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 Modelo de datos del desarrollo de software para el Centro FORTES

**Autores:**

Keimis Figueras Lores

Ernest Alvarez Calderón

**Tutor:**

Ing. Lizardo Ramírez Taboada

Ing. Nellis Margarita Cabrera Mallea

Dr.C. María Caridad Valdés Rodríguez

La Habana, 2015

"Año 57 de la Revolución"

# Declaración de Autoría

**Declaración de Autoría**

Declaramos ser los autores del presente trabajo de diploma y otorgamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____ _____

Ernest Alvarez Calderón Keimis Figueras Lores

_____ _____

Ing. Lizardo Ramírez Taboada Ing. Nellis Margarita Cabrera Mallea

Tutor Tutor

_____

Dr. C. María Caridad Valdés Rodríguez

Tutor

# Dedicatoria

*Keimis*

*El presente trabajo va dedicado a mi mami por el amor, cariño y comprensión que me ha brindado todo este tiempo. Además, a todas las personas que pensaron que no podíamos llegar hasta aquí.*

*Ernest*

*A mi mamá y a mi papá por su inmenso amor, comprensión, apoyo y por confiar en mí; por su sacrificio, entrega y por ser mis guías en este largo camino recorrido.*

# Agradecimientos

*Keimis*

*En primer lugar, le agradezco a Dios, luego a mi mami por estar conmigo siempre en las buenas y las malas. También a todos mis amigos y familiares.*

*Ernest*

*Gracias, en primer lugar, a toda mi familia (...) que siempre han estado presente con todo su apoyo y cariño, en especial a mis padres, mis hermanos (Thalia, Osmany y Yumi) por todo lo que han hecho por mí y por inculcarme que el estudio siempre es de gran importancia en la vida.*

*Agradezco a todos los que han tenido que ver de una forma u otra con la realización de este Trabajo de Diploma; a todos mis amigos, por todos los momentos que compartimos durante los cinco años de la carrera.*

## Resumen:

El Centro de Tecnologías para la Formación brinda servicios para el aseguramiento, verificación y validación de los productos desarrollados. Entre los servicios se encuentran las auditorías, las pruebas de *software* en sus diferentes etapas de desarrollo, la capacitación, orientación y las revisiones técnicas formales; durante la ejecución de estas últimas se presentan insuficiencias, específicamente en el área de Modelo de datos del desarrollo del *software*, al comprobar en su estructura, la organización de los datos de la aplicación y de los requisitos. No existe correspondencia entre los datos y tipos de datos, al verificar el conjunto de operaciones entre los mismos. Como solución a la anterior situación se propone la informatización del procedimiento, el cual agilizará las revisiones en el área, por la importancia y utilidad en el *software*. Su propósito fundamental es minimizar el tiempo de revisión de cada *software* y contribuir a que los mismos se liberen con la calidad requerida. En su primera versión, la herramienta permite, mediante criterios de revisión, crear, eliminar, modificar y generar un Dictamen Técnico. También, contiene la incorporación de criterios elaborados por los especialistas en base de datos del Centro, lo cual permite la comprensión con una mayor claridad de los criterios de revisión, establecidos por la Universidad. Durante la implementación de la aplicación se realizaron pruebas de aceptación y unitarias, con el objetivo de demostrar el correcto desarrollo de la misma.

**Palabras clave:** Dictamen Técnico, Modelo de datos, Revisión Técnica Formal.

# Índice

## Índice de tabla

## Ilustraciones

## Introducción

La calidad de cualquier producto, siempre ha sido uno de los objetivos importantes a tener en cuenta en la evolución de las empresas, a lo largo de los años, para lograr satisfacer las necesidades de los clientes y con ello, ocupar una buena posición en el mercado. A tales efectos se han escrito y desarrollado metodologías y técnicas capaces de ayudar en este proceso.

La Universidad de las Ciencias Informáticas (UCI) desempeña un importante rol como proveedora de *software* a diferentes entidades cubanas y busca situarse al alcance de clientes extranjeros, siendo la producción uno de sus objetivos principales. La Universidad posee diferentes centros de producción, distribuidos por sus facultades. En la Facultad 4 se encuentra el Centro de Tecnologías para la Formación (FORTES), el cual está dedicado a la producción de *software* educativo, a ofrecer servicios y productos para la implementación de soluciones de formación.

El principal objetivo del Centro es ofrecer servicios para el aseguramiento, verificación y validación a los productos desarrollados, a los proyectos productivos del Centro e interesados. Entre estos servicios se encuentran las auditorías, las pruebas de *software* en sus diferentes etapas de desarrollo, la capacitación y orientación. Además, en el Centro se ejecutan las Revisiones Técnicas Formales (RTF), que incluyen actividades como: inspecciones, revisiones cíclicas y evaluaciones técnicas del *software*.

Las RTF son inspecciones que se realizan en todas las áreas del desarrollo de *software*, dígase: Requerimiento, Planificación, Arquitectura y Modelo de datos. Estas actividades tienen como objetivo:

- 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.
- Conseguir un *software* desarrollado uniformemente (1).

El crecimiento de productos informáticos hace preciso prestar especial atención al área Modelo de datos, pues a través de este se describen las entidades y sus relaciones para almacenar y tratar la información. Las buenas prácticas de trabajo con las base de datos requieren efectuar las RTF en esta área, para poder puntualizar su evolución en artefactos que hacen comprender mejor su fin y utilización. Pues de ahí,

la claridad de un modelo de datos, el código final en el gestor de base de datos y la documentación que se genere, tiene que estar almacenado en algún tipo de descripción que ayude en un futuro a realizar nuevos cambios.

Para realizar las RTF en el área Modelo de datos se llevan a cabo revisiones evaluadas mediante listas de chequeo, a través de estas los revisores técnicos o especialistas en el área son los encargados de revisar que la documentación exista, además de dar consistencia de que su ejecución haya sido un éxito. La gran documentación a revisar, la contrapartida que se realiza de solución con documentos recibidos del proyecto, se dificulta para quienes realizan esta tarea, pues se consume más tiempo de lo estimado, lo cual puede influir en la entrega del producto en general. Presentándose como posibles problemas que afectan la calidad de cada producto de *software* en el Modelo de datos los siguientes:

- Existen insuficiencias a la hora de comprobar la estructura del modelo de datos, en cuanto a la organización de los datos de la aplicación y de sus requisitos.
- Existen deficiencias en cuanto a la verificación de la correspondencia del conjunto de operaciones entre los datos y tipo de datos.
- Se hace complejo revisar la estandarización en las bases de datos, lo cual puede influir en la liberación de los productos de *software* con la calidad requerida.

Por lo antes expuesto se propone el siguiente **problema a resolver** ¿Cómo informatizar el procedimiento de Revisiones Técnicas Formales en el área Modelo de datos del desarrollo de *software* para el Centro FORTES?

Se define como **objeto de estudio** el proceso de Revisiones Técnicas Formales en el desarrollo de *software*.

Para dar cumplimiento al problema planteado se establece como **objetivo general** informatizar el procedimiento de Revisiones Técnicas Formales, en el área Modelo de datos del desarrollo de *software* para el Centro FORTES.

A partir del objetivo general se derivan los siguientes **objetivos específicos:**

- Investigar el proceso de Revisiones Técnicas Formales, conceptos fundamentales, artefactos y diversas fuentes que analizan los procedimientos en el Modelo de datos del desarrollo de *software*.

- Diseñar una herramienta para la informatización del procedimiento de las Revisiones Técnicas Formales en el área Modelo de datos del desarrollo de *software*.
- Realizar la implementación y pruebas de la herramienta para la informatización del procedimiento de las Revisiones Técnicas Formales en el área Modelo de datos del desarrollo de *software*.

E**l campo de acción** lo constituye el proceso de informatización del procedimiento de Revisiones Técnicas Formales en el área Modelo de datos del desarrollo de *software* en el Centro FORTES.

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

1. ¿Qué fundamentos teóricos, metodológicos y tecnológicos sustentan las tendencias actuales del estudio de las Revisiones Técnicas Formales en el área Modelo de datos del desarrollo de *software*?

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

3. ¿Qué tecnologías se deben tener presente en el diseño de una herramienta para informatizar el procedimiento de Revisiones Técnicas Formales en el área Modelo de datos del desarrollo de *software* en el Centro FORTES?

4. ¿Qué resultados se alcanzan con la implementación y la aplicación de las pruebas de *software* a la herramienta de informatización del procedimiento de Revisiones Técnicas Formales en el área de modelo de datos en el Centro FORTES?

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

- Investigación del proceso de Revisiones Técnicas Formales, conceptos fundamentales, artefactos y diversas fuentes que analizan los procedimientos en el Modelo de datos del desarrollo de *software*.
- Diseño de una herramienta para la informatización del procedimiento de las Revisiones Técnicas Formales en el área Modelo de datos del desarrollo de *software*.
- Implementación y pruebas de la herramienta para la informatización del procedimiento de las Revisiones Técnicas Formales en el área Modelo de datos del desarrollo de *software*.

Para el desarrollo de la investigación se hizo uso de los siguientes métodos científicos:

**Métodos Teóricos:**

- **Histórico-Lógico:** se utilizó para el análisis de la trayectoria completa de las RTF, las etapas principales de su desenvolvimiento y las conexiones históricas fundamentales.
- **Analítico-Sintético:** se utilizó para analizar toda la documentación existente sobre el tema de investigación y arribar a conclusiones después del procesamiento de la información.

**Métodos Empíricos:**

- **Entrevista**: se aplicó una entrevista formal (planificada) a la especialista general del departamento de calidad UCI, los especialistas en base de datos y los encargados de RTF del Centro, para obtener información del estado actual y el procedimiento que actualmente se ejecuta de las RTF en el área Modelo de datos del desarrollo de *software*, con vistas a determinar las funcionalidades que debe cumplir el sistema informático. Ver **Anexo 1**
- **Observación**: es la apreciación planificada de un objeto, se realiza de forma consciente y orientada al procedimiento del desarrollo de las RTF en el área Modelo de datos del desarrollo de *software*, con el objetivo de identificar las funcionalidades que debe tener la propuesta solución.

La investigación se ha organizado en tres capítulos. Se estructura de la siguiente forma:

**Capítulo 1: Fundamentación teórica**

En este capítulo se presenta el marco teórico de la RTF en el área Modelo de datos del desarrollo de *software* y se definen conceptos que se relacionan con la misma, identificando metodologías y tecnologías a usar.

**Capítulo 2: Propuesta de solución**

En este capítulo se describen las funcionalidades y características de la herramienta, se crean las historias de usuarios, se define el plan de entrega y el de iteraciones, utilizando la metodología seleccionada. Además, se realiza el diseño de la herramienta.

**Capítulo 3: Implementación y Pruebas**

# Introducción

En este capítulo se presentan los Casos de Prueba, a los cuales fue sometida la herramienta y las pruebas utilizadas para comprobar las funcionalidades en cada una de las iteraciones. También se plantean los resultados detectados durante el período de implementación.

# Capítulo 1: Fundamentación Teórica

## 1.1 Introducción

En el capítulo se aborda sobre los principales contenidos relacionados con la investigación. Además, se realiza un estudio sobre las principales características de las metodologías ágiles y robustas para a partir de los resultados arrojados por el mismo, seleccionar una de ellas para regir el proceso de desarrollo del sistema. Se especifican además las características fundamentales de las tecnologías y herramientas a utilizar en la implementación de la aplicación.

**Principales conceptos**

Según la norma ISO-8402, la calidad es “la totalidad de aspectos y características de un producto o servicio, que tienen que ver con su habilidad para satisfacer las necesidades declaradas o implícitas” (2). Es decir, la calidad se adquiere cuando se satisfacen las necesidades de un cliente o usuario.

Se entiende por calidad de *software*, la “concordancia con los requisitos funcionales, desempeño explícitamente establecidos, estándares de desarrollo documentados de forma correcta y características implícitas que se espera de todo *software* desarrollado profesionalmente” (3).

Se puede inferir que si no se ejecutan los requisitos establecidos entonces el *software* no queda con la calidad esperada. En ocasiones el cliente no resalta funcionalidades o características implícitas, que pueden llegar a ser importantes para el desarrollo del *software*. Los estándares especificados que definen los criterios de desarrollo, constituyen la forma en que se aplica la Ingeniería de *Software*.

Las revisiones técnicas, son una forma de verificar el trabajo de los desarrolladores, pues se pueden introducir errores, y ningún producto está exento de ellos. Otra razón por la que son necesarias estas revisiones es porque resulta factible examinar el trabajo desde otra perspectiva que no sea la de los propios desarrolladores, ya que existen algunos errores que suelen ser complejos de detectar por el propio autor (4).

Una Revisión Técnica Formal es una actividad de garantía de calidad del *software* que es llevada a cabo por profesionales de la Ingeniería de *Software*. Es importante definir sus objetivos con antelación, estos expresarán el alcance que tendrá la realización de cada revisión. Su beneficio más obvio es el pronto

descubrimiento de los defectos del *software*, de tal forma, que cada defecto pueda ser corregido antes de llegar al siguiente paso del proceso de ingeniería de *software*, pues reduciría el costo de su corrección (1).

Análisis de datos: es la ciencia que examina datos en bruto con el propósito de sacar conclusiones sobre la información. Se distingue de la extracción de datos por su alcance, su propósito y su enfoque sobre el análisis (5).

Modelo de base de datos: es un conjunto de ideas utilizadas para representar la estructura de datos y las relaciones en la base de datos. Es la representación simple relativa, generalmente gráfica, de estructuras complejas de datos en el mundo real. El modelado de datos es el análisis de los objetos de datos que se utilizan en una empresa u otro contexto y la identificación de las relaciones existentes entre éstos. Este es un primer paso para hacer la programación orientada a objetos (6).

## 1.2 Análisis de situaciones similares

La calidad de todo producto de *software* desarrollado es de importancia en las empresas, pues para poder comercializar estos productos es preciso que pasen por un proceso de revisión. La revisión de éstos se realiza durante las cuatro etapas del desarrollo de *software*. Actualmente no se cuenta con una herramienta informática que apoye la inspección en el área Modelo de datos del desarrollo de *software*, por lo que es necesario realizar un estudio de otros sistemas similares existentes, para así tomar las características que se puedan adaptar a la solución que se desea implementar. Se muestran a continuación ejemplos de éstos.

**Propuesta de la Facultad de Estadística e Informática de la Universidad Veracruzana, México**

La Facultad de Estadística e Informática de la Universidad Veracruzana en México desarrolló una aplicación *desktop* para la informatización de los procesos de RTF que se desarrollan en dicho centro. Con la herramienta se pretende aligerar la carga de los revisores al realizar una evaluación preliminar de aspectos automatizables de diferentes artefactos, de acuerdo con reglas predefinidas. Se espera además que se favorezca la realización de revisiones para mejorar la calidad del *software*, al reducir la difusión de defectos (7).

A continuación se muestran las funcionalidades del sistema; las mismas pueden servir como base para la implementación de las funcionalidades de la propuesta de solución:

- Administración de documentos y archivos.
- Sistema de acceso basado en roles.
- Generación de reportes.

***Rational RequisitePro***

Herramienta centrada en documentos, que almacena los requisitos asociándolos a documentos 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 casos de uso, mejorar la trazabilidad, fortalecer la colaboración, reducir la reanudación del proyecto, y aumentar la calidad. Tiene como principales características (8):

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

Luego del estudio de soluciones similares a nivel internacional es posible concluir lo siguiente:

- Las herramientas no pueden ser utilizadas para dar solución al problema de la investigación, debido a que las mismas se enmarcan en el área de requisitos. Otro factor que dificulta la adopción de estos sistemas en el Centro es que las mismas no cumplen con los parámetros que se evalúan en las listas de chequeo establecidas por el grupo de expertos de la universidad.
- La adaptación de la Propuesta de la Facultad de Estadística e Informática de la Universidad Veracruzana, México, hacia el área Modelo de datos resultaría en gran medida una tarea compleja, fundamentalmente por ser una herramienta *desktop*, lo que traería grandes costos en cuanto a fuerza de trabajo y uso de recursos. Además, no se tiene conocimiento acerca de la factibilidad de su uso, porque no existen resultados registrados de dicha herramienta que evidencien su puesta en práctica.

- Las principales funcionalidades de la herramienta *Rational RequisitePro* están guiadas a la gestión de requerimientos. La misma no realiza una revisión de los requisitos sino que los crea con la calidad demandada.

**Propuesta del procedimiento de la Universidad de la Ciencias Informáticas**

La UCI comenzó el programa de mejora, donde se implantan las áreas de proceso del nivel 2 de CMMI, y específicamente de PPQA (*Process and Product Quality Assurance*[1]). Las Revisiones Técnicas Formales entran en el nivel 3. El área de proceso encargada de verificar la adherencia de los procesos, los productos de trabajo, de garantizar la información y resolución de las no conformidades, ha redefinido el proceso de revisiones que se estaba siguiendo hasta el momento en tres etapas de revisión que a continuación se describen (9):

Primera etapa: Planear las revisiones

Durante la planeación de las revisiones se elabora un plan de revisiones, se seleccionan los procesos y productos a evaluar por los revisores y se envían las notificaciones necesarias. Además se verifica con tiempo suficiente de antelación la viabilidad de la revisión.

Salidas: plan de revisiones y notificaciones que pueden ser de revisión, de revisores y del personal.

Segunda etapa: Desarrollo de la revisión

La etapa de desarrollo de la revisión corresponde a confirmar la disponibilidad de recursos, asignación de las tareas al equipo revisor, seguida de la revisión documental, preparar el informe preliminar con las no conformidades y las acciones correctivas o de mejoras, y para finalizar la reunión de cierre.

Salidas: no conformidades detectadas, informe preliminar, acciones correctivas o de mejoras y reunión de cierre.

Tercera etapa: Fin de las RTF

La etapa de fin de las RTF corresponde a tramitar el informe técnico, preparar expediente de la revisión, almacenar la información, evaluar desempeño de los revisores.

Salidas: informe final de la revisión y evaluaciones de desempeño de los revisores.

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

Se muestra a continuación el procedimiento para realizar las RTF al desarrollo de *software* en la UCI, tomándose las Figuras 1, 2, 3 del documento Procedimiento de Revisiones Técnicas Formales del Departamento de Calidad de *Software* (9).

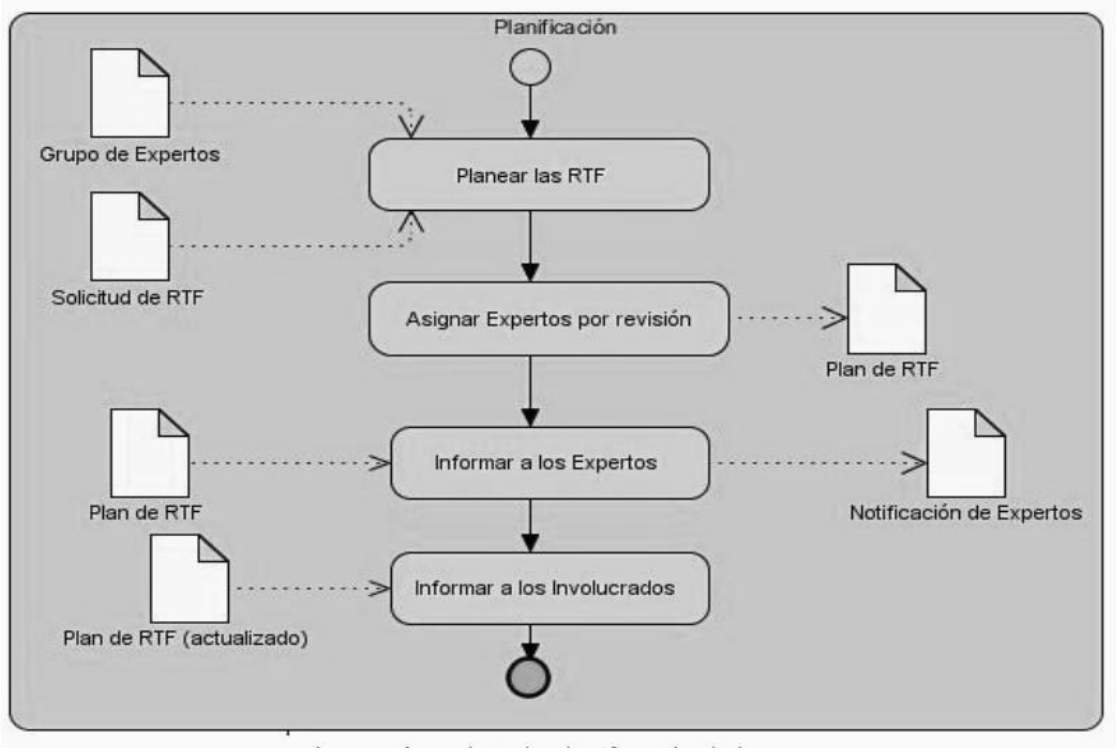


**Figura 1.Flujo de planificación de la RTF**

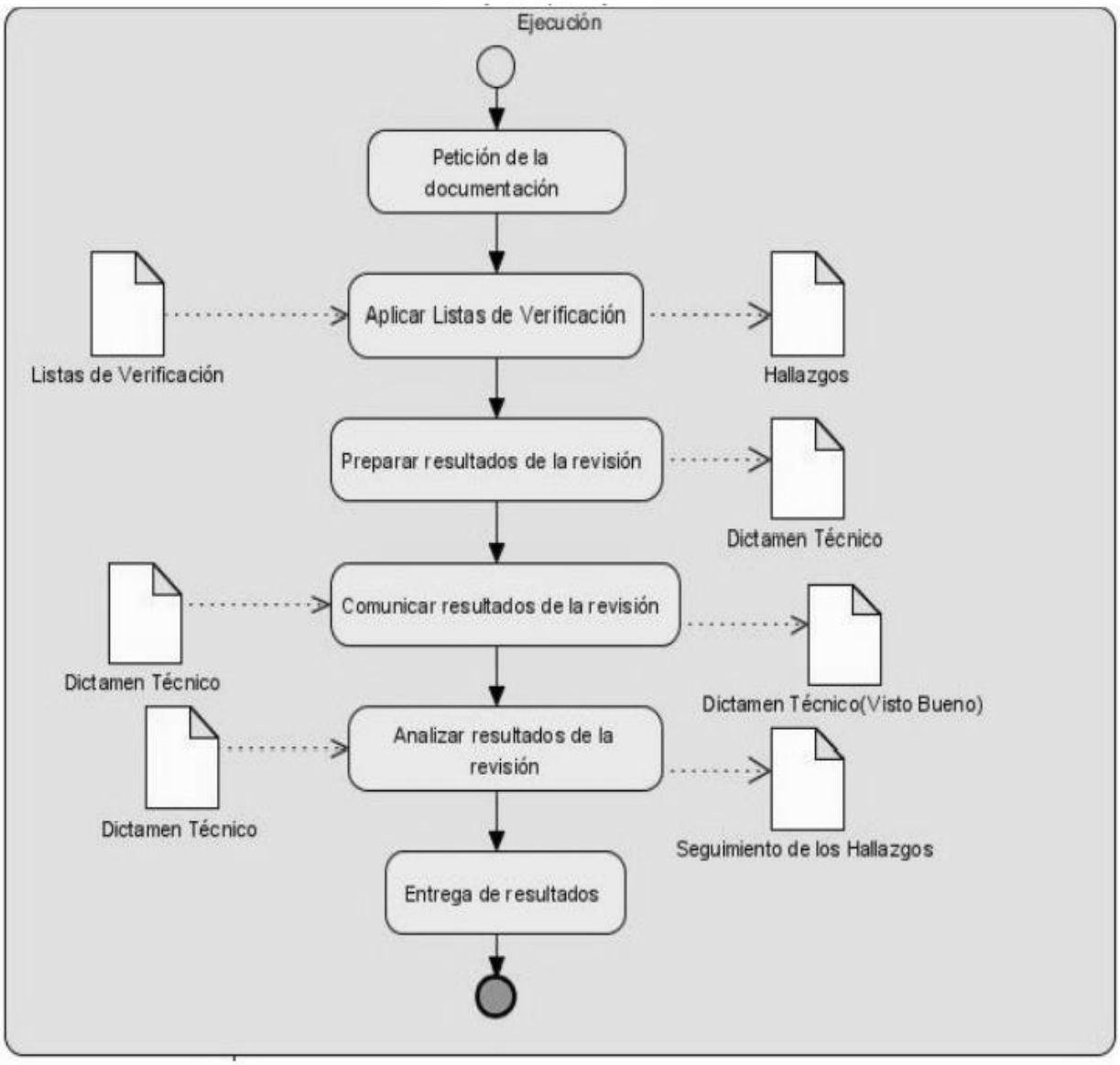


**Figura 2.Flujo de ejecución de la RTF**

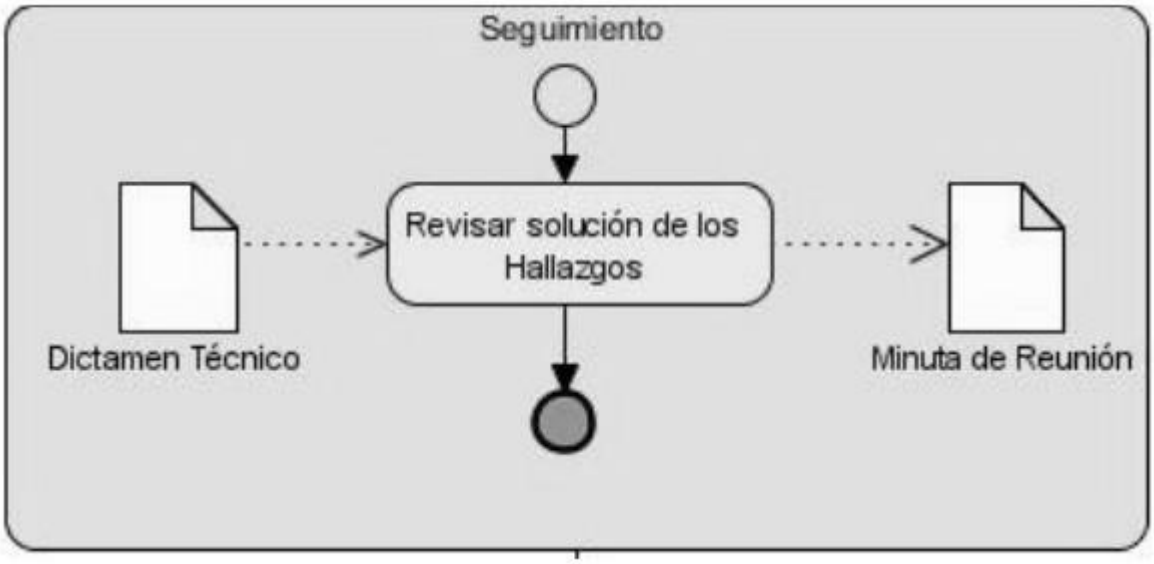


**Figura 3.Flujo de seguimiento de la RTF**

### 1.3 Descripción del proceso de RTF en el Centro FORTES

El procedimiento a tener en cuenta, se lleva a cabo mediante la confección de tres fases: Planificación de la revisión, Ejecución de la revisión y Análisis de los resultados. Estas fases que se describen a continuación son las etapas donde se realizan un sin número de actividades generando artefactos de entrada y salida (10).

Planificación de la revisión: su objetivo principal es darle cumplimiento al cronograma y acuerdos planteados en la reunión de inicio. En esta fase la preparación del equipo revisor constituye una de las tareas más importantes para garantizar una mayor familiarización y un ambiente de trabajo más cómodo al realizar la RTF.

Ejecución de la revisión: es la encargada de realizar las RTF. Ésta se centra específicamente, en la etapa de planificación, de arquitectura, requerimiento y modelo de datos. Su objetivo es detectar las no conformidades que puedan existir en cuanto a: descripción y captura de requisitos, relaciones entre clases, entregables, interfaces y diagramas.

Análisis de los resultados: es la encargada de integrar los resultados de las revisiones por etapas, de elaborar el registro de las no conformidades con los comentarios realizados por los revisores técnicos y de dar a conocer el estado del proyecto revisado.

A continuación se muestra en la Figura 4 el diagrama del proceso de negocio de las RTF en el Centro.

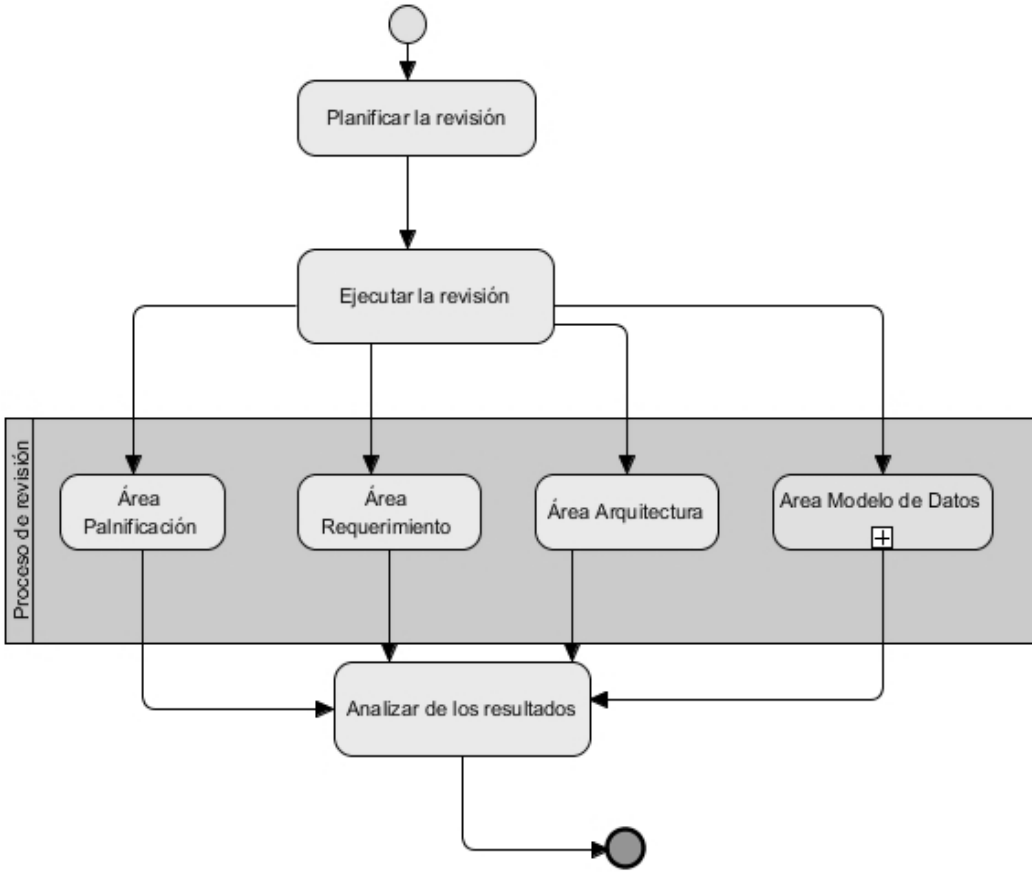


**Figura 4.Diagrama de proceso de negocio de la RTF en el Centro**

Las RTF en el área Modelo de datos se enmarcan en la segunda fase del proceso de RTF en el Centro FORTES. Estas revisiones son realizadas por un conjunto de especialistas en base de datos del Centro a todos los proyectos productivos desarrollados por el mismo. Su ejecución es llevada a cabo mediante una lista de chequeo contenida en un Excel con doce puntos a evaluar.

## 1.4 Descripción del procedimiento de las RTF en el área Modelo de datos del desarrollo del *software*

Los especialistas en el área, al recibir el expediente de proyecto del producto a revisar con la vista de despliegue, vista de datos y de seguridad, proceden a inspeccionar estos documentos teniendo en cuenta cada criterio de la lista. Las salidas de esta revisión son las no conformidades encontradas y las observaciones.

Luego de recoger todas no conformidades encontradas se conforma el informe o dictamen técnico con las observaciones y un conjunto de recomendaciones formuladas a consideración del especialista o revisor técnico. Se realiza el análisis del estado del proyecto en cuanto al área y se archivan los resultados obtenidos en el expediente del proyecto revisado.

**Lista de chequeo**

La dirección de calidad UCI utiliza una lista de chequeo para realizar las revisiones técnicas formales a las distintas áreas del desarrollo de *software*. Ésta es creada para ejecutar actividades de control de calidad, para obtener información de forma ordenada y sistemática. La lista está conformada por un conjunto de criterios que evalúan el estado del proyecto, a continuación se muestran los criterios que componen la lista del Modelo de datos.

- Gestores de base de datos utilizados por la aplicación: este criterio realiza un análisis del gestor de base de datos utilizado, especificando la versión y el sistema operativo sobre el que se encuentra instalado.
- Modelo de seguridad lógica y física: este criterio realiza una descripción de los métodos utilizados para tener un acceso eficiente a los datos, de la estructura e implementación de la base de datos.
- Utilización de recursos de la base de datos: este criterio verifica los procedimientos almacenados y vistas materializadas.
- Control de la concurrencia de usuarios. Herramientas utilizadas para su gestión: este criterio verifica qué mecanismos se utilizaron para garantizar la ejecución de transacciones en paralelo por parte de los usuarios.

- Consideraciones si el sistema implementado es distribuido: este criterio realiza una descripción de la distribución de los datos en diferentes nodos de una red de computadoras, la arquitectura que presenta y los componentes involucrados tanto en *hardware* como en *software*.
- Escenario de réplica de datos: este criterio realiza una caracterización de los mecanismos utilizados para propagar y diseminar los datos.
- Herramientas utilizadas para la realización de la réplica de datos.
- Herramientas para la realización de *Clústers*[2] y lograr la alta disponibilidad.
- Políticas de respaldo y recuperación. Herramientas utilizadas para lograrlo.
- Cálculo de estimación de crecimiento de la base de datos.
- Almacenamiento de BLOBs[3] en la base de datos.

**Necesidad de realizar una aplicación para el Centro**

Este proceso de revisión en el área Modelo de datos resulta una actividad engorrosa para quienes realizan esta labor pues requiere de mucho tiempo, conocimiento y mucha dedicación. El Centro al no contar con una herramienta que agilice este proceso y que asegure de manera eficiente la correcta evaluación de los productos a través de las listas de chequeo propuestas, tiene como resultado que se omitan errores y se arrastren hacia el desarrollo de las siguientes iteraciones del producto. Además, de que se hace complejo dar seguimiento a las estrategias de mitigación de las no conformidades, así como también establecer un estudio del comportamiento de los parámetros a evaluar, tanto a nivel de proyecto, como en el Centro, ya que no se cuenta con un método para almacenar, clasificar y consultar los datos de la revisión.

## 1.5 Análisis de las tendencias y tecnologías actuales

Al no existir un sistema capaz de solucionar la situación problemática existente, es preciso desarrollar una nueva solución y garantizar que el proceso de RTF en el área Modelo de datos sea satisfactorio. Para esto, se procede a efectuar un estudio de las metodologías, herramientas y tecnologías actuales, identificando las más adecuadas para desarrollar la propuesta de solución.

[2] Conjuntos de computadoras unidos entre sí que se comportan como si fuesen una única computadora.
[3] Objetos binarios grandes del inglés (*Binary Large Objects*) son elementos utilizados en las bases de datos para almacenar datos de gran tamaño que cambian de forma dinámica.

### 1.5.1 Metodologías de Desarrollo

Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de *software* en sus esfuerzos por implementar nuevos sistemas de información (11).

Con el estudio realizado de las metodologías (ágiles o robustas), fue posible recoger en la Tabla 1 las principales características de las mismas, propiciando a que se escoja la metodología más adecuada para guiar el desarrollo de la propuesta de solución.

**Tabla 1.Comparación entre Metodologías**

| Metodologías Ágiles | Metodologías Robustas |
|---|---|
| Se basan en heurísticas provenientes de prácticas de producción de código | Se basan en normas provenientes de estándares seguidos por el entorno de desarrollo |
| Preparados para cambios | Cierta resistencia al cambio |
| Impuestas internamente | Impuestas externamente |
| Proceso menos controlado con menos principios | Proceso controlado, numerosas normas |
| Contrato flexible e incluso inexistente | Contrato prefijado |
| El cliente es parte del desarrollo del proyecto | Interacción del cliente con el equipo de desarrollo mediante reuniones |
| Grupos pequeños menores que 10 | Grupos grandes |
| Pocos artefactos | Más artefactos |
| Menor énfasis en la arquitectura de | La arquitectura del *software* es esencial |

| | |
|---|---|
| *software* | |

Con el objetivo de conseguir un *software* eficiente, las metodologías robustas se enfocan en el control y planificación del proyecto, demostrando su efectividad en proyectos de gran tamaño. Éstas no se adaptan a cambios continuos, por lo que la relación con el cliente no es constante, mientras que las ágiles se adecuan a los cambios, así mantienen el vínculo con el cliente estableciendo un comportamiento agradable entre él y el equipo de trabajo. Además las metodologías ágiles son utilizadas por equipos de trabajo pequeños.

Mediante la comparación anterior, se llega a la conclusión que para el desarrollo de la investigación se va a hacer uso de las metodologías ágiles, debido a que el tema a desarrollar requiere del vínculo con el cliente y se acoge a la programación en pareja. Además, de que se ajusta a los cambios que pudieran ocurrir en el transcurso del proyecto.

Entre las metodologías ágiles más conocidas y utilizadas por la universidad se pueden mencionar las siguientes:

**Scrum**

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de *software* se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto. La verdadera protagonista es la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (12).

**XP (Extreme Programming)**

Metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito del desarrollo de *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes y simplicidad en las soluciones implementadas (12).

XP se define especialmente adecuada para proyectos con requisitos imprecisos y cambiantes donde existe un alto riesgo técnico. Esta metodología no genera gran cantidad de artefactos a modelar o diseñar para la representación del sistema, pero si posibilita la creación de los artefactos necesarios para que se entienda el funcionamiento del negocio.

Utiliza las historias de usuario como la técnica para especificar los requisitos del *software*, en estas se describe brevemente las características que el sistema debe poseer.

**Prácticas en XP** (12):

- El juego de la planificación: es un espacio frecuente de comunicación entre el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.
- Entregas pequeñas: la idea es producir rápidamente versiones del sistema que sean operativas, aunque obviamente no cuenten con toda la funcionalidad pretendida para el sistema.
- Diseño simple: se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra debe ser removido inmediatamente.
- Pruebas: la producción de código está dirigida por las pruebas unitarias. Estas son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Los clientes son los encargados de escribir las pruebas funcionales para cada historia de usuario que deba validarse.
- Refactorización: la refactorización es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. La refactorización mejora la estructura interna del código sin alterar su comportamiento externo.
- Programación en parejas: la implementación de código debe realizarse en parejas de programadores. Las principales ventajas de introducir este estilo de programación consiste en detectar muchos errores conforme son introducidos en el código, para reducir la tasa de errores del producto final, para que los diseños sean mejores, el tamaño del código menor y los problemas de programación se resuelven más rápido. Además de posibilitar la transferencia de

conocimientos entre los miembros del equipo y de que varias personas entienden las diferentes partes del sistema.

- Propiedad colectiva del código: cualquier programador puede cambiar parte del código en cualquier momento. Esta práctica motiva a contribuir con nuevas ideas en todos los segmentos del sistema, evitando a la vez, que algún programador sea imprescindible para realizar cambios en alguna porción de código.
- Integración continua: cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día. Todas las pruebas son ejecutadas y tienen que ser aprobadas para que el nuevo código sea incorporado definitivamente.
- Cliente *in-situ*: el cliente tiene que estar presente y disponible todo el tiempo para el equipo de desarrollo.

Los principios y prácticas son de sentido común pero llevadas al extremo. La metodología no obliga, ni exige la aplicación estricta de las prácticas, a pesar de los beneficios que se pueden obtener con el uso de todas en conjunto. La principal ventaja de la metodología XP está en su alto grado de adaptabilidad, y su principal desventaja es su elevado costo en caso de no cumplir las metas.

**Selección de la metodología**

Luego del estudio de las metodologías antes expuestas, se aprecia que son muy parecidas, pero se escoge la metodología XP porque a diferencia de Scrum provee sencillez, tanto en su aprendizaje como en su aplicación, reflejando un ambiente que permita la colaboración entre todos los miembros del equipo de trabajo y el cliente. Es adaptable a la propuesta de solución por presentar un desarrollo iterativo incremental o sea pequeñas mejoras, unas tras otras en menos tiempo, la programación en parejas propiciando la calidad del código escrito, pues es revisado y discutido mientras se escribe. Además de un control de la simplicidad y la comunicación conllevando a que sea fácil identificar qué se debe y qué no se debe hacer.

### 1.5.2 Tipo de aplicación

Una aplicación es todo programa informático que permite a un usuario utilizar una computadora con un fin específico. Son programas escritos en cualquier lenguaje de programación, pueden tener varios tipos de

interfaz, que pueden ser de texto, gráfica o ambas. Existen dos tipos de aplicaciones, las web y las de escritorio.

Se llama **aplicación web** a aquella que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una Intranet mediante un navegador. Las aplicaciones web son fáciles de actualizar y mantener sin distribuir e instalar *software* en las PC clientes. Éstas proveen de información centralizada, movilidad, seguridad y copias de seguridad.

Se llama **aplicación de escritorio** a aquella que los usuarios instalan en las PC, las cuales son solo usadas en una red local. Son utilizados varios lenguajes de programación para su creación. Las mismas tienen mayor rendimiento y seguridad que las aplicaciones web. Además de ser robustas y estables poseen movilidad al igual que las aplicaciones web.

**Selección del tipo de aplicación**

Aunque las aplicaciones de escritorio son más rápidas que las web, la propuesta de solución se adecúa más a esta última, porque para la propuesta de solución se hace necesario que sea accesible desde cualquier lugar y no desde donde se tenga instalado previamente el *software* como en las de escritorio, que sea portable, o sea, que el usuario pueda acceder a la misma desde un navegador web.

### 1.5.3 Lenguaje informático

Un lenguaje informático es un lenguaje empleado por el ordenador. Existen varios tipos de lenguajes informáticos como por ejemplo los lenguajes de descripción (CSS), de marcado (HTML, XML) y de programación (PHP, JavaScript, Perl, Ruby, Python y Java).

**Lenguajes de programación**

En la actualidad para el desarrollo de sistemas web existen muchos lenguajes de programación. Algunos de los más importantes y utilizados son PHP, Perl, Ruby, Python, JavaScript y Java. La elección de alguno de éstos se basaría en el conocimiento que se tenga de dicho lenguaje, además de las particularidades de la aplicación a desarrollar.

Se escoge PHP v5.4.32 como lenguaje del lado del servidor, debido a su carácter libre, su simplicidad para aprender, su estabilidad y por su aplicación en la construcción de páginas web. Se utiliza del lado del

cliente JavaScript para simplificar los comandos comunes, además de que es utilizado actualmente para el desarrollo de páginas web.

**Lenguaje del lado del servidor**

PHP es un lenguaje interpretado, de alto nivel, embebido en páginas HTML, ejecutado en el servidor y de programación multiplataforma. Además es empleado en la construcción de páginas web de contenido dinámico, permitiendo el manejo eficiente de excepciones y el empleo de técnicas de programación orientada a objetos (13). Lenguaje más extendido en la web, que ha tenido una gran aceptación en la comunidad de desarrolladores, debido a la potencia y simplicidad que lo caracteriza, así como el soporte generalizado en la mayoría de los servidores de *hosting*. PHP permite embeber sus pequeños fragmentos de código dentro de la página HTML y realizar determinadas acciones de una forma fácil y eficaz, combinando lo que ya se sabe del desarrollo HTML. Es decir, con PHP escribimos *scripts* dentro del código HTML. Por otra parte, y es aquí donde reside su mayor interés con respecto a los lenguajes pensados para los CGI[4], PHP ofrece un sinfín de funciones para la explotación de bases de datos de una manera sencilla y sin complicaciones. Proporciona estabilidad a la aplicación pues utiliza su propio sistema de administración de recursos y dispone de un método de manejo de variables, conformando así un sistema robusto. Posibilita configurar diferentes niveles de seguridad para evitar ataques al código (14).

**Lenguaje del lado del cliente**

JavaScript es un lenguaje ideado para dotar de dinamismo, rapidez y agilidad a las páginas web. Puede tener distintas aplicaciones, pero la más común es la de ser un lenguaje de programación de propósito específico que se ejecuta del lado del cliente. En los desarrollos web JavaScript se mezcla de alguna manera con HTML y CSS o con lenguajes de programación como PHP. Lenguaje de programación completo, con una sintaxis, un conjunto de sentencias e instrucciones similares a las de otros lenguajes. Como peculiaridades se tiene que JavaScript es un lenguaje normalmente interpretado por el navegador web y por tanto se puede obtener en ocasiones resultados diferentes al pasar de un navegador a otro (15).

**Lenguaje de marcado HTML**

[4] **Interfaz de entrada común** (en inglés *Common Gateway Interface*, abreviado **CGI**) es una importante tecnología de la World Wide Web que permite a un cliente solicitar datos de un programa ejecutado en un servidor web.

HTML es un lenguaje de etiquetas. Indica al navegador como tiene que mostrar el contenido. Este separa el "contenido" (palabras, imágenes, audio, video) de la "presentación" (la definición del tipo de contenido y las instrucciones ellos tienen que mostrar). El mismo emplea un conjunto de elementos predefinidos que contienen una o más etiquetas, las cuales suelen ir encapsuladas entre los símbolos <>, y las etiquetas de cierre (que indican el final de un determinado contenido) están precedidas por una barra / (16).

Es el lenguaje básico de casi todo el contenido web. Específicamente, es el lenguaje con el que se escribe la estructura y la semántica del contenido de un documento web. El contenido dentro de una página web es etiquetado con elementos HTML como <img>, <title>, <p>,  (16).

La selección del lenguaje HTML5 para el maquetado de la propuesta de solución fue respaldada porque posee numerosas herramientas para el desarrollo web además de ser un estándar definido por W3C.

**Lenguaje de descripción CSS**

Tecnología desarrollada por el *World Wide Web Consortium* (W3C) con el fin de separar la estructura de la presentación. CSS posee el control de características gráficas tales como imágenes, colores de fondo, márgenes exactos y bordes, para evitar el trabajo del diseño de tablas complejas (17). Es decir, mecanismo que describe cómo se va a mostrar un documento en la pantalla.

Los estilos definen la forma de mostrar los elementos HTML y XML. Las hojas de estilo en cascada permiten a los desarrolladores web controlar el estilo y el formato de múltiples páginas web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento. Ésta funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos (18).

Se selecciona CSS3 para definir el estilo visual de la propuesta de solución y por ser un estándar definido por W3C.

### 1.5.4 *Frameworks*

Un *Framework* (marco de trabajo) ofrece componentes como una librería. Además, provee de plantillas o esqueletos que definen el funcionamiento de las aplicaciones. Éstos permiten manejar y controlar prácticamente toda la aplicación sin escribir mucho código (19).

**Selección del *framework* de desarrollo**

Symfony es uno de los *frameworks* más completos para el lenguaje PHP, de uso casi extensivo en el desarrollo web en la UCI, lo que brinda la posibilidad de que su comprensión se encuentre al alcance del conocimiento que se pueda adquirir de la extensa documentación que existe en la web. Además de ser rápido, flexible, sencillo de aprender y dominar, posee mejor rendimiento en aplicaciones reales. Este marco entre sus características proporciona la integración con otras aplicaciones adaptándose al propósito de la propuesta de solución. La versión a utilizar será la 2.3.7.

Aparte de las características mencionadas este brinda las siguientes particularidades que le hacen el marco de trabajo más adecuado para la investigación (20):

- Soporte: sigue una política de tipo LTS (*Long Term Support*). Las versiones estables se mantienen con una continua corrección de los errores.
- Licencia: utiliza una licencia MIT (*Massachusetts Institute of Technology*), con la que se puede hacer aplicaciones web comerciales, gratuitas y de *software* libre.
- Código: desde su primera versión este ha sido creado exclusivamente para PHP 5.
- Seguro: se puede controlar hasta el último acceso a la información e incluye por defecto protección contra ataques XSS y CSRF, contiene además un sistema potente de archivos *Logs*.
- Documentado: se trata del *framework* PHP mejor documentado.
- Calidad: su código fuente incluye pruebas unitarias y funcionales.
- Fácil de instalar y configurar en distintos sistemas operativos.
- Funciona con todas las bases de datos comunes (MySQL, PostgreSQL, SQLite, Oracle, MS SQL Server).

El *framework* es uno de los más usados para el lenguaje PHP con la implementación de ORM (*Object Relational Maping* o Mapeo de Objeto Relacional), patrones de diseño, herramientas de pruebas, conceptos de Modelo Vista Controlador (MVC) y totalmente caracterizado por ser orientado a objetos. Es diseñado para optimizar el desarrollo de aplicaciones web, proporcionando herramientas para agilizar aplicaciones complejas y guiando al desarrollador a acostumbrarse al orden y buenas prácticas dentro del proyecto. Reutiliza conceptos y desarrollos exitosos de terceros para integrarlos como librerías (20).

Es flexible, escalable y un potente *frameworks* PHP para la aplicación del patrón MVC, unos de los

patrones más eficaces en cuanto a programación web. Utiliza un sin número de componentes PHP reutilizables como seguridad de plantillas, traducción, validador y formularios de configuración. El *framework* es modularizador con *Composer*. Su objetivo es hacer que la creación de aplicaciones web y el mantenimiento sea rápido y con codificación menos repetitivas (21).

En la Figura 1 se muestra una comparación entre los *frameworks* PHP más usados, donde se puede apreciar que Symfony es uno de ellos.

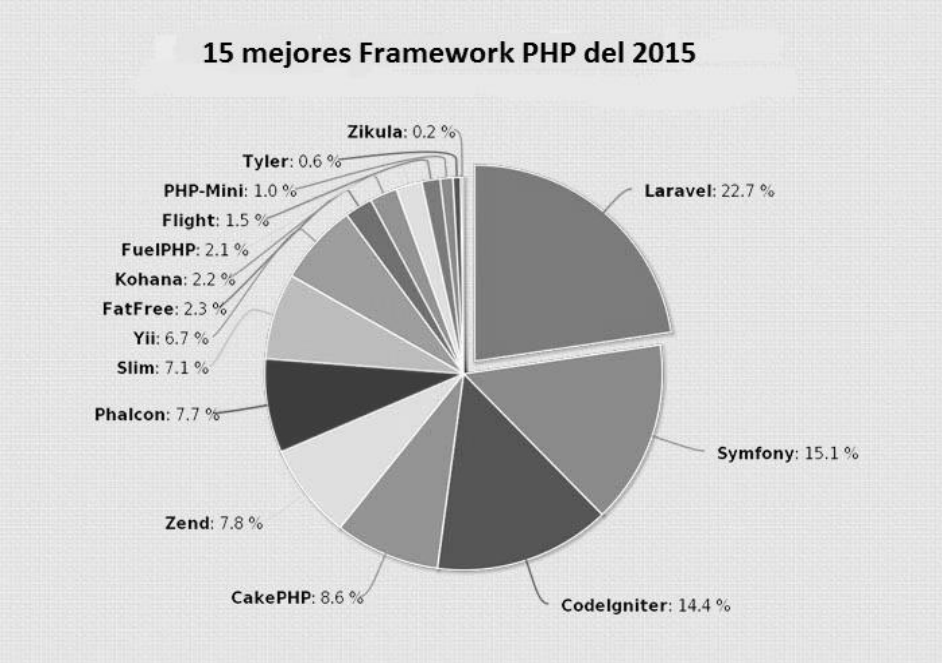


**Figura 5.Comparación entre los 15 *frameworks* PHP más usados (21)**

### 1.5.5 Lenguaje Unificado Modelado

UML es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de *software*. UML ofrece una forma de modelar objetos conceptuales como son los procesos de negocio y funciones del sistema, además de escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de *software* reusables (22). La versión de UML que será utilizada es la 2.0.

### 1.5.6 Visual Paradigm for UML

Plataforma de modelado mediante UML, diseñada para apoyar a los arquitectos de sistemas, desarrolladores y diseñadores UML, para acelerar el proceso de análisis y diseño de aplicaciones, durante todos los pasos del ciclo de vida de desarrollo de un *software* (23). Aunque XP no genere gran cantidad de artefactos, la herramienta CASE es utilizada para generar los diseños necesarios que esclarecerán el funcionamiento del negocio. La versión a utilizar es la 8.0.

### 1.5.7 PostgreSQL

El Sistema Gestor de Base de Datos (SGBD) PostgreSQL ofrece estabilidad, potencia, robustez, facilidad de administración e implementación de estándares, además está bajo licencia BSD[5]. También es de código abierto y brinda un control de concurrencia multiversión que permite trabajar con grandes volúmenes de datos, soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación (24).

Posee características significativas del motor de datos, entre las que se pueden incluir las subconsultas, los valores por defecto, las restricciones a valores en los campos (*constraints*) y los disparadores (*triggers*). Dispone de funcionalidades tales como, claves primarias, identificadores entrecomillados, entrada de enteros, binarios y hexadecimales (24).

El código fuente se encuentra disponible para todos sin costo alguno. Está disponible para 34 plataformas con la última versión estable. Es totalmente compatible con ACID (acrónimo de *Atomicity, Consistency, Isolation and Durability*; en español: Atomicidad, Consistencia, Aislamiento y Durabilidad). Funciona en los sistemas operativos Linux, Unix, y Windows. Debido a la liberación de la licencia, PostgreSQL se puede usar, modificar y distribuir de forma gratuita para cualquier fin, ya sea privado, comercial o académico (24).

**Selección de PostresSQL**

La elección de PostgreSQL en su versión 9.2.4 está dada por la facilidad de trabajo que brinda con las bases de datos, por el conocimiento previo que se tiene del mismo al funcionar en diferentes sistemas operativos y por dar soporte para el lenguaje de programación PHP, además de su carácter libre que le hace poder ser usado de forma gratuita.

[5] BSD: *Berkeley Software Distribution License*. Licencia de Distribución de *Software*.

### 1.5.8 Cliente del Sistema Gestor de Base de Datos

PgAdmin III: aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo el más completo y popular con licencia *Open Source*. Existen versiones disponibles para varios sistemas operativos. Es capaz de ejecutarse en cualquier plataforma. Está diseñado para responder a las necesidades de todos los usuarios. La aplicación también incluye un editor SQL con resaltado de sintaxis y un editor de código de la parte del servidor. La conexión al servidor puede hacerse mediante conexión TCP/IP y puede encriptarse mediante SSL para mayor seguridad (25). La versión a utilizar será 1.14.

### 1.5.9 Entorno de desarrollo

Un entorno de desarrollo (IDE) ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Existen una variedad de IDE Open Source ente los más usados a nivel internacional se encuentran PhpStorm y NetBeans (26).

**NetBeans** es un entorno de desarrollo (IDE) gratuito y de código abierto. Permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones web. Da soporte a varias tecnologías como Java, PHP, HTML5. El IDE puede ser instalado en varios sistemas operativos. Cuenta con un potente editor de código que permite el completamiento y resaltado del código. Mediante este es posible diseñar aplicaciones con solo arrastrar y soltar objetos sobre la interfaz de un formulario (27).

El IDE NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Se puede conectar a distintos sistemas gestores de bases de datos, como pueden ser Oracle, MySql y PostgreSQL, ver tablas, realizar consultas y modificaciones (28).

**PhpStorm** es uno de los IDE de programación más completo de la actualidad, desarrollado por Jetbrains. Cuenta con un editor que entiende su código y su estructura, brinda soporte para PHP, HTML, CSS y JavaScript. El IDE ofrece los mejores autocompletado de código y refactorizaciones en la prevención de errores. Tiene incorporado un formato de código compatible para Symfony, comprobación de errores sobre la marcha, además de variables locales relevantes al contexto y definidas por el usuario. Ofrece la posibilidad de evaluar una expresión en tiempo de ejecución (29).

El IDE posee diferentes tipos de licencias:

- Licencia Personal (a pagar para los desarrolladores individuales).
- Licencia Comercial (a pagar para las empresas y organizaciones).
- Licencia Académica (gratis para los estudiantes y profesores).
- Licencia Aula (gratis para los formadores e instituciones educativas).
- Licencia proyecto de código abierto (libre para proyectos de código abierto) (29).

**Selección del IDE**

A pesar que el PhpStorm supera al NetBeans en cuanto a rapidez de respuesta y que además son muy similares, se selecciona el IDE NetBeans en su versión 8.0 por la familiarización y experiencia de trabajo con esta herramienta, además de poseer las siguientes características:

- La plataforma NetBeans puede ser usada para desarrollar cualquier tipo de aplicación.
- Reutilización de Módulos.
- Instalación y actualización simple.
- Brinda un amplio soporte al desarrollo de aplicaciones web usando el *framework* Symfony.
- Producto libre y gratuito sin restricciones de uso (28).

### 1.5.10 Apache

El servidor de aplicaciones Apache, es un servidor web que por su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa (30). Este es de *software* libre y su popularidad sigue incrementándose debido a que gran cantidad de sitios web en internet están manejados por él. Apache es un proyecto de la Fundación de *Software* Apache, su objetivo es suministrar un servidor seguro, eficiente, y extensible que proporcione servicios HTTP en sincronía con los estándares HTTP actuales (31).

La flexibilidad, rapidez, eficiencia y la adaptabilidad a nuevos protocolos HTTP, a diferentes entornos y necesidades para el desarrollo de módulos específicos, le hacen a Apache en su versión 2.2.22 el servidor web portador de las características adecuadas para el desarrollo de la propuesta de solución. Además del popular uso que posee en la red.

### 1.5.11 PHPUnit

*Framework* que permite realizar las pruebas pertinentes al código verificando que el funcionamiento de las aplicaciones PHP sea el esperado y a su vez encontrando fallos y errores que una vez solucionados mejorarán la calidad del desarrollo web. Este *software* basado en la familia de *frameworks* xUnit constituye uno de los principales *frameworks* de pruebas unitarias en PHP. Symfony2 lo integra como librería independiente para desarrollar pruebas (32).

Posibles respuestas que se pueden obtener luego de ejecutar cada prueba unitaria:

- Se observa un punto (.) si se tiene éxito.
- Una “F” si ocurrió un error.
- Una “I” si la prueba está marcada como incompleta.
- Una “S” si se ha marcado como Omitida (*Skipped*) (33).

Existen muchas y diferentes condiciones (aserciones) que pueden ayudar a probar los resultados de los tipos de llamadas en las aplicaciones. Las aserciones proporcionadas por PHPUnit cubren la mayoría de los casos que se quieran probar. A continuación se muestran en la Tabla 2 las principales aserciones con las que trabaja PHPUnit (33).

**Tabla 2.Tipos de aserciones del *framework* PHPUnit (33)**

| Aserción | Descripción |
|---|---|
| AssertTrue/AssertFalse | Comprueba la entrada para verificar si es igual a true/false |
| AssertEquals | Comprueba el resultado frente a otra entrada en busca de coincidencias |
| AssertGreaterThan | Comprueba el resultado para ver si es mayor que un valor (también hay LessThan, GreaterThanOrEqual, y LessThanOrEqual) |

| | |
|---|---|
| AssertContains | Comprueba que la entrada contiene un valor específico |
| AssertType | Comprueba que una variable es de un cierto tipo |
| AssertNull | Comprueba que una variable sea nula |
| AssertFileExists | Comprueba que un archivo exista |
| AssertRegExp | Comprueba la entrada con una expresión regular |

### 1.5.12 *Framework* de interfaz JQuery

**JQuery** es una serie de funciones y métodos de JavaScript, utilizado en el desarrollo de páginas web. Su objetivo es simplificar los comandos comunes de JavaScript. La librería presenta numerosas ventajas, dentro de estas se puede mencionar la compatibilidad con navegadores como Mozilla Firefox, Internet Explorer y Google Chrome (34).

La versión a utilizar en el desarrollo de este sistema será la 1.11.4 teniendo en cuenta que sus componentes visuales permiten desarrollar en poco tiempo una aplicación agradable a la vista de los usuarios. Está bien documentado y cuenta con componentes para la interfaz de usuario y galerías. Tiene una gran cantidad de *plugins* que pueden ser adicionados fácilmente, lo cual permite ahorrar tiempo y esfuerzo.

**Conclusión**

Tras el análisis de las soluciones similares, se toman en cuenta algunas de las características de los sistemas informáticos como aporte a la investigación del tema y su posterior desarrollo. Se construyó el marco teórico para servir de guía a la informatización de las RTF en el área Modelo de datos del desarrollo de *software,* poniendo en práctica los diferentes conceptos que lo componen. El estudio de las tecnologías permitió la selección de las herramientas necesarias para el desarrollo de la propuesta de solución.

## Capítulo 2: Propuesta de solución

En este capítulo se describe la propuesta de solución dándose a conocer los objetivos, características y roles de la misma. Basándose en las primeras fases de la metodología XP se conforman las historias de usuarios correspondientes a las funcionalidades descritas por el cliente, el plan de entregas y el plan de iteraciones.

**Criterios elaborados por especialistas en base de datos del Centro**

Para una mejor comprensión de los criterios de revisión, que contiene la lista de chequeo del Modelo de datos, los especialistas en el área conformaron una serie de criterios que se tomarán como aporte a la realización de la propuesta de solución. Estos criterios elaborados se enfocan en la vista de datos, de seguridad y despliegue del expediente de proyecto de arquitectura. A continuación se exponen los criterios con una descripción de cada uno de estos.

Vista de Datos

- Se encuentra detallada la caracterización del producto según la vista: este parámetro verifica que se identifiquen correctamente los diferentes tipos de datos que se quieren almacenar.
- Se encuentran descritos los escenarios y patrones de solución: este parámetro verifica que haya una correcta descripción de los escenarios y patrones de descripción en el documento.
- Se encuentra en el diccionario de datos los elementos necesarios para identificar los conceptos para la modelación de la base de datos: este parámetro verifica que los términos se encuentren con la definición correcta y en caso de referenciar otro documento, verificar que en el mismo se encuentre correctamente.
- Se encuentran las especificaciones del diseño físico de la solución en el Modelo de datos: este parámetro verifica que se encuentren las especificaciones del diseño físico en el modelo de datos, en caso de referenciar algún documento, verifica que el mismo esté correcto.

Vista de Seguridad

- Se identifican los requisitos de seguridad del producto: este parámetro verifica que se especifiquen en las preguntas mínimas que deben quedar respondidas y que deben permitir identificar requisitos de seguridad.

- Se identifica la autenticación y la capa de presentación: este parámetro verifica que se evalúe asignándole la puntuación máxima a los elementos que debe utilizar para el diseño y desarrollo de la aplicación.
- Se especifica el nivel de la capa de negocio: este parámetro verifica que se evalúe el nivel de la capa de negocio asignándole la puntuación máxima a los elementos que se deben utilizar en el diseño y desarrollo de la aplicación.
- Se especifica el nivel de la capa de datos: este parámetro verifica que se evalúe asignándole la puntuación máxima a los elementos que debe utilizar para el diseño y desarrollo de la aplicación.
- Se especifica la seguridad del servidor de la aplicación: este parámetro verifica que se identifique el estado de seguridad de la aplicación vista desde el entorno.
- Se especifica la tecnología para el sellado del código: este parámetro verifica que se especifique la tecnología que utilizará para la protección del código.

Vista de Despliegue

- Se especifica la solución vista desde las licencias de los componentes que la forman: este parámetro verifica que se seleccione qué tipos de licencias soportan los componentes de la solución.
- Se especifican los requerimientos mínimos de instalación: este parámetro verifica que se seleccionen los requerimientos mínimos para la instalación.

## 2.1 Descripción de la propuesta de solución

Con el presente trabajo se pretende lograr una aplicación web que posibilite la ejecución de las RTF y el procesamiento de los resultados de las revisiones a los distintos proyectos del Centro FORTES. El sistema propuesto tiene como objetivo facilitar el trabajo a los revisores técnicos o especialistas en el área de Modelo de datos, minimizando el tiempo de revisión y entrega del producto. El mismo contará con la opción de registrar todas las RTF realizadas en dicha área, para facilitar la obtención de conocimiento de las mismas y almacenar los resultados encontrados como evidencia en el dictamen técnico que se generará automáticamente al finalizar dicha revisión.

El proceso de revisión es manejado por una lista de chequeo correspondiente a comprobar los requisitos

que posee el producto recibido. Esta lista contiene características específicas del modelo de datos de cada proyecto. Los indicadores se basan en preguntas, las cuales generan un método de evaluación, especificando la descripción de la puesta en práctica de cada criterio según se encuentre en los documentos del expediente del proyecto. Cada revisión, para poder crearse tiene que tener registrado los objetivos, el alcance, el cronograma de actividades, así como también, el revisor que ejecuta la inspección. Dada la necesidad de conocer de forma más detallada datos sobre el manejo de los diversos parámetros, estándares, buenas prácticas, problemas y recomendaciones se conformó una nueva lista de chequeo con otra forma de evaluación diferente a la propuesta por el departamento de calidad UCI. Todo esto posibilitará planificar y ejecutar nuevas estrategias de seguimiento a las no conformidades detectadas en cada proyecto.

El sistema permite desempeñar a los usuarios los siguientes roles: Revisor y Administrador. Además es capaz de administrar los usuarios, gestionar sus permisos, sus datos específicos y los proyectos. El acceso al sistema será mediante la autenticación de los roles que interactuarán con el sistema, donde se permite a los usuarios registrados realizar búsquedas y ordenamientos de elementos conforme a su consideración. También brinda la posibilidad de valorar los criterios generales y los criterios específicos, consultar vista previa del dictamen y exportar a pdf el documento obtenido.

Se muestra en la Figura 6 el proceso de negocio general de la ejecución la RTF al área Modelo de datos de la propuesta de solución. Además se describen las actividades que contienen este proceso.

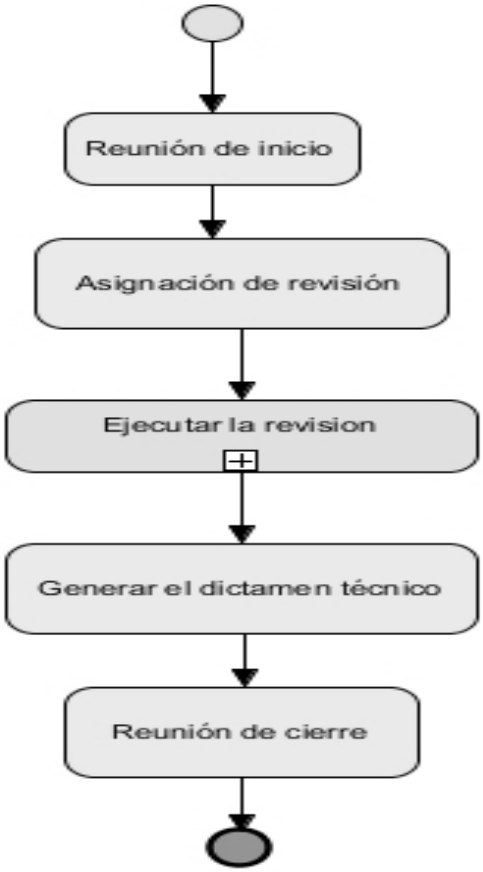


**Figura 6.Diagrama de proceso de negocio**

Reunión de inicio:

- Se analiza la necesidad de realizar una RTF a alguna organización productiva. Se selecciona el personal con posibilidad de llevar a cabo las revisiones. Se notifica a los involucrados la realización de la RTF.
- Análisis de disponibilidad de información suficiente y apropiada para ejecutar la RTF.
- Se determinan los objetivos de revisión.
- Se define el alcance de la revisión.
- Se definen los criterios de la revisión.

Asignación de revisión:

- Se establece el líder del equipo revisor.

- Se establece el equipo revisor.

Ejecución revisión:

- Se recopila y se verifica la información del expediente del proyecto.
- Se generan los hallazgos o no conformidades de las RTF.
- Se elabora y se presentan las acciones correctivas o de mejora.
- Se genera el dictamen técnico
- Se elabora el informe o dictamen técnico de las RTF con las observaciones encontradas.

Reunión de cierre:

- Se presentan los resultados finales de la revisión y se establecen las estrategias de seguimiento.
- Se prepara el expediente de las RTF.
- Se almacena la información de las RTF.

En la Figura 7 se muestra el proceso de negocio para el rol Administrador de la propuesta de solución, evidenciándose la secuencia de actividades que debe realizar el mismo.

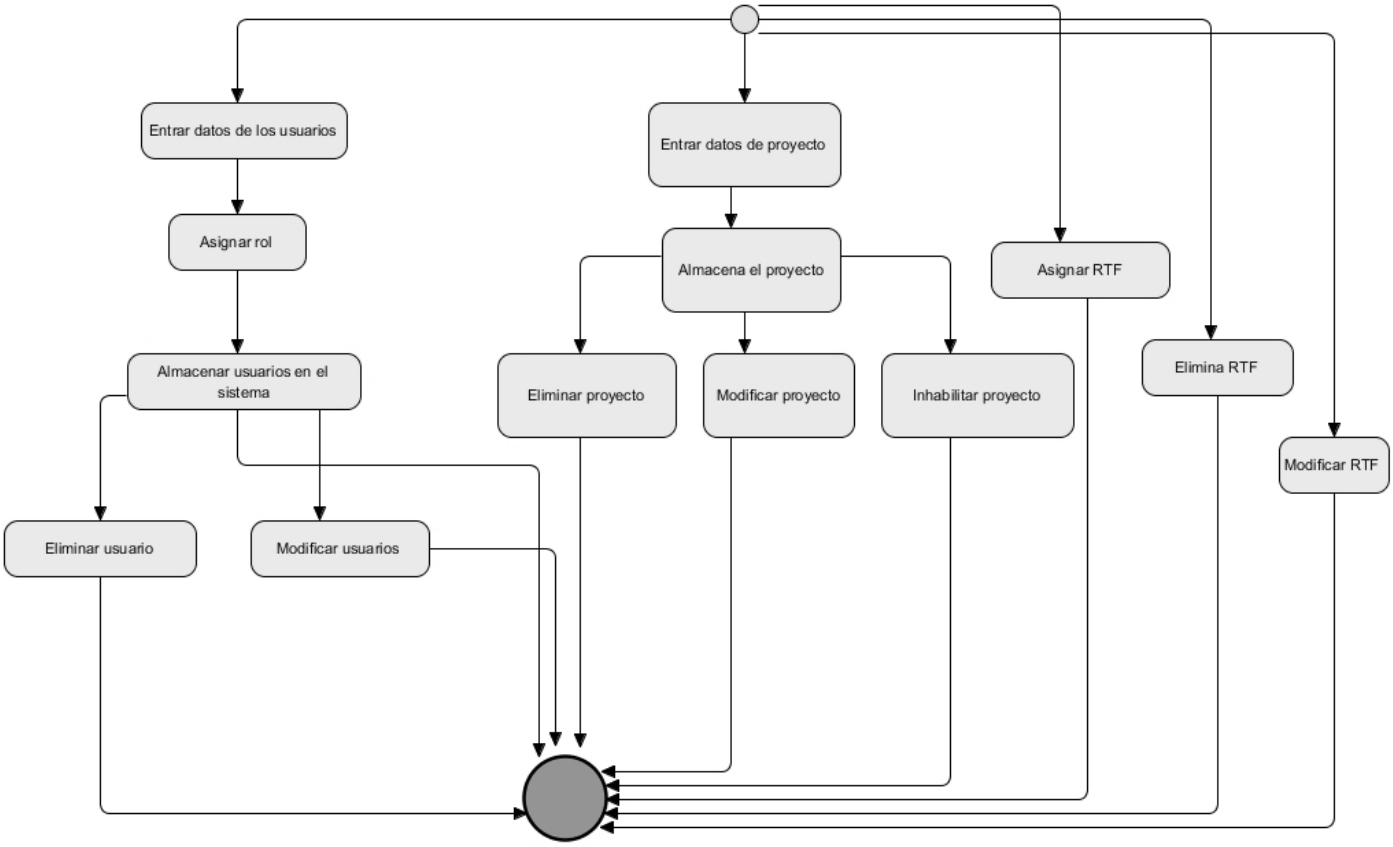


**Figura 7.Diagrama de proceso de negocio para el rol Administrador**

En la Figura 8 se muestra el proceso de negocio para el rol Revisor de la propuesta de solución, donde se muestra la secuencia de pasos que debe ejecutar el revisor para revisar los documentos y generar el informe técnico.

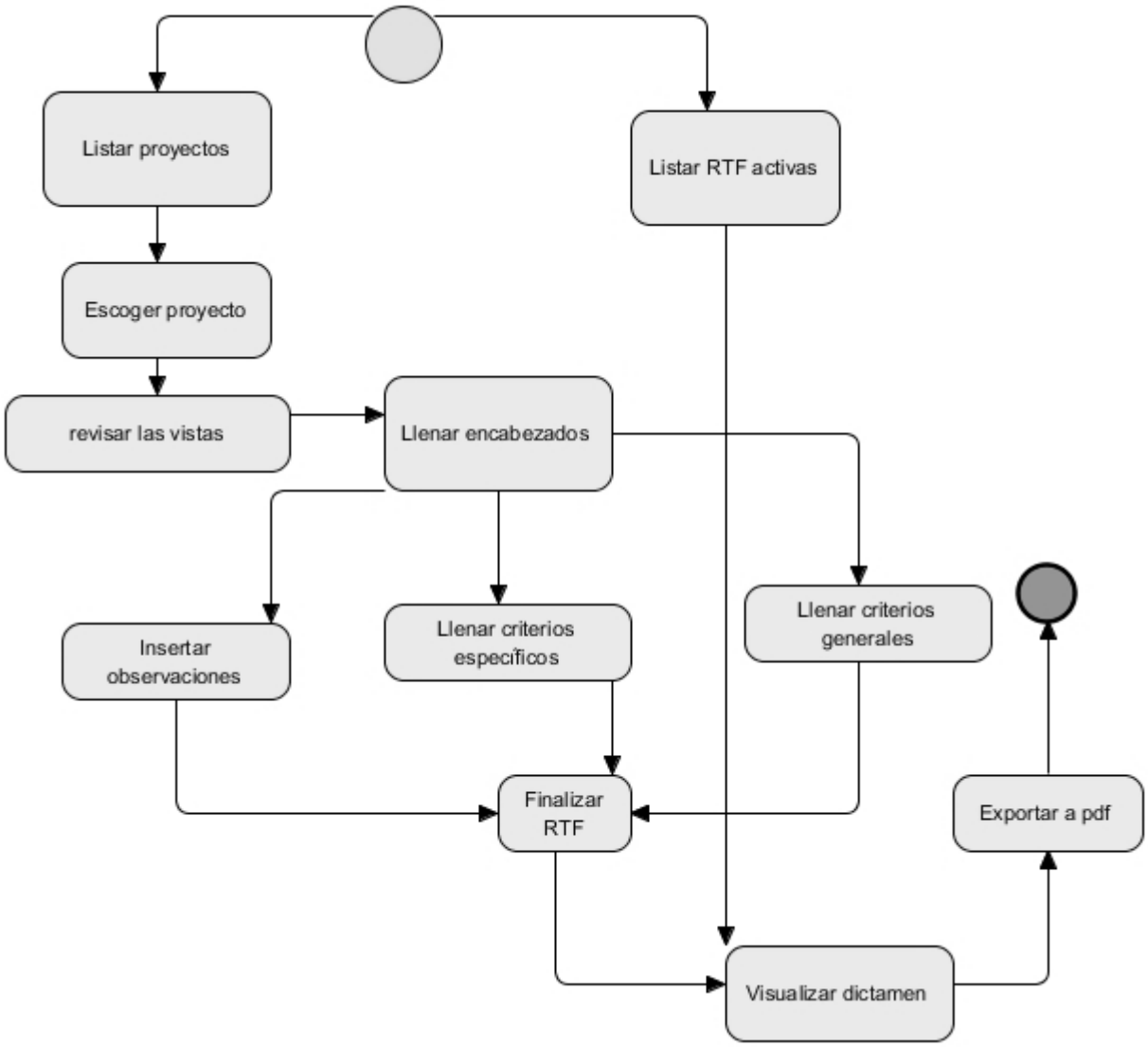


**Figura 8.Diagrama de proceso de negocio para el rol Revisor**

**Diagrama conceptual del dominio**

Su relevancia radica en que si no se comprende el dominio del problema y las reglas de negocio no es posible sugerir y desarrollar un sistema acorde a las necesidades del cliente. El diagrama muestra las relaciones entre los conceptos principales, clases y módulos que intervienen en el proceso de negocio. Además se tiene en cuenta los roles, multiplicidad y generalizaciones. A continuación se muestra el

diagrama conceptual del dominio, Figura 9, creado con el objetivo de esclarecer el funcionamiento de la propuesta de solución.

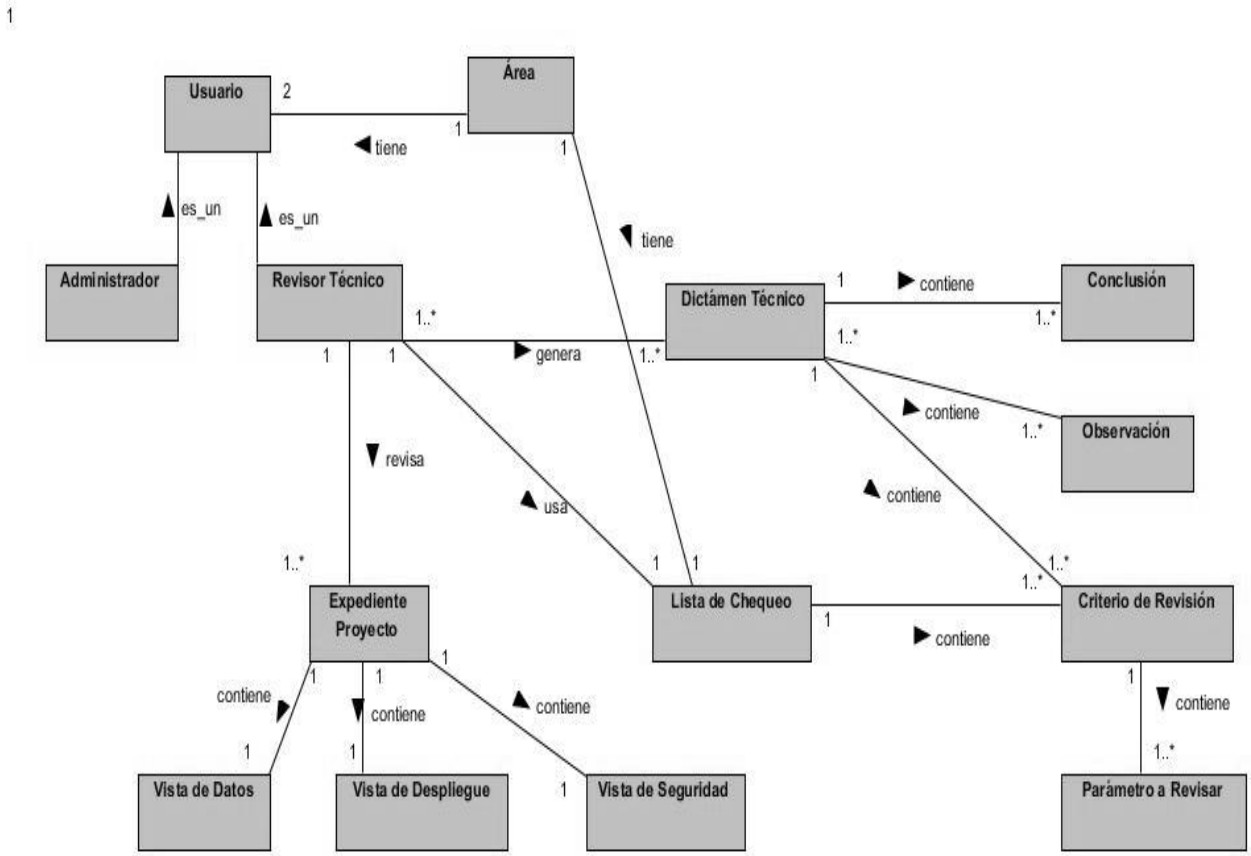


**Figura 9.Diagrama conceptual del dominio**

## 2.2 Fases Exploración y Planificación de Entrega

En estas fases los clientes en conjunto con el equipo de desarrollo plantean y construyen el prototipo (tarjeta) de las historias de usuarios (HU) que serán de interés para la entrega del producto, se establece la prioridad de cada HU, se realiza la estimación del esfuerzo de desarrollo necesario de cada historia en un rango de una a tres semanas. Se llegan a acuerdos sobre el contenido de las entregas y se realiza el cronograma de entrega. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.

## 2.3 Funcionalidades del sistema

- Autenticar usuario: esta funcionalidad permite autenticar a un usuario al introducir usuario y contraseña.
- Gestionar usuario: esta funcionalidad englobará los requisitos crear, eliminar y modificar usuario.
- Gestionar Proyecto: esta funcionalidad permitirá crear, modificar y eliminar los datos principales de los proyectos a realizar RTF.
- Revisión activa: esta funcionalidad permitirá mostrar el listado de las revisiones que se encuentran activas.
- Revisión abortada: esta funcionalidad permitirá mostrar el listado de las revisiones que se encuentran abortadas.
- Gestionar revisión técnica: esta funcionalidad permitirá crear, eliminar, mostrar y modificar una RTF.
- Gestionar criterios de revisión: esta funcionalidad permitirá crear, eliminar y modificar los parámetros de cada criterio de revisión.
- Generar un DT: esta funcionalidad permitirá generar la plantilla del Dictamen Técnico (DT) asociado a cada revisión con sus respectivos campos rellenos.
- Exportar a pdf: esta funcionalidad permitirá exportar el DT como pdf.
- Visualizar DT: esta funcionalidad permitirá ver el informe conformado.
- Buscar: esta funcionalidad permitirá ejecutar la búsqueda por Dictamen Técnico y por proyecto.
- Solicitar prórroga de tiempo: dado que puede surgir cualquier imprevisto esta funcionalidad permite extender el plazo de revisión.
- Reporte: permitirá recoger las observaciones hechas, el estado de las revisiones y las RTF realizadas.

## 2.4 Características del sistema:

- *Software*: las computadoras deben tener instalado un navegador web
- Interfaz: debe ser amigable para usuarios. Su funcionamiento debe ser intuitivo, y requerir de información mínima.
- Usabilidad: claridad y buena organización de la información, permitiendo la interpretación correcta

e inequívoca de la información.

- Rendimiento: ejecución de acciones de una manera rápida, minimizando los pasos a dar en cada proceso.
- Configuración y seguridad: la aplicación requiere validación de uso por el servidor local o central. Si no obtiene la validación, no puede iniciar. La validación permite al servidor determinar, los datos que enviará a la aplicación, dependiendo del nivel de acceso inscrito al usuario. Es decir, el usuario recibe solo el contenido y material necesario para realizar las RTF.
- Funcionalidad: reducir al mínimo el tiempo en que carga la aplicación.
- Legales: una vez terminado la aplicación web debe ser sometida a una evaluación y certificación por parte del cliente del producto.
- Derecho de autor: creación de un documento en el que quede plasmada la autoría de las personas que desarrollaron la aplicación web.

**Historias de usuarios**

Técnica que utiliza la metodología XP para representar el levantamiento de requisitos del sistema. Según las búsquedas hechas no existe un prototipo específico para definir las HU. Son elaboradas por los clientes según las necesidades que el sistema requiera todas escritas en lenguaje de usuarios. Se realiza una por cada funcionalidad.

La metodología no posee un estándar fijo para la estimación de una HU. Las mismas se consideran sencillas cuando su esfuerzo se realiza en menos de una semana (5 días), en este caso se agruparía. Si fuese mayor de tres semanas se dividiría en HU de menores esfuerzos.

Se propone como prototipo de HU para la aplicación el siguiente, conformado por el cliente conjuntamente con el equipo de desarrollo:

**Tabla 3. Descripción de HU**

| **Historia de usuario** | |
|---|---|
| **Número:** número de HU. | **Nombre:** nombre de HU. |
| **Usuario:** rol que interactúa con el sistema. | |

| | |
|---|---|
| **Prioridad:** nivel de prioridad en el negocio (Alto, Medio, Bajo). | **Riesgo:** criticidad de la HU para el negocio (Alto, Medio, Bajo). |
| **Estimación:** tiempo de desarrollo (en días) de la HU. | **Iteración:** número de iteración en que será implementada. |
| **Descripción:** se hace una breve descripción de la HU. | |
| **Observación:** es para agregar alguna observación si es necesario. | |

A continuación se muestran algunas de las HU de la propuesta de solución, las restantes se pueden encontrar en el **Anexo 2**.

**Tabla 4. HU Autenticar usuario**

| **Historia de usuario** | |
|---|---|
| Número: 1 | Nombre: Autenticar usuario. |
| Usuario: Administrador, Revisor | |
| Prioridad: Alta | Riesgo: Media |
| Estimación: 3 días | Iteración: 1 |
| Descripción: el sistema debe permitir autenticar a un usuario. El usuario Administrador o Revisor debe autenticarse para realizar cualquier tarea dentro del sistema. | |
| Observación: el sistema mostrará en cada caso el perfil de usuario o indicará un error en caso de datos incorrectos. | |

**Tabla 5.HU Revisión activa**

| Historia de usuario | |
|---|---|
| **Número:** 4 | **Nombre:** Revisión activa |
| **Usuario:** Revisor, Administrador | |
| **Prioridad:** Baja | **Riesgo:** Baja |
| **Estimación:** 2 días | **Iteración:** 1 |
| **Descripción:** el sistema mostrará un listado con las revisiones que se encuentren en estado activa. | |
| **Observación:** esta actividad solo se puede realizar después de autenticarse. | |

**Tabla 6.HU Revisión abortada**

| Historia de usuario | |
|---|---|
| **Número:** 5 | **Nombre:** Revisión abortada |
| **Usuario:** Revisor, Administrador | |
| **Prioridad:** Baja | **Riesgo:** Baja |
| **Estimación:** 2 días | **Iteración:** 2 |
| **Descripción:** el sistema mostrará un listado con las revisiones que se encuentren en estado abortadas. | |
| **Observación:** esta actividad solo se puede realizar después de autenticarse. | |

**Tabla 7.HU Generar Dictamen Técnico**

| Historia de usuario | |
|---|---|
| **Número:** 8 | **Nombre:** Generar DT |
| **Usuario:** Revisor | |
| **Prioridad:** Alta | **Riesgo:** Alta |
| **Estimación:** 5 días | **Iteración:** 3 |
| **Descripción:** esta funcionalidad debe permitir generar el informe técnico conformado y estructurado. | |
| **Observación:** esta actividad solo se puede realizar después de autenticarse. Se deben haber insertado todos los datos necesarios en la funcionalidad crear revisión técnica para que el informe obtenido contenga las especificaciones requeridas. | |

**Tabla 8.HU Solicitar prórroga**

| Historia de usuario | |
|---|---|
| **Número:** 12 | **Nombre:** Solicitar prórroga |
| **Usuario:** Revisor | |
| **Prioridad:** Media | **Riesgo:** Media |
| **Estimación:** 5 días | **Iteración:** 4 |
| **Descripción:** esta funcionalidad debe permitir solicitar un tiempo de revisión en caso de que exista un problema determinado. | |

| |
|---|
| **Observación:** esta actividad solo se puede realizar después de autenticarse. |

**Tabla 9.HU Reporte**

| **Historia de usuario** | |
|---|---|
| **Número:** 13 | **Nombre:** Reporte |
| **Usuario:** Revisor | |
| **Prioridad:** Media | **Riesgo:** Media |
| **Estimación:** 5 días | **Iteración:** 4 |
| **Descripción:** esta funcionalidad debe permitir generar un reporte según las RTF realizadas, observaciones y estado de revisión de las mismas. | |
| **Observación:** esta actividad solo se puede realizar después de autenticarse. | |

**Plan de entregas**

El cronograma de entregas es conformado por los actores del proyecto (cliente, desarrolladores), establece qué historias de usuario serán agrupadas según sus prioridades para conformar una entrega, y el orden de las mismas. El mismo se realiza en base a las estimaciones de tiempo de desarrollo.

El Plan de Entregas propuesto constará con el nombre del entregable, con tres entregas cada una en su versión correspondiente, además de las HU que serán implementadas en cada entrega. De esta manera queda conformado el mismo.

**Entregable:** Informatización del procedimiento de RTF en el área Modelo de datos para el Centro FORTES

**Versión 0.1:** La primera entrega de la aplicación será en la 4$^{ta}$ semana de marzo.

**Versión 0.2:** La segunda entrega de la aplicación será en la 4ta semana de abril.

**Versión 0.3:** La tercera entrega de la aplicación será en la 3ra semana de mayo.

**Tabla 10.Funcionalidades por entregas**

| Historia de Usuario | 1ra entrega | 2da entrega | 3ra entrega |
|---|---|---|---|
| HU1: Autenticar usuario | X | | |
| HU2: Gestionar usuario | X | | |
| HU3: Gestionar Proyecto | X | | |
| HU4: Revisión activa | X | | |
| HU5: Revisión abortada | X | | |
| HU6: Gestionar revisión técnica | X | | |
| HU7: Gestionar criterios de revisión | X | | |
| HU8:Generar un DT | | X | |
| HU9: Exportar a pdf | | X | |
| HU10: Visualizar DT | | X | |
| HU11: Buscar | | X | |
| HU12: Solicitar prórroga de tiempo | | | X |
| HU13: Reporte | | | X |

**Iteraciones por entregas**

En esta fase se procede a definir cuantas iteraciones serán necesarias para el proyecto. La metodología XP plantea que la duración de cada iteración debe estar comprendida de una a cuatro semanas. Se toman para la primera iteración las HU que suelen definir la arquitectura de todo el sistema, para la elección de las siguientes iteraciones el cliente es quien decide su orden. Al final de las implementaciones se debe también efectuar todas las pruebas de aceptación creadas por el cliente.

**Tabla 11.Plan de iteraciones**

| Iteraciones | Orden de las HU a implementar | Cantidad de tiempo de trabajo |
|---|---|---|
| Iteración 1 | HU1. Autenticar usuario<br>HU2. Gestionar usuario<br>HU3. Gestionar Proyecto<br>HU4. Revisión activa | 4 semanas |
| Iteración 2 | HU5. Revisión abortada<br>HU6. Gestionar revisión técnica<br>HU7. Gestionar criterio de revisión | 4 semanas |
| Iteración 3 | HU8. Generar un DT<br>HU9. Exportar a pdf<br>HU10. Visualizar DT<br>HU11. Buscar | 4 semanas |
| Iteración 4 | HU12. Solicitar prórroga de tiempo<br>HU13. Reporte | 3 semanas |

## 2.5 Fase de Diseño

Esta fase propone un diseño simple y sencillo, fácil de implementar con una correcta especificación de los nombres de métodos y clases para ayudar a comprender y facilitar la reutilización de los códigos. Además XP sugiere el uso de las tarjetas CRC (Clases, Responsabilidades y Colaboración) siguiendo el uso de la POO (programación orientada a objeto).

**Tarjetas Clase-Responsabilidades-Colaboradores (CRC)**

La metodología XP propone como técnica de diseño orientada a objetos, las tarjetas CRC procurando de esta forma un diseño lo más simple posibles. Para ello, las mismas crean una relación de las clases a utilizar en la implementación del sistema. Además identifican las responsabilidades como métodos y sus colaboradores como llamadas a otras clases.

Se muestran a continuación tres tarjetas CRC de la propuesta de solución correspondiente a las clases controladoras, ya que son las encargadas de manejar gran parte de las funcionalidades en el sistema. Las restantes tarjetas se pueden encontrar en el **Anexo 3.**

**Tabla 12.Descripción de la tarjeta CRC**

| **Nombre de la clase :** nombre de clase | |
|---|---|
| **Responsabilidades:** métodos de la clase. | **Colaboradores:** llamadas a otras clases. |

**Tabla 23.Tarjeta CRC AdminController**

| **Nombre de la clase :** AdminController | |
|---|---|
| **Responsabilidades** | **Colaboradores** |
| usuariosAction() | UserManager,Form |
| editRolAction() | UserManager, Controller, Form |
| salvarRoleAction() | UserManager, Form, UserInterface, Controller |

| | |
|---|---|
| blockAction() | UserManager, UserInterface, Controller |
| unblockAction() | UserManager, UserInterface, Controller |
| deleteAction() | UserManager, ContainerInterface, AbstractManagerRegistry, Controller |
| usuarioByUsernameAction() | ParameterBag, ContainerInterface, UserManager |

**Tabla 34.Tarjeta CRC DefaultController**

| **Nombre de la clase :** DefaultController | |
|---|---|
| **Responsabilidades** | **Colaboradores** |
| indexAction() | Date, Controller |

**Tabla 45.Tarjeta CRC ObservacionController**

| **Nombre de la clase :** ObservacionController | |
|---|---|
| **Responsabilidades** | **Colaboradores** |
| indexAction() | AbstractManagerRegistry, Controller |
| createAction() | Observacion, ObservacionController, Form, AbstractMAnagerRegistry |
| createCreateForm() | Controller, ObservacionType, Form |
| newAction() | Observacion |
| showAction() | ObservacionController |
| EditAction() | ObservacionController, Form |

| | |
|---|---|
| createEditForm() | Controller, Observacion |
| updateAction() | Controller, ObservacionController, Form, AbstractMAnagerRegistry |
| deleteAnteriorAction() | Controller, ObservacionController |
| deleteAction() | AbstractManagerRegistry, Controller |
| createDeleteForm() | FormConfigBuilder, Controller |
| addObservacionAction() | ObservacionController, Controller, Form |

**Conclusión**

Con el desarrollo de este capítulo se logró conformar las historias de usuarios correctamente descritas, el plan de entregas y el plan de iteraciones mediante la comunicación del cliente con el equipo de trabajo, sirviendo esto para asegurar la implementación de las funcionalidades determinadas y brevemente detalladas. Además de generar el diagrama conceptual del negocio para un mejor entendimiento y dominio de la aplicación, se realizó la descripción de la propuesta de solución donde se establecieron los roles Administrador y Revisor que intervendrán en el proceso de ejecución de la misma haciendo de esta segura y confiable. Se hizo un análisis del diseño y se generaron las tarjetas CRC de la aplicación.

# Capítulo 3: Implementación y Pruebas

En este capítulo se describe la arquitectura que seguirá la propuesta de solución conjuntamente con los patrones que acompañarán el proceso de diseño de la misma. Se detallarán varios artefactos como Diagrama Entidad Relación. Durante el proceso de codificación de las funcionalidades se definen las tareas en que se dividió cada HU y las pruebas realizadas a estas. Además se genera por cada funcionalidad su correspondiente caso de prueba.

## 3.1 Descripción de la arquitectura

Symfony determina como patrón arquitectónico el Modelo Vista Controlador (MVC) porque concede a la aplicación una estructura más visible y ayuda al programador web a organizar y a construir aplicaciones altamente modulares, modificables y mantenibles (que pueda evolucionar durante mucho tiempo) (35). El proceso de ejecución del patrón se muestra en la Figura 10 y se describe seguidamente la secuencia de pasos del mismo.

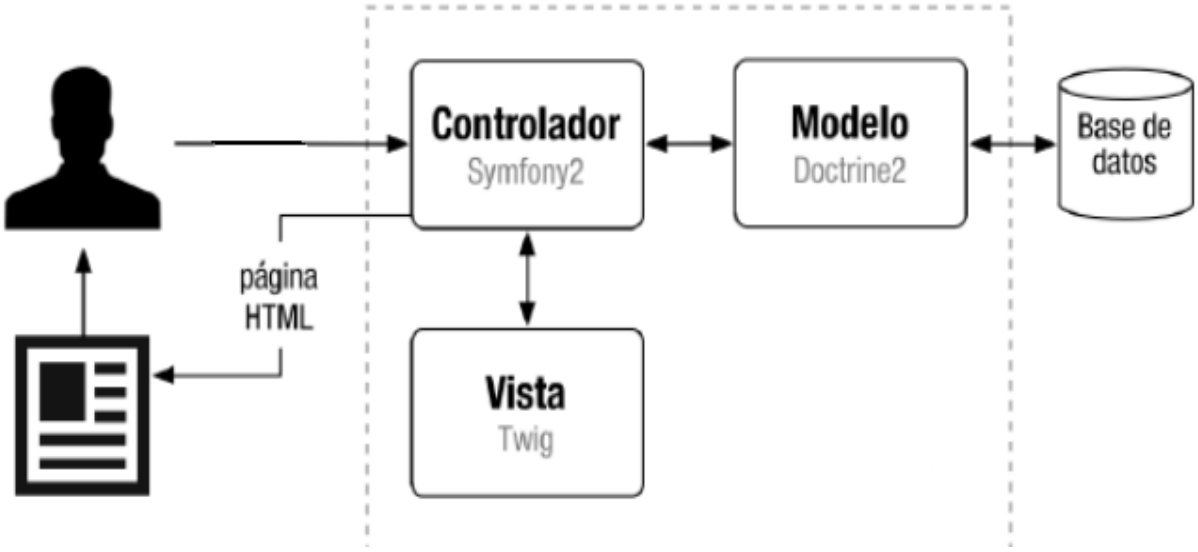


**Figura 10.Proceso de ejecución del patrón MVC (33)**

**Secuencia de pasos del proceso de ejecución del patrón MVC**

- El usuario envía una petición al controlador vía url.
- El controlador solicita al modelo los datos y este a la base de datos.

- La base de datos envía al modelo los datos solicitados.
- El modelo devuelve los datos al controlador.
- El controlador selecciona la vista adecuada para los datos solicitados por el usuario.
- La vista envía al controlador la página html seleccionada.
- El controlador devuelve al usuario una página html con los datos solicitados.

El patrón arquitectónico MVC permite mejorar la calidad y organización del código dividiéndose en tres capas diferentes: el **Modelo** contiene la información almacenada en una base de datos junto con las reglas de negocio, este transforma esa información teniendo en cuenta las acciones de los usuarios, la **Vista** es la página HTML o interfaz gráfica que interactúa con el usuario y el **Controlado**r es el encargado de responder y manejar las solicitudes de los usuarios procesando la información almacenada y generando el contenido HTML.

La programación se puede simplificar si se utilizan otros patrones de diseño. De esta forma, las capas del modelo, la vista y el controlador se pueden subdividir en más capas.

A continuación se muestra en la Figura 11 como se evidencia el patrón arquitectónico Modelo Vista Controlador en la aplicación:

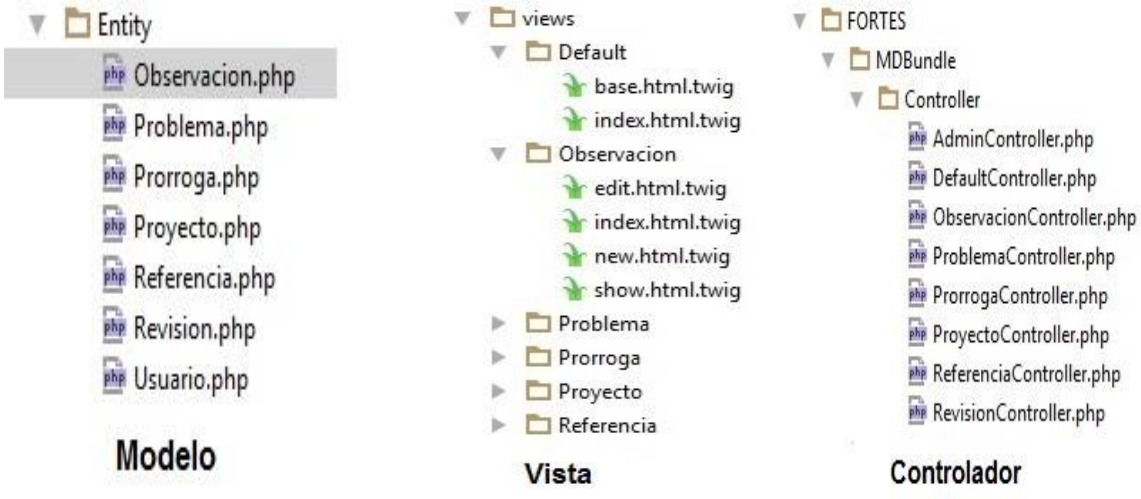


**Figura 11.Aplicación del patrón MVC**

## 3.2 Patrones de diseño

Los patrones de diseño son utilizados para la búsqueda de soluciones de problemas que se pueden encontrar a la hora de desarrollar una aplicación. Son considerados por los programadores como buenas prácticas en el diseño orientado a objeto. A continuación serán descritos los patrones utilizados para dar mayor estilo y calidad de diseño al desarrollo del sistema propuesto.

### 3.2.1 Patrones GRASP[6]

- **Experto:** resuelve los problemas de asignar responsabilidades a los expertos en información (36). Se utilizó en la capa de abstracción de la base de datos. Con el uso del ORM Doctrine. Las clases que representan las entidades del modelo de datos son generadas automáticamente por el *framework*.
- **Bajo Acoplamiento:** resuelve los problemas de asignar responsabilidades de forma tal que el acoplamiento permanezca bajo (36). Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.
- **Alta Cohesión:** resuelve el problema de asignar una responsabilidad de manera que la cohesión permanezca alta (36). Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. La alta cohesión se evidencia en los controladores que poseen un conjunto de funcionalidades, existiendo estrecha relación entre algunas.
- **Creador:** resuelve el problema de asignar a una clase la responsabilidad de crear una instancia de otra clase (36). Se evidencia este patrón en las clases controladoras, pues en las mismas se crean los modelos y las vistas.
- **Controlador:** resuelve el problema de asignar una responsabilidad, de recibir o manejar una solicitud de evento del sistema a una clase (36). Este patrón se ve claramente debido a que se utiliza el *framework* Symfony, el cual está basado en el patrón MVC y las clases controladoras son las encargadas de controlar el flujo de información en el sistema.

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

### 3.2.2 Patrones GOF

- **Decorador:** es un patrón de tipo estructura, utilizado en la capa Vista para reducir los elementos de vistas comunes. El fichero layout.php es la plantilla global que utiliza el *framework* para decorar las plantillas asociadas a cada una de las acciones de la aplicación.
- **Comando:** es un patrón de comportamiento en el que un objeto se utiliza para encapsular toda la información necesaria para llevar a cabo una acción o activar un evento en un momento posterior. Esta información incluye el nombre del método, el objeto que posee el método y los valores de los parámetros del método. El patrón hace que la implementación de los métodos *Execute()* de una interfaz *Command* encapsule la invocación del método de otro objeto distinto.

**Diagrama Entidad Relación (DER)**

El DER mostrado en la Figura 12 representa la estructura de la base de datos de la propuesta de solución y la relación que existe entre los datos, el mismo fue generado de forma automática por la herramienta Visual Paradigm for UML.

A continuación se muestra una descripción de las entidades que conforma la base de datos de la aplicación y la relación que existe entre ellas:

**tb_fos_user:** es la encargada de almacenar la información común entre un usuario y un proyecto. Posee una relación de uno a muchos con la tabla tb_proyecto. Un usuario estará relacionado únicamente con un proyecto.

**tb_proyecto:** es la encargada de almacenar la información común entre un usuario y una determinada revisión. Posee una relación de uno a muchos con las tablas tb_fos_user y tb_revisión. Un proyecto estará relacionado únicamente con un usuario o con una revisión. El sistema solo permite relacionar a un revisor con un proyecto.

**tb_prorroga:** almacena la información relacionada con las prórrogas a solicitar por un revisor.

**tb_problema:** almacena la información relacionada con los problemas que encuentre el revisor durante el proceso de revisión técnica formal asignado.

**tb_referencia:** almacena la información relacionada con la referencia que se quiera realizar con un determinado documento a revisar.

**tb_observacion:** almacena la información relacionada con las observaciones planteadas por el revisor.

**tb_revision:** se utiliza para almacenar los datos de las revisiones técnicas formales a realizar por los revisores. Tiene una relación de uno a mucho con las tablas tb_proyecto, tb_problema, tb_referencia, tb_observacion y tb_prorroga.

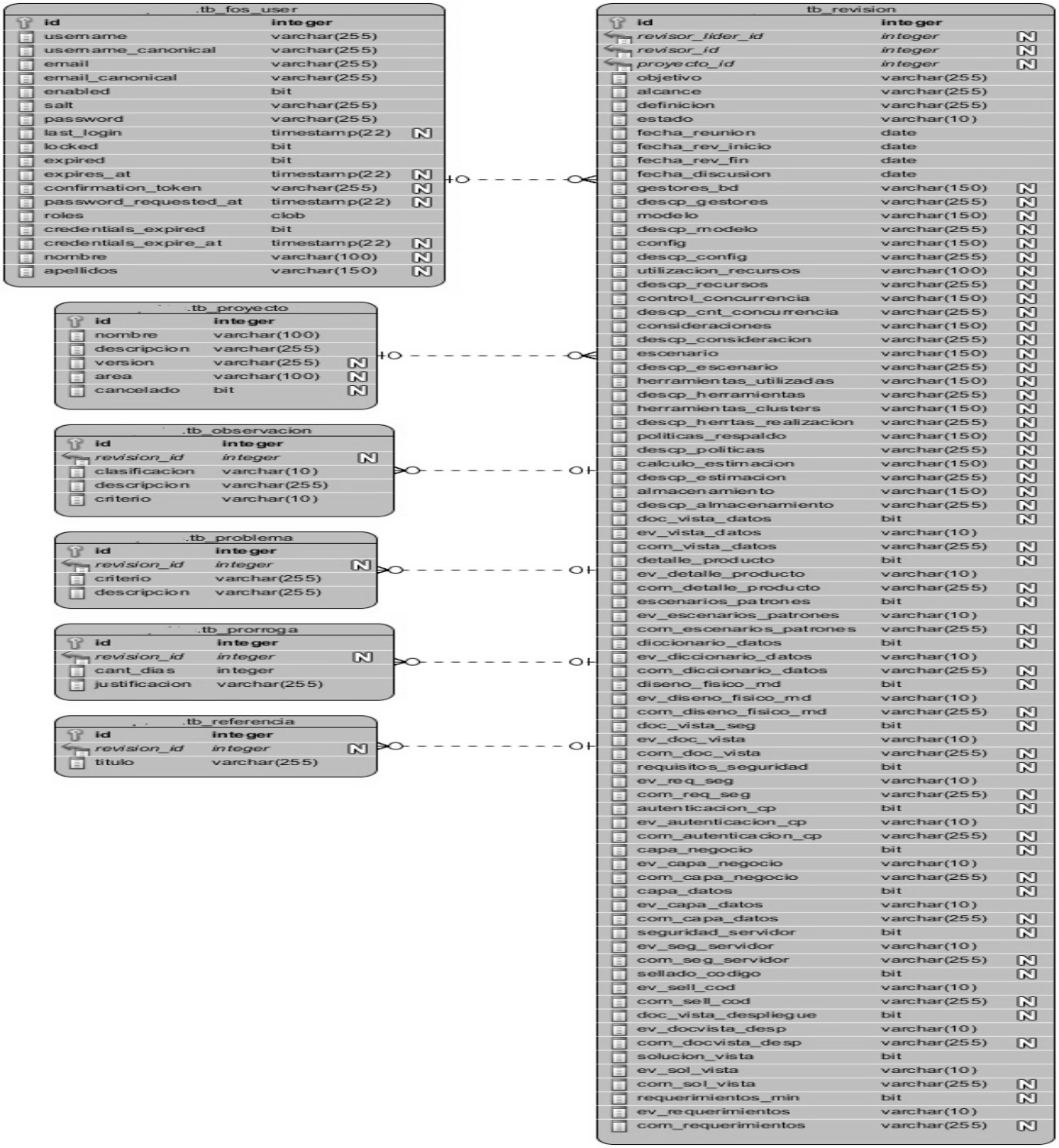


**Figura 12. Modelación del DER**

## 3.3 Fase de Implementación

En esta fase se realizan las codificaciones de las HU por iteración, dado que las descripciones de las HU fueron muy breves en la etapa de planificación, además se requiere tener la completa disponibilidad del cliente para discutir los detalles de cada funcionalidad con el equipo de desarrollo. Se analiza el plan de iteraciones y se modifica en caso de ser necesario. Se realiza la estandarización del código para facilitar el análisis y la modificación del mismo por parte de cualquier miembro del equipo. Para la implementación de cada iteración propuesta en la planificación se descomponen las HU en tareas de tipo investigativa y de desarrollo, las mismas asociadas a un equipo de trabajo o a una persona. Al concluir cada iteración se obtiene una versión del producto que será entregado al cliente.

Siguiendo la planificación realizada anteriormente, se llevaron a cabo cuatro iteraciones de desarrollo sobre el sistema, obteniéndose un producto con todos los requisitos cumplidos.

**Tareas de ingeniería**

Las tareas de ingeniería parten de la división de cada HU con el objetivo de facilitar la implementación de las mismas, la duración de estas debe de estar comprendida entre uno y cinco días aproximadamente. Se muestran algunas de las tareas elaboradas por cada HU, las restantes se pueden encontrar en el **Anexo 4.**

**Tabla 16.Descripción de la tarea**

| **Tarea** | |
|---|---|
| **Número:** número de la tarea. | **Número de HU:** número de la historia de usuario. |
| **Nombre:** nombre de la tarea. | |
| **Tipo de tarea:** clasificación de la tarea (Desarrollo o Investigación). | **Estimación:** tiempo de esfuerzo de la tarea en días. |
| **Fecha inicio:** fecha de inicio de la | **Fecha fin:** fecha de concluida la tarea. |

| | |
|---|---|
| tarea. | |
| **Programador responsable:** nombre de (el o los) programadores responsables de cada tarea. | |
| **Descripción:** breve descripción de lo que debe realizar la tarea. | |

**Tabla 27.Autenticar usuario**

| **Tarea** | |
|---|---|
| **Número:** 1 | **Número de HU:** 1 |
| **Nombre:** Autenticar usuario | |
| **Tipo de tarea:** Desarrollo | **Estimación:** 3 días |
| **Fecha inicio:** 9 de febrero de 2015 | **Fecha fin:** 11 de febrero de 2015 |
| **Programador responsable:** Keimis Figueras Lores, Ernest Alvarez Calderón | |
| **Descripción:** Se autenticará un usuario en el sistema. | |

**Tabla 18.Crear revisión técnica**

| **Tarea** | |
|---|---|
| **Número:** 10 | **Número de HU:** 5 |
| **Nombre:** Crear revisión técnica | |
| **Tipo de tarea:** Desarrollo | **Estimación:** 1 día |

| | |
|---|---|
| **Fecha inicio:** 19 de marzo de 2015 | **Fecha fin:** 20 de marzo de 2015 |
| **Programador responsable:** Keimis Figueras Lores, Ernest Alvarez Calderón | |
| **Descripción:** creará una revisión técnica en el sistema. | |

**Tabla 19.Crear criterio**

| **Tarea** | |
|---|---|
| **Número:** 14 | **Número de HU:** 6 |
| **Nombre:** Crear criterio | |
| **Tipo de tarea:** Desarrollo | **Estimación:** 1 días |
| **Fecha inicio:** 24 de marzo de 2015 | **Fecha fin:** 24 de marzo de 2015 |
| **Programador responsable:** Keimis Figueras Lores, Ernest Alvarez Calderón | |
| **Descripción:** Añadir un nuevo criterio a la lista. | |

**Tabla 20.Generar dictamen técnico**

| **Tarea** | |
|---|---|
| **Número:** 18 | **Número de HU:** 8 |
| **Nombre:** Generar dictamen técnico | |
| **Tipo de tarea:** Desarrollo | **Estimación:** 5 días |
| **Fecha inicio:** 6 de abril de 2015 | **Fecha fin:** 10 de abril de 2015 |

| |
|---|
| **Programador responsable:** Keimis Figueras Lores, Ernest Alvarez Calderón |
| **Descripción:** Genera un informe técnico. |

**Tabla 21.Solicitar prórroga**

| **Tarea** | |
|---|---|
| **Número:** 22 | **Número de HU:** 12 |
| **Nombre:** Solicitar prórroga | |
| **Tipo de tarea:** Desarrollo | **Estimación:** 5 días |
| **Fecha inicio:** 4 de mayo de 2015 | **Fecha fin:** 8 de mayo de 2015 |
| **Programador responsable:** Keimis Figueras Lores, Ernest Alvarez Calderón | |
| **Descripción:** Solicitar un tiempo de revisión en caso de situación inesperada. | |

**Tabla 22.Reporte**

| **Tarea** | |
|---|---|
| **Número:** 23 | **Número de HU:** 13 |
| **Nombre:** Reporte | |
| **Tipo de tarea:** Desarrollo | **Estimación:** 5 días |
| **Fecha inicio:** 11 de mayo de 2015 | **Fecha fin:** 15 de mayo de 2015 |
| **Programador responsable:** Keimis Figueras Lores, Ernest Alvarez Calderón | |

**Descripción:** Genera un reporte sobre las RTF revisadas, las observaciones y el estado de revisiones hechas.

**Estrategia de codificación. Estándares y estilos**

La metodología XP promueve la programación basada en estándares, de manera que sea entendible por todo el equipo, y que facilite la recodificación.

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física, para facilitar la lectura, comprensión y mantenimiento del código (37).

Symfony sigue los estándares definidos en los documentos del PSR-0, PSR-1 y el PSR-2 (37). Resumidos en:

- Añadir un solo espacio después de cada delimitador de coma.
- Añadir un único espacio alrededor de los operadores binarios (==, &&,...), con la excepción de la concatenación operador (.).
- Colocar los operadores unarios, adyacente a la variable afectada (,, ...!).
- Añadir una coma después de cada elemento de matriz en una matriz de varias líneas, incluso después de la última.
- Añadir una línea en blanco antes de las declaraciones de retorno, a menos que el retorno es solo dentro de un grupo de declaración (como una sentencia if).
- Definir una clase por archivo: esto no se aplica a las clases de ayuda privadas que no están destinados a ser una instancia desde el exterior y por lo tanto no están afectados por la norma PSR-0.
- Declarar propiedades de clase antes de los métodos.
- Declarar métodos públicos, luego protegidos y finalmente privados. Las excepciones a esta regla son el constructor de la clase y los métodos de montaje y desmontaje de las pruebas PHPunit, que siempre deben ser los primeros métodos para aumentar la legibilidad.
- Utilizar paréntesis para instanciar clases sin importar el número de argumentos que el constructor tenga.

### 3.4 Fase de Pruebas

La metodología XP propone la realización de pruebas a lo largo de todo el desarrollo del *software*, para así lograr un producto con calidad, reduciendo el número de errores. La metodología exige la ejecución de las pruebas unitarias encargadas de verificar el código y pruebas de aceptación que están orientadas a probar que las funcionalidades desarrolladas, sean las esperadas por el cliente.

#### 3.4.1 Pruebas Unitarias

Las pruebas unitarias en XP son realizadas a todos los módulos, pues se deben de efectuar antes de ser liberados o publicados. Las mismas deben de ser definidas antes de realizar el código. La propiedad colectiva del código funciona si todo código liberado pasa correctamente las pruebas, es decir, si el resultado es satisfactorio. Después se realiza la refactorización que consiste en limpiar y organizar el código, adaptarlo a los patrones y aumentar su legibilidad, sin modificar su comportamiento externo.

El desarrollo dirigido por pruebas (TDD) es una característica de la metodología XP para comprobar el funcionamiento de los códigos que se van implementando. Se centra en tres objetivos fundamentales:

- La implementación de las funciones que el cliente necesita.
- La reducción del número de defectos que tenga el *software* durante su desarrollo.
- La producción de *software* modular, altamente reutilizable y preparado para el cambio.

Para realizar las pruebas unitarias o de caja blanca a la aplicación se utilizó la librería PHPUnit que integra Symfony. Las pruebas fueron ejecutadas a través de un comando (*$ phpunit -c app/*) durante todo el proceso de implementación, lo que permitió que los errores detectados fueran corregidos a medida que avanzaba el desarrollo de la solución propuesta. En la Figura 13 se muestra un ejemplo de uno de los resultados obtenidos con la ejecución del *framework*.

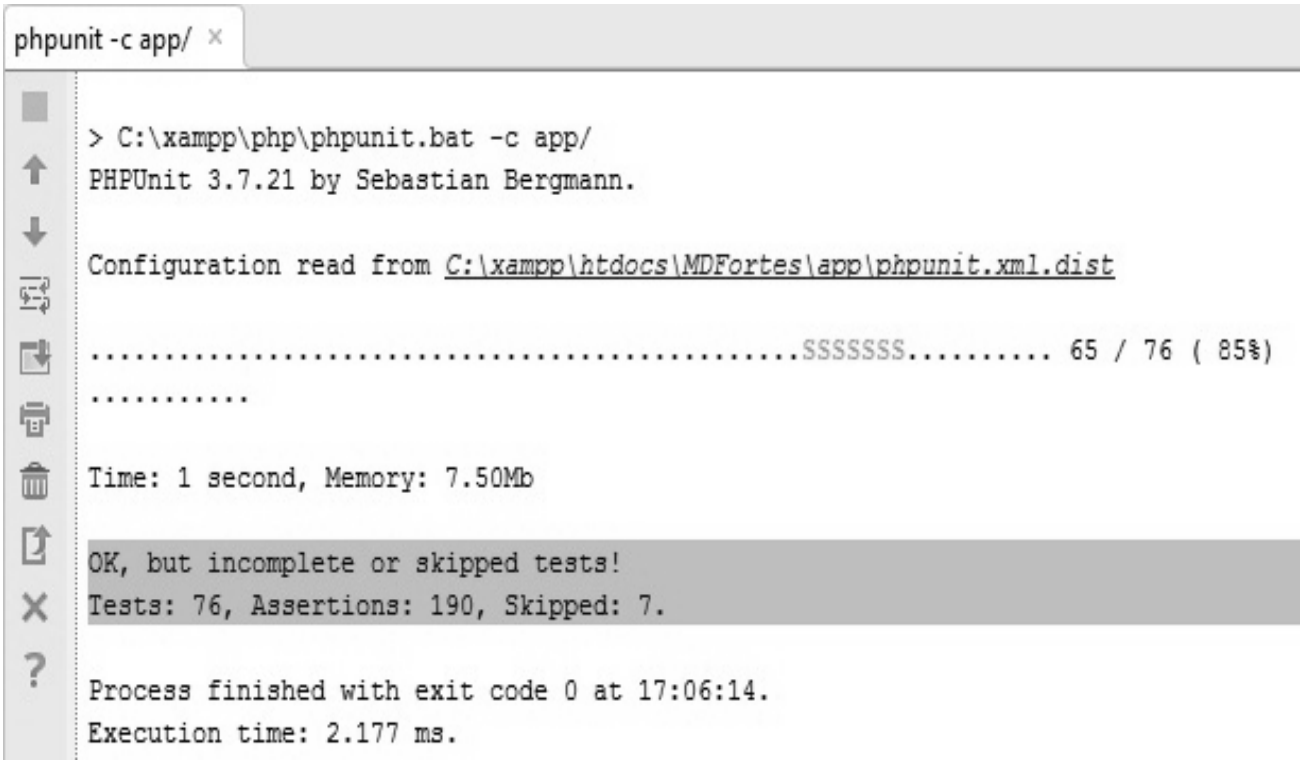


**Figura 13. Resultado del trabajo con PHPUnit**

Los errores más comunes detectados principalmente fueron en la gestión de datos, específicamente en la incompatibilidad de los tipos datos entrados por parámetros y los que se esperaba que devolviera el método.

### 3.4.2 Pruebas de Aceptación

Las pruebas de aceptación son ejecutadas por el cliente para comprobar que las HU creadas han sido correctamente implementadas. Estas son consideradas como pruebas de caja negra. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información. Se describen mediante una tabla llamada Caso de Prueba de Aceptación.

A continuación se muestran 5 casos de pruebas de aceptación realizadas a las funcionalidades del *software*. Los restantes se pueden encontrar en el **Anexo 5**.

**Tabla 23.Descripción del caso de prueba**

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** compuesto por el número del caso de prueba seguido por un guion bajo y el número de historia de usuario. | **Historia de usuario:** nombre de la funcionalidad. |
| **Descripción:** descripción de lo que debe realizar la funcionalidad. | |
| **Condición de ejecución:** condición para que se realice la funcionalidad. | |
| **Entrada/Pasos de ejecución:** secuencia de pasos que se deben de tener en cuenta para llegar a la funcionalidad. | |
| **Resultado esperado:** resultado que se espera de la funcionalidad. | |

**Tabla 24.CPA1 Autenticar usuario**

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** CPA1_HU1 | **Historia de usuario:** Autenticar usuario |
| **Descripción:** Prueba para la funcionalidad que permite autenticar un usuario. | |
| **Condición de ejecución:** nombre de usuario y contraseña correcta | |
| **Entrada/Pasos de ejecución:**<br>• El usuario introduce su nombre de usuario y contraseña. Da clic en la opción “Entrar”.<br>• El usuario introduce su nombre de usuario incorrecto y contraseña. Da clic en la opción “Entrar”.<br>• El usuario introduce su nombre de usuario y contraseña incorrecta. Da clic en la opción “Entrar”. | |

| |
|---|
| **Resultado esperado:**<br>• Se autentica al usuario permitiendo que entre a su perfil.<br>• Se muestra un mensaje de usuario y contraseña incorrecta en caso de que algunos de estos datos sean inválidos. |

**Tabla 25.CPA8 Revisión activa**

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** CPA8_HU4 | **Historia de usuario:** Revisión activa |
| **Descripción:** Prueba para la funcionalidad que permite listar las revisiones activas. | |
| **Condición de ejecución:**<br>• El usuario debe estar autenticado con el rol Revisor. | |
| **Entrada/Pasos de ejecución:** para realizar esta funcionalidad, ir al menú en la parte superior izquierda en Revisiones/ Activas se muestra una vista con un listado de las revisiones activas. | |
| **Resultado esperado:** Se muestra un listado de las revisiones activas en el sistema. | |

**Tabla 26.CPA9 Revisión abortada**

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** CPA9_HU5 | **Historia de usuario:** Revisión abortada |
| **Descripción:** Prueba para la funcionalidad que permite listar las revisiones abortadas. | |
| **Condición de ejecución:**<br>El usuario debe estar autenticado con el rol Revisor. | |

| |
|---|
| **Entrada/Pasos de ejecución:** para realizar esta funcionalidad, ir al menú en la parte superior izquierda en Revisiones/ Abortadas se muestra una vista con un listado de las revisiones abortadas. |
| **Resultado esperado:** Se muestra un listado de revisiones abortadas en el sistema. |

**Tabla 27.CPA10 Crear revisión técnica**

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** CPA10_HU6 | **Historia de usuario:** crear revisión técnica |
| **Descripción:** Prueba para la funcionalidad que permite crear una revisión técnica | |
| **Condición de ejecución:** El usuario debe estar autenticado con el rol Revisor. | |
| **Entrada/Pasos de ejecución:**<br>✓ Para crear una revisión, ir al proyecto creado que aparece en el listado de la vista Proyecto. Luego dar clic en la opción mostrar que aparece al final de la fila del proyecto a crear la revisión. Una vez dado el clic aparece una vista con los datos del proyecto y los botones “Editar”, “Eliminar” y “Nueva Revisión”. Al dar clic en el botón “Nueva Revisión” aparece la vista Nueva RTF para realizar la revisión que incluye un menú con tres formularios a llenar encabezado, Criterios generales y específicos. Al dar clic en la opción del menú “Encabezado” y rellenar los campos dar clic en el botón “Crear”. Si desea cancelar la operación dar clic en el botón “Cancelar”.<br>✓ Para crear una revisión, ir al menú que se encuentra en la parte superior a la izquierda en Vista/Archivo Dictamen, donde aparece un listado con los dictámenes hechos y los botones “Editar”, “Eliminar” y “Nueva Revisión”. Al dar clic en el botón “Nueva Revisión” aparece la vista Nueva RTF para realizar la revisión que incluye un menú con tres formularios a llenar encabezado, Criterios generales y específicos. Al dar clic en la opción del menú “Encabezado” y rellenar los campos dar clic en el botón “Crear”. Si | |

| |
|---|
| desea cancelar la operación dar clic en el botón “Cancelar”. |
| **Resultado esperado:** Se crea la revisión en el sistema. |

**Tabla 28.CPA17 Generar DT**

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** CPA17_HU8 | **Historia de usuario:** Generar DT |
| **Descripción:** prueba para la funcionalidad que permite generar un dictamen. | |
| **Condición de ejecución:** solo puede ejecutar esta funcionalidad el usuario con rol Revisor. | |
| **Entrada/Pasos de ejecución:**<br>✓ Para generar un dictamen dirigirse al menú en la parte superior izquierda, la opción Revisiones/ Registro RTF, donde aparece la vista RTF Modelo de Datos con un listado de las revisiones realizadas. Al seleccionar la opción mostrar que se encuentra al final de la fila de cada RTF, se muestra la vista con la RTF seleccionada y los botones “Editar”, “Finalizar” y “Eliminar”. Al dar clic en el botón “Finalizar” automáticamente se genera el dictamen. | |
| **Resultado esperado:** Se genera un dictamen. | |

**Resultados generales de las pruebas de aceptación**

Luego de ejecutadas todas las pruebas de aceptación, se detectaron un conjunto de no conformidades (NC) que afectaban el funcionamiento y apariencia del sistema. En la primera iteración se detectaron 5 NC cuatro fueron clasificadas como no significativas, ya que constituían faltas de ortografía y una significativa la cual constituyó el nombre no deseado por el cliente de un botón. En la segunda y tercera iteración todas las NC encontradas se clasificaron de no significativa. En la cuarta iteración se detectó una no conformidad significativa en la interfaz gráfica, las restantes constituyeron no significativas. A continuación

se muestra en la Figura 14 un gráfico con la información correspondiente a las cuatro iteraciones de prueba realizadas.

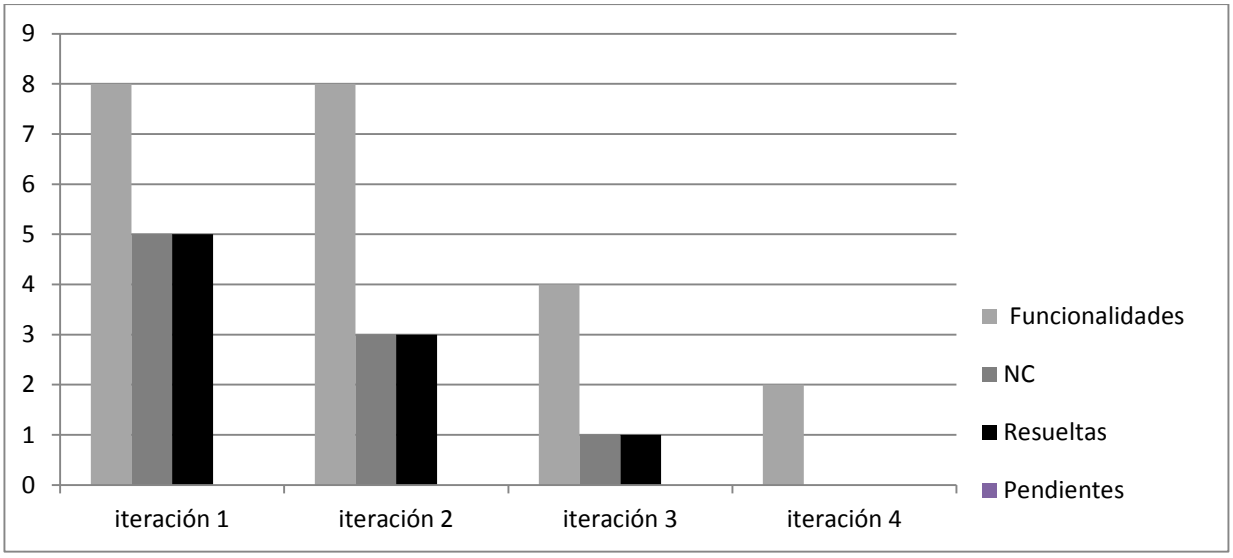


**Figura 14.Resultado de las pruebas de aceptación**

Todas las no conformidades fueron resueltas lo cual valida en cierto grado la calidad de la propuesta de solución, logrando la satisfacción por parte del cliente.

**Conclusión**

En el desarrollo del capítulo se describe la arquitectura a utilizar escogiendo como patrón arquitectónico el MVC implementado por Symfony. Se generó uno de los artefactos relevantes de la propuesta de solución, el DER para llevar a cabo la implementación de la aplicación. Para hacer menos complejo el diseño se tuvieron en cuenta los patrones de diseño GRAPS y GOF. Además se presentan los estilos y estándares más significativos, utilizados durante el proceso de codificación. Se conforman las tareas de ingenierías necesarias para desarrollar cada HU. Se ejecutan las pruebas necesarias para que el funcionamiento de la solución propuesta tenga menor probabilidad de fallo. Se detectan las NC y se les da solución a las mismas entregándole al cliente el *software* deseado.

## Conclusiones

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

- La presente investigación se centró en el análisis y desarrollo de una aplicación web para gestionar las Revisiones Técnicas Formales en el área Modelo de datos.
- La construcción del marco teórico que guio el desarrollo de la aplicación fue sustentada por los métodos científicos y las técnicas de recopilación de datos empleadas.
- El estudio del estado del arte a sistemas que gestionan las RTF permitió comprobar que existen características propias que diferencian al área Modelo de datos de otras áreas. También se realizó un estudio de las diferentes metodologías, herramientas y tecnologías, escogiendo la metodología XP como guía para el proceso de desarrollo.
- La metodología seleccionada permitió satisfacer las necesidades del cliente haciendo uso de las HU definidas y codificadas. También se realizó un diseño acorde a las especificaciones del cliente.
- Se obtuvieron índices aceptables para cada prueba realizada, por lo que se considera un resultado general positivo, cumpliendo con un nivel alto de aceptación por parte del cliente.

## Recomendaciones

Tomando como base la investigación realizada y el análisis de los resultados obtenidos se recomienda:

- La integración con los módulos Requisitos, Código y Arquitectura. Además con el eXcriba y el Gespro.
- La utilización de este servicio por otros Centros productivos de la universidad.

## Referencias bibliográficas


1. Yinimary Ortega Montoya y Isis Margarita Blanco Zamora. “Estrategia de Control de la Calidad mediante revisiones y auditorías para el proyecto CICPC. S.l.: s.n”, 2007.
2. Manuel F. Bertoa, José M. Troya y Antonio Vallecillo. Aspectos de Calidad en el Desarrollo de Software Basado en Componentes.
3. Calidad de Software. [En línea]. Disponible en: http://www.ub.edu.ar/catedras/ingenieria/ing_software/ubftecwwwdfd/calidadsw/calidad.htm.
4. Indira Díaz Pérez. *Procedimiento de Revisiones de Software para los Proyectos de FORTES*. S.l.: s.n., 2012.
5. Margaret Rouse. ¿Qué es Análisis de datos? - Definición en WhatIs.com. [En línea]. noviembre 2012.
6. María G. Rosa-Rosario. *Modelos de bases de datos*. S.l.
7. Juan M Fernández Peña, Lizbet A Hernández González. Herramienta para la Automatización de Revisiones Técnicas Formales como apoyo al desarrollo orientado a objetos. RETO. [En línea]. Disponible en: http://www.uv.mx/mis/files/2012/11/HernandezFernandez.pdf.
8. Informer Technologies, [En línea].Disponible en: http://rational-requisitepro.software.informer.com/7.1/
9. Luz María Gutiérrez Feria, 2014, *Procedimiento de Revisiones Técnicas Formales*. 2014.
10. Elena González Revilla. *Procedimiento de Revisión Técnica Formal en el Grupo de Calidad de FORTES de la facultad 4*. Universidad de Ciencias Informáticas, 2014.
11. Pedro Pablo Rosález López, Ing. Oscar Tinoco Gómez and Ing. Julio Salas Bacalla. Industrial Data - <B>Selection criteria of methodologies of software development</B>.

12. Patricio Letelier, Ma Carmen Penadés. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea]. 2006.

13. Christian Van Der Henst, 2001, ¿Qué es el PHP? [En línea]. 23 Mayo 2001. Disponible en: http://www.maestrosdelweb.com/phpintro/

14. Rubén Alvarez. Introducción a la programación en PHP.

15. César Krall. "Tutorial básico del programador web: JavaScript desde cero" (CU01102E). [En línea]. 2006. Disponible en: http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=776:orientacion-sobre-el-curso-qtutorial-basico-del-programador-web-javascript-desde-ceroq-cu01102e&catid=78:tutorial-basico-programador-web-javascript-desde-&Itemid=206

16. Contributors, Mozilla Developer Network and individual. Introducción al HTML - Guía de Desarrollo Web | MDN. [En línea]. 2005 2015. Disponible en: https://developer.mozilla.org/es/docs/Web/Guide/HTML/Introduction_alhtml

17. Diego Barcia. ¿Qué es CSS? In: *Maestros del Web* [En línea]. 8 noviembre 2003. Disponible en: http://www.maestrosdelweb.com/introcss/.

18. Guía Breve de CSS. [En línea]. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo

19. Javier J. Gutiérrez., 2015, *1* ¿Qué es un framework web? [En línea]. 2015. Disponible en: http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf

20. Luis Chavez. 10 razones para utilizar Symfony | Super Tecla. [En línea]. 29 noviembre 2010. Disponible en: http://www.supertecla.com.ar/2010/11/10-razones-para-utilizar-symfony/.

21. Beebom. 15 Best PHP Frameworks of 2015. [En línea]. Copyright 2015 2011. Disponible en: http://beebom.com/2015/02/best-free-php-frameworks.

22. Patricio Salinas Caro, Nancy Histchfeld K. Tutorial de UML. [En línea]. Disponible en: http://users.dcc.uchile.cl/~psalinas/uml/introduccion.html.

23. Copyright SOFT112. Visual Paradigm for UML Standard 8.0 - Free Download. [En línea] 15 Febrero 2014.

24. Rafael Martinez. Sobre PostgreSQL | www.postgresql.org.es. [En línea]. Disponible en: http://www.postgresql.org.es/sobre_postgresql.

25. Guía Ubuntu. PgAdmin III - Guía Ubuntu. [En línea]. otoño 2008.

26. WordPress, 2013, ENTORNO DE DESARROLLO INTEGRADO (IDE). [En línea]. 25 de enero 2013 Disponible en: https://alanss18.wordpress.com/entorno-de-desarrollo-integradoide/

27. Oracle Corporation. Información NetBeans IDE 6.1. [En línea]. 2015.

28. María Alejandra Giménez. NETBEANS accesible: Información Básica sobre NetBeans. [En línea]. Disponible en: http://netbeansaccesible.blogspot.com/2012/07/caracteristicas-la-plataforma-netbeans.html

29. Pikuro.com. PhpStorm Editor de PHP. [En línea]. Copyright 2012-2014. Disponible en: http://www.pikuru.com/index.php?a=prog&os=windows&cat=desarrolladores&subcat=herramientas-de-programaci%C3%B3n&det=phpstorm-editor-de-php

30. Guillermo González-Vallés Saco. Una Introducción a Apache. [En línea]. 2014.

31. Novell, Inc. and others. Apache - openSUSE. [En línea]. 2011.

32. LibrosWeb.es. El framework de pruebas PHPUnit. [En línea]. Copyright 2015. Disponible en: http://librosweb.es/libro/symfony_2_x/capitulo_10/el_framework_de_pruebas_phpunit.html.

33. Javier Eguiluz. *Desarrollo webágil con Symfony2*. Copyright 2014.

34. Desarrolloweb.com. Manual de jQuery. [En línea]. Copyright 2015. Disponible en: http://www.desarrolloweb.com/manuales/manual-jquery.html

35. Juan David Rodríguez García. Profundizando en la arquitectura MVC de Symfony. [En línea]. Copyright 2012. Disponible en: http://juandarodriguez.es/cursosf14/unidad7.html.

36. Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos. [En línea]. Prentice Hall, México, 1999. ISBN 970-17-0261-1. Disponible en: https://docs.google.com/file/d/0B_MWRyBDHOk-UDhZUzNFU2V4OWs/edit?pli=1.

37. Fabien Potencier. Coding Standards. [En línea]. 2015. Disponible en: http://symfony.com/doc/current/contributing/code/standards.html.