Universidad de las Ciencias Informáticas

Facultad 4

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

Informatización del procedimiento de Revisiones Técnicas Formales en el área de Requisitos del desarrollo de software para el Centro FORTES

Autoras: Yarisleidy Soca Hernández

Roxana Sánchez Quiala

Tutores: Ing. Lizardo Ramírez Taboada

Dr.C. María Caridad Valdés Rodríguez

Co-tutora: Ing. Lisbeth Oslé Suárez

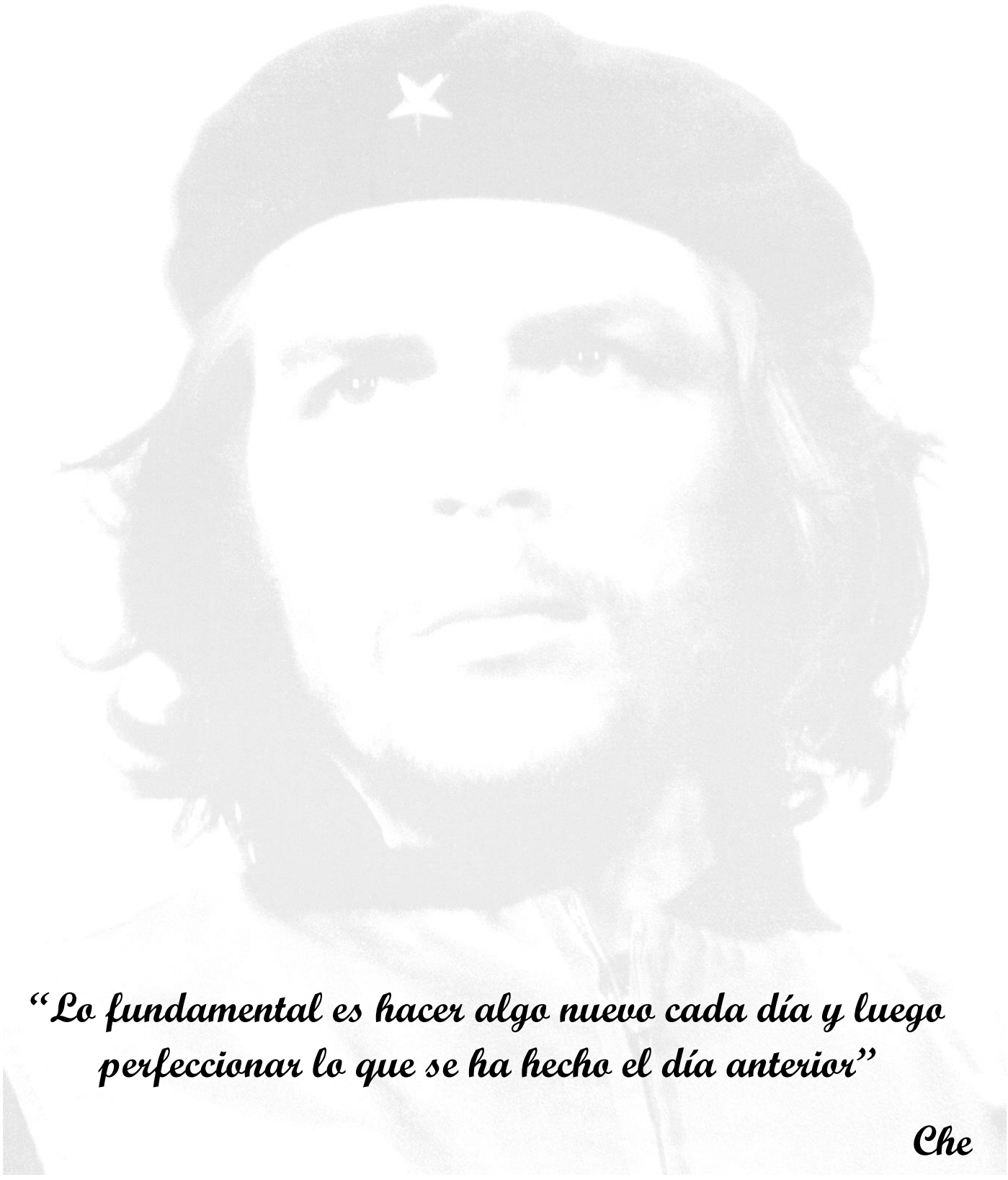
La Habana, junio de 2015
¨Año 57 de la Revolución¨

Declaración de autoría

Declaramos ser las autoras de la presente tesis, reconociendo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de manera exclusiva.

Para que así conste, firmamos la presente a los ___ días del mes de _____ de _____

_____	_____
Yarisleidy Soca Hernández	Roxana Sánchez Quiala
Firma Autor	Firma Autor
_____	_____
Ing. Lizardo Ramírez Taboada	Dr.C. María Caridad Valdés Rodríguez
Firma Tutor	Firma Tutora



"Lo fundamental es hacer algo nuevo cada día y luego perfeccionar lo que se ha hecho el día anterior"

Che

Dedicatoria y Agradecimientos

A la UCI por darnos la oportunidad de convertirnos en profesionales.

A nuestros tutores por guiarnos y confiar en nosotros.

Dedico todo este año de trabajo a la autora de mi vida: mi madre, por apoyarme y confiar en mí cuando más lo necesitaba, a Mayito que aunque no sea mi padre de sangre es mi padre que me ha apoyado y guiado por el buen camino, a mis abuelos por sus consejos, a mi Tía Nieves, por ser mi segunda madre; a toda mi familia por creer en mí, a mi novio por todo su apoyo, dedicación y paciencia, a mis amigos y a todas las personas que de una forma u otra se preocuparon por mí.

Yarisleidy Soca Hernández

Dedicatoria y Agradecimientos

A la UCI por darnos la oportunidad de convertirnos en profesionales.

A nuestros tutores por guiarnos y confiar en nosotros.

A mi madre, por haberme impulsado para lograr mi sueño, a mi hermano y a mis abuelos que con su cariño y dedicación me enseñan a enfrentar mejor la vida, a mis tíos, agradcerle a mi novio, amigos y a todas aquellas personas que me apoyaron y se preocuparon por mí.

Roxana Sánchez Quiala

Resumen

En la Universidad de las Ciencias Informáticas (UCI), se encuentra el Centro de Tecnologías para la Formación (FORTES) donde se desarrollan varios proyectos productivos, a los cuales se le realizan varias pruebas para validar la completitud y las deficiencias que pueden afectar al *software*, permitiendo que al final el cliente pueda recibir un producto con la calidad requerida. En este trabajo se da a conocer la justificación del diseño y la implementación de una herramienta que informatiza parte de una Revisión Técnica Formal (RTF) apegada al desarrollo de *software*, como un apoyo a la demanda de calidad que se requiere del mismo en la etapa de análisis del *software*. Actualmente, en el Centro FORTES de la Facultad 4 no se utiliza una herramienta para realizar el procedimiento del desarrollo de las RTF, en el área de Requisitos, siendo este proceso muy engorroso y hace que se pierda mucho tiempo para realizarlo con alta calidad. Para la elaboración de la aplicación se hizo necesario una investigación sobre el desarrollo del procedimiento de las RTF en la UCI, identificando sus fases, etapas y actividades, dónde posteriormente se analiza en que fase se desarrollan las RTF, en área de Requisitos. Se selecciona la metodología de desarrollo de *software*, además de elegir las herramientas y tecnologías a utilizar. Se determinaron e implementaron todas las funcionalidades pertinentes para el funcionamiento del sistema y se realizaron las pruebas a la solución propuesta, donde se obtuvieron resultados favorables para el equipo de desarrollo, que muestran un alto grado de satisfacción por parte del cliente.

Palabras clave: calidad, desarrollo de *software,* Requisitos, Revisiones Técnicas Formales, *software*.

Índice de contenido

Índice de contenido

Índice de figuras

Índice de tablas

Introducción

En la actualidad se evidencia una evolución considerable de las Tecnologías de la Información y las Comunicaciones (TIC), con el crecimiento de la tecnología, el uso y manejo de la misma, ha cambiado vertiginosamente con respecto a años anteriores. Esto ha permitido un avance considerable en la utilización de la Informática como ciencia, que maneja la información, donde el desarrollo del *software* se ha convertido en uno de los pilares fundamentales. La necesidad de construir un producto con la calidad máxima requerida y que además satisfaga al cliente en la mayor medida posible, es un reto que tienen hoy en día las empresas productoras de *software* [1].

Cuba no ha quedado exenta del uso de las TIC para desarrollar y transformar el país, con el objetivo de lograr una sociedad informatizada. Bajo estas premisas surge la Universidad de las Ciencias Informáticas (UCI), institución que desempeña un papel importante en su utilización; se crea en el año 2010 el Centro de Tecnologías para la Formación (FORTES), el cual brinda servicios y desarrolla productos para tales fines.

Cuando se desarrolla un proyecto de *software*, se pueden introducir defectos, tales como: errores en la captura y especificación de requisitos y en la redacción y ortografía; siendo los mismos arrastrados a las etapas que le continuán, aún sin percibirlo y sin ser detectados en pruebas, tales como: las de unidad, de integración o incluso cuando el *software* ya está en uso. Esto suele ser una práctica común en los ingenieros de *software,* que repercute en la calidad del producto final y en la cantidad del tiempo que incluye en la corrección de los defectos encontrados.

Para minimizar los costos del trabajo en las pruebas y corrección de errores del *software*, es necesario implementar mecanismos que permitan prevenirlos o encontrarlos tan pronto como hayan sido introducidos en el producto de *software*. Muchos de los errores detectados en la construcción o pruebas de los sistemas se deben a las dificultades en las etapas de análisis, especificación o diseño. Por lo que la realización de revisiones formales de los documentos asociados a las distintas etapas del ciclo de desarrollo del *software* permite: obtener alertas tempranas sobre riesgos potenciales y problemas de calidad, reducir los tiempos de desarrollo y pruebas al evitar repetición de trabajos y generar estándares útiles para los distintos ciclos del desarrollo.

Actualmente, el Departamento de Calidad de *Software* de la Universidad de las Ciencias Informáticas, es el responsable de la elaboración del procedimiento para realizar revisiones técnicas formales a la actividad productiva en la UCI. El mismo cuenta con un mecanismo para verificar errores en el área de requisitos, el cual está semi-informatizado. Cuando un especialista ejecuta una RTF a dicha área, es guiado por un documento excel, que contiene una lista de chequeo con la información necesaria para verificar que los entregables de un proyecto han sido cumplidos satisfactoriamente.

Una vez realizada la revisión se procede a conformar un dictamen técnico, el cual incluye los resultados de las revisiones a dichos documentos, donde queda reflejado si la Especificación de Requisitos del Software (ERS) puede ser entregada al cliente. Cuando se lleva a cabo este procedimiento manualmente se pierde mucho tiempo en cada revisión al procesar los resultados y ejecutar el informe del dictamen técnico, lo que impide que no se reutilice la experiencia de otros revisores. No se tiene un resumen de revisiones anteriores realizadas a un proyecto, imposibilitando el seguimiento de las deficiencias encontradas y la verificación de su solución, por lo que en próximas revisiones se pueden encontrar incidencias ya detectadas con anterioridad.

Debido a la situación anteriormente expuesta, se plantea el siguiente **problema científico**: ¿Cómo informatizar el procedimiento de Revisiones Técnicas Formales, en el área de Requisitos del desarrollo de *software* para el Centro FORTES?

El **objeto de estudio** en el cual se centrará la presente investigación lo constituye el proceso de Revisiones Técnicas Formales en el desarrollo de *software*.

Para dar cumplimiento al problema científico se establece como **objetivo general** de la presente investigación: Informatizar el procedimiento de Revisiones Técnicas Formales, en el área de Requisitos del desarrollo de *software* para el Centro FORTES.

A partir del objetivo general se derivan como **objetivos específicos:**

- Investigar acerca del proceso de Revisiones Técnicas Formales, conceptos fundamentales, artefactos y diversas fuentes que analizan el procedimiento en el área de Requisitos del desarrollo de *software*.

- Diseñar una herramienta basada en la captura de los requisitos y las características del sistema que permita la informatización del procedimiento de Revisiones Técnicas Formales, en el área de Requisitos del desarrollo de *software* para el Centro FORTES.

- Realizar la implementación y las pruebas de *software* para la informatización del procedimiento de Revisiones Técnicas Formales, en el área de Requisitos del desarrollo de *software* para el Centro FORTES.

El **campo de acción** lo constituye el proceso de informatización del procedimiento de Revisiones Técnicas Formales, en el área de Requisitos del desarrollo de *software* para el Centro FORTES.

En correspondencia con el problema científico, el objeto de estudio, el objetivo general y los especifícos y el campo de acción, se determinaron las siguientes **preguntas científicas**:

1. ¿Qué fundamentos teóricos, metodológicos y tecnológicos sustentan las tendencias actuales del estudio del desarrollo de las Revisiones Técnicas Formales?

2. ¿Cuál es el estado actual del procedimiento de Revisiones Técnicas Formales, en el área de Requisitos del desarrollo de *software* para el Centro FORTES?

3. ¿Qué tecnologías deben tenerse presente en el diseño de una herramienta para el procedimiento de Revisiones Técnicas Formales, en el área de Requisitos del desarrollo de *software* para el Centro FORTES?

4. ¿Qué resultados se alcanzan con la implementación y la aplicación de las pruebas de *software* para la informatización del procedimiento de Revisiones Técnicas Formales, en el área de Requisitos del desarrollo de *software* del Centro FORTES?

Para garantizar el cumplimiento de los objetivos específicos se proponen las siguientes **tareas de investigación:**

1. Estudio de los fundamentos teóricos, metodológicos y tecnológicos que sustentan las tendencias actuales acerca del proceso de informatización de las Revisiones Técnicas Formales y de del procedimientos en el área de Requisitos del desarrollo de *software*.

2. Diseño de una herramienta basada en los requisitos y las características del sistema que permita la informatización del procedimiento de Revisiones Técnicas Formales, en el área de Requisitos del desarrollo de *software* del Centro FORTES.

3. Implementación y aplicación de las pruebas de *software* para la informatización del procedimiento de Revisiones Técnicas Formales, en el área de Requisitos del desarrollo de *software* del Centro FORTES.

Durante el desarrollo de esta investigación se hizo necesario profundizar en el estudio de los temas abordados, por lo que se utilizaron varios **métodos de investigación:**

Métodos teóricos: seleccionados para crear condiciones de las características superficiales de la realidad y posibilitar el conocimiento del estado del arte del fenómeno, su evolución en una etapa determinada y su relación con otros fenómenos. Los métodos teóricos que se emplearon:

- **Analítico-Sintético:** para estudiar las teorías y documentos necesarios, permitiendo la extracción de los elementos más importantes, para analizar la situación problémica existente y determinar las características que debe tener la solución propuesta, así como para analizar y sintetizar la teoría y la documentación relacionadas con el proceso de pruebas en todos los niveles.
- **Histórico-Lógico:** se utilizó para constatar teóricamente la evolución de las RTF en el área de Requisitos. Para el estudio crítico de los antecedentes asociados a la propuesta que se pretende construir y utilizar estos como punto de referencia y estableciendo una comparación entre los mismos.
- **Inducción-Deducción:** es una forma de razonamiento, mediante el cual se pasa de un conocimiento general del procedimiento de las RTF, sus fases, etapas y actividades, explicando su desarrollo en el área de requisitos.
- **Modelación:** se empleó para analizar de forma general el procedimiento de RTF y así modelar el proceso de revisiones en los requisitos mediante el diagrama de proceso.

Método Empírico: se empleó para el conocimiento del objeto mediante su conocimiento directo y el uso de la experiencia. El método empírico que se aplicaron:

- **Observación científica:** es la percepción planificada dirigida a un fin y relativamente prolongada de un hecho o fenómeno, se realiza de forma consciente y orientada a un objetivo determinado, en este caso dirigido al procedimiento de RTF, en el área de Requisitos del desarrollo de *software* en el Centro FORTES, con el objetivo de determinar las funcionalidades y la arquitectura de la información que debe tener la solución propuesta.

- **Entrevista**: Realizada a la especialista Tamara Rodríguez Sánchez del Centro de Informatización de Entidades, subdirección de Investigaciones la cual es la encargada de realizar las Revisiones Técnicas Formales, en el área de Requisitos, obteniendo información sobre el estado actual del proceso en la organización, con el fin de comprender cómo es su funcionamiento.

La presente investigación se ha organizado en tres capítulos, los cuales abordan la investigación del proceso a informatizar. Se estructura de la siguiente forma:

Capítulo 1: Fundamentación teórica

En este capítulo se presenta la fundamentación teórica de la investigación, describiendo el procedimiento de Revisiones Técnicas Formales, en el área de Requisitos del desarrollo de *software*. Se caracterizan los conceptos que se relacionan con la misma, identificando la metodología, herramientas y tecnologías a usar en el proceso de desarrollo de la aplicación.

Capítulo 2: Propuesta de solución

Se describen, en este capítulo, las funcionalidades y características del sistema y se realiza el diseño de la aplicación, utilizando la metodología de desarrollo seleccionada.

Capítulo 3: Implementación y pruebas

En este capítulo se presenta la fase de implementación y los tipos de pruebas de *software,* utilizados en cada una de las iteraciones, se exponen los resultados y las funcionalidades alcanzadas, durante el período de desarrollo.

Capítulo1. Fundamentación teórica

Introducción

En el presente capítulo se aborda un estudio de los fundamentos teóricos, metodológicos y tecnológicos que sustentan las tendencias actuales, acerca del proceso de informatización del procedimiento de Revisiones Técnicas Formales, en el área de Requisitos. Se conceptualizan varios términos; se identifican las fases, etapas y actividades de cómo se procede en las revisiones técnicas formales, para anallizar el proceso en el área de Requisitos. Se detallan las metodologías de desarrollo, seleccionando la más adecuada para darle cumplimiento a la solución propuesta. Para el desarrollo de la herramienta se analizan varias tecnologías, dentro de ello se especifican los lenguajes de programación, identificando el framework de desarrollo, todo ello costituyen aspectos fundamentales para la solución propuesta.

1.1 Principales conceptos asociados al dominio del problema

Calidad del software

Se estudiaron diversas definiciones existen en determinadas fuentes:

- Roger Pressman (2010) define que es "*La concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente*" [2].
- La norma ISO[1] 8402:1994 plantea que es "*El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas*" [3].
- Según la IEEE[2] estándar 610 – 1900 "*Es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario*" [4].

Para la presente investigación se escoge la definición de calidad de *software* del profesor Roger Pressman, debido a que le da más relevancia a los requisitos y estándares que el *software* debe cumplir.

[1] Del inglés International Organization of Standardization (Organización Internacional de Normalización).
[2] Del inglés Institute of Electrical and Electronics Engineers (Instituto de Ingeniería Eléctrica y Electrónica).

Revisión técnica formal

La Revisión Técnica Formal purifica las actividades de Ingeniería de *Software*, se aplica en diversos momentos del desarrollo para detectar defectos. Valida la completitud y corrección de los entregables de un proyecto, previniendo en forma temprana sobre potenciales problemas y riesgos que puedan derivarse en etapas posteriores del proyecto como inconsistencias, ambigüedades y no cumplimiento de estándares. El servicio consiste en la revisión formal de la documentación de arquitectura, diseño, requerimiento y modelo de datos, con el objetivo de [6]:

- Verificar la consistencia interna de la documentación y su coherencia con los requerimientos.
- Verificar el cumplimiento de estándares del cliente.
- Validar la completitud y facilidad de lectura de la documentación.
- Proponer mejoras, agregados y estándares nuevos.

Los objetivos de la RTF son [7]:

- Descubrir errores en la función, la lógica o la implementación de cualquier representación de *software*.
- Verificar que el *software* bajo su revisión alcanza sus requisitos funcionales.
- Garantizar que el *software* ha sido desarrollado de acuerdo con los estándares predefinidos.

Análisis del software

El análisis del *software* se define de diversas formas, fueron consultadas varias fuentes como:

- Según Roger Pressman: *"Describe como el sistema será realizado a partir de la funcionalidad previa y de las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar"* [8].
- Según la IEEE: *"Es el proceso del estudio de las necesidades de los usuarios para llegar a una definición de los requerimientos del sistema, hardware o software, así como el proceso de estudio y refinamiento de dichos requisitos"* [9].

El objetivo que persigue el análisis del *software*, es identificar con precisión las necesidades de información de una organización y establecer la alternativa de solución más conveniente para satisfacerla [10].

En la Informática, la ingeniería de *software* comienza con una serie de tareas de modelado que conducen a una especificación de requisitos y a una representación completa del diseño del *software* que se

construirá. El modelado del análisis, que en realidad es una serie de modelos, es la primera representación técnica de un sistema.

Modelado del análisis

El modelado del análisis utiliza una combinación de formatos en textos y diagramas para representar los requisitos de los datos, las funciones y el comportamiento de una manera que es relativamente fácil de entender y, aún más importante, conduce a una revisión para lograr la corrección, la integridad y la consistencia [8].

El objetivo del modelado del análisis es crear una variedad de representaciones que muestran los requisitos del *software* para la información, la función y el comportamiento. Esto se logra aplicando dos filosofías diferentes de modelado [8]:

- **Análisis estructurado:** Los datos y el proceso que transforman los datos son entidades separadas. Los objetos de datos se modelan en una forma que define sus atributos y relaciones.
- **Análisis orientado a objeto:** Se centra en la definición de clases y en la manera que estas colaboran entre ellas para efectuar los requisitos del cliente. El UML y el proceso unificado están orientados a objeto de forma predominante.

Requisitos

- Es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio. Se usa en un sentido formal en la ingeniería de sistema, ingeniería de software e ingeniería de requisitos.
- Condición o capacidad que un usuario necesita para poder resolver un problema o lograr un objetivo.
- Condición o capacidad que debe exhibir o poseer un sistema para satisfacer un contrato, estándar, especificación, u otra documentación formalmente impuesta.
- Una condición o capacidad que debe ser conformada por el sistema.
- Algo que el sistema debe hacer o una cualidad que el sistema debe poseer [11].

Los requisitos bien formulados deben satisfacer varias características. Si no lo hacen, deben ser reformulados hasta hacerlo, ejemplos de estas caractersticas [11]:

- **Necesario**: Lo que pida un requisito debe ser necesario para el producto.
- **No ambiguo**: El texto debe ser claro, preciso y tener una única interpretación posible.
- **Conciso**: Debe redactarse en un lenguaje comprensible por los inversores en lugar de uno de tipo técnico y especializado, aunque aun así debe referenciar los aspectos importantes.
- **Consistente**: Ningún requisito debe entrar en conflicto con otro requisito diferente, ni con parte de otro. Asimismo, el lenguaje empleado entre los distintos requisitos debe ser consistente también.
- **Completo**: Los requisitos deben contener en sí mismos toda la información necesaria, y no remitir a otras fuentes externas que los expliquen con más detalle.
- **Alcanzable**: Un requisito debe ser un objetivo realista, posible de ser alcanzado con el dinero, el tiempo y los recursos disponibles.
- **Verificable**: Se debe poder verificar con absoluta certeza, si el requisito fue satisfecho o no. Esta verificación puede lograrse mediante inspección, análisis, demostración o testeo.

Análisis de requisitos

- Según la IEEE estándar 830: *"El análisis de requisitos se puede definir como el proceso del estudio de las necesidades de los usuarios para llegar a una definición de los requisitos del sistema, hardware o software, así como el proceso de estudio y refinamiento de dichos requisitos"* [12].

Especificación de requisitos de software(ERS)

Es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye un conjunto de casos de uso que describe todas las interacciones que tendrán los usuarios con el software. Los casos de uso también son conocidos como requisitos funcionales. Además de los casos de uso, la ERS también contiene requisitos no funcionales (o complementarios). Los requisitos no funcionales son requisitos que imponen restricciones en el diseño o la implementación, como, por ejemplo, restricciones en el diseño o estándares de calidad [13].

Lista de chequeo

Es un formato creado para realizar actividades repetitivas, controlar el cumplimiento de una lista de requisitos o recolectar datos ordenadamente y de forma sistemática. Se usa para hacer comprobaciones sistemáticas de actividades o productos asegurándose de que el trabajador o inspector no olvide nada importante [14].

Los principales usos de una lista de chequeo son los siguientes [14]:

- ✓ Realización de actividades en las que es importante que no se olvide ningún paso y/o deben hacerse las tareas con un orden establecido.
- ✓ Realización de inspecciones donde se debe dejar constancia de cuáles han sido los puntos inspeccionados.
- ✓ Verificar o examinar artículos.
- ✓ Examinar o analizar la localización de defectos. Verificar las causas de los defectos.
- ✓ Verificación y análisis de operaciones.
- ✓ Recopilar datos para su futuro análisis.

Por lo anteriormente expuesto se llega a la conclusión de que las Revisiones Técnicas Formales en el área de Requisitos no es más que una validación y corrección de todas las actividades de la etapa del modelado de análisis de la ingeniería de *software* para prevenir de forma temprana sobre problemas y riesgos que puedan derivarse en etapas posteriores del proyecto.

1.2 Estudio de las tendencias actuales del proceso de Revisiones Técnicas Formales en el desarrollo de software.

Para la presente investigación se realizó un estudio para analizar herramientas centradas en las revisiones técnicas formales que presenten similitudes con la solución propuesta. Se describe el desarrollo del procedimiento de revisiones en la universidad, analizando aquellas particularidades que se necesiten para darle cumplimiento al problema científico.

1.2.1 Sistemas similares

Las soluciones varían desde las más específicas a las más generales, a continuación se muestran algunos ejemplos de las mismas:

Algunas de las herramientas se centran en la inspección del código fuente[3], como:

- ✓ ***CheckMate*** que tiene disponible un grupo de inspecciones de *software* para verificar código fuente en C y C++ de acuerdo a políticas de codificación predeterminadas, es un sistema basado en inspecciones, desarrollado por la Universidad de Minnesota [15].

[3] Texto escrito en un lenguaje de programación que ha de ser compilado o interpretado para ejecutarse en una computadora.

- ✓ **CodeChecker** permite ejecutar reglas de diferentes lenguajes como Java[4], .NET[5], Delphi[6], e integrarlas en un repositorio común y brinda una interfaz de análisis y reportes que facilita la tarea del revisor humano, desarrollada por Pragma Consultores [16].

Unas de las herramientas que llegan más a fondo en cuanto a revisiones de modelos, conformidad o trazabilidad son:

- ✓ **DocTesting** es una herramienta de administración, gestión y planificación de actividades de desarrollo de *software* y aseguramiento de la calidad creada por Pragma Consultores para dar soporte y facilitar el trabajo de los múltiples equipos de proyecto (tanto internos como de cliente). Dispone de múltiples funcionalidades para gestionar requerimientos, pedidos de cambio, casos de prueba e incidentes asociados a distintas aplicaciones y/o proyectos de *software*. Permite obtener información estadística sobre la productividad, la eficiencia y la eficacia del proceso de desarrollo de *software* en su totalidad [16]. A partir de la amplia experiencia con que Pragma Consultores cuenta en el área de Aseguramiento de la Calidad, es que desarrolla docTesting como su herramienta para administración de la actividad de Testing, el mismo facilita la administración y seguimiento de:
  - Los Proyectos que se realizan sobre los distintos productos de *software*.
  - Los requerimientos involucrados en cada proyecto.
  - Los casos de prueba asociados a cada requerimiento.
  - Los incidentes detectados para dichos casos y requerimientos.
  - Los pedidos de cambio a realizar en los distintos proyectos.
- ✓ **Rational RequisitePro** es una herramienta centrada en documentos, que almacena los requisitos asociándolos a documentos (aunque también permite guardarlos directamente en la base de datos). Se auxilia especialmente en el control de cambio de requisitos, con trazabilidad para especificaciones de *software* y pruebas, también ayuda a los equipos de proyectos para gestionar sus necesidades, escribir buenos casos de uso, mejorar la trazabilidad, fortalecer la colaboración,

[4] Lenguaje de programación de propósito general, concurrente y orientado a objetos.
[5] Es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.
[6] Es un entorno de desarrollo de software diseñado para la programación de propósito general con énfasis en la programación visual.

reducir la reanudación del proyecto, y aumentar la calidad. Tiene como principales características [17]:

- Evitar trabajar demás utilizando la integración avanzada, en tiempo real con Microsoft Word.
- Administrar la complejidad con vistas detalladas de trazabilidad que muestran relaciones padres / hijos.
- Mitigar el riesgo del proyecto con la exhibición de los requisitos que pueden ser afectados por los cambios ascendentes o descendientes de requisitos.

Tras haber estudiado las soluciones similares existentes a nivel internacional, se llegó a la conclusión que las mismas constituyen herramientas eficaces en sus correspondientes áreas, pero ninguna resuelve la problemática planteada anteriormente, puesto que cada uno de los sistemas analizados presentan características diferentes a la solución propuesta. Por otra parte, las aplicaciones son diseñadas con herramientas propietarias o no contienen módulos que permitan gestionar la información de los procesos que se tratan en la presente investigación.

1.2.2 Desarrollo del procedimiento de RTF en la Universidad de las Ciencias Informáticas (UCI)

La Universidad de las Ciencias Informáticas comenzó el programa de mejora, donde se ponen en practicas las áreas de proceso del nivel 2 de CMMI, realizando las revisiones de PPQA (Process and Product Quality Assurance[7]), las cuales se encargan de comprobar que están hechas las actividades que se necesitan para desarrollar un *software*, pero no verifica que esten bien. La UCI ha comezado a realizar las Revisiones Técnicas Formales las cuales entran en el nivel 3, este procedimiento se ejecuta con el objetivo de verificar que las actividades de las diferentes etapas del ciclo de desarrollo de un *software* se desarrollen bien.

El procedimiento de RTF permite gestionar de manera adecuada las actividades que se desarrollan y corresponden a un proceso RTF, posibilitando que las mismas sean constantemente examinadas, evaluadas y mejoradas. A continuación se describen en detalles las fases, etapas y actividades correspondientes al procedimiento para el proceso de RTF a la actividad productiva de la UCI.

Fase I: Planificación de la revisión

[7] Traducción al español Aseguramiento de la Calidad del Proceso y del Producto.

1. Coordinar la revisión
   1.1. Analizar programa de la RTF: Se analiza necesidad de realizar una RTF a alguna organización productiva. Se selecciona el personal con posibilidad de llevar a cabo las auditorías. Se notifica a los involucrados la realización de la RTF.
   1.2. Designar el revisor líder.
   1.3. Determinar Viabilidad: disponibilidad de información suficiente y apropiada para planificar la RTF; la cooperación adecuada del revisado, el tiempo y los recursos adecuados.
2. Caracterizar la RTF.
   2.1. Determinar objetivos de revisión
   2.2. Definir alcance de la revisión
   2.3. Definir criterios de la revisión

Acontinuación en la figura 1 se muestra el flujo de planificación de las RTF:

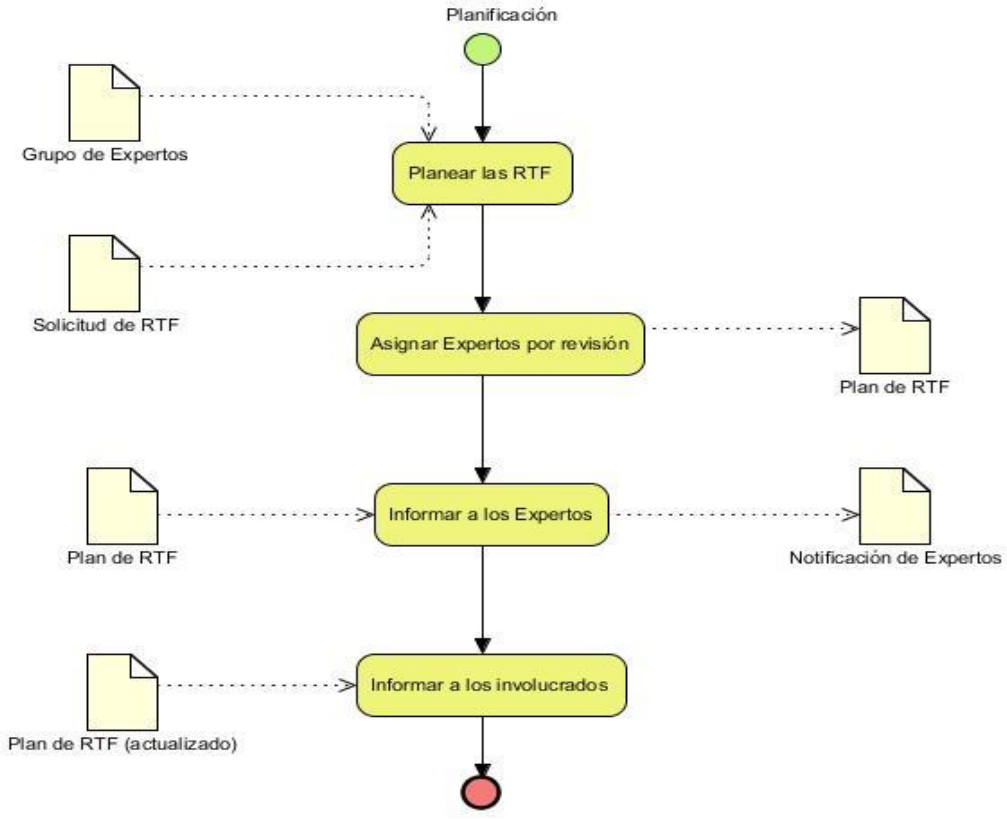


Figura 1. Flujo de Planificación de las RTF

Fase II: Desarrollo de la RTF.

1. Preparar las actividades:

    1.1. Confirmar disponibilidad de recursos

    1.2. Definir los métodos y las técnicas a utilizar

    1.3. Asignar las tareas al equipo revisor

    1.4. Preparar documentos

    1.5. Preparar el plan de la revisión

        Elementos:

        - Proyecto a revisar
        - Líder del Equipo revisor
        - Equipo revisor
        - Objetivo de la RTF
        - Alcance de la RTF
        - Criterios de la RTF
        - Cronograma de las actividades con fecha, hora, lugar y responsables
        - Asignación de los recursos necesarios a las áreas y actividades críticas de la revisión

2. Realizar las actividades:

    2.1. Preparar y realizar reunión de apertura

    2.2. Recopilar y verificar la información

    2.3. Generar los Hallazgos de la RTF

    2.4. Elaborar el Informe preliminar de la RTF

    2.5. Elaborar y presentar acciones correctivas o de mejora

    2.6. Preparar y realizar reunión de cierre

La figura 2 muestra el flujo de ejecución de las RTF:

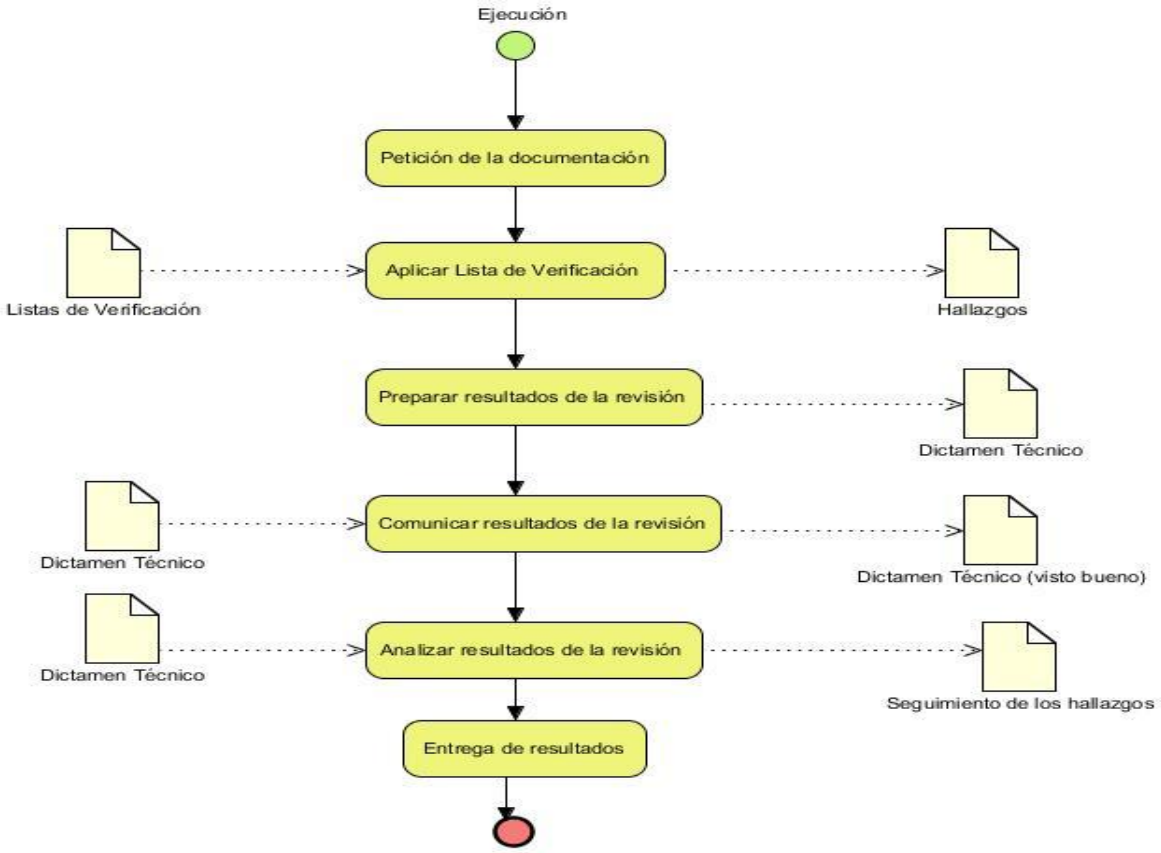


Figura 2. Flujo de Ejecución de las RTF

Fase III: Finalizar la RTF

Tramitar informe de la RTF.

Preparar expediente de la RTF

Almacenar la información

Evaluar desempeño de los revisores

Las evaluaciones de los revisores pueden ser:

- Bien: cuando se cumple con todas las tareas asignadas

- Regular: Cuando se cumple con las tareas asignadas pero con deficiencias
- Mal: Cuando no se cumple con las tareas asignadas

A continuación se muestra en la figura 3 el flujo de seguimiento de las RTF:

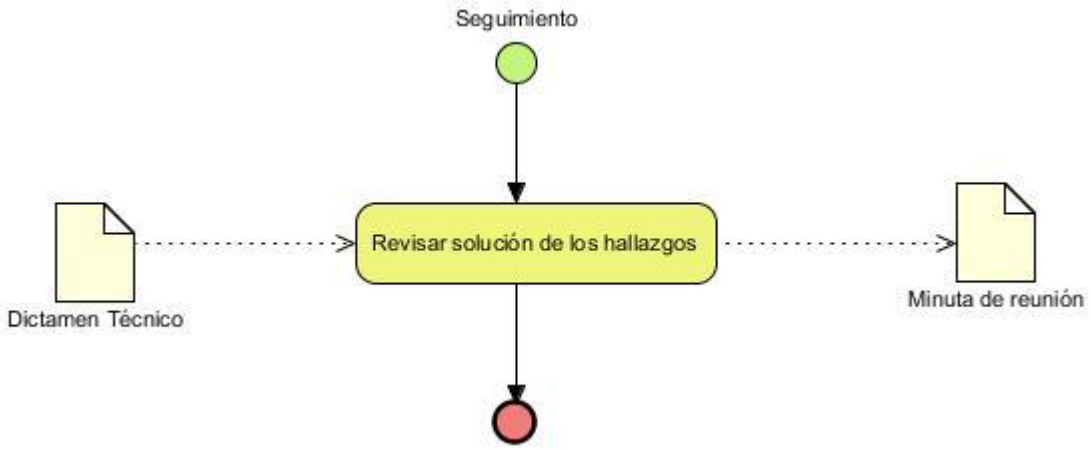


Figura 3. Flujo de Seguimiento de las RTF

Análisis del procedimiento de RTF en el área de Requisitos

El procedimiento de RTF en el área de requisitos que se pretende informatizar comienza en la fase II: Desarrollo de la RTF especifícamente en la etapa dos, realizando las actividades dos, tres y cuatros. Las cuales son las encargadas de responder la lista de chequeo correspondiente a esa área, la misma esta contenida en un documento excel lo cual resulta engorroso por parte de los revisores ejecutar la revisión a un proyecto determinado. Mientras se realiza la revisión se van generando los hallazgos detectados y al final se elabora manualmente un informe del dictamen técnico como resumen de las revisiones.

Las RTF en el área de requisitos se desarrollan al concluir la etapa de requerimiento verificando que los documentos pertenecientes a los requisitos esten descritos correctamente. Este proceso es realizado por un revisor técnico y un revisor líder, los mismos son los encargados de verificar que los documentos que se generar en esta etapa estén completos ya que las siguientes etapas dependen de estos artefactos para una correcta implementación del *software.*

Por lo expuesto anteriormente, se reafirma la necesidad de la creación de un una herramienta que informatice el procedimiento de las Revisiones Técnicas Formales en el área de Requisitos, para dar solución al problema existente.

1.3 Análisis y selección de la metodología de desarrollo

La metodología de desarrollo de *software* es el marco usado para estructurar, planear y controlar el proceso de desarrollo de un proyecto, que en dependencia de la plataforma de desarrollo y el lenguaje de programación, utiliza herramientas que permiten obtener los diferentes artefactos y el producto final. Existen dos tipos de metodologías de desarrollo de *software* conocidas, las cuales tienen similitudes y diferencias entre sí. Estas metodologías son las conocidas como robustas y ágiles, a continuación se analizan cada una de ellas, haciendo referencia a sus características principales [18].

1.3.1 Metodologías robustas

Las metodologías tradicionales o robustas sirven para desarrollar soluciones de *software* complejas. Se basan en años de experiencia en el uso de la tecnología orientada a objetos y en el desarrollo de *software*. Guían a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado mientras se balancean los requerimientos del negocio, el tiempo, el mercado y los riesgos del proyecto. Las metodologías tradicionales buscan seguir una secuencia, en etapas validadas con tecnologías o manuales. Fueron pensadas para soportar la complejidad técnica de proyectos grandes, en los cuales se dificulta la comunicación y sincronización entre los miembros del equipo, precisamente por el tamaño de este [18].

Aunque se ha expresado que también pueden ser utilizadas en proyectos pequeños, la gran cantidad de artefactos, roles y tareas que ellas definen generalmente no se adecuan a las características de dichos proyectos [18].

Dentro de las metodologías de desarrollo tradicionales se encuentra **Rational Unified Process (RUP)**, a continuación se muestran varias características de la misma:

- Centrado en la arquitectura: la arquitectura muestra una visión completa del sistema y se describen los elementos más importantes para la construcción del *software*.
- Dirigido por casos de uso: los casos de uso muestran lo que los usuarios necesitan y desean, esto se obtiene a partir de la modelación del negocio, quedando plasmado en la especificación de requisitos.

- Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Es muy útil dividir el proyecto en mini proyectos ya que el desarrollo de cada uno de ellos es una iteración que contribuye al incremento del proyecto general.
- RUP propone 9 flujos de trabajo: 6 de ingeniería y 3 de apoyo. Los flujos de trabajo ingenieriles son: Modelamiento del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, y Despliegue [18].

Microsoft Solution Framework (MSF) es considerada una metodología de desarrollo tradicional empleada por Microsoft en sus procesos de desarrollo. Es un modelo basado principalmente en los ciclos de vida espiral y cascada (hitos y fases). Tiene como principios fundamentales la comunicación entre cliente y desarrollador es decir acepta cambios en etapas avanzadas del proceso, capacitación constante de las personas, cumple con el proceso de formación de personal, compartir los roles entre todo el equipo de trabajo. Además es un proceso que se basa en la producción de versiones para el negocio de cada cliente, debe ser ágil, ya que es menos abultado que RUP.

MSF se compone de 2 modelos y 3 disciplinas. Modelo de Equipo y de Proceso. Disciplina de Administración de Proyecto, Administración de Riesgos y Administración de la Preparación, es un proceso muy largo, pero a la vez es muy solvente para la solución de un problema, este modelo no utiliza UML. [19]

1.3.2 Metodologías ágiles o ligeras

Las metodologías ágiles cambian significativamente algunos de los énfasis de los métodos ingenieriles. La diferencia inmediata es que son menos orientadas a la documentación de todo el proceso de desarrollo, exigiendo una cantidad más pequeña de documentos para una tarea dada. De muchas maneras son más bien orientadas al código, siguiendo un camino que dice que la parte más importante de la documentación es el código fuente [18].

Estas características hacen a las metodologías ágiles idóneas para el desarrollo que se propone, ya que se cuenta con un equipo pequeño (solo dos personas), poco tiempo de desarrollo, no es necesaria una documentación extensa, es decir, solo se generan los documentos necesarios en cada momento del desarrollo, la colaboración e intercambio con el cliente será constante debido a que el mismo se encuentra cerca y disponible, convirtiéndose en otro miembro del equipo de desarrollo.

Utilizar una metodología de desarrollo ágil además brinda la posibilidad de adquirir habilidad para responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, la tecnología, etc.), ya que estas metodologías tienen como principio una planificación flexible y abierta. A continuación se analizan algunas de ellas:

Programación extrema (XP)

Dentro de las metodologías ágiles, se encuentra XP. Esta es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de *software*, ya que promueve el trabajo en equipo, se preocupa por el aprendizaje de los desarrolladores y propicia un buen clima de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

Esta metodología se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico [20]. Para la especificación de las funcionalidades del sistema se puede utilizar algo tan sencillo como tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer (Historias de Usuarios (HU)), sean requisitos funcionales o no funcionales. El tratamiento de las HU es muy dinámico y flexible. Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla a lo sumo en varias semanas.

Una característica distintiva de XP es la programación en parejas, con el objetivo de que el código sea revisado y validado antes de ser escrito; la refactorización de código[8] está presente durante todo el desarrollo, y no es más que escribir el mismo código fuente nuevamente buscando claridad, pero sin cambiar la funcionalidad resultante. En la figura 4 se muestra el ciclo de desarrollo de XP:

[8] Es una técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.

Figura 4. Ciclo de desarrollo de XP (20).

Metodologías cristal

Se trata de un conjunto de metodologías para el desarrollo de *software* caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn[9]. El desarrollo de *software* se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (tres a ocho miembros) y Crystal Orange (veinticinco a cincuenta miembros) [21].

Para realizar un proyecto de desarrollo de *software* con éxito deben estudiarse las metodologías que guiarán el proceso, teniendo en cuenta las características del propio proyecto en sí para poder adaptar dicha metodología al contexto donde se empleará la misma (recursos técnicos y humanos, tiempo de desarrollo, tipo de sistema, etc.). Históricamente, las metodologías tradicionales o robustas han intentado abordar la mayor cantidad de situaciones de contexto del proyecto, exigiendo un esfuerzo considerable para ser adaptadas, sobre todo en proyectos pequeños y con requisitos muy cambiantes. Una de las cualidades más destacables de las mismas es su sencillez, tanto en su aprendizaje como en su aplicación, reduciéndose así los costos de implantación en un equipo de desarrollo.

[9] Alistair Cockburn es uno de los iniciadores del movimiento ágil en el desarrollo de software.

Al estudiar las metodologías ágiles de desarrollo de *software* se llegó a la conclusión que se deben tener en cuenta varios puntos definitorios tales como: la magnitud del proyecto, la cantidad de miembros del equipo, el tiempo de que se dispone para su culminación, la disponibilidad del cliente y la facilidad de intercambio con los desarrolladores, así como las expectativas y la experiencia de los mismos, entre otros.

A continuación se muestra una tabla con las características del equipo de desarrollo:

Características	**Valoración**
Composición del equipo de desarrollo	3 personas ✓ 1 Cliente ✓ 2 Desarrolladores
Experiencia productiva	(bajo, medio, alto) ✓ Medio
Conocimiento de metodologías ágiles	(bajo, medio, avanzado) ✓ Cliente (medio) ✓ Desarrolladores (medio)
Conocimiento de metodologías tradicionales	(bajo, medio, avanzado) ✓ Cliente (medio) ✓ Desarrolladores (bajo)
Cliente	Se encuentra dentro de la universidad, específicamente en el Centro FORTES
Relación con el cliente	Forma parte del equipo de desarrollo

Disponibilidad de tiempo	Limitada
Retroalimentación cliente-equipo desarrollo	(continua, discontinua) ✓ continua

Tabla 1. Características del equipo de desarrollo (Elaboración propia)

Según el estudio realizado, se decidió utilizar la metodología ágil XP, porque se adapta a las necesidades debido a que no sigue un régimen muy estricto para ponerla en uso, es más conocida y usada que las demás metodologías ágiles por lo que está más probada y la documentación de consulta para el aprendizaje de la misma es amplia y fácil de encontrar. Además los requisitos tienden a cambiar frecuentemente, el cliente puede agregar nuevas historias de usuarios, dividirlas o simplemente eliminarlas. El cliente forma parte del equipo de desarrollo, logrando una mejor retroalimentación, corrección de errores y así un producto que satisfaga todas las necesidades del mismo. Esta metodología brinda la posibilidad de desarrollar un producto y probarlo, permitiendo terminarlo e integrarlo. Además, al tener un enfoque en el trabajo en grupo, es más fácil dividir las contribuciones al proyecto.

1.4 Lenguaje Unificado de Modelado (UML)

UML: *"…es un lenguaje de modelado visual que se usa para especificar, visualizar, construir, y documentar artefactos de un sistema de software".* [22]

El Lenguaje Unificado de Modelado proporciona a los desarrolladores un vocabulario que incluye tres categorías: elementos, relaciones y diagramas. En la siguiente figura se muestra el vocabulario de UML:

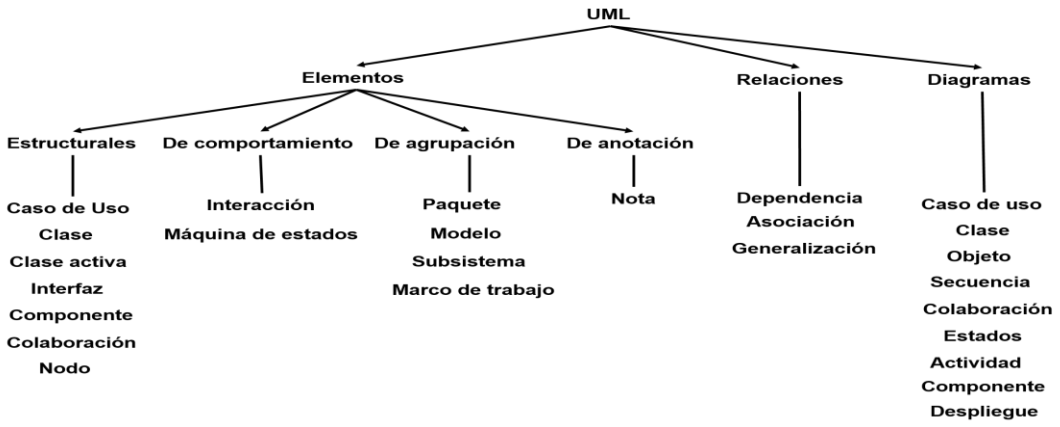


Figura 5. Vocabulario UML (22)

UML está pensado para ser empleado en herramientas interactivas de modelado visual que tengan generadores de código y/o generadores de informes. La especificación de UML no define un proceso estándar pero está ideado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos [22].

La capacidad de UML para modelar cualquier sistema de software y el hecho de que haya sido adoptado por el Grupo de Gestión de Objetos como un estándar desde noviembre de 1997, permitieron determinarlo como el lenguaje de modelado para desarrollar la propuesta de solución.

1.5 Análisis y selección del framework de desarrollo

"Un frameworks de desarrollo es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software" [23].

Los *framework* suelen incluir soportes de programas, bibliotecas, lenguajes de scripting y *software* para desarrollar y unir diferentes componentes de un proyecto de desarrollo de programas. Los mismos permiten facilitar el desarrollo de *software* y evitar los detalles de bajo nivel, permitiendo concentrar más esfuerzo y tiempo en identificar los requerimientos de *software*

Symfony2

Es la versión más reciente de Symfony, el mismo se ha convertido en uno de los *frameworks* de PHP más populares gracias a sus características avanzadas y su gran documentación. Se anunció por vez primera en el 2009, y supone un cambio radical tanto en la arquitectura interna como en la filosofía de trabajo respecto a sus versiones anteriores. Symfony2 ha sido ideado para exprimir al límite todas las características de PHP5.5 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 tu proyecto. En aplicaciones web reales, la diferencia de rendimiento entre Symfony 2.0 y 1.2 no es tan grande, la clave del nuevo Symfony 2.0 es el "gestor de peticiones" o Request Handler[10], que a su vez es la pieza clave de la renovada parte del controlador. Symfony 2.0 es siete veces más rápido que Symfony 1.2 y cinco veces más rápido que Zend *Framework*. [24].

Yii

Es un *framework* PHP de alto rendimiento basado en componentes que promueve el diseño limpio, seco y fomenta el rápido desarrollo web, viene con una herramienta de línea de comandos denominada "yiic" que puede crear una aplicación Yii esqueleto. Sus principales características son [25]:

- Modelo-Vista-Controlador (MVC)
- Objetos de acceso a base de datos (DAO), Query Builder[11], Active Record, y DB Migración
- Reproductores compatibles con AJAX[12]
- Potente soporte de almacenamiento en caché - el almacenamiento en caché de datos, páginas caché, caché de fragmentos y contenido dinámico
- Autenticación
- Control de acceso basado en roles
- Las pruebas unitarias y funcionalidad
- Validación de entrada
- Filtrado de salida
- Inyección SQL[13]

[10] Significado: Controlador de solicitudes.
[11] Significado: Generador de consultas.
[12] Del inglés **A**synchronous **J**avaScript **A**nd **X**ML.

Al analizar varios *framework* de desarrollo se llega a la conclusión de usar Symfony2, la versión a utilizar será la 2.3.0, liberada el 1 de junio del 2014. La selección se basó en las características que posee, entre las que se destacan:

- Organiza el código fuente en una estructura de tipo proyecto y almacena los archivos del proyecto en una estructura estandarizada de tipo árbol. Permitiendo a un proyecto poseer una o varias aplicaciones, a una aplicación poseer uno o varios módulos, y a un módulo poseer una o varias acciones. Dicha estructura permite escribir un código comprensible para un futuro mantenimiento.
- Las distintas aplicaciones que conformen un proyecto pueden compartir el mismo modelo de datos y las librerías.
- Implementa el patrón arquitectónico Modelo-Vista-Controlador, el cual permite separar la lógica del negocio, la lógica del servidor y la presentación, obteniendo así un producto que cumpla con las buenas prácticas del desarrollo de aplicaciones web actuales [24].

1.6 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"* [26] Los lenguajes de programación para el desarrollo de aplicaciones web se agrupan en dos: lenguajes del lado del cliente y del lado del servidor.

1.6.1 Lenguaje de programación del lado del cliente

Los lenguajes de programación del lado del cliente son aquellos que pueden ser directamente "digeridos" por el navegador y no necesitan un pre tratamiento [27]. Existen varios lenguajes de programación del lado del cliente, como son: VBScript, JavaScript, HTML, DHTML, CSS; sin embargo, en la presente investigación se analizan JavaScript, HTML y CSS pues son los utilizados en Symfony2.

JavaScript: Es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como aparición y desaparición de texto, animaciones, acciones que se activan al pulsar botones u otros elementos y ventanas con mensajes

[13] Es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación.

de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos, es decir, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios [28].

HTML: El Lenguaje de Etiquetado de Hipertexto es un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado W3C[14]. Como se trata de un estándar reconocido por todas las empresas relacionadas con el mundo de Internet, una misma página HTML se visualiza de la misma manera en cualquier navegador de cualquier sistema operativo. La W3C define el lenguaje HTML como "Un lenguaje comúnmente utilizado para la publicación de hipertexto en la Web y desarrollado con la idea de que cualquier persona o tipo de dispositivo pueda acceder a la información en la Web. HTML utiliza etiquetas que marcan elementos y estructuran el texto de un documento" [29].

La última versión de este lenguaje es la conocida HTML5, la misma está pensada con una mayor integración con los lenguajes CSS y JavaScript. HTML5 ha revolucionado la web y no solo se ve como el presente, sino como el futuro, por las numerosas novedades que trae con respecto a la versión anterior. La selección de HTML5 como el lenguaje para realizar el maquetado de la propuesta de solución está sustentada en las novedosas herramientas que incorpora para el desarrollo web.

CSS: Las hojas de estilo en cascada (CCS) son complementos de código añadidos al HTML que se encargan de la apariencia del documento. El concepto de hojas de estilo reside en el principio de la separación entre el contenido y el formato en la elaboración de documentos HTML. Las hojas de estilo pueden gestionar así todo lo que concierne a la apariencia, dejando al HTML la función de estructurar y de codificar la información bruta [30].

Las CSS ofrecen propiedades para ampliar el lenguaje HTML en la representación visual de las páginas web. Es el más conocido y utilizado para definir las propiedades de formato de los diferentes elementos HTML.

La última versión del lenguaje CSS es la versión 3 (CSS3), la misma fue seleccionada para escribir el estilo visual de la propuesta de solución, teniendo en cuenta que posee un amplio uso y difusión por parte de maquetadores, diseñadores y desarrolladores web.

[14] Del inglés World Wide Web Consortium.

1.6.2 Lenguaje de programación 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. Dentro de estos lenguajes se encuentran PHP, Perl, ASP[15], JSP[16]. En la presente investigación se analiza el lenguaje PHP pues es el utilizado en Synfony2 y sus módulos.

PHP: Es un lenguaje de código abierto interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. PHP puede hacer cualquier tarea, como procesar la información de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies [31]. La versión a utilizar será la 5.5. Este lenguaje brinda una serie de ventajas que se citan a continuación:

- Muy fácil de aprender
- Se caracteriza por ser un lenguaje muy rápido
- Soporta en cierta medida la orientación a objeto, clases y herencias
- Es un lenguaje multiplataforma
- Capacidad de conexión con la mayoría de los manejadores de base de datos: MysSQL, PostgreSQL y Oracle
- Capacidad de expandir su potencial utilizando módulos
- Posee documentación en su página oficial la cual incluye descripción y ejemplos de cada una de sus funciones
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos
- Incluye gran cantidad de funciones
- No requiere definición de tipos de variables ni manejo detallado de bajo nivel

1.7 Análisis y selección del sistema de gestión de base de datos

Existen varias formas de guardar información, una de las más conocidas y seguras son las bases de datos las cuales desempeñan un papel crucial en casi todas las áreas de aplicaciones de las computadoras.

[15] Del inglés Microsoft ActiveServer Pages
[16] Del inglés Java Server Pages

En la actualidad existen muchos sistemas gestores de bases de datos, tanto libres como propietarios. Se tuvieron en cuenta dos gestores de datos muy populares, MySQL y PostgreSQL que a continuación se describen:

1.7.1 MySQL

Es un sistema de gestión de base de datos relacional, multihilo, que aprovecha la potencia de sistemas multiprocesador. Es además multiusuario, soporta gran cantidad de tipos de datos. MySQL es un gestor de base de datos muy rápido en la lectura, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación [32].

En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones. MySQL presenta un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor [33].

1.7.2 PostgreSQL

Es un sistema de gestión de base de datos relacional orientado a objetos y libre. Como muchos otros proyectos de código abierto, el desarrollo de postgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, el mismo permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Esta estrategia es superior al uso de bloqueos por tabla o por filas comunes en otras bases de datos, eliminando la necesidad del uso de bloqueos explícitos, soporta distintos tipos de datos, además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC[17], IP[18]), cadenas de bits [34].

También permite la creación de tipos propios e incorpora una estructura de datos de arreglos, funciones de diversa índole como manejo de fechas, geométricas, orientadas a operaciones con redes, etc. Permite la declaración de funciones propias, así como la definición de disparadores, soporta el uso de índices,

[17] Del inglés Media Access Control (control de acceso al medio)
[18] Del inglés Internet Protocol

reglas y vistas, incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales [34].

PostgreSQL es un magnífico gestor de bases de datos, capaz de competir con muchos gestores comerciales, aunque carezca de alguna característica casi imprescindible como un conjunto de herramientas que permitan una fácil gestión de los usuarios y de las bases de datos que contenga el sistema. Por otro lado, la velocidad de respuesta que ofrece este gestor con bases de datos relativamente pequeñas puede parecer un poco deficiente, aunque esta misma velocidad la mantiene al gestionar bases de datos realmente grandes [34].

Con respecto a MySQL, PostgreSQL tiene mayor escalabilidad, integridad de los datos, necesita mayores requerimientos de hardware, ambos gestores de bases de datos se integran con Symfony 2. PostgreSQL intenta ser un sistema de bases de datos a la altura de oracle, sybase o Interbase. Si bien es cierto que la velocidad en responder consultas simples es un poco más lenta, ofrece mejor rendimiento bajo grandes cargas de trabajo. Además, escala mucho mejor, tanto en términos de la utilización de un hardware de alto rendimiento, como al hacer frente a la concurrencia, de igual forma, ofrece una garantía de integridad en los datos mucho más fuerte, lo cual es necesario, ya que no se puede permitir que se corrompa o se pierda ni un solo registro.

Con la evidencia anterior y agregándole además la experiencia del equipo de desarrollo con este gestor de base de datos, se decidió usar postgreSQLv9.2.4 por ser la versión estable más recientemente liberada.

1.8 Análisis y selección del servidor de aplicaciones

Un servidor web es un tipo de servidor que permite el procesamiento de datos de una aplicación de cliente [35]. A continuación solo se realiza el análisis de los servidores web Apache e Internet Information Server (IIS); dado que el lenguaje de programación PHP (elegido para la elaboración del sistema) para su funcionamiento necesita tener instalado uno de estos dos servidores.

Internet Information Server

"Internet Information Server o IIS es un servidor de Microsoft destinado a la publicación, mantenimiento y gestión de páginas y portales web" [36]. IIS es privativo, razón suficiente por la cual no se procede al posterior análisis del mismo, teniendo en cuenta que la UCI aboga por el uso de tecnologías gratuitas y de código libre.

Apache

Es un servidor web que se distingue por las siguientes características [37]:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Es una tecnología gratuita de código fuente abierta.
- Altamente configurable de diseño modular. Es muy sencillo ampliar sus capacidades, actualmente existen muchos módulos para Apache que son adaptables, con facilidades de instalación según se necesiten.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.
- Permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor.

Por tanto se elige Apache como servidor web, teniendo en cuenta que es gratuito, multiplataforma, eficiente y seguro. Además su selección también viene respaldada por la posibilidad de descarga a través de internet, lo que posibilita la actualización del mismo. En este caso se utiliza la versión 2.2.22.

1.9 Herramientas para el proceso de desarrollo

Una herramienta de desarrollo de software es un programa informático que usa un programador para crear, depurar, gestionar o mantener un programa [38]. A continuación se hará un breve resumen de las herramientas necesarias para el desarrollo del software que le dará respuesta a la situación problémica.

1.9.1 Análisis y selección del entorno de desarrollo integrado (IDE)

Un IDE es un sistema que facilita el trabajo del desarrollador de *software*, integrando sólidamente la edición orientada al lenguaje, la compilación o interpretación, la depuración y las medidas de rendimiento [39]. Dentro de los IDEs más utilizados a nivel mundial se encuentran NetBeans y Eclipse, descritos a continuación:

Eclipse

Es una herramienta para desarrollo de *software*, creado por IBM[19] inicialmente para desarrollar en Java, y posteriormente ha brindado soporte a otros lenguajes de programación como PHP, Ruby on Rails y C/C++. [40].

Características:

- Potente editor de texto con la capacidad de resaltar sintaxis, así como también una compilación en tiempo real y soportes para (AWT/SWING[20]) y web (Servlets[21], jsp).
- Vinculación con herramientas de control de versiones como Subversion, Git y Bazaar.
- Capacidad para la integración de plugins.

NetBeans

Netbeans es un entorno de desarrollo integrado libre, diseñado principalmente para el lenguaje de programación Java, es un producto libre y gratuito sin restricciones de uso [41]. Es multiplataforma lo que le permite a los desarrolladores crear con rapidez aplicaciones utilizando la plataforma Java, así como PHP, JavaScript, Ajax, Ruby on Rails, Groovy[22], y C/C++.

Características:

- Cuenta con un potente editor de código que permite el completamiento y resaltado del código. Además el editor brinda soporte a una gran variedad de lenguajes, como Java, C/C++, XML, HTML, PHP, Groovy, Javadoc, JavaScript y JSP[23].
- Ofrece integración con herramientas de control de versiones como Subversion, Mercurial y Git que permiten gestionar eficientemente proyectos de menor y mayor envergadura.
- Posee un depurador de código muy completo, el cual permite crear puntos de ruptura, tracear funciones, observar el estado de variables y objetos almacenados en memoria. Incluye también un depurador visual para aplicaciones Java SE, lo que le permite depurar las interfaces de usuario sin mirar el código fuente.

[19] Del inglés International Business Machines Corp. Es una empresa multinacional estadounidense de tecnología y consultoría con sede en Armonk, Nueva York

[20] Es una API para proporcionar una interfaz gráfica de usuario.

[21] Es una clase en el lenguaje de programación Java-

[22] Es un lenguaje ágil y dinámico para la máquina virtual de Java

[23] Del inglés JavaServer Pages

- Fácil instalación de plugins.

Teniendo en cuenta las características analizadas de cada IDE, aunque son bastante similares, se selecciona NetBeans en su versión 8.0 para llevar a cabo la implementación de la solución propuesta. El uso de Netbeans facilita el trabajo para el equipo de desarrollo, debido a la experiencia que se tiene con el mismo.

1.9.2 Cliente del gestor de bases de datos

PgAdmin III: Es una herramienta de código abierto para la administración de bases de datos PostgreSQL y derivados (EnterpriseDB Postgres Plus Advanced Server y Greenplum Database). Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, soporte para el motor de replicación Slony-I. La conexión al servidor puede hacerse mediante conexión TCP/IP[24] o Unix Domain Sockets (en plataformas *nix), y puede encriptarse mediante SSL[25] para mayor seguridad. Está disponible en más de una docena de lenguajes y para varios sistemas operativos, incluyendo Microsoft Windows ™, Linux, FreeBSD, Mac OSX y Solaris [42].

1.9.3 Herramienta de modelado

Visual Paradigm for UML 8.0: es una Herramientas de Ingeniería de Software Asistida por Computadora (CASE: Computer Aided Software Engineering), diseñada para la ayuda al desarrollo de software. Soporta estándares de la industria clave, tales como Lenguaje de Modelado Unificado (UML). Ofrece un completo conjunto de herramientas de los equipos de desarrollo de software necesarios para la captura de requisitos, la planificación de programas, la planificación de controles, la clase de modelado así como el modelado de datos. Permite realizar ingeniería tanto directa como inversa. Además, la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como Web o Pdf, y permite control de versiones.

[24] TCP del inglés Transmission Control Protocol (Protocolo de Control de Transmisión).
IP del inglés Internet Protocol (Protocolo de Internet).
[25] Del inglés Secure Sockets Layer (capa de conexión segura).

Conclusiones del capítulo

Luego del análisis del objeto de estudio, apoyado en los métodos de la investigación científicos, se concluye lo siguiente:

- ✓ Se requiere de una fundamentación teórica, metodológica y tecnológica para el analísis, diseño e implementación de una herramienta que imformatize el procedimiento del desarrollo de las Revisiones Técnicas Formales, en el área de Requisitos, así como las actividades más importantes para sustentar la propuesta de solución informática que se asume.
- ✓ Con el analísis y selección de la metodología XP se determinó que la misma constituiría la guía del proceso de desarrollo por sus características como la disponibilidad con el cliente, el tiempo que se necesita para la implementación de la solución, así como la magnitud y su complejidad.
- ✓ La integración del *framework* Symfony 2.0 soportado por los lenguajes del lado del cliente HTML 5, CSS3, y JavaScript, y del lado del servidor PHP 5.5, así como del sistema de gestor de base de datos PostgresSQL 9.2.4, el servidor web Apache 2.2.22, el IDE NetBeans 8.0.1 y el Visual Paradigm 8 para el modelado; constituye una base sólida para el desarrollo de la solución, su adaptabilidad, optimización como tecnologías libres.

Capítulo2. Propuesta de solución

Introducción

Se describen, en este capítulo, la propuesta de solución, las funcionalidades y características del sistema y se explica acerca del diseño de la aplicación, con el uso de la metodología de desarrollo, identificando sus fases con las correspondiente actividades, lo cual permite la informatización del procedimiento del desarrollo de las Revisiones Técnicas Formales, en el área de Requisitos del Centro FORTES. Se hace referencia a lo relacionado con las fases de Exploración y Planificación, en las que se detallan las Historias de Usuario (HU) para establecer el orden en que serán implementadas, atendiendo a su prioridad. Se determinan las fases de Diseño e Implementación, detallando los principales patrones y modelos a usar para lograr la realización de la solución.

2.1 Propuesta del sistema

Con el presente trabajo se pretende obtener una aplicación web que facilite el procesamiento de los resultados de las revisiones realizadas a los distintos proyectos del Centro FORTES, la misma brindará la opción de registrar todas las RTF, en el área de Requisitos realizadas en el centro, posibilitando tener conocimiento de las mismas y permitiendo guardar todos los resultados encontrados en cada revisión, al realizar dicha revisión se genera automáticamente el dictamen técnico como resumen de la revisión y se guarda otro documento con un resumen de los hallazgos detectados.

El procedimiento de revisión es ejecutado una vez que se introduzca el proyecto que se desea revisar, unas vez registrado se procede a adicionar la RTF, en la misma se escoge la lista de chequeo correspondiente a verificar los requisitos, la cual contiene características específicas de los proyectos tales como: corrección, completitud, no ambigüedad, consistencia, comprobabilidad, modificabilidad, requisitos clasificados, abstracción y rastreabilidad, donde en las mismas se describen los indicadores necesarios para evaluar los documentos de requisitos. Los indicadores se basan en preguntas, las cuales generan un método de evaluación, especificando si los requisitos están: cumplido, no cumplido, parcialmente cumplido y no aplica. También permite a los revisores con menos experiencia guiarse a partir de comentarios más usados y poder tener un mejor conocimiento de los problemas que pueden existir al evaluar un indicador, permitiendo registrar hallazgos detectados en ese indicador. Cada característica además de la descripción

y los indicadores que en ellas se definen presenta sus propios datos, los mismos son definidos al crear una nueva característica y sus valores pueden ser modificados.

Además cuenta con un control de versiones, el cual se crea automáticamente cuando se incluye una característica en el sistema, tomando como versión inicial del mismo el número 0.1 y este control de versiones puede ser, a partir de esta versión inicial, modificado a razón del cliente. El sistema además es capaz de administrar los usuarios, gestionando de cada uno sus permisos, datos específicos y los proyectos a los que podrá acceder y por ende, agregarle o eliminarle proyectos.

El cliente definió una serie de funcionalidades que debe cumplir la aplicación, por cada funcionalidad definida se hace una HU, especificando de forma sencilla como resolver el problema planteado.

2.2 Funcionalidades del sistema

La Informatización del proceso de HRTFR debe permitir.

- **RF1 Gestionar usuario**
    1. RF Insertar un nuevo usuario
    2. RF Modificar usuario
    3. RF Eliminar usuario
    4. RF Consultar usuario
    5. RF Ver los detalles de un usuario
- **RF2 Gestionar permisos.**
    1. RF Insertar permiso
    2. RF Modificar permiso
    3. RF Eliminar permiso
    4. RF Consultar permiso
    5. RF Ver los detalles de los permisos
- **RF3 Gestionar proyecto**
    1. RF Insertar un nuevo proyecto
    2. RF Modificar proyecto
    3. RF Eliminar proyecto
    4. RF Consultar proyecto
    5. RF Ver los detalles de un proyecto

- **RF4 Gestionar lista de chequeo**
    1. RF Insertar lista de chequeo
    2. RF Modificar lista de chequeo
    3. RF Eliminar lista de chequeo
    4. RF Consultar lista de chequeo
    5. RF Ver los detalles de la lista de chequeo
- **RF5 Gestionar revisiones técnicas formales**
    1. RF Insertar revisiones técnicas formales
    2. RF Modificar revisiones técnicas formales
    3. RF Eliminar revisiones técnicas formales
    4. RF Consultar revisiones técnicas formales
    5. RF Ver los detalles de la revisiones técnicas formales
- **RF6 Gestionar hallazgos**
    1. RF Insertar hallazgos
    2. RF Modificar hallazgos
    3. RF Eliminar hallazgos
    4. RF Consultar hallazgos
    5. RF Ver detalles de los hallazgos
- **RF7 Gestionar nomenclador_importancia**
    1. RF Insertar NImportancia
    2. RF Modificar NImpostancia
    3. RF Eliminar NImportancia
    4. RF Consultar NImportancia
    5. RF Ver detalles de los NImportancia
- **RF8 Gestionar nomenclador_tipohallazgo**
    1. RF Insertar NTipohallazgo
    2. RF Modificar NTipohallazgo
    3. RF Eliminar NTipohallazgo
    4. RF Consultar NTipohallazgo
    5. RF Ver detalles de los NTipohallazgo

- **RF9 Gestionar nomenclador_estadohallazgo**
    1. RF Insertar NEstadohallazgo
    2. RF Modificar NEstadohallazgo
    3. RF Eliminar NEstadohallazgo
    4. RF Consultar NEstadohallazgo
    5. RF Ver detalles de los NEstadohallazgo
- **Rf10 Gestionar nomenclador_evaluación**
    1. RF Insertar NEvaluación
    2. RF Modificar NEvaluación
    3. RF Eliminar NEvaluación
    4. RF Consultar NEvaluación
    5. RF Ver detalles de los Nevaluación
- **RF11 Gestionar documentos**
    1. RF Insertar documentos
    2. RF Modificar documentos
    3. RF Eliminar documentos
    4. RF Consultar documentos
    5. RF Ver detalles de los documentos
- **RF12 Generar dictamen técnico**
- **RF13 Generar resumen de hallazgos**
- **RF14 Mostrar comentarios mas usados en una revisión técnica formal**
- **RF15 Exportar reporte a formato Word**

2.3 Características del sistema

Para un correcto funcionamiento de la aplicación se deben tener en cuenta las características del sistema, a continuación se muestra un listado de estas características para la solución propuesta:

Usabilidad

No se necesitará una preparación previa para operar con el sistema. Se requiere un nivel bajo de conocimientos de computación, el manejo de la aplicación es sencillo, permitiendo la fácil comprensión por el usuario.

Disponibilidad

Es necesaria la ejecución de la aplicación durante toda la jornada de trabajo en el proyecto donde se utilice, en este caso el Centro FORTES, para la actualización de la base de datos en tiempo real. Esta característica es fundamental debido a que continuamente se obtendrán datos importantes en los registros generados por el equipo de revisión del proyecto.

Eficiencia

La eficiencia del servicio estará determinada en su mayoría por la velocidad de transmisión de datos de la red donde se desplegará la aplicación.

Hardware

Para la instalación de la aplicación se debe disponer de una computadora de 512 MB de RAM o superior, 80 GB de disco duro o superior.

Software

La máquina que actuará como servidor de aplicaciones requiere la instalación del Apache v2 o posterior. La máquina que actuará como servidor de Base de Datos requiere la instalación de PostgreSQLv9 o superior.

Las máquinas clientes requieren la instalación de un navegador web como Mozilla Firefox v3.0 o superior o Internet Explorer 7 o superior. Como sistema operativo puede utilizarse Windows Xp, Windows Server, Windows 7, Windows 8 o Linux.

Interfaz de usuario

La aplicación propuesta poseerá una interfaz sencilla dirigida a las personas que se relacionen con el sistema.

Usuarios del sistema

Se define como persona relacionada con el sistema aquella que administra todos los servicios que brinda la aplicación (**Revisor Líder**), y como (**Revisor Técnico**) las personas encargadas de realizar las RTF a un proyecto.

2.4 Exploración

El ciclo de vida ideal de un proyecto realizado con XP se inicia con la fase de Exploración, donde el equipo de desarrollo recopila la información que necesitará para producir una primera entrega del sistema; esta información se traduce en Historias de Usuario. Al mismo tiempo, el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo y se prueba la tecnología. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología [43][44].

2.4.1 Historias de usuario

Las HU son la técnica utilizada para especificar los requisitos del *software* que se desea desarrollar; son tarjetas de papel, que no necesitan un formato fijo ni específico, donde el cliente describe brevemente y con su lenguaje las características que debe poseer el sistema, ya sean requisitos o características del sistema. Una de las principales característica de estas HU es su flexibilidad y claridad en el lenguaje en el que han sido escritas, permitiendo una mejor y rápida comprensión por parte de los programadores a la hora de implementarlas [44].

Como se explicó anteriormente no existe definido un formato o una plantilla específica para la elaboración de las HU, Beck en su libro [44] presenta un ejemplo de ficha (*customer story and task card*[26]) en la cual pueden reconocerse los siguientes contenidos: fecha, tipo de actividad (nueva, corrección, mejora), prueba funcional, número de historia, prioridad técnica y del cliente, referencia a otra historia previa, riesgo, estimación técnica, descripción, notas y una lista de seguimiento con la fecha, estado, tareas por terminar y comentarios.

Clasificación de las historias de usuario

La prioridad en el negocio:

Alta: se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema.

[26] Traducido al español: Historia del cliente y tarjeta de tarea

Media: se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.

Baja: se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo.

El riesgo en su desarrollo:

Alta: cuando en la implementación de las HU se consideran la posible existencia de errores que lleven a la inoperatividad del código.

Media: cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.

Baja: cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.

Teniendo en cuenta los datos necesarios para la planificación y estimación de las Historias de Usuario y las características propias del sistema a desarrollar, así como las opiniones del equipo de desarrollo, se propone entonces la utilización de la siguiente tabla:

Historia de Usuario	
Número: Número sucesivo a partir de 1	**Usuario**: el usuario del sistema que utiliza o protagoniza la historia
Nombre historia: identifica la historia de usuario	
Prioridad en negocio: define la relevancia e impacto de la historia de usuario para el negocio de acuerdo a las necesidades del usuario	**Riesgo en desarrollo:** define la dificultad técnica que supone desarrollar la historia de usuario desde el punto de vista del programador.
Puntos estimados: permiten estimar duración de	**Iteración asignada:** precisa la iteración a la

implementación.	que pertenece la historia de usuario
Descripción: explica en qué consiste la historia de usuario, teniendo en cuenta las acciones realizadas por el usuario y la respuesta brindada por el sistema.	
Observaciones: información extra que se estime agregar para hacer más comprensible la historia de usuario. Por ejemplo: conceptos, precondiciones, etc.	

Tabla 2. Ejemplo de una Historia de Usuario (44)

A continuación se muestran algunos ejemplos de HU definidas para la implementación de la herramienta en cuestión, las demás se pueden ver en los anexos:

Historia de Usuario	
Número: 4	**Usuario:** Revisor líder
Nombre historia: Gestionar lista de chequeo.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Descripción: permite realizar varias operaciones sobre la Lista de chequeo (Incluir, Modificar, Ver y Eliminar). En caso de acceder a Incluir una Lista de chequeo se debe permitir introducir los datos de la lista de chequeo, el sistema valida los mismos y si son correctos el sistema crea la nueva lista de chequeo. En caso de acceder a Modificar la Lista de chequeo, el sistema muestra todos los datos de la lista de chequeo y permite modificarlos, actualizando el registro del sistema. En caso de seleccionar la opción Ver Lista de chequeo, el sistema muestra todos los datos de la lista de chequeo. En caso de acceder a Eliminar Lista de chequeo, el sistema brinda la posibilidad de eliminar la lista de chequeo.	
Observaciones: Datos de la lista de chequeo: • Características. • Descripción.	

• Indicadores.
• Método de evaluación.
• Evaluación.
  ○ Cumplido.
  ○ Parcialmente cumplido.
  ○ No cumplido.
  ○ No aplica.
Ver Anexo 20: IU Crear Lista de chequeo

Tabla 3. HU 4: Gestionar lista de chequeo (Elaboración propia)

Historia de Usuario	
Número: 5	**Usuario:** Revisor líder, Revisor técnico
Nombre historia: Gestionar revisiones técnicas formales.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Descripción: permite realizar varias operaciones sobre las revisiones técnicas formales (Incluir, Modificar, Ver y Eliminar). En caso de acceder a Incluir una revisiones técnicas formales se debe permitir introducir los datos de las revisiones técnicas formales, el sistema valida los mismos y si son correctos el sistema crea una nueva revision técnica formal. En caso de acceder a Modificar las revisiones técnicas formales, el sistema muestra todos los datos de las revisiones técnicas formales y permite modificarlos, actualizando el registro del sistema. En caso de seleccionar la opción Ver revisiones técnicas formales, el sistema muestra todos los datos de las revisiones técnicas formales. En caso de acceder a Eliminar revisiones técnicas formales, el sistema brinda la posibilidad de eliminar las revisiones técnicas formales.	
Observaciones: Datos de las revisiones técnicas formales: • Proyecto a revisar	

• Fecha • Objetivo de la revisiones técnicas formales • Alcance de la revisiones técnicas formales • Lista de chequeo • Resultados • Hallazgos Ver Anexo 19: IU Crear revisiones técnicas formales.

Tabla 4. HU 5: Gestionar revisiones técnicas formales (Elaboración propia)

Historia de Usuario	
Número: 12	**Usuario:** Revisor técnico
Nombre historia: Generar dictamen técnico (DT).	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 3
Descripción: brinda la opción de generar automáticamente un informe con todos los datos asociados a las RTF.	
Observaciones: Informar el resultado de la RTF a la Especificación de Requisitos de *Software*.	

Tabla 5. HU 12: Generar dictamen técnico. (Elaboración propia)

2.5 Planificación

En esta fase el cliente establece la prioridad de cada HU y los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente.

Para realizar esta estimación XP propone métodos como el Planning Poker [44], el cual permite incrementar la comunicación entre los desarrolladores y con el cliente, además una de sus principales características es darle valor a la opinión individual de cada miembro del equipo de desarrollo. Gracias a la flexibilidad de XP se pueden usar otros métodos para la estimación de las Historias de Usuario, en este caso el equipo decidió usar como medida para el esfuerzo asociado a la implementación de las HU, el punto. Un punto, equivale a una semana ideal de programación, sin considerar posibles incidentes que afecten el trabajo. En la estimación debe incluirse todo el esfuerzo asociado, por ejemplo: probar unitariamente el código, integrar con el resto del código, preparar y aplicar las pruebas de aceptación, refactorización del código y de la base de datos cuando sea necesario.

No obstante, para no dejar gran margen a errores, el equipo fue consciente de que aún con la debida planificación y teniendo en cuenta que cada punto es una semana ideal de programación los riesgos son altos debido al corto tiempo, la complejidad de las HU y las posibles modificaciones y afectaciones que pueden surgir en el proceso de desarrollo. Debido a estas circunstancias, se tomaron medidas que ayudan al desarrollo de la solución de forma segura y eficiente.

Estas medidas son:

Mantener control de versiones del desarrollo: para poder regresar a una versión anterior y funcional si ocurriese algún problema.

Uso frecuente de la aplicación: para comprobar el funcionamiento de los componentes y evitar que algunos dejen de funcionar debido a la incorporación de las nuevas funcionalidades de la aplicación o a la constante refactorización del código.

2.5.1 Estimación de esfuerzo por historia de usuario

Teniendo en cuenta lo antes expuesto, la estimación de esfuerzos por HU quedó de la siguiente manera:

No	Historias de Usuario	Puntos de Estimación
1	Gestionar usuario.	2
2	Gestionar permisos	2
3	Gestionar proyecto.	1

4	Gestionar lista de chequeo.	2
5	Gestionar RTF.	2
6	Gestionar hallazgos.	2
7	Gestionar nomenclador_importancia.	1
8	Gestionar nomenclador_tipohallazgo.	1
9	Gestionar nomenclador_estadohallazgo.	1
10	Gestionar nomenclador_evaluación.	1
11	Gestionar documentos	1
12	Generar dictamen técnico.	2
13	Generar resumen de hallazgos	2
14	Mostrar comentarios mas usados en una RTF	2
15	Exportar reporte a formato Word.	1

Tabla 6. Puntos de Estimación por Historias de Usuario

2.5.2 Plan de iteraciones

Después de ser identificadas y descritas las HU y estimar el esfuerzo dedicado a la realización de cada una de ellas se procede a la planificación de la fase de implementación estableciendo una división de tres iteraciones.

A continuación se muestra el plan de iteraciones con el objetivo seguido en cada una de ellas y su contenido de trabajo.

Iteración primera: tiene como finalidad administrar los valores iniciales e imprescindibles para crear un sistema.

Iteración segunda: el propósito es terminar las funcionalidades deseadas por el cliente, obteniendo una versión completa del sistema.

Iteración tercera: se pretende entregar un sistema funcional que permita al cliente realizar operaciones sobre las RTF en el área de Requisitos del desarrollo de *software*. En esta fase también se pretende comenzar con las pruebas finales de aceptación del sistema.

2.5.3 Plan de liberaciones

El plan de liberaciones (entregas) tiene como objetivo definir el número de liberaciones que se realizarán en el transcurso del proyecto y las iteraciones que se requieren para desarrollar cada una. El cliente se encarga de decidir cuáles HU comprende la primera entrega según sus prioridades para darle valor a su negocio y que por tanto justifique su ejecución y así sucesivamente para las demás.

Para realizar el plan de entregas, el sistema propuesto se dividió en módulos que contienen las HU relacionadas lógicamente de acuerdo a su propósito. Cada una de las entregas se correspondió con una iteración de desarrollo.

El plan de liberaciones previsto finalmente quedó estructurado en 3 iteraciones a lo largo de las cuales se trabajará en los diferentes módulos definidos, expuestos en la siguiente tabla:

Módulos	Historias de Usuarios
RTF	Gestionar proyecto.
	Gestionar lista de chequeo.
	Gestionar revisiones técnicas formales.
	Gestionar hallazgos.
	Gestionar nomenclador_tipohallazgo.
	Gestionar nomenclador_importancia.
	Gestionar nomenclador_estadoohallazgo.
	Gestionar nomenclador_evaluación.
Plantilla	Gestionar documentos
	Generar dictamen técnico.
	Generar resumen de hallazgos.
	Mostrar comentarios mas usados en una revisión técnica formal.
	Exportar reporte a formato Word.
Seguridad	Consultar Usuario

	Gestionar permisos.

Tabla 7. Historias de Usuarios por módulos (Elaboración propia)

A modo de resumen se presenta la siguiente tabla que muestra las tres iteraciones de implementación del sistema, además se muestran las HU que se implementarán en cada iteración.

Iteración	Historias de Usuarios
Iteración 1	Gestionar usuario.
	Gestionar permisos.
	Gestionar proyecto.
	Gestionar lista de chequeo.
	Gestionar revisiones técnicas formales.
	Gestionar hallazgos.
Iteración 2	Gestionar nomenclador_importancia.
	Gestionar nomenclador_tipohallazgo.
	Gestionar nomenclador_estadohallazgo.
	Gestionar nomenclador_evaluación.
Iteración 3	Gestionar documentos
	Generar dictamen técnico.
	Generar resumen de hallazgo.
	Mostrar comentarios mas usados en una revisión técnica formal.
	Exportar reporte a formato Word.

Tabla 8. Historias de Usuarios por iteraciones (Elaboración propia)

Para hacer esta estimación se tuvieron en cuenta la complejidad de las HU y su prioridad en el negocio. Es por esto que los módulos no se implementan completos en una sola iteración, dejando para la próxima iteración aquellas HU que son de menos prioridad para el funcionamiento de la aplicación.

Módulo	Iteración 1	Iteración 2	Iteración 3
RTF	50%	20%	100%
Plantilla	40%	40%	100%
Seguridad	50%	30%	100%

Tabla 9. Iteraciones por módulos (Elaboración propia)

2.6 Diseño

Durante la fase de diseño se confeccionan las tarjetas clase-responsabilidad-colaborador para la descripción de cada una de las entidades y se realiza el diagrama de clases para tener mejor visión de la estructura del sistema.

2.6.1 Tarjetas clase-responsabilidad-colaborador (CRC)

El uso de este tipo de tarjetas es una técnica de modelado que permite identificar las clases, sus atributos responsabilidades. El objetivo es obtener un diseño simple, elegante y fácil de comprender por parte de los programadores. A continuación se muestran algunas de las tarjetas del sistema a desarrollar, el resto se encuentra en los anexos del trabajo.

Tarjeta CRC	
Datos de la clase	
Nombre de la clase: Hallazgo	
Descripción: Gestiona la información de los hallazgos	
Responsabilidades	**Colaboradores**
setDescripción()	Rtf, Documento
getDescripción()	Rtf, Documento
setRecomendaciones()	Rtf
getRecomendaciones()	Rtf

setUbicación()	Rtf, Documento
getUbicación()	Rtf, Documento
setEstado()	Rtf, EstadoHallazgo
getEstado()	Rtf, EstadoHallazgo
setTipo()	Rtf, TipoHallazgo
getTipo()	Rtf, TipoHallazgo
setImportancia()	Rtf, Importancia
getImportancia()	Rtf, Importancia

Tabla 10. Tarjeta CRC: Hallazgo (Elaboración propia)

Tarjeta CRC	
Datos de la clase	
Nombre de la clase: Rtf	
Descripción: Gestiona la información de las RTF	
Responsabilidades	**Colaboradores**
setObjetivo()	ListaChequeo, Proyecto
getObjetivo()	ListaChequeo, Proyecto
setAlcance()	ListaChequeo, Proyecto
getAlcance()	ListaChequeo, Proyecto

Tabla 11. Tarjeta CRC: Rtf (Elaboración propia)

Tarjeta CRC
Datos de la clase
Nombre de la clase: ListaChequeo

Descripción: Gestiona los datos de la lista de chequeo	
Responsabilidades	**Colaboradores**
setNombre()	Rtf, característica
getNombre()	Rtf, característica
setDescripción()	Rtf, característica
getDescripción()	Rtf, característica

Tabla 12. Tarjeta CRC: ListaChequeo (Elaboración propia)

2.6.4 Modelo de dominio

El modelo de dominio es esencial para un correcto entendimiento de lo que se quiere que haga la aplicación que se implementará, la misma muestra todas las entidades del sistema y las relaciones entre sí. Este modelo ayuda al equipo de desarrollo, ya que muestra un esbozo inicial de todas las entidades con las que se trabajará en el proceso de desarrollo del sistema. A continuación se muestra el modelo de dominio definido para el desarrollo de la HRTFR. Ver Figura 6:

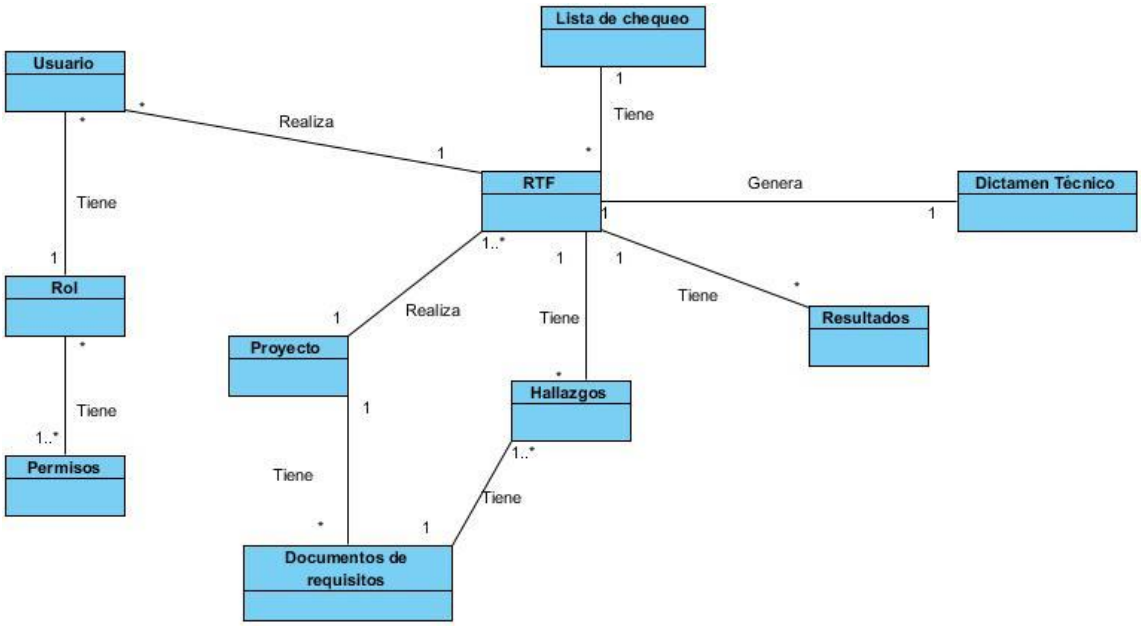


Figura 6. Modelo de dominio

2.6.5 Modelo de datos

El modelo de datos mostrado representa el modelo físico correspondiente a la base de datos de la propuesta de solución. Ver Figura 7:

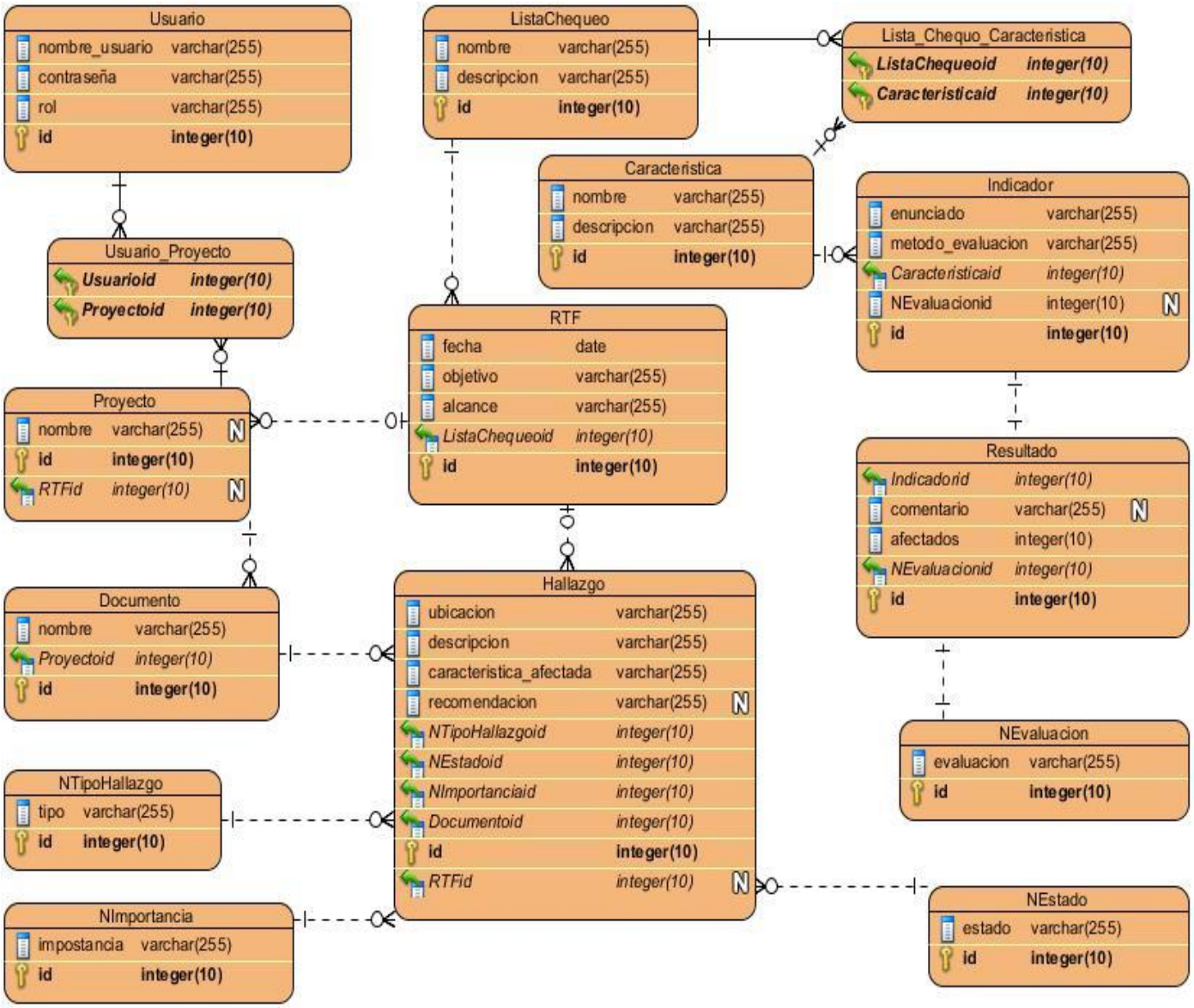


Figura 7. Modelo de datos

2.6.6 Mapa de navegación

El Mapa de Navegación o Diagrama de flujo de las interfaces muestra de manera sencilla tanto para el usuario como para el propio equipo de desarrollo las conexiones existentes entre todas las pantallas que tendrá el sistema, más conocidas como Interfaz de Usuario (IU). En este diagrama se pueden definir concretamente las IU a las que podrá acceder el usuario según sus permisos y rol en el negocio desde que es logueado en el sistema. A continuación se muestran las IU definidas para la HRTFR con sus respectivas conexiones. Ver Figura 8:

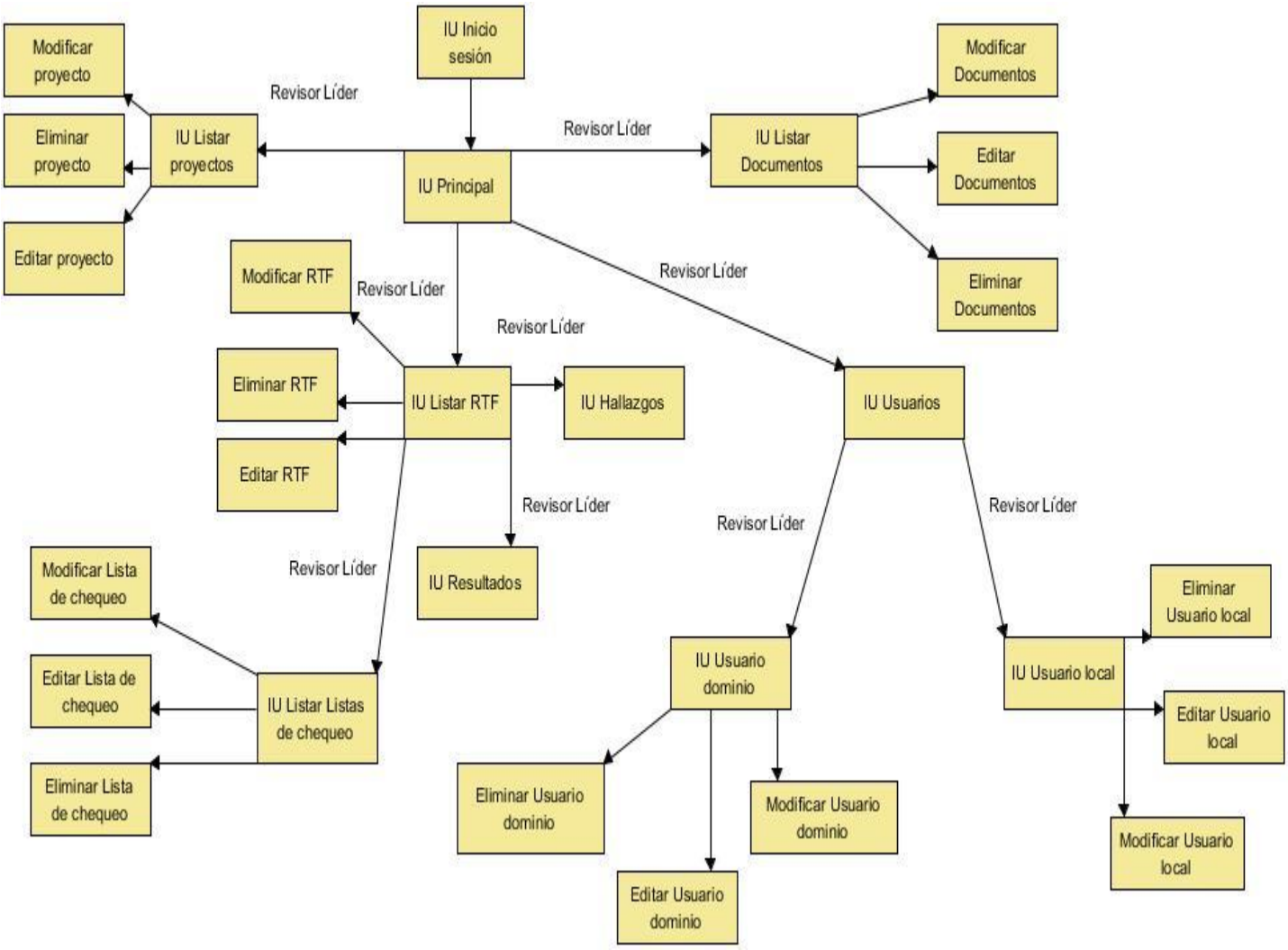


Figura 8. Mapa de navegación

2.6.7 Modelo de proceso

El Modelo de proceso o diagrama de flujo de procesos es una forma gráfica de presentar las actividades involucradas en el procedimiento del desarrollo de las RTF, en el área de Requisitos. Ofrece una descripción visual de las actividades implicadas, mostrando la relación secuencial entre ellas, facilitando la rápida comprensión de cada actividad y su relación con las demás. Ver figura 9:

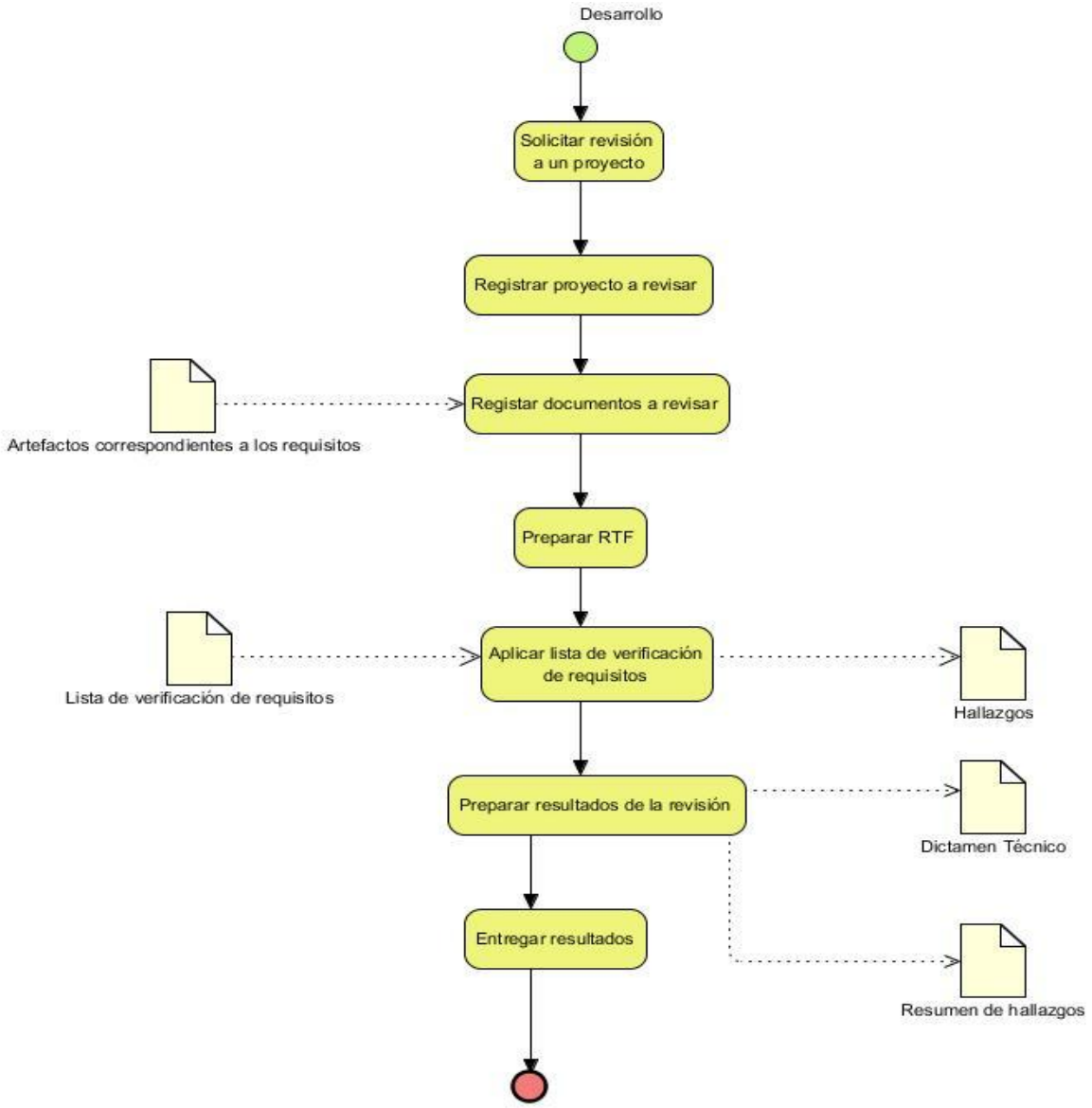


Figura 9. Flujo del desarrollo de las RTF en el área de Requisitos

Conclusiones del capítulo

En este capítulo se arribaron a las siguientes conclusiones:

- ✓ El uso de la metodología XP posibilitó organizar por fases de desarrollo la solución propuesta, describiendo las historias de usuario, en las cuales se especificó el orden de prioridad definido por el cliente para la implementación de la herramienta y se elaboró el plan de iteraciones para la definición del momento y tiempo necesario.
- ✓ Con el análisis del diseño se posibilitó la elaboración de los artefactos como las tarjetas CRC; se analizaron los patrones de diseño y el patrón de arquitectura, con lo que se facilitó el desarrollo de la herramienta. Al modelar los diagramas de dominio, de datos y de proceso se alcanzó un mejor entendimiento del negocio al aportar una representación, lo cual facilita su comprensión.

Capítulo3. Implementación y pruebas

Introducción

En este capítulo se presenta la face de implementación y los tipos de pruebas de software como las unitarias y funcionales, esta última se llevó a cabo mediante los Casos de Prueba de Aceptación, exponiendo los resultados y las funcionalidades alcanzadas, durante el período de la informatización del procedimiento del desarrollo de las Revisiones Técnicas Formales, en el área de Requisitos del Centro FORTES.

3.1 Implementación

La metodología XP plantea que la implementación de un producto debe realizarse de forma iterativa, esta característica trae consigo que después del desarrollo de cada iteración se obtenga un producto funcional que debe ser mostrado al cliente y previamente probado para incrementar la visión de los desarrolladores y clientes de posibles cambios.

3.1.1 Patrón de arquitectura

Para el desarrollo de la HRTFR se utilizó el patrón de arquitectura Modelo Vista Controlador (MVC) [45], que ofrece como ventajas: soporte de vistas múltiples dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación web pueden utilizar el mismo modelo de objetos, mostrado de maneras diferentes, adaptación al cambio, los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos como teléfonos celulares. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo. Este patrón separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes [45].

Cuando un usuario solicita a cierta aplicación web ver el sitio, internamente sucede lo siguiente:

1. El sistema de enrutamiento determina qué controlador está asociado con la página inicial del sitio.

2. Symfony2 ejecuta el controlador asociado a la portada. Un controlador no es más que una clase PHP en la que puedes ejecutar cualquier código que quieras.
3. El controlador solicita al Modelo los datos. El modelo no es más que una clase PHP especializada en obtener información, normalmente de una base de datos.
4. Con los datos devueltos por el Modelo, el controlador solicita a la Vista que cree una página mediante una plantilla y que inserte los datos del Modelo.
5. El controlador entrega al servidor la página creada por la Vista.

A pesar de que puedes llegar a hacer cosas muy complejas con Symfony2, el funcionamiento interno siempre es el mismo:

1. El Controlador manda y ordena
2. El Modelo busca la información que se le pide
3. La Vista crea páginas con plantillas y datos.

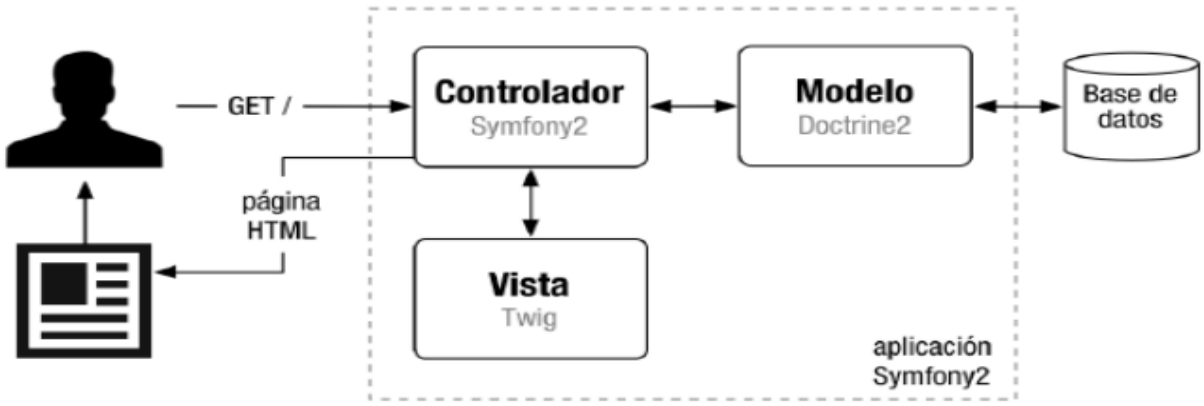


Figura 10. Patron MVC en Symfony 2 (45)

2.6.3 Patrones de diseño

Craig Larman define en la segunda edición de su libro “UML y Patrones” a un patrón como *“… un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos.”*. Los patrones de diseño comunican los estilos y soluciones consideradas como "buenas prácticas", que los expertos en el diseño

orientado a objetos utilizan para la creación de sistemas [46]. El *framework* Symfony utiliza en su implementación un conjunto de patrones de diseño definidos por su equipo de desarrollo para dar solución a problemas específicos del diseño orientado a objetos durante el flujo de ejecución de una petición. Para el desarrollo de la propuesta de solución que se describe en la investigación solo fueron usados los implementados por el *framework*, por tal motivo serán los descritos a continuación.

Patrones GRASP[27]

1. **Experto:** Este patrón resuelve el problema de *"asignar una responsabilidad al experto en información -la clase que tiene la información necesaria para realizar la responsabilidad -."* [46]. Es utilizado en la capa de abstracción del modelo de datos. Symfony genera automáticamente las clases que representan las entidades de nuestro modelo de datos. Asociado a cada una de estas clases son generadas un conjunto de funcionalidades que las relacionan de forma directa con la entidad que representan. Estas clases contienen toda la información necesaria de la tabla que representan en la base de datos [47].
2. **Creador:** Este patrón resuelve el problema de *asignar responsabilidades relacionadas con la creación de objetos* [46]. 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 [47].
3. **Alta cohesión:** Este patrón resuelve el problema de *"asignar una responsabilidad de manera que la cohesión permanezca alta."* La cohesión es una medida de la fuerza con que se relacionan y del grado de focalización de las responsabilidades de un elemento[28]. Un elemento con responsabilidades altamente relacionadas, y que no hace una gran cantidad de trabajo, tiene alta cohesión [46]. La alta cohesión se evidencia en los controladores que poseen un conjunto de funcionalidades, existiendo estrecha relación entre algunas. Ejemplo de ello lo constituyen las acciones *create* y *update* que al crear o actualizar un objeto realizan las validaciones mediante la acción *processForm* [47].

[27] Del inglés General Responsibility Assignment Software Patterns, en español Patrones Generales de Software para Asignar Responsabilidades.

[28] Estos elementos pueden ser clases, subsistemas, etc.

4. **Bajo acoplamiento:** Este patrón resuelve el problema de *"asignar una responsabilidad de manera que el acoplamiento permanezca bajo."* El acoplamiento es una medida de la fuerza con que un elemento está conectado a, tiene conocimiento de, confía en, otros elementos. Un elemento con bajo (o débil) acoplamiento no depende de demasiados elementos [46]. El bajo acoplamiento se evidencia en el hecho de que los controladores heredan únicamente de la clase sfActions. 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 [47].
5. **Controlador:** Este patrón resuelve el problema de *asignar la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clas*e. Un controlador sirve como intermediario entre una interfaz y la acción que se desee ejecutar [46]. Dentro del *framework* el patrón se evidencia en las clases que forman la capa Controlador del patrón arquitectónico MVC. En el mismo todas las peticiones son procesadas por un solo controlador frontal, este es el único punto de entrada de una aplicación en un entorno determinado [47].

Patrones GoF[29]

1. ***Singleton***[30]: Es un patrón de tipo creación, ya que abstrae el proceso de creación de instancias. Resuelve el problema de que exista una instancia única de una clase, proporcionando un punto de acceso global a la misma [46]. En Symfony se hace visible en clases como sfContext la cual permite interactuar con los objetos que son únicos en el núcleo del *framework,* ya que posee una referencia a cada uno de ellos; a través del método getInstance que posee esta clase es posible obtener una instancia de la misma. Otro ejemplo lo constituye la clase Routing la cual se encarga de enrutar todas las peticiones que se hagan a la aplicación, la misma permite a través del método getInstance que se obtenga la instancia única de ella existente en el núcleo del *framework* [47].
2. ***Decorator***: Es un patrón de tipo estructura, ya que permite que clases y objetos sean utilizados para componer estructuras de mayor tamaño. Resuelve el problema de añadir responsabilidades adicionales a un objeto dinámicamente. En Symfony este patrón es utilizado en la capa Vista del

[29] Del inglés Gang of Four, en español Banda de los Cuatro. Es el nombre con el que se conoce comúnmente a los autores del libro Design Patterns (en español Patrones de Diseño) (ISBN 0-201-63361-2), los mencionados autores son: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.
[30] Conocido en español como "Instancia Única".

patrón arquitectónico MVC, ejemplo de ello lo constituye la clase View que es padre de todas las vistas de la aplicación. El fichero layout.php[31] es la plantilla global que utiliza el *framework* para decorar las plantillas asociadas a cada una de las acciones de la aplicación [47].

3. ***Command*:** Es un patrón de tipo comportamiento, ya que ayuda a definir la comunicación e interacción entre los objetos del sistema, reduciendo el acoplamiento entre los mismos. Resuelve el problema de *"¿Cómo gestionar las solicitudes o tareas que necesitan funciones como: ordenar (estableciendo prioridades), poner en cola, retrasar, anotar en registro o deshacer?"* [46].

3.1 Pruebas

Tienen como objetivo determinar si los requisitos y el sistema final cumplen los objetivos para los que se construyó el producto. En la metodología XP las pruebas son un elemento importante a lo largo de todo el desarrollo del *software*, estimulando a los desarrolladores a probar constantemente, permitiendo aumentar la calidad de los sistemas, reduciendo el número de errores no detectados y a su vez disminuyendo el tiempo transcurrido entre la aparición de un error y su corrección. En la metodología de desarrollo XP, las pruebas constituyen una parte fundamental que no se deben obviar bajo ningún concepto y que deben aplicarse de manera frecuente y temprana.

La actividad de probar el sistema y verificar que se encuentra libre de defectos tiene muchos beneficios como: garantiza la calidad, que es una variable muy importante para todo producto, la confianza en el equipo y por último, proporciona una medida del progreso del trabajo.

XP divide las pruebas del sistema en tres grupos: **pruebas unitarias**, encargadas de verificar el código y diseñadas por los programadores y **pruebas funcionales** o pruebas de aceptación destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida por el cliente. Tanto el método de caja blanca como de caja negra pueden aplicarse, aunque la combinación de ambos es ideal para validar tanto la interfaz del *software* como su desempeño interno.

3.1.1 Pruebas unitarias

Las pruebas unitarias son un tipo de pruebas de caja blanca que prueba a cada clase en aislamiento como elemento unitario. El objetivo fundamental de estas pruebas es asegurar el correcto funcionamiento de las interfaces y el flujo de datos entre componentes [48].

[31] Por defecto el framework genera el fichero layout.php, aunque pueden ser definidas otras plantillas globales.

Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular. Las pruebas unitarias se encargan de un único caso cada vez, lo que significa que un único método puede necesitar varias pruebas unitarias si su funcionamiento varía en función del contexto. Estas pruebas aisladas proporcionan cinco ventajas básicas, fomentan el cambio, simplifican la integración, documentan el código, separan la interfaz del código y hacen que los errores estén más acotados y sean fáciles de localizar.

XP aconseja que la realización de las pruebas unitarias al sistema sea automatizada y en etapas tempranas del desarrollo permitiendo disminuir la ocurrencia de defectos y aprovechar las ventajas de la retroalimentación que se produce en el proceso. Aunque la tarea de crear casos de prueba manuales, que luego serán ejecutados y analizados sus resultados, puede utilizarse complementariamente a las anteriores. [48].

Para la validación de HRTFR, se realizaron las pruebas unitarias las cuales fueron ejecutadas a través de un comando haciendo uso de la librería PHPUnit, estas pruebas fueron realizadas por el equipo de desarrollo a medida que se fue implementando la solución por lo que no existen resultados registrados ya que las deficiencias detectadas fueron solucionadas durante la ejecución de las mismas.

3.1.2 Pruebas funcionales

Las pruebas funcionales no solo validan la transformación de una entrada en una salida, sino que validan una característica completa. De modo que las pruebas funcionales validan procesos y requieren de un escenario. Por consiguiente, se derivan directamente de las HU. Su número está limitado por la cantidad de escenarios que deban probarse para cada historia con que se cuenta, hasta asegurar su funcionamiento adecuado.

Estas pruebas simulan la navegación del usuario, realizan peticiones y comprueban los elementos de la respuesta, tal y como lo haría manualmente un usuario para validar que una determinada acción hace lo que se supone que tiene que hacer. Debido a la relevancia de las pruebas funcionales se determinó efectuar casos de prueba de aceptación. La plantilla utilizada y los resultados obtenidos se describen a continuación:

Caso de prueba de aceptación	
Historia de usuario: *Número de la historia de usuario relacionada con la prueba que se realiza.*	**Número Caso Prueba:** *Código que identifica la prueba.*
Descripción de la Prueba: *Definición de la prueba que se realiza.*	
Condiciones de Ejecución: *Condiciones que deben cumplirse para que se pueda realizar la Prueba.*	
Entrada \ Pasos de Ejecución: *Se describen los pasos de ejecución de la prueba.*	
Resultados de la Prueba: *Proporciona las expectativas ideales para las cuales fue pensada la prueba.*	
Evaluación de la Prueba: *Calificación que recibe la prueba de acuerdo a los resultados obtenidos*	

Tabla 13. Ejemplo de plantilla. Caso de prueba de aceptación

A continuación se muestran algunos ejemplos de los Caso de pruebas de aceptación realizados a la herramienta en cuestión, las demás se pueden ver en los anexos:

Caso de prueba de aceptación	
Historia de usuario: 1	**Número Caso Prueba:** HU1-CPA1
Descripción de la Prueba: Gestionar Usuario.	
Condiciones de Ejecución: El usuario autenticado debe tener el rol de Revisor líder.	
Entrada \ Pasos de Ejecución: el Usuario identifica el índice “Usuario” en el menú principal y escoge el tipo de usuario (local, dominio) accede a la opción que le permite Incluir Usuario. Si escoge usuario local introduce el nombre de usuario, el nombre completo, el 1er apellido, el 2do apellido, contraseña, selecciona el rol del nuevo usuario en el sistema y el solapin. Accede a la opción de *Incluir* y el sistema muestra el mensaje “*Elemento creado satisfactoriamente*”. Si el usuario desea agregar un usuario de tipo dominio introduce el usuario y el rol. Accede a la opción de *Incluir* y el	

sistema muestra un mensaje "*Elemento creado satisfactoriamente*". El usuario puede acceder a la opción de *Cancelar* en cualquier momento regresando al Escritorio de trabajo. El usuario puede ver, modificar y eliminar un usuario. Ver CPA2: Consultar Usuario.
Resultados de la Prueba: Si la información proporcionada por el Usuario es correcta y el nuevo Usuario no ha sido registrado previamente, entonces es añadido al sistema.
Evaluación de la Prueba: Satisfactoria**.**

Tabla 14. Caso de prueba de aceptación 1

Caso de prueba de aceptación	
Historia de usuario: 1	**Número Caso Prueba:** HU1-CPA2
Descripción de la Prueba: Consultar Usuario.	
Condiciones de Ejecución: El usuario autenticado debe tener el rol de Revisor líder.	
Entrada \ Pasos de Ejecución: el usuario logueado identifica el índice "Usuario" en el menú principal y accede a la opción que le permite Consultar Usuario. Introduce los criterios de búsqueda por los cuales desea consultar, el sistema muestra un listado de coincidencias y permite Ver, Modificar o Eliminar al usuario seleccionado. Accede a la opción de *Ver* y el sistema muestra los datos del usuario seleccionado, si el usuario logueado selecciona la opción de *Modificar* el sistema permite modificar los datos del mismo y si accede a la opción de *Eliminar*, se muestra el mensaje de advertencia ""*¿Está seguro de que quiere borrar el elemento seleccionado?*", si el usuario logueado accede nuevamente a la opción de Eliminar, se elimina el usuario seleccionado y se muestra el mensaje de información *"Elemento eliminado satisfactoriamente"*. El usuario puede acceder a la opción de *Limpiar* y se limpiarán todos los criterios de búsqueda así como a la opción de *Cancelar* en cualquier momento regresando al Escritorio de trabajo.	
Resultados de la Prueba: si la información proporcionada por el usuario logueado es correcta se podrá consultar, modificar, ver y/o eliminar un usuario.	
Evaluación de la Prueba: satisfactoria**.**	

Tabla 15. Caso de prueba de aceptación 2

Caso de prueba de aceptación	
Historia de usuario: 4	**Número Caso Prueba:** HU4-CPA3
Descripción de la Prueba: Gestionar lista de chequeo.	
Condiciones de Ejecución: El Usuario debe haberse autenticado y tener los permisos para gestionar la lista de chequeo.	
Entrada \ Pasos de Ejecución: el Usuario identifica el índice "Lista de chequeo" en el menú principal y accede a la opción que le permite Agregar nuevo. Introduce el nombre de la nueva lista de chqueo que desea crear, selecciona el nombre de la caracteristica que desea usar en dicha lista (esta caracteristica debe existir en el sistema), de no existir en el sistema, el usuario puede seleccionar la opción de introducir una nueva caracteristica al sistema. Accede a la opción de *Crear* y el sistema muestra el mensaje "*Elemento creado satisfactoriamente*". El usuario puede acceder a la opción de *Cancelar* en cualquier momento regresando al Escritorio de trabajo. El usuario puede ver, modificar y eliminar un glosario. Ver CPA4: Consultar lista de chequeo.	
Resultados de la Prueba: si la información proporcionada por el Usuario es correcta y la nueva lista de chequeo no ha sido registrado previamente, entonces es añadido al sistema.	
Evaluación de la Prueba: satisfactoria**.**	

Tabla 16. Caso de prueba de aceptación 3

Caso de prueba de aceptación	
Historia de usuario: 4	**Número Caso Prueba:** HU4-CPA4
Descripción de la Prueba: Consultar lista de chequeo.	
Condiciones de Ejecución: El usuario autenticado debe tener el rol de Administrador	
Entrada \ Pasos de Ejecución: el usuario logueado identifica el índice "lista de chequeo" en el menú principal y accede a la opción lista de chequeo. Muestra un listado con todos las listas de chequeo que existen en la aplicación y permite Ver, Modificar o Eliminar a la lista seleccionada, así como ver las caracteristicas asociada a la misma. Accede a la opción de *Ver* y el sistema muestra los datos de	

la lista seleccionada, si el usuario logueado selecciona la opción de *Modificar* el sistema permite modificar los datos de la misma y si accede a la opción de *Eliminar,* se muestra el mensaje de advertencia "*¿Está seguro de que quiere borrar el elemento seleccionado?*", si el usuario logueado accede nuevamente a la opción de Eliminar se elimina la lista seleccionada y se muestra el mensaje de información *"Elemento eliminado satisfactoriamente"*. El usuario puede acceder a la opción *Cancelar* en cualquier momento regresando al Escritorio de trabajo.
Resultados de la Prueba: si la información proporcionada por el usuario logueado es correcta se podrá consultar, modificar, ver y/o eliminar una lista.
Evaluación de la Prueba: satisfactoria**.**

Tabla 17. Caso de prueba de aceptación 4

3.2 Resultado de las pruebas

A lo largo del desarrollo de todo el sistema se realizaron los casos de prueba, con el objetivo de comprobar la eficiencia del código escrito y el cumplimiento de los requisitos del sistema planteados por el cliente. Por cada flujo principal se realizó un caso de prueba, en algunos casos se necesitó más de un caso de prueba debido a la complejidad del mismo, para un total de 15 HU se realizaron 10 casos de prueba de aceptación, de manera que fueron comprobados todos los posibles escenarios.

Al finalizar con todas las pruebas se detectaron errores funcionales en la aplicación, los mismos fueron corregidos adecuadamente y en el momento en el que se detectaron, logrando con esto que un error no persistiera y afectara de alguna manera al funcionamiento general del *software*.

Los casos de prueba de aceptación fueron diseñados por cada HU, traduciendo las mismas desde la perspectiva del cliente para así abarcar todos los escenarios posibles y verificar que se cumpliera con todos los requisitos funcionales y no funcionales presentados por el cliente y poder de esta forma liberar la historia de usuario correspondiente.

A continuación, se muestra un gráfico con las no conformidades detectadas y resueltas en cada iteración del proceso de desarrollo del *software*, recordar que la metodología XP propone realizar pruebas en cada iteración:

Figura 11. Incidencias detectadas por Iteración

Las principales incidencias detectadas en la Iteración uno fueron debidas a la gestión de la lista de chequeo y la gestión de los hallazgos, detectando un total de 14 no conformidades de ellas siete de ortografía y siete de validaciones de las cuales fueron resueltas 11, quedando cinco no conformidades por resolver. En la Iteración dos, los principales problemas fueron detectados a la hora de generar los nomencladores detectando nueve no conformidades, tres de ortografía y cuatro de validaciones de las cuales se resolvieron seis quedando tres no conformidades sin resolver y en la Iteración tres y final las incidencias radicaron en la funcionabilidad de generar los informes del dictamen técnico y el resumen de los hallazgos, detectando cuatro no conformidades de validaciones, las cuales fueron resultas.

Conclusiones del Capítulo

Al finalizar este capítulo se arribaron a las siguientes conclusiones:

- ✓ Se ejecutaron las pruebas de la metodología XP, para comprobar que en cada una de las iteraciones del desarrollo se han alcanzado los objetivos propuestos al inicio de estas.

- ✓ Finalmente se realizaron pruebas de aceptación a la herramienta, obteniendo como resultado en la última iteración una correspondencia entre las funcionalidades desarrolladas y los requisitos funcionales antes descritos.

Conclusiones generales

Después de desarrollar el presente trabajo y analizar los resultados obtenidos, se arriba a las siguientes conclusiones:

- ✓ La utilización de los métodos científicos y las técnicas de recopilación de datos empleadas, permitieron desarrollar la teoría que sustenta la investigación.
- ✓ La selección de la metodología de desarrollo, las herramientas, lenguajes de programación y tecnologías más apropiadas para dar cumplimiento al objetivo general, es un resultado del análisis del estado del arte realizado.
- ✓ La metodología seleccionada, permitió satisfacer las necesidades del cliente mediante la definición de las HU y la construcción de las Tarjetas CRC, que dieron paso a la fase de codificación.
- ✓ El diseño desarrollado para la aplicación, permitió la implementación de funcionalidades que dieron solución a la problemática descrita.
- ✓ La validación de la solución, demostró el correcto funcionamiento de la misma y el nivel de satisfacción del cliente.

Recomendaciones

Después de todo el estudio realizado para la construcción de la aplicación propuesta y en vista de seguir perfeccionando la herramienta en el futuro, para lograr una mayor eficiencia en el trabajo que se realiza con ella, sería recomendable tener en cuenta los siguientes aspectos.

- ✓ Ampliar la aplicación añadiéndole una nueva funcionalidad, la opción de corregir la ortografía automaticamente de los documentos que se revisan.
- ✓ Proponer la utilización de la herramienta en otros proyectos productivos de la Universidad de las Ciencias Informáticas y la integración con la herramienta Gespro para en conjunto realizar la planificación de las revisiones.

Referencias bibliográficas

1. **Editorial Ciencias Médicas,** [En línea], ACIMED, 2015.
2. **Pressman, Roger.** *Reflexión de Calidad del Software de Roger Pressman.* 2010. Quinta edición.
3. **8402-1994, ISO.** *Calidad de Software.* 1994. 8402.
4. **610-1900, IEEE estándar.** *Calidad de Software.* 1900. 610.
5. **Díaz, Indira Pérez.** *Procedimiento de Revisiones de Software.* La Habana, Cuba. : s.n., 2012. Universidad de las Ciencias Informáticas.
6. **9001:2008, ISO.** *Revisión Técnica Formal.* 2008. 9001.
7. **García, Yinimary Ortega Montoya y Zayli Noda.** *Estrategia de Control de la Calidad mediante Revisiones y Auditor as.* La Habana, Cuba. : s.n., 2007. Universidad de las Ciencias Informáticas.
8. **Pressman, Roger.** *Modelado del análisis, Parte, capítulo 8.* 2010. Sexta edición.
9. **Según el estándar IEEE 830** (Piattini, 1996), http://www.academia.edu/6647065/Especificacion_de_Requisitos_Software_segun_el_estándar_de_IEEE_830.
10. **Ayala, Alejandro Peña.** *Ingeniería de Software. Una Guía para Crear Sistemas de Información*. 2006. Primera Edición. http://www.wolnm.org/apa/articulos/ingenieria_software.pdf?target=.
11. **Sommerville, Ian.** *Requerimientos Parte II*. 2005. Séptima Edición
12. **Según el estándar IEEE 830** (Piattini, 1996), http://es.slideshare.net/AldoMamani/norma-830.
13. **La enciclopedia libre. Wikipedia**. *Especificación de requisitos del software,* http://es.wikipedia.org/wiki/Especificaci%C3%B3n_de_requisitos_de_software
14. **PdcaHome,** [En línea], http://www.pdcahome.com/check-list/

15. **Philip Johnson.** *Tools for Formal Technical Review*, Department of Information and Computer Sciencies. Mayo 1998. http://inspection.iese.fhg.de/old/tools.html.

16. **Pragma Consultores,** [En línea], http://www.pragmaconsultores.com/herramientas/Paginas/code_checker.aspx.

17. **Informer Technologies,** [En línea], http://rational-requisitepro.software.informer.com/7.1/.

18. **Lara Rodríguez, José Ernesto y Hernández Bernal, Yurisbel.** *Trabajo de Diploma: Análisis, Diseño e Implementación de IDE Online.* Ciudad de la Habana : s.n., 2010.

19. **Camacho, Lorenzo Hernan,** *Análisis de metodologías ágiles,* 2012, http://lorenzohernancamacho.blogspot.com/2012/05/analisis-de-metodologias-agiles-vs.html.

20. **Grupo ISSI y Universidad Politécnica de Valencia.** *Metodologías Ágiles en el Desarrollo de Software.* Alicante : s.n., 2003.

21. **Fernández Enrich, Margarita.** *Presentación Crystal Methodologies.* Laboratorio de Sistemas de Información Facultad de Informática Universidad Politécnica de Valencia : s.n., 2003.

22. **Rumbaugh, Jacobson y Booch**, 2000, p.3

23. **Alegsa, Leandro**, *¿Qué significa Framework? - Información sobre Framework,* 2015, http://www.alegsa.com.ar/Dic/framework.php#sthash.0FCWtdqf.dpuf.

24. **Eguiluz**, **Javier,** *Desarrollo web agil con symfony2*, 2011, Primera Edición, http://sunshine.prod.uci.cu/gridfs/sunshine/books/desarrollo-agil-symfony2.pdf

25. **Alcalde, Alejandro,** *el baúl del programador,* 2014, http://elbauldelprogramador.com/los-10-mejores-frameworks-gratis-de-aplicaciones-web/.

26. **Plataforma Aprende en línea**, *Definición de lenguaje de programación,* http://aprendeenlinea.udea.edu.co/lms/moodle/mod/glossary/showentry.php?courseid=297&concept=lenguaje+de+programaci%C3%B3n.

27. **Prezi Inc.,** [En línea],

https://prezi.com/dokabhjjebim/copy-of-estructura-de-las-aplicaciones-web/.

28. **Aprende a programar,** [En línea], 2015, http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=777:ique-es-javascript-principales-usos-servidor-y-cliente-html-css-y-programacion-efectos-cu01103e&catid=78:tutorial-basico-programador-web-javascript-desde-&Itemid=206.

29. **Jaffe**, **Jeffrey M**, *Definición de HTML5*, [En línea], http://www.w3c.com/.

30. **Libros web**, [En línea], http://librosweb.es/libro/css/capitulo_1/breve_historia_de_css.html.

31. **Achour**, **Mehdi y otros**, *Manual de PHP*, 2010.

32. **Otero, Abraham.** MySQL vs PostgreSQL. ¿Cuándo emplear cada una de ellas? *JavaHispano.* [En línea] 2007. [Citado el: 18 de Diciembre de 2010.] http://www.javahispano.org/contenidos/es/mysql_vs_postgresql_cuando_emplear_cada_una_de_ellas_11/.

33. **Otero, Abraham.** MySQL vs PostgreSQL. *Guatewireless.* [En línea] 2007. [Citado el: 18 de Diciembre de 2010.] http://www.guatewireless.org/articulos/mysql-vs-postgresql/.

34. **ArPUG.** PgAdmin . *ArPug - PostgreSQL Argentina.* [En línea] 2008. [Citado el: 11 de Febrero de 2011.] http://www.arpug.com.ar/trac/wiki/PgAdmin.

35. **Alegsa**, **Leandro**, *¿Qué significa servidor de aplicaciones?, Información sobre servidores de aplicaciones*, 2015, http://www.alegsa.com.ar/Dic/servidor%20de%20aplicaciones.php.

36. **Alegsa**, **Leandro**, *¿Qué significa IIS?, Información sobre IIS*, 2015, http://www.alegsa.com.ar/Dic/iis.php.

37. **Alegsa**, **Leandro,** *¿Qué significa Apache?, Información sobre apache*, 2015, http://www.alegsa.com.ar/Dic/apache.php.

38. **Mastermagazine,** [En línea], Definición de herramienta. http://www.mastermagazine.info/termino/5234.php.

39. **Jesús M. González Barahona, Joaquín Seoane Pascual, Gregorio Robles,** *Introducción al software libre.* Madrid, España: s.n., 2010.

40. **Genbetadev,** [En línea], Herramienta Eclipse IDE, 2014. http://www.genbetadev.com/herramientas/eclipse-ide.

41. **Oracle Corporation.** NetBeans. *Netbeans.* [En línea] 2012. [Citado el: 14 de 12 de 2014.] http://netbeans.org/features/platform/index.html.
42. **Guía Ubuntu**, *PgAdmin III*, [En línea], http://www.guia-ubuntu.com/index.php?title=PgAdmin_III.
43. **Grupo ISSI y Universidad Politécnica de Valencia.** *Metodologías Ágiles en el Desarrollo de Software.* Alicante : s.n., 2003.
44. **Beck, K.** *Extreme Programming Explained. Embrace Change.* [trad.] Adison Wesley. s.l. : Pearson Education, 1999. Traducido al español como: Una explicación de la programación extrema. Aceptar el cambio.
45. **Reynoso, Carlos y Kiccillof, Nicolás.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Universidad de Buenos Aires : s.n., 2004.
46. **Larman, Craig.** *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2003. Segunda Edición.
47. **Potencier, F y Zaninotto, F.** *Symfony 1.1, la guía definitiva*. [En línea] 2009. librosweb.es. http://www.librosweb.es/symfony_1.1.
48. **Nazcasoft.** Pruebas Unitarias. *CalidadySoftware.* [En línea] [Citado el: 15 de Diciembre de 2010.] http://www.calidadysoftware.com/testing/pruebas_unitarias1.php