAction
Mostrar informe	respuestasPorUsuarioAction

2.7 Modelo de datos

El diseño de la estructura de las tablas que se agregaron a la base de datos de Symfony es el primer paso para comenzar la codificación de la solución informática. La estructura de las tablas se muestra en la Figura 10.

Para el desarrollo de la aplicación se establece una relación de mucho a uno entre la tabla {Users} (esta contiene la información referente a los usuarios de la aplicación) y la tabla {Proyecto} (contiene los proyectos a los cuales se le aplicará las pruebas de usabilidad).

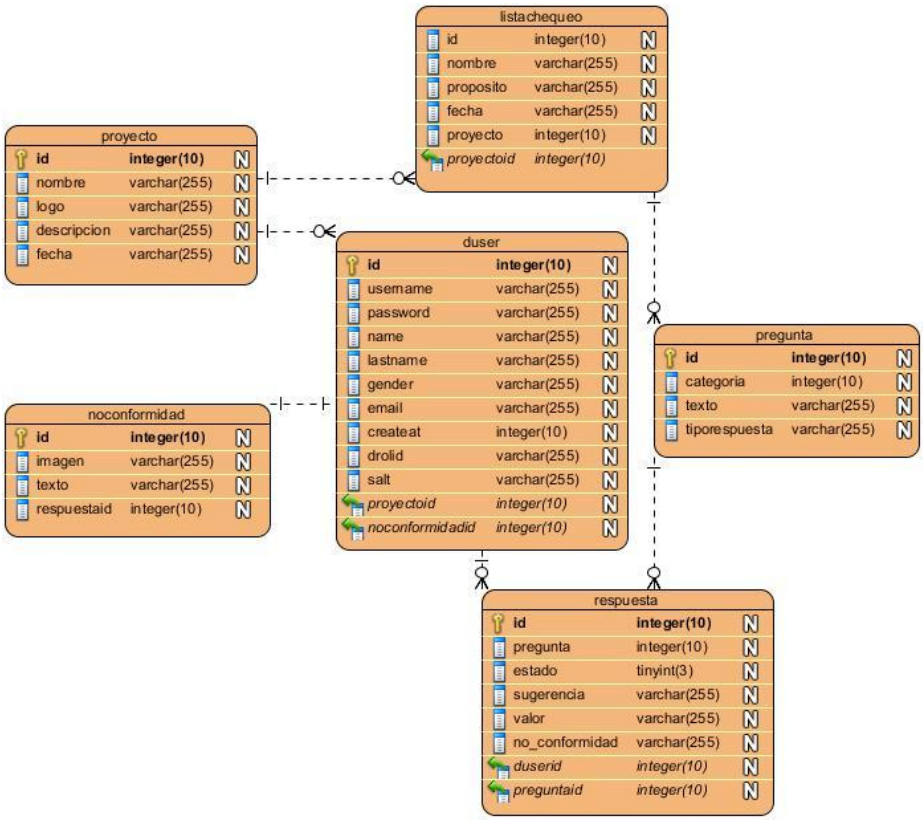

Figura 11. Tablas adicionadas a la base de datos.

Conclusiones

- ✓ Se definieron las HU que describen los aspectos principales a tener en cuenta para el desarrollo de la solución, contemplando una visión futura de las funcionalidades que debe cumplir el sistema.
- ✓ Asociado a estas HU se construyó el plan de entregas que enmarcó el tiempo de desarrollo en ocho semanas, determinando un cronograma que especifica las entregas que deben hacerse y conjuntamente se elaboró el prototipo de interfaz no funcional de la solución y una tarjeta CRC que representa las funcionalidades a implementar definiendo las responsabilidades y los colaboradores.
- ✓ Los principios, prácticas y técnicas que propone la metodología XP aportó los artefactos necesarios que darán paso a la implementación de la solución.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

Introducción

En el capítulo anterior se describió la propuesta del sistema y se presentaron los artefactos generados que ayudan al desarrollador a entender las funcionalidades que busca el cliente para así poder implementar el sistema. Igual de importante para un sistema es el proceso de pruebas, que tiene como objetivo determinar si el *software* se ajusta a los requisitos o a las condiciones impuestas.

Este capítulo está enmarcado en las fases de iteraciones hasta la entrega final y producción que propone la metodología XP, por lo tanto, se explica someramente como se desarrollaron las funcionalidades de la herramienta, comenzando desde la creación de las tablas añadidas a la base de datos, hasta la implementación de la herramienta. Para concluir, se analizan los resultados de las pruebas diseñadas para cada una de las HU y la herramienta en general.

3.1 Implementación

Todas las HU se desarrollan según la planificación de las iteraciones, por lo cual XP plantea que se verifique el plan de iteraciones para proceder así a la implementación. Todo esto se manifiesta de igual manera con las tareas de cada HU.

Para el presente trabajo se propone realizar tres iteraciones y a continuación se detallan cada una de ellas.

3.1.1 Iteración 1

Para la presente iteración se implementaron las HU de mayor prioridad y a continuación se detallan cada una de ellas.

Tabla 9. Funcionalidades abordadas en la primera iteración.

Módulo / componente	Historias de usuario	Tiempo de implementación (semanas)	
		Estimación	Real
Herramienta	Gestionar categoría	1	0.98
	Gestionar pregunta	1	0.16
	Gestionar proyecto	1	0.98

Se muestran ejemplos de las tareas de ingeniería efectuadas para las funcionalidades implementadas en esta iteración (Ver anexo II).

Tabla 10 Tareas de ingeniería para la primera iteración.

Tarea	
No. de tarea: 1	**No. de HU: 1**
Nombre: Estudiar el framework Symfony.	
Tipo de tarea: estudio	Puntos de estimación: 0.4
Fecha inicio: 10/12/2014	Fecha fin: 12/12/2014
Programador responsable: Carlos Ernesto Deniz Peña	
Descripción: Estudiar el funcionamiento del framework Symfony, en especial el trabajo con los módulos, los formularios y el modelo.	
Tarea	
No. de tarea: 2	**No. de HU: 1**
Nombre: Estudiar funciones del framework Extjs.	
Tipo de tarea: estudio	Puntos de estimación: 0.8
Fecha inicio: 13/12/2014	Fecha fin: 17/12/2014
Programador responsable: Carlos Ernesto Deniz Peña	
Descripción: Estudiar el funcionamiento del framework Extj y su integración con Symfony.	

3.1.2 Iteración 2

Para esta iteración se implementaron las HU que tienen prioridad media.

Tabla 11. Funcionalidad abordada en la segunda iteración.

Módulo / componente	**Historias de usuario**	**Tiempo de implementación (semanas)**

		Estimación	Real
Herramienta	Gestionar usuario	2	2
	Gestionar respuesta	1	1
	Adicionar no conformidad	1	1
	Eliminar no conformidad	1	1

Seguidamente se muestran las tareas de ingeniería efectuadas para la funcionalidad implementada en esta iteración.

Tabla 12 Tareas de ingeniería para la segunda iteración.

Tarea	
No. de tarea: 1	**No. de HU: 4**
Nombre: Desarrollar CRUD de Gestionar usuario.	
Tipo de tarea: desarrollo	Puntos de estimación: 1
Fecha inicio: 07/01/2015	Fecha fin: 11/01/2015
Programador responsable: Carlos Ernesto Deniz Peña	
Descripción: Se implementa la funcionalidad de gestionar usuario donde se podrá adicionar, eliminar o modificar un usuario.	
Tarea	
No. de tarea: 2	**No. de HU: 5**
Nombre: Desarrollar CRUD de Gestionar respuesta.	
Tipo de tarea: desarrollo	Puntos de estimación: 1
Fecha inicio: 14/01/2015	Fecha fin: 16/01/2015
Programador responsable: Carlos Ernesto Deniz Peña	
Descripción: Se implementa la funcionalidad de gestionar respuesta donde se podrá	

adicionar, eliminar o modificar una respuesta.

3.1.3 Iteración 3

Para esta iteración se implementaron las HU que tienen prioridad baja.

Tabla 13. Funcionalidad abordada en la tercera iteración.

Módulo / componente	Historias de usuario	Tiempo de implementación (semanas)	
		Estimación	Real
Herramienta	Autenticar usuario	1	0.10
	Responder pregunta	1	1
	Mostrar informe	1	1
	Asignar proyecto a usuario	1	1

Se muestran las tareas de ingeniería efectuadas para la funcionalidad implementada en esta iteración (Ver anexo III):

Tabla 14 Tareas de ingeniería para la tercera iteración.

Tarea	
No. de tarea: 1	**No. de HU: 8**
Nombre: Autenticar usuario.	
Tipo de tarea: desarrollo	Puntos de estimación: 1
Fecha inicio: 14/01/2015	Fecha fin: 16/01/2015
Programador responsable: Carlos Ernesto Deniz Peña	
Descripción: Se implementa la funcionalidad que permitirá a un usuario autenticarse.	
Tarea	

No. de tarea: 2	**No. de HU: 10**
Nombre: construcción de la funcionalidad mostrar informe.	
Tipo de tarea: desarrollo	Puntos de estimación: 1
Fecha inicio: 17/01/2015	Fecha fin: 18/01/2015
Programador responsable: Carlos Ernesto Deniz Peña	
Descripción: Se implementa la funcionalidad que permitirá mostrar un informe de las respuestas.	

3.2 Patrón arquitectónico en Symfony

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura Modelo Vista Controlador, en lo adelante MVC, que está formado por tres niveles:

- ✓ **Modelo**: representación específica del dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica del dominio añade significado a los datos.
- ✓ **Vista**: esta presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.
- ✓ **Controlador**: Se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

El principio más importante de la arquitectura MVC es la separación del código del programa en tres capas, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador. La programación se puede simplificar si se utilizan otros patrones de diseño. De esta forma, las capas del modelo, la vista y el controlador se pueden subdividir en más capas.

3.3 Aplicación de los patrones de diseño en Symfony

Los patrones de diseño son la descripción de un problema y su solución, los mismos nombran, abstraen e identifican los aspectos claves de un diseño estructurado, que lo hace útil para la creación de diseños orientados a objetos que pueden emplearse en otros contextos (LARMAN, 2002).

Patrones GRASP (General Responsibility Asignment *Software* Patterns):

Patrones para asignar responsabilidades, son parejas de problema/solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades (LARMAN, 2002).

Experto: este patrón se encarga de asignar responsabilidad al experto en información, es decir, a la clase que cuenta con la información necesaria para cumplir la responsabilidad. Es utilizado en la capa de abstracción del modelo de datos. Symfony crea automáticamente las clases que generan las entidades del modelo de datos. Asociado a cada una de estas clases son creadas un conjunto de funcionalidades que la relacionan de forma directa con la entidad que representan y contienen toda la información necesaria de la tabla en la base de datos.

Creador: este patrón resuelve el problema de asignar responsabilidades relacionadas con la creación de objetos. Es utilizado en los controladores, en ellos se encuentran las acciones definidas para el sistema. En la implementación de las acciones se crean instancias de las clases del modelo y de los formularios que representan a estas clases.

Alta cohesión: es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas para que no realicen un trabajo enorme. La alta cohesión se evidencia en los controladores que poseen un conjunto de funcionalidades existiendo estrecha relación entre algunas. Ejemplo de ellos lo constituyen las acciones *create* y *update* que al crear o actualizar un objeto realiza las validaciones mediante la acción *processForm*.

Bajo acoplamiento: es una medida de la fuerza con que una clase está conectada a con otras. Mientras menos componentes dependan de otros, mas reusable y flexible se vuelve el sistema. El bajo acoplamiento se evidencia en el hecho de que los controladores heredan únicamente de la clase *Controller*. Además, las clases que implementan la lógica del negocio y de acceso a datos no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia entre las clases, en este caso sea baja.

Controlador: este patrón resuelve el problema de asignar la responsabilidad, recibir o manejar un mensaje de evento del sistema a una clase. Un controlador sirve como intermediario entre una interfaz y la acción que se desee ejecutar. Se muestra en las clases que forman la capa controlador entre ellas se encuentran: ListaChequeoController, PreguntasController, ProyectoController y en el index.php del ambiente que se está ejecutando.

Patrones GOF (Gang of Four) (Gamma, y otros, 1995).

Decorator: se encarga de crear una plantilla de diseño base que incorpora todos los componentes comunes del sistema (cabecera, pie de página, menú y secciones) que luego puede ser empleada en cualquier página de la aplicación. En Symfomy este patrón es utilizado en la capa vista del patrón arquitectónico MVC, ejemplo de ello lo constituye la clase base.html.twig que es padre de todas las vistas de la aplicación.

3.4 Estructura de la aplicación

La aplicación quedo estructurada en el framework Symfony de la siguiente forma (ver Figura 11).

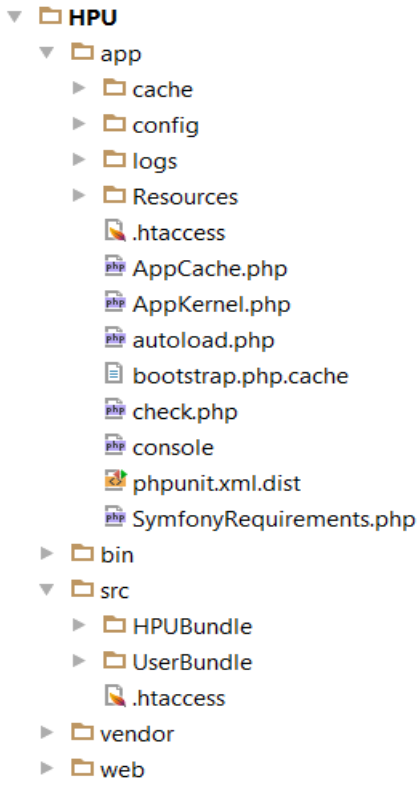

Figura 12. Ubicación de la herramienta en la estructura de archivos de Symfony.

El módulo terminó con 3070 líneas. Entre las funciones más complejas está responder_pregunta que posee una complejidad de O(n^2).

3.5 Pruebas

En esta sección se presenta el estudio relacionado con las pruebas para asegurar la calidad del *software*, se evidencian además las utilizadas para probar las funcionalidades del sistema en cuestión con el objetivo de encontrar y documentar los defectos que pueden afectar la calidad del mismo.

3.6.1 Pruebas unitarias

En las pruebas unitarias, se analiza una porción de código que pueda ser analizada de manera aislada, como por ejemplo funciones y métodos. A los que después de pasarle parámetros de entrada se deben obtener otros parámetros de salida claramente definidos.

A partir de la versión 2, Symfony ha optado por utilizar la librería PHPUnit, para crear y ejecutar todas las pruebas unitarias.

PHPUnit es un marco de trabajo *open source* para el desarrollo, orientado a pruebas para cualquier código PHP. Fue creado por Sebastian Bergman y su objetivo es crear pequeñas unidades que revisen funcionalidades puntuales del código y probar que funcionen correctamente, además de la posibilidad de automatizar estas pruebas para ejecutarlas frecuentemente, tanto como el código cambie (Bergmann, 2014).

En un proyecto Symfony es muy fácil implementar las pruebas unitarias. El ORM viene con su propio conjunto de herramientas para facilitarlas y simular todo lo que se necesite, tal como una conexión y un gestor de entidades. Si se desea probar la ejecución real de las consultas, se necesita una prueba funcional. Las pruebas unitarias solo son posibles cuando se prueba un método que construye una consulta (Pacheco, 2011).

A continuación se muestran algunas imágenes que muestran la realización de estas pruebas.

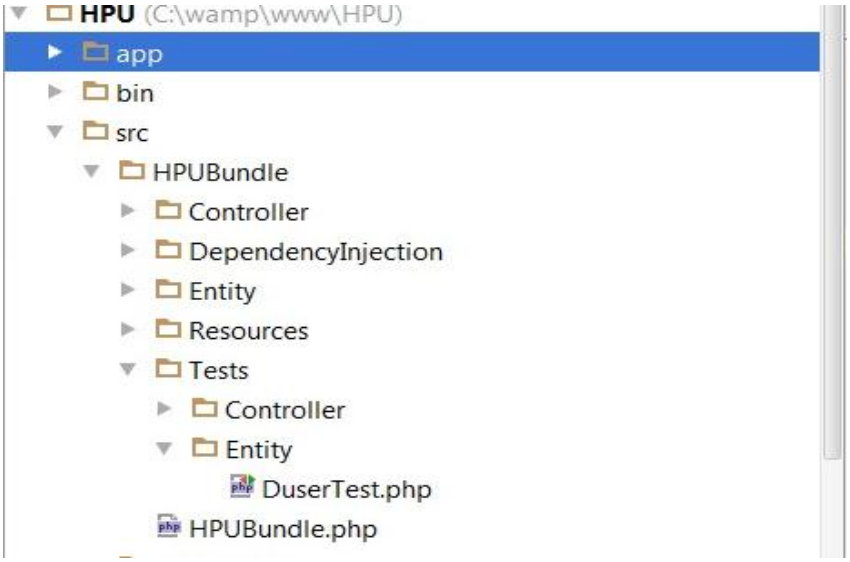

Figura 13. Estructura donde se guardan las pruebas a las entidades.

\UserBundle\Test\Entity\DuserTest

```
7
8   class DuserTest extends \PHPUnit_Framework_TestCase
9   {
10
11      private $validator;
12
13      protected function setUp()
14      {
15          $this->validator = Validation::createValidatorBuilder()
16              ->enableAnnotationMapping()
17              ->getValidator();
18      }
19
20      public function testValidarUsuario()
21      {
22          $usuario = new Duser();
23          $usuario->setUsername('Usuario de prueba');
24          $user = $usuario->getUsername();
25          $this->assertEquals('usuario-de-prueba', $user, 'El usuario se asigna automáticamente a partir del username'
26          );
27      }
28
29      public function testValidarApellidos()
30      {
31          $usuario = new Duser();
32          $usuario->setName('usuario de prueba');
33          $listaErrores = $this->validator->validate($usuario);
34          $this->assertGreaterThan(0, $listaErrores->count(), 'Los apellidos no deben dejarse en blanco');
35          $error = $listaErrores[0];
36          $this->assertEquals('This value should not be blank.', $error->getMessage());
37
38          $this->assertEquals('descripcion', $error->getPropertyPath());
39          $usuario->setLastname('apellidos de prueba');
40          $listaErrores = $this->validator->validate($usuario);
41          $this->assertGreaterThan(0, $listaErrores->count(), 'Los apellidos debe tener al menos 50 caracteres'
42          );
```

Figura 14. Descripción de la clase DuserTest.

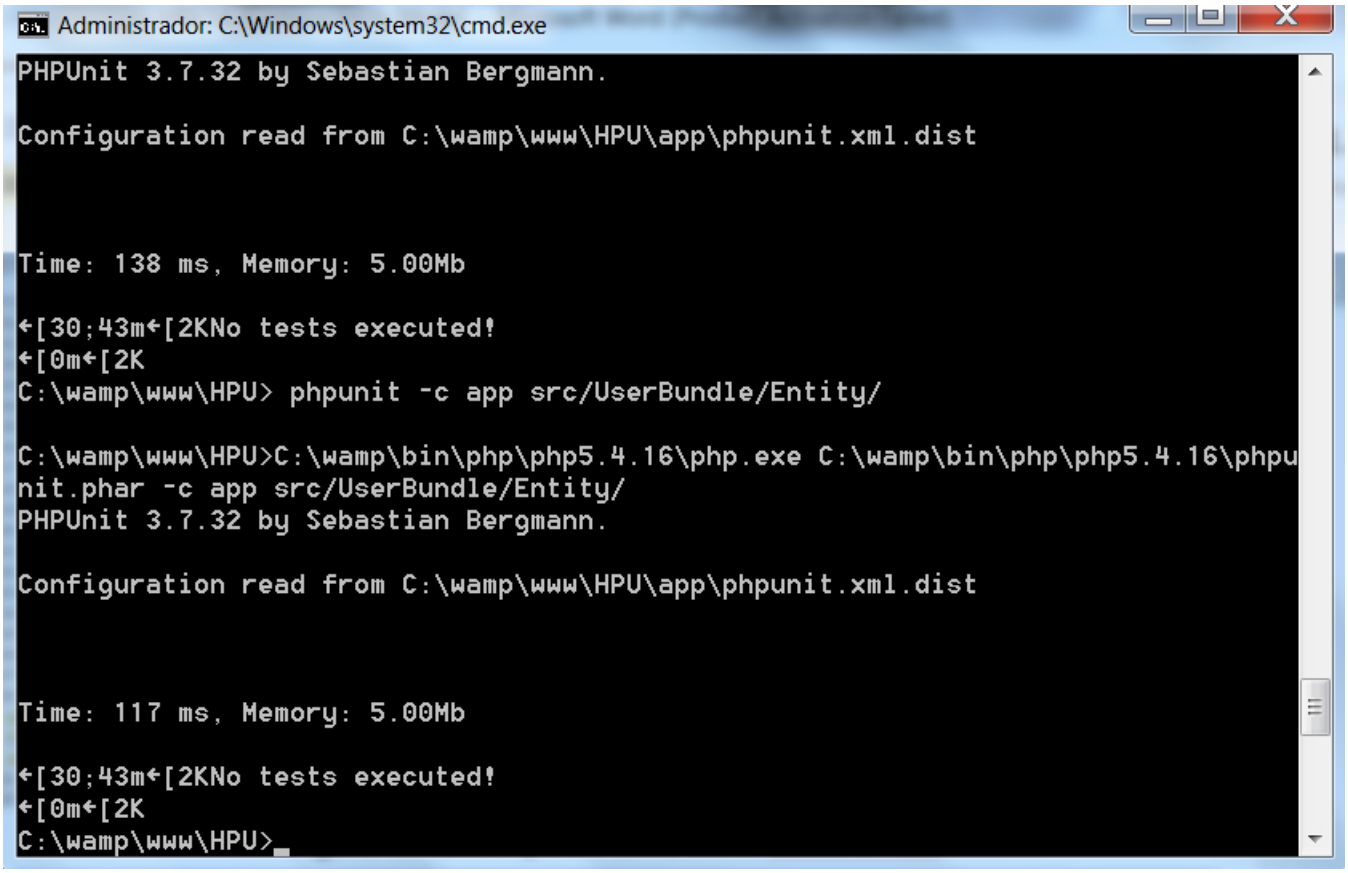

Figura 15. **Prueba realizada al *UserBundle Entity.***

En el siguiente grafico se muestra un estudio estadístico de la cantidad de no conformidades detectadas con la realización de las pruebas unitarias.

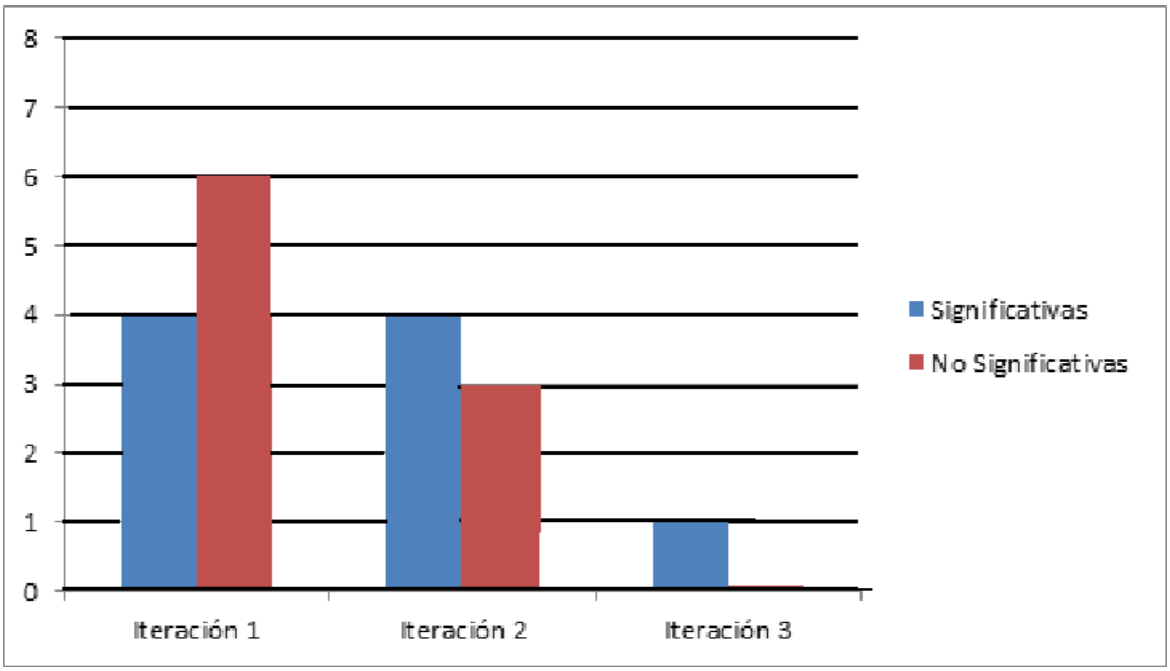

Figura 16. Resultados de las pruebas unitarias.

Se realizaron tres iteraciones obteniéndose en la primera y en la segunda cuatro no conformidades significativas y en la tercera una. Además, se identificaron seis y tres no significativas en la primera y segunda iteración respectivamente. Todas quedaron resueltas al finalizar las pruebas.

3.6.2 Pruebas de aceptación

En las pruebas de aceptación se deben especificar uno o varios escenarios para comprobar que una HU ha sido correctamente implementada. Las pruebas de aceptación son consideradas como pruebas de caja negra. Una HU no se puede dar por terminada hasta tanto pase correctamente todas las pruebas de aceptación. Los escenarios que se han definido para probar la herramienta se enumeran en los siguientes casos de prueba:

Tabla 15. Caso de prueba de aceptación HU1_P1.

Caso de prueba de aceptación.	
Código: HU1_P1	**No. de HU: 1**
Nombre: Gestionar categoría	
Descripción: el usuario ya se ha autenticado anteriormente en el sistema, se le mostrarán las categorías existentes, además podrá adicionar otras categorías.	

Condiciones de ejecución: el usuario debe estar autenticado como administrador.
Pasos de ejecución: Se autentica en la página principal, da clic en la pestaña categorías, luego selecciona el botón adicionar.
Resultado esperado: el sistema permitirá que el usuario registrado adicione, las categorías, en caso contrario mostrará un mensaje con el mínimo de información.
Evaluación de la prueba: prueba satisfactoria.

Tabla 16 Caso de prueba de aceptación HU1_P2.

Caso de prueba de aceptación.	
Código: HU1_P2	**No. de HU: 1**
Nombre: Gestionar categoría	
Descripción: el usuario ya se ha autenticado anteriormente en el sistema, se le mostrarán las categorías existentes, además podrá eliminar una o varias categorías.	
Condiciones de ejecución: el usuario debe estar autenticado como administrador.	
Pasos de ejecución: se autentica en la página principal, da clic en la pestaña categorías, selecciona la categoría deseada y luego da clic en el botón eliminar.	
Resultado esperado: el sistema permitirá que el usuario registrado elimine las categorías, en caso contrario mostrará un mensaje con el mínimo de información.	
Evaluación de la prueba: prueba satisfactoria.	

Tabla 17 Caso de prueba de aceptación HU1_P3.

Caso de prueba de aceptación.	
Código: HU1_P3	**No. de HU: 1**
Nombre: Gestionar categoría	
Descripción: el usuario ya se ha autenticado anteriormente en el sistema como administrador, se le mostrará las categorías, además podrá modificar una o varias categorías	

si así lo desea.
Condiciones de ejecución: el usuario debe estar autenticado como administrador.
Pasos de ejecución: se autentica en la página principal, da clic en la pestaña categorías, da doble clic sobre la categoría deseada, modifica la categoría y luego da clic en el botón actualizar.
Resultado esperado: el sistema permitirá que el usuario registrado modifique las categorías.
Evaluación de la prueba: Prueba satisfactoria.

Los restantes casos de prueba de aceptación se pueden observar en el (Ver anexo IV).

Las pruebas a las funcionalidades se realizaron en tres iteraciones detectándose siete, cuatro y dos no conformidades significativas en la primera, segunda y tercera iteración respectivamente. También, se encontraron cuatro no significativas en la primera y segunda iteración.

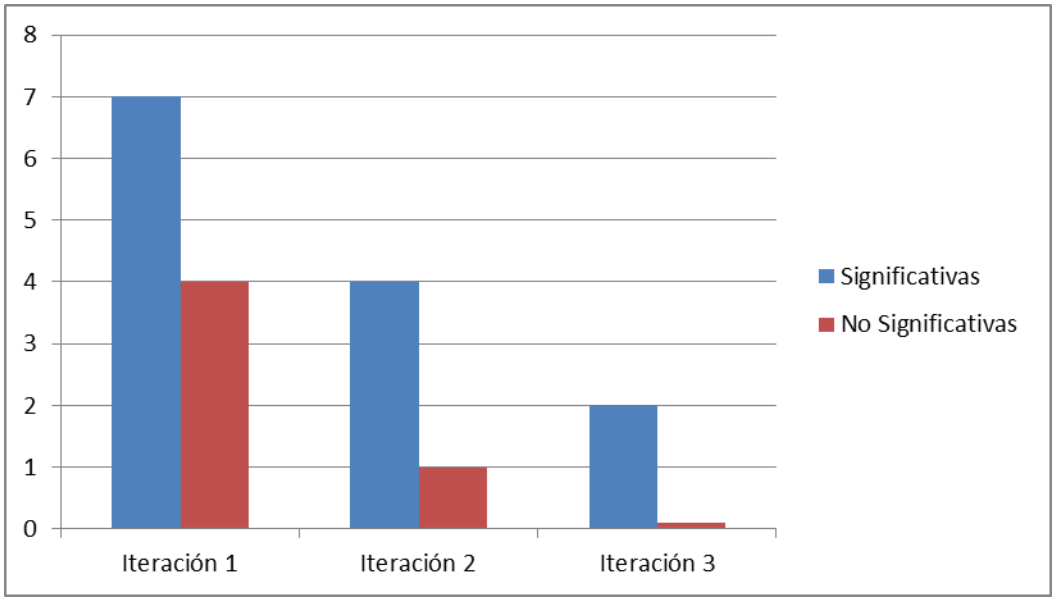

Figura 17. No conformidades significativas y no significativas.

Las no conformidades no significativas, se centraron en errores ortográficos como: omisiones de tildes, paréntesis, cambio de mayúscula por minúscula. Las significativas, en errores de validación y cambios en el diseño. Las no conformidades encontradas después de concluida cada iteración de pruebas se

analizaron por parte del equipo de desarrollo para corregir los errores detectados en la iteración anterior, lo que contribuyó a mejorar la calidad y funcionalidad del *software*. Al final de la tercera iteración quedaron resueltas todas las no conformidades detectadas.

Conclusiones

- ✓ El desglose de las historias de usuario en tareas de ingeniería facilitó la implementación y posibilitó a los programadores obtener en cada iteración, una versión funcional del producto.
- ✓ Las pruebas de aceptación y unitarias aplicados permitieron probar el correcto funcionamiento de la aplicación, tanto a nivel de código como en la interacción con el usuario comprobándose que se satisfacen las necesidades del cliente.

CONCLUSIONES

Después de desarrollar el presente trabajo y analizar los resultados obtenidos, las conclusiones esenciales a las que se arriban son las siguientes:

- ✓ El análisis de diferentes conceptos y elementos teóricos relacionados con el objeto de estudio permitió adquirir el conocimiento necesario sobre las pruebas de usabilidad y así identificar la metodología de desarrollo, las herramientas y tecnologías seleccionadas para darle cumplimiento al objetivo general propuesto.
- ✓ El diseño de la propuesta de solución, guiado por la metodología de desarrollo XP, posibilitó identificar los requisitos y características del sistema que agilizará, controlará y realizará un seguimiento de las pruebas usabilidad realizadas a los proyectos del Centro FORTES.
- ✓ La implementación y el análisis de las pruebas de aceptación y unitarias realizadas al sistema desarrollado permitieron garantizar la solidez del sistema, además de comprobar que los requisitos planteados fueron cumplidos y que las necesidades del cliente fueron satisfechas.

RECOMENDACIONES

Concluida la investigación se proponen las siguientes recomendaciones:

- ✓ Que se extienda el uso de la herramienta en toda la universidad.
- ✓ Continuar con el estudio del proceso de las pruebas de usabilidad en el Centro FORTES e identificar nuevas funcionalidades, así como hacer uso de algunas de las técnicas de evaluación como la indagación y el *test* de usuario.
- ✓ Ampliar el sistema con técnicas de Inteligencia Artificial como el diseño de una base de conocimientos.

BIBLIOGRAFÍA

Arellano Moncayo, Fabricio Gerardo. 2011. Space Escuela Politécnica Nacional. *Desarrollo e implantación del sistema de reservación de laboratorios para el Laboratorio de la Facultad de Ingeniería de Sistemas.* [En línea] 3 de Mayo de 2011. [Citado el: 7 de Marzo de 2013.] http://bibdigital.epn.edu.ec/handle/15000/3824.

Asenjo Sánchez, Jorge. 2009. *Sistemas Gestores de Bases de Datos.* España : Creative Commons, 2009.

autores, colectivo. 2006. Artefacto: Prototipo de interfaz de usuario. [En línea] 2006. [Citado el: 3 de 6 de 2015.] http://betaniatech.com/SmallProjects/core.base_rup/workproducts/rup_user_interface_prototype_7237E5AA.html.

—. 2013. Bienvenido a CodeIgniter : Guía de Usuario de CodeIgniter. [En línea] 8 de 1 de 2013. [Citado el: 3 de 6 de 2015.] http://escodeigniter.com/guia_usuario/.

Avison, D.E. y Fitzgerald, G. 1995. *Information Systems Development: Methodologies, Techniques, and Tools.* s.l. : McGraw-Hill, 1995.

Bajo de Luque, M.J. y et.al. 2011. Metodologías ágiles. *Agile Alliance.* [En línea] 2011. [Citado el: 23 de Febrero de 2013.] http://metodologiasagiles.herobo.com/index.php/es/2011-12-05-16-09-55/metodologia-xp.

Baquía, Redacción de. 2007. Charla con Alfonso de la Nuez, CEO de Xperience Consulting. [En línea] 31 de 1 de 2007. [Citado el: 15 de 2 de 2015.] http://www.baquia.com/tecnologia-y-negocios/entry/emprendedores/charla-con-alfonso-de-la-nuez-ceo-de-xperience-consulting.

Beck, Kent. 1999. *Embracing Change with Extreme Programming.* s.l. : Addison Wesley Longman, Inc, 1999. ISBN-10: 0201616416 | ISBN-13: 978-0201616415.

Beck, Kent y Andres, Cynthia. 2005. *"Extreme Programming Explained—Embrace Change".* s.l. : Second Edition,Addison Wesley, 2005.

Bergmann, Sebastian. 2014. PHPUnit Manual. [En línea] 2014. [Citado el: 15 de 5 de 2015.] http://phpunit.de/manual/current/en/phpunit-book.pdf.

Bevan, N., Kirakowski, J. y Maissel, J. 1991. *What is Usability?* Elsevier : Proceedings of the 4th International Conference on HCI, Stuttgart, 1991.

Bevan, Nigel. 1995. *Usability is quality of use.* s.l. : Advances in human factors ergonomics, 1995. vol. 20, p. 349-349.

Cabezón, Carolina. 2014. La usabilidad como concepto clave para el éxito de tu web. *ICEMD.* [En línea] 2014. [Citado el: 10 de 2 de 2015.] http://blogs.icemd.com/blog-socializa-en-on/sabilidad-exito-web/.

Carreras, O. 2007. Usable y Accesible. “Disciplinas relacionadas con la usabilidad”. [En línea] 2007. [Citado el: 6 de 1 de 2015.] http://olgacarreras.blogspot.com/2007/01/disciplinas-relacionadas-con-la.html.

Carvajal Riola, J.C. 2008. *Metodologías ágiles: Herramientas y modelo de desarrollo para aplicaciones Java EE como metodología empresarial.* Barcelona : Tesis de maestría, 2008.

Carvajal, M. y Saab, J. 2010. *Regulación de usabilidad para sitios web estatales.* 2010.

Código, Editores de. 2014. Descargar PhpStorm Full IDE de programación web. [En línea] 2014. [Citado el: 15 de 2 de 2015.] http://www.editoresdecodigo.com/2014/06/descargar-phpstorm-full-ide-para-php-y-mas.html.

Converse, Tim, Park, Joyce y Morgan, Clark. 2004. *PHP5 and MySQL Bible.* Indiana : Wiley Publishing,Inc, 2004. ISBN: 0-7645-5746-7.

De Luca, Damián y De Luca, A.A. 2012. CSS3 & HTML5. *Los nuevos estándares para el diseño y desarrollo web.* [En línea] 2 de Septiembre de 2012. [Citado el: 14 de Diciembre de 2012.] http://css3html5.com.ar/.

Díaz, Jesús Bustamante. 2010. *La herramienta de tests de usabilidad a distancia Loop11.* s.l. : Revista internacional de Información y Comunicación, 2010. vol. 19, núm. 4 .

DIS, ISO. 9241-11. 1998. *Ergonomic requirements for office work with visual display terminals (VDTs)-Part 11-Guidance on usability.* s.l. : International Organisation for Standardisation, 1998.

Duque González, Raúl. 2011. Mundo geek. [En línea] 2011. [Citado el: 9 de Enero de 2013.] http://mundogeek.net/etiqueta/servidor-web/.

Eguíluz Pérez, J. 2008. *Introducción a Javascript.* 2008.

Eguizabal Alonso, D. y Latorre, P. M. 2009. *Una Herramienta Para La Obsevación Remota De Usuarios.* 2009.

etnassoft.com. 2012. Manual de Symfony2. [En línea] 2012. [Citado el: 3 de 6 de 2015.] http://www.etnassoft.com/biblioteca/manual-de-symfony2/.

Gamma, E., y otros. 1995. *Design Patterns.* Massachusetts : Addison-W esley. ISBN 0-201-63361-2, 1995.

Gilfillan, Ian. 2003. *La Biblia de MySQL.* España : Kluwer Academic, 2003.

González Barahona, J.M., Seoane Pascual, Joaquín y Robles, Gregorio. 2003. Introducción al *software* libre. [En línea] 2003. [Citado el: 10 de Enero de 2013.] http://curso-sobre.berlios.de/introsobre/2.0.1/sobre.html/sec-ide.html.

Granollers, T., Perdrix, F. y Lorés, J. 2004. *Incorporación de usuarios en la evaluación de la usabilidad por recorrido cognitivo.* s.l. : Interacción’04, 2004.

Grenning, J. 2001. Launching Xp at a Process-Intensive Company. s.l. : IEEE *Software*, 2001. Vol. 18.

Hassan Montero, Y. y Martín Fernández, F. J. 2003. Guía de Evaluación Heurística de Sitios Web. [En línea] 2003. [Citado el: 15 de 2 de 2015.] http://www.nosolousabilidad.com/. nº 2, ISSN 1886-8592.

HERRERA, B. y LATAPIE, V. 2010. *Diseñando para la educación. No solo Usabilidad.* 2010. Revista multidisciplinar sobre diseño, personas y tecnología.

ISO, 9241-11. 1998. *Ergonomic requirements for office work with visual display terminals (VDT)s – Part 11 Guidance on usability.* Londres : s.n., 1998.

ISO/IEC, 9126. 1991. *Software Product Evaluation.* 1991.

Kabir, Mohammed J. 1999. *La Biblia Servidor Apache 2.* s.l. : Anaya Multimedia, 1999. ISBN: 8441514682.

Kada*Software*. 2013. Kada*Software*. [En línea] Universidad Tecnologica de la Mixteca, 2013. [Citado el: 14 de Enero de 2013.] http://www.kada*software*.com/index.php/tecnologias-de-desarrollo.html.

LARMAN, Craig. 2002. *UML y Patrones.* s.l. : Prentice-Hall, 2002.

Letelier, P. y Penadés, Mª C. 2008. Métodologías ágiles para el desarrollo de *software*:eXtreme Programming (XP). Valencia : Universidad de Valencia, 2008. Vol. 05, 26. ISSN 1666-1680.

López, Jorge. 2008. *Symfony, el framework PHP" de moda" para desarrollo web.* s.l. : Todo linux: la revista mensual para entusiastas de GNU/LINUX , 2008. no 93, p. 20-23.

Maurer, Donna. 2004. ¿Qué es Usabilidad?, Y Como conseguirla. *SG Buzz.* [En línea] 1 de 11 de 2004. [Citado el: 10 de 2 de 2015.] http://www.steptwo.com.au/papers/kmc_whatisusability/index.html.

Morales, Ángel Luis. 2006. *Manual sobre el uso de iTALC en los centros TIC.* Fabra : s.n., 2006.

NETBEANS.ORG. 2011. Bienvenido a NetBeans y www.netbeans.org, Portal del IDE Java de Código Abierto. [En línea] 2011. [Citado el: 20 de Noviembre de 2012.] http://netbeans.org/index_es.html.

NETCRAFT. 2011. December 2011 Web Server Survey. [En línea] 2011. [Citado el: 15 de 2 de 2015.] http://news.netcraft.com/archives/2011.

Nielsen, J. 1993. *Usability Engineering.* s.l. : Morgan Kaufmann Publishers Inc., 1993.

Nielsen, Jakob. 1994. *Usability inspection methods.* s.l. : En Conference companion on Human factors in computing systems. ACM, 1994. p. 413-414.

Nuez, Alfonso de la. 2009. *Tips and Tricks for Remote Unmoderated Usability Testing.* 2009.

Pacheco, Nacho. 2011. *Manual de Symfony2.* 2011.

Padilla Matos, Reynier. 2008. Programación Web y Tecnologías Informáticas. [En línea] 28 de Febrero de 2008. [Citado el: 14 de Enero de 2013.] http://zenkius.blogspot.com/2008/02/tecnologias-del-lado-del-cliente.html.

Palacio, J. 2007. *Flexibilidad con Scrum, principios de diseño e implantación en campos Scrum.* s.l. : SafeCreative, 2007.

Pérez Eguíluz, Javier. 2009. *Introducción a JavaScript.* Madrid : Autoedición, 2009.

Perurena Cancio, L. y Moráguez Bergues, M. 2013. *Usabilidad de los sitios Web, los métodos y las técnicas para la evaluación.* s.l. : Revista Cubana de Información en Ciencias de la Salud, 2013. vol. 24, no 2, p. 176-194.

Piattini Velthuis, M. y al., et. 2013. Alarcos. [En línea] Universidad de Castilla La Mancha, 2013. [Citado el: 14 de Enero de 2013.] alarcos.inf-cr.uclm.es/doc/I*SOFTWARE*I/Tema04.pdf.

Picasso, Ruiz. 2012. *Usabilidad.* Valle del Grijalva : s.n., 2012.

Potencier, Fabien y Zaninotto, François. 2015. 1.2. Conceptos básicos . *LIBROSWEB.* [En línea] 2015. [Citado el: 21 de 4 de 2015.] http://librosweb.es/libro/symfony_1_4/capitulo_1/conceptos_basicos.html.

Prieto, Martínez y Torrente, Suárez. 2010. *Towards the evaluation of usability in educative websites.* s.l. : International Journal of Technology Enhanced Learning, 2010. vol. 2, no 1, p. 145-161.

Rivera, E.A., Zamora, R.G. y Soria, M.G. 2012. *Sistema de Educación a Distancia.* s.l. : IV Congreso de Tecnología en Educación, 2012.

Rodríguez, Daniel y Barredo, Eduard. 2009. usabilidad. [En línea] 2009. [Citado el: 15 de 2 de 2015.] http://www.trucosoptimizacion.com/index.php/2011/02/02/test-remoto-usuarios-3-herramientas-cambiaran-usabilidad/.

Shore, James y Warden, Shane. 2008. *The Art of Agile Development.* Estados Unidos : O'Reilly Media, Inc., 2008. ISBN-13: 978-0-596-52767-9.

Soto, Jose Sanchez. 2012. Arquitectura Mvc Ext JS. *Developer's Muni Miraflores.* [En línea] 16 de 4 de 2012.

Suárez Jorge, Alinoet . 2009. *Estudio Comparativo sobre 11 Metodologías de Desarrollos Web.* La Habana : s.n., 2009.

symfony.es. 2014. Conoce el nuevo plugin de Symfony2 para PHPStorm. [En línea] 25 de 9 de 2014. [Citado el: 15 de 2 de 2015.] http://symfony.es/noticias/2014/08/25/conoce-el-nuevo-plugin-de-symfony2-para-phpstorm/.

Travis, B. 2011. *Drupal 7 for Windows Developers.* s.l. : Apress, 2011.

Walker, Craig. 2015. Announcing Sencha Pivot Grid. *Sencha.* [En línea] Sencha Inc., 2015. [Citado el: 30 de 3 de 2015.] http://www.sencha.com/.

GLOSARIO DE TÉRMINOS

API: Del inglés *Application Programming* Interface - Interfaz de Programación de Aplicaciones: es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.

Artefacto: Un artefacto es un producto tangible resultante del proceso de desarrollo de *software*.

Código Abierto: Término con el que se conoce al *software* distribuido y desarrollado libremente.

CSS: Las hoja de estilo en cascada (en inglés *Cascading Style Sheets*) contienen un conjunto de etiquetas que definen el formato que se aplicará al contenido de las páginas de una Web. Se llama "cascada" porque una hoja puede heredar los formatos definidos en otra hoja de forma de que no hace falta que vuelva a definirlos. Estas hojas permiten la separación entre el contenido y la presentación de un Sitio Web.

Entorno de desarrollo integrado (IDE): Programa compuesto por un conjunto de herramientas para un programador que puede dedicarse en exclusiva a un sólo lenguaje de programación o bien puede utilizarse para varios. Consisten en un editor de código, un compilador, un depurador y un GUI.

Frameworks: Es una estructura de soporte definida en la cual otro proyecto de *software* puede ser organizado y desarrollado. Típicamente puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros *software*, para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de *software* que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

HTML: HyperText Markup Language (Lenguaje de marcado hipertextual), es el método más común de intercambio de información en la *World Wide We*b, mediante el cual se transfieren las páginas web a un ordenador.

HTTP: *HyperText Transfer Protocol* (Protocolo de transferencia de hipertexto). Es el protocolo usado para intercambiar archivos (texto, gráfica, imágenes, sonido, video y otros archivos multimedia) en la *World Wide Web*.

JavaScript: Lenguaje de programación interpretado, utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Actualmente todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas Web.

Licencia GNU/GPL: *Free Software Foundation* (Licencia Pública General). Es una licencia creada por GNU y orientada principalmente a los términos de distribución, modificación y uso de *software*. Su propósito es declarar que el *software* cubierto por esta licencia es *software* Libre.

PHP: PHP (*Hypertext Preprocessor*) es un lenguaje multiplataforma, multiparadigma, script (no se compila para conseguir códigos máquina sino que existe un intérprete que lee el código y se encarga de ejecutar las instrucciones que contiene éste código) para el desarrollo de páginas web dinámicas del lado del servidor, cuyos fragmentos de código se intercalan fácilmente en páginas HTML, debido a esto y a que es de código.

TIC: Tecnologías de la Información y la Comunicación, se encargan del estudio, desarrollo, implementación, almacenamiento y distribución de la información mediante la utilización de hardware y *software* como medio de sistema informático.

UCI: Universidad de las Ciencias Informáticas.

Web: Es un sistema para presentar información en Internet basado en hipertexto. Cuando se utiliza en masculino (el Web, un Web) se refiere a un sitio Web entero, en cambio si se utiliza en femenino (la Web, una Web) se refiere a una página Web concreta dentro del sitio Web.

XHTML: *Extensible HyperText Markup Language* (Lenguaje extensible de marcado de hipertexto), es el lenguaje pensado para sustituir a HTML como estándar para las páginas web.

XML: *Extensible Markup Language* (Lenguaje de marcado extensible), conjunto de reglas para definir etiquetas semánticas que organizan un documento en diferentes partes.

ANEXOS

ANEXO I.

Historias de Usuarios.

Tabla 18. HU Gestionar usuario.

Historia de Usuario	
No: 4	Nombre: Gestionar usuario
Usuario: Usuario autenticado	
Prioridad en el negocio: Baja	Puntos Estimados: 2
Riesgo de desarrollo: Media	Iteración asignada: 3
Descripción: El administrador podrá adicionar y eliminar un usuario en la aplicación.	
Observaciones: El usuario debe estar autenticado para realizar la acción.	

Tabla 19. HU Gestionar respuesta.

Historia de Usuario	
No: 5	Nombre: Gestionar respuesta
Usuario: Usuario autenticado	
Prioridad en el negocio: Alta	Puntos Estimados: 2
Riesgo de desarrollo: Alta	Iteración asignada: 1
Descripción: Da la posibilidad al administrador de adicionar, modificar y eliminar una respuesta a la aplicación.	
Observaciones: El usuario debe estar autenticado para realizar la acción.	

Tabla 20. HU Adicionar no conformidad.

Historia de Usuario	
No: 6	Nombre: Adicionar no conformidad
Usuario: Administrador	

Prioridad en el negocio: Alta	Puntos Estimados: 3
Riesgo de desarrollo: Alta	Iteración asignada: 1
Descripción: Se podrá adicionar una no conformidad en la aplicación.	
Observaciones: La realiza el usuario al efectuar la prueba.	

Tabla 21. HU Eliminar no conformidad.

Historia de Usuario	
No: 7	Nombre: Eliminar no conformidad
Usuario: Usuario autenticado	
Prioridad en el negocio: Media	Puntos Estimados: 2
Riesgo de desarrollo: Alta	Iteración asignada: 2
Descripción: Se podrá eliminar una no conformidad en la aplicación.	
Observaciones: El usuario debe estar autenticado como administrador.	

Tabla 22. HU Autenticar usuario.

Historia de Usuario	
No: 8	Nombre: Autenticar usuario
Usuario: Usuario autenticado	
Prioridad en el negocio: Alta	Puntos Estimados: 1
Riesgo de desarrollo: Alta	Iteración asignada: 1
Descripción: Da la posibilidad al usuario de autenticarse en la aplicación.	
Observaciones: El individuo debe introducir su usuario y contraseña para autenticarse. Según sus privilegios tendrá acceso a las acciones correspondientes.	

Tabla 23. HU Responder pregunta.

Historia de Usuario

No: 9	Nombre: Responder pregunta
Usuario: Usuario autenticado	
Prioridad en el negocio: Media	Puntos Estimados: 1
Riesgo de desarrollo: Alta	Iteración asignada: 2
Descripción: Se podrá responder una pregunta en la aplicación.	
Observaciones: El usuario podrá responder las preguntas de cada categoría.	

Tabla 24. HU Mostrar informe.

Historia de Usuario	
No: 9	Nombre: Mostrar informe
Usuario: Usuario autenticado	
Prioridad en el negocio: Media	Puntos Estimados: 1
Riesgo de desarrollo: Alta	Iteración asignada: 2
Descripción: Se podrá mostrar un listado de todas las categorías en la aplicación.	
Observaciones: El usuario autenticado como administrador podrá ver un informe de las respuestas por usuario.	

Tabla 25. HU Asignar proyecto a usuario.

Historia de Usuario	
No: 9	Nombre: Asignar proyecto a usuario
Usuario: Usuario autenticado	
Prioridad en el negocio: Media	Puntos Estimados: 1
Riesgo de desarrollo: Alta	Iteración asignada: 2
Descripción: Se podrá asignar un proyecto a uno o varios usuarios en la aplicación.	
Observaciones: El usuario debe estar autenticado como administrador.	

Tabla 26 HU Apariencia.

Historia de Usuario	
No: 10	Nombre: Apariencia
Descripción: El sistema contará con una interfaz amigable, manteniendo el formato en los navegadores.	
Observaciones: El sistema contará con enlaces bien identificados, permitiendo acceder a cualquiera de las secciones con un número mínimo de clics y estará optimizado para una resolución de 1024x768.	

Tabla 27. HU Seguridad.

Historia de Usuario	
No: 11	Nombre: Seguridad
Descripción: La autenticación está garantizada a través de los roles y permisos definidos a cada usuario en el sistema.	
Observaciones: Los usuarios no necesariamente deberán estar autenticados para visualizar algunos de los contenidos que brinda la aplicación. Sin embargo los administradores y moderadores se comportarán como usuarios, pero para poder además, realizar configuraciones de los contenidos o algún cambio en el sistema, tendrán obligatoriamente que estar autenticados. Cada usuario autenticado podrá realizar las operaciones correspondientes con su rol definido en el sistema.	