# Universidad de las Ciencias Informáticas


# Título: Aplicación web para la gestión de las necesidades de innovación tecnológica en la Universidad de las Ciencias Informáticas

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

**Autor:** Danny Morejón Macías

**Tutores:** Lic. Raynel Batista Téllez

Ing. Leidy Ramos González

La Habana, Cuba
Junio, 2015
"Año 57 de la Revolución"

# Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

**Danny Morejón Macías**

_____

**Autor**

**Lic. Raynel Batista Téllez**

_____

**Tutor**

**Ing. Leidy Ramos González**

_____

**Tutor**

# Agradecimientos


## *Agradezco:*

*A mis padres Beatriz y Abel por permitirme tomar mis propias decisiones, por su confianza, su sacrificio, por guiarme y apoyarme siempre.*

*A mis hermanos Lisandra y Daxiel por ser el motivo de mis esfuerzos para logar ser un buen ejemplo para ustedes.*

*A mi novia Maybelis por estar a mi lado, apoyarme y hacerme feliz cada día.*

*A todos mis amigos, que de una forma u otra han compartido conmigo estos años de estudio.*

*A mis tutores, por su guía durante la realización de este trabajo y por confiar en mí durante todo este tiempo.*

*A los profesores que a lo largo de mi vida como estudiante han hecho de mi un profesional y sobre todo una mejor persona.*

*A todos, muchas gracias…*


## *Dedicatoria:*

*A las cuatro personas más importantes de mi vida: mi mamá, mi papá, mi hermano y mi hermanita.*

*…muchas gracias.*

# Resumen

**RESUMEN**

En la Universidad de las Ciencias Informáticas, a pesar de poseer una red telemática avanzada, vincular a miles de especialistas, académicos e investigadores en más de 200 proyectos I+D y aportar el 76% de las soluciones implantadas en el país y el 99% de las soluciones informáticas exportadas por la nación, no ha logrado aprovechar al máximo las ventajas de las Tecnologías de la Información y las Comunicaciones para promover la comunicación, el intercambio y la colaboración científica para elevar la excelencia de sus aportes. En tal sentido, los especialistas, académicos e investigadores que participan en los grupos, proyectos y resultados de investigación se desarrollan de forma aislada, comprometiendo la solución de las necesidades por las que existen. Sin la requerida sinergia los resultados pudieran estar limitados en cuanto a utilidad, novedad, impacto, visibilidad y competitividad.

El presente trabajo tiene como objetivo el desarrollo de una aplicación web basada en tecnologías de código abierto para la gestión de las necesidades de investigación en la red de investigadores de la Universidad de las Ciencias Informáticas, la cual tiene como particularidad que se ajusta sobre todo a los intereses y estrategias de trabajo de la Dirección de Investigaciones de la Universidad.

El sistema permite realizar una adecuada gestión de las distintas problemáticas originadas en la Universidad de las Ciencias Informáticas. Para su desarrollo fueron utilizadas varias tecnologías actuales como el lenguaje de programación PHP, el marco de trabajo Symfony2, el cual reutiliza las siguientes tecnologías libres: Twig y Doctrine2, además se utilizó como servidor de base de datos PostgresSQL y como servidor de aplicaciones web Apache.

El desarrollo del sistema logró realizar de forma confiable y segura el proceso de gestión de las distintas problemáticas que afectan a la universidad, posibilitando un seguimiento y control a las mismas por parte de varios especialistas, desde el momento de su creación, hasta su solución.

La aplicación web fue validada por las pruebas unitarias y de aceptación.

**Palabras claves:** gestión de información, innovación tecnológica, métricas, problemáticas

# Índice

## Tabla de contenidos

# Índice

# Introducción

## INTRODUCCIÓN

Desde los inicios de la civilización humana, el hombre siempre ha tenido la necesidad de investigar para explicar los fenómenos que ocurrían a su alrededor. Estas investigaciones propiciaron la creación, búsqueda y obtención de conocimientos. Los estudios de Demócrito, Hipócrates, Aristóteles y sus seguidores sentaron las bases del conocimiento actual y fueron precursores de lo que hoy entendemos cómo ciencia (Fowler, 2009). Sin embargo, el auge de la ciencia en la antigüedad no puede sino identificarse después de la creación de un centro de investigación científica, el *Museum*, que fue el centro de reunión para todos los sabios del mundo griego (Historia de la ciencia, 2014), demostrando la importancia de la **comunicación**, el **intercambio** y la **colaboración** para obtener nuevos conocimientos: un método de socialización que su demostrada efectividad ha trascendido siglos de evolución de la sociedad humana.

El surgimiento y constante desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) ha impulsado el intercambio de información y conocimiento. Las TIC han logrado satisfacer las necesidades de la vida cotidiana de prácticamente todos los sectores sociales en la mayoría de los países del mundo, formando parte de la cultura tecnológica con la que se debe convivir (Graells, 2008). Las TIC, en ese sentido, han creado un nuevo espacio para la colaboración entre investigadores, superando incluso las barreras del idioma y la geografía. La tendencia a la colaboración regional se ha notado en el análisis de la cohesión de la red de colaboración científica de las TIC (J. M. Russell, y otros, 2007), que funciona efectivamente como un espacio de intercambio de conocimiento con un nivel de interacción interno mayor que el promedio general de la red de co-publicación científica mundial (Barrere, y otros, 2012).

Las redes de colaboración científica han potenciado el pensamiento crítico y la creatividad a través del proceso de aprendizaje para la generación de nuevos conocimientos. La sistematización de estos propósitos ha devenido en la creación de instituciones académicas y científicas que persiguen nuevos hallazgos del conocimiento a partir de las necesidades sociales. La universidad constituye sin dudas, el centro por excelencia para crear conocimientos y formar profesionales que puedan innovar y crear los nuevos estándares de vida en nuestros países (Ramírez, 2010).

# Introducción

Cuba ha mostrado interés de aprovechar los beneficios de las TIC para elevar el impacto de su tradición científica. El proceso de informatización de la sociedad cubana impacta a todas las instituciones, especialmente las académicas y de innovación tecnológica. La creación de la Universidad de la Ciencias Informáticas (UCI), como parte de las tareas para la informatización de los servicios, demuestra tales propósitos. Sin embargo, la UCI, a pesar de poseer una red telemática avanzada, vincular a miles de especialistas, académicos e investigadores en más de 200 proyectos I+D y aportar el 76% de las soluciones implantadas en el país y el 99% de las soluciones informáticas exportadas por la nación (Dirección General de Producción, 2011), no ha logrado aprovechar al máximo las ventajas de las TIC para promover la comunicación, el intercambio y la colaboración científica para elevar la excelencia de sus aportes (UNESCO, 2010).

En tal sentido, los especialistas, académicos e investigadores que participan en los grupos, proyectos y resultados de investigación se desarrollan de forma aislada, comprometiendo la solución de las necesidades por las que existen. Por lo que, sin la requerida sinergia los resultados pudieran estar limitados en cuanto a utilidad, novedad, impacto, visibilidad y competitividad.

Teniendo en cuenta ese contexto cabría entonces afirmar como **problema de investigación:** el modo en que se gestionan las necesidades de innovación tecnológica en la UCI limita el trabajo colaborativo entre especialistas, académicos e investigadores.

Se define como **objeto de estudio**: aplicaciones web para sistemas de gestión de información científico-técnica. Enmarcado en el **campo de acción**: aplicaciones web para la gestión de necesidades de innovación tecnológica en la UCI.

Para dar solución al problema científico planteado se define como **objetivo general** de la investigación**:** desarrollar una aplicación web para la gestión de las necesidades de innovación tecnológica que eleve el trabajo colaborativo entre especialistas, académicos e investigadores.

**Objetivos específicos**

Para complementar el objetivo general se trazaron los siguientes objetivos específicos:

1. Analizar herramientas para la gestión de las necesidades de investigación en el contexto universitario que permita comparar sus características e identificar sus potencialidades.
2. Identificar tendencias y enfoques en aplicaciones web para la gestión de las necesidades de investigación que permitan establecer patrones de diseño en las herramientas estudiadas.
3. Realizar el análisis y diseño la propuesta de solución.
4. Implementar una aplicación web que a partir de enfoques novedosos permita elevar el trabajo colaborativo en la red de investigadores de la UCI.
5. Validar la propuesta de solución a partir de pruebas de software.

**Idea a defender**

El desarrollo de una aplicación web para la gestión de las necesidades de innovación tecnológica elevará el trabajo colaborativo entre especialistas, académicos e investigadores.

La investigación será guiada por los siguientes **métodos de investigación**:

**Histórico-Lógico:** este método se utilizó para analizar los antecedentes, evolución y desarrollo de la colaboración científica a partir de sistemas de gestión de información hasta la actualidad, así como los principales conceptos asociados a estos.

**Analítico-Sintético:** este método permitió el análisis de los documentos, procesos y teorías para la extracción de los elementos más importantes que se relacionan con los sistemas de gestión de información que promueven la colaboración científica. Además, para buscar y analizar la información acerca de las tecnologías, metodologías y herramientas a utilizar en el desarrollo del sistema, seleccionando los principales elementos y características.

**Hipotético-Deductivo:** para la realizar generalización de la teoría estudiada y llegar a conclusiones específicas.

**Modelación:** permitió la creación de modelos que representan abstracciones de la realidad y de la cual se obtienen los artefactos con el objetivo de obtener una representación visual del sistema que se va a desarrollar.

**Métodos empíricos:**

**Entrevista:** se emplea este método con el objetivo de obtener información sobre la gestión de las necesidades científicas de la red de investigadores de la universidad, intercambiando directamente con las personas involucradas. Además permite reunir y determinar la información básica necesaria para la captura de requisitos del sistema.

# Estructura del documento

**ESTRUCTURA DEL DOCUMENTO**

El presente trabajo de diploma se encuentra estructurado de la siguiente forma: introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos.

**Capítulo 1: Fundamentación teórica**

El capítulo contiene una base teórica para entender el problema a solucionar así como algunos conceptos fundamentales, incluyéndose el estudio de otras soluciones informáticas existentes en la actualidad. Se realiza una selección de la metodología, lenguajes de programación, servidor de base de datos, marco de trabajo y herramientas a utilizar en el desarrollo de la solución. Por último se estudian las pruebas que serán aplicadas una vez terminado el trabajo.

**Capítulo 2: Análisis y diseño del sistema**

El capítulo describe la propuesta de solución del sistema a desarrollar y los procesos que serán informatizados con el desarrollo de esta aplicación. También se presentan los requisitos funcionales y no funcionales con los que debe cumplir el sistema propuesto. Se presentan las técnicas empleadas en la captura y validación de los requisitos, así como los patrones utilizados en el diseño de la aplicación. Además, muestra el modelo de datos del sistema y los artefactos del diseño de los procesos mencionados anteriormente.

**Capítulo 3: Implementación y prueba**

El capítulo muestra el desarrollo de la solución al problema planteado con anterioridad, cumpliendo con los requisitos especificados para el mismo. Se definen los prototipos de interfaces y los artefactos generados en la implementación. Además se muestran los resultados de las pruebas realizadas al software para comprobar su correcto funcionamiento.

## CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

### Introducción

El estudio y análisis de las principales aplicaciones que se encargan de brindar un ambiente propicio para la colaboración y desarrollo del trabajo científico e investigativo en el mundo, en Cuba y en la universidad posibilita entender cómo han evolucionado estos sistemas y cuáles son las tendencias actuales en el área.

El capítulo comprende un análisis de los conceptos y sistemas relacionados con los sistemas de gestión de información científico-técnica. Se definen la metodología, arquitectura de software, las herramientas y tecnologías adecuadas teniendo en cuenta el marco tecnológico establecido para la integración con otras aplicaciones pertenecientes a la Dirección de Investigaciones de la Universidad. (UCI, 2014).

### 1.1 Sistema de gestión de la información

La gestión de la información es un proceso que incluye operaciones como extracción, manipulación, tratamiento, depuración, conservación, acceso y/o colaboración de la información adquirida por una organización a través de diferentes fuentes y que gestiona el acceso y los derechos de los usuarios sobre la misma (Curto, 2006).

La gestión de la información propicia la generación, apropiación, intercambio y uso de conocimientos necesarios para el incremento de la eficacia de las organizaciones. Además, utiliza la información para el apoyo a la toma de decisiones, organizándola de manera tal que se cumplan los objetivos de la organización demostrando ser una estructura probada para la gestión y prosperidad de las políticas, los procedimientos y procesos de la organización (San Juan Rosabal, y otros, 2011).

#### 1.1.1 Gestión de información y conocimiento en un ambiente de comunicación

Por lo que respecta al conocimiento explícito, es posible y conveniente almacenarlo en bases de datos, concebidas como sistemas diseñados para la gestión de información. Sin embargo, no es factible estructurar y almacenar el conocimiento tácito de igual modo, por lo que la estrategia mayormente aceptada para gestionarlo se ha basado en fomentar la

creación de redes de colaboración entre las personas que componen la organización e incluso, con personas que no laboran directamente para ella que puedan enriquecer el aprendizaje organizacional y la elaboración de un mapa del conocimiento al que todos pueda acceder (Serradell López, y otros, 2003).

El conocimiento tácito una vez localizado, será tanto más valioso cuanto mayor sea su incorporación y uso en el proceso productivo de la organización. Por ello, es una condición fundamental de la gestión del conocimiento, la socialización del conocimiento tácito que poseen los miembros de cualquier organización. Esto es posible mediante el intercambio con expertos, situados dentro y fuera de la organización, la creación de un clima de solidaridad, pertenencia y profesionalismo dentro de la organización. Una vez convertido el conocimiento tácito en explícito, debe colocarse a disposición de quienes lo requieren, diseminarlo e interiorizarlo, al tiempo que se hace corresponder con las metas y objetivos de la organización. La gestión del conocimiento no sólo enfatiza en las fuentes documentales, sino también en las no documentales (recursos humanos) y en el incremento y factibilidad de la comunicación (Mesa, 2006).

### 1.1.2 Innovación tecnológica

Aunque la innovación y su tipología han sido ampliamente estudiadas[1], dos aspectos han sido comúnmente mencionados en su definición: **novedad** y **aplicación.** De este modo una invención o idea creativa no se convierte en una innovación hasta que no se utiliza para cubrir una necesidad concreta. El objetivo principal es el de convertir esas mejoras individuales en mejoras o cambios globales para la sociedad y, para ello, es esencial que se dé difusión a la innovación (Barranco, y otros, 2001).

[1] Una revisión de la literatura de innovación, muestra una variedad de ángulos desde donde este tema ha sido estudiado. Por mencionar algunos ejemplos: SCHUMPETER (1939) estudia el proceso como un todo; TUSHMAN (1977) analiza la innovación como un proceso de información; ROGERS (1983) se concentra en la difusión como parte del proceso de innovación; COOPER (1984) lo enfoca desde la perspectiva del éxito de las estrategias de la innovación de productos; VON HIPPEL (1988) subraya la importancia de los usuarios como fuentes de innovación; VAN DE VEN (1989) investiga la dirección de la innovación; PORTER (1990) relaciona la innovación con la competividad; MUÑOZ SECA (1992) vincula la innovación con el aprendizaje y la formación.

# Capítulo 1: Fundamentación Teórica

La Dra. Nilia Victoria Escobar Yéndez define la innovación como la transformación de una idea en un producto o equipo vendible, nuevo o mejorado; en un proceso operativo en la industria o el comercio, o en una nueva metodología para la organización social. Cubre todas las etapas científicas, técnicas, comerciales y financieras, necesarias para el desarrollo y comercialización exitosa del nuevo o mejorado producto, proceso o servicio social. El acto por el cual se introduce por primera vez un cambio tecnológico en un organismo o empresa se denomina innovación. Por tanto, la **innovación tecnológica** es la que comprende los nuevos productos y procesos y los cambios significativos, desde el punto de vista tecnológico, en productos y procesos (Dra. Escobar Yéndez, 2000).

La innovación tecnológica puede definirse como el proceso de creación, desarrollo, producción, introducción, comercialización y difusión de nuevos y mejores procesos-productos y procedimientos en la sociedad (Castro Díaz-Balart, y otros, 2001).

La I+D no es más que una de estas actividades y puede ser llevada a cabo en diferentes fases del proceso de innovación, siendo utilizada no sólo como la fuente de ideas creadoras sino también para resolver los problemas que pueden surgir en cualquier fase hasta su culminación.

### 1.2 Comunicación y colaboración en el ámbito científico e investigativo

Hasta la fecha, se han desarrollado muchos productos de software para empresas, instituciones y organizaciones con el objetivo de gestionar la información de las investigaciones que se realizan en ellas. Estos sistemas propician un punto de encuentro para todos los actores que participan en el proceso investigativo y demás personas interesadas en darle seguimiento. A continuación se realiza un análisis a varios sistemas nacionales e internacionales abordados en la literatura consultada, seleccionados a partir de intereses del estudio y por su similitud con el contexto del problema enunciado (ver ***Anexo 1***).

#### 1.2.1 Sistema de Gestión de la Investigación

La universidad chilena de Talca se planteó el desarrollo del Sistema de Gestión de la Investigación (SGI) con el objetivo de apoyar las investigaciones que realizan sus

estudiantes, profesores y trabajadores por medio de un sitio web que sirva de punto de encuentro entre la oferta investigativa de la universidad y la demanda de investigación de la sociedad y las empresas, de manera tal que éstas planteen temas y problemas susceptibles para ser resueltos a través de programas y/o proyectos de investigación ( Palomo, y otros, 2007).

El sistema desarrollado gestiona los investigadores, postulaciones, evaluaciones, oficializaciones, ejecuciones, cierres y resultados de los programas de investigación.

### 1.2.2 Sistema de Gestión de Investigación en Línea

Desde el año 2004 la comunidad de investigadores de la Universidad Pedagógica Nacional de Colombia cuenta con un completo sistema de gestión administrativa y académica de las convocatorias internas para proyectos de investigación. Dicho sistema ha sido desarrollado según los requerimientos de la División de Gestión de Proyectos-Centro de Investigaciones (DGP–CIUP).

El sistema, conocido por la comunidad académica como Proceso de Gestión de la Investigación en Línea (PGIL), le permite a los docentes investigadores participar en línea en las convocatorias internas gestionadas por el CIUP.

PGIL facilita que mediante un nombre de usuario y una contraseña todos y cada uno de los investigadores participen con diferentes roles en las convocatorias; como investigadores, co-investigadores, monitores, decanos y gestores del CIUP.

Acciones a realizar con la aplicación:

- Publicación de los términos de referencia y cronograma de la convocatoria
- Inscripción de líneas y grupos de investigación
- Conformación de equipos de trabajo
- Actualización de las hojas de vida de cada investigador
- Diligencia de proyectos, solicitudes de presupuesto y cargas académicas
- Ingreso de las evaluaciones de las propuestas de investigación

Posee además un servicio de soporte en línea para atender cualquier inquietud que pueda presentarse y con un sistema de reportes para apoyar la gestión interna administrativa y académica del CIUP (DGP-CIUP, 2014).

### 1.2.3 Gestión de la investigación de la Universidad de Valencia

Para gestionar la producción científica, la Universidad de Valencia adquirió en octubre de 2003 la aplicación GREC.

GREC es una plataforma privativa para la gestión, seguimiento y evaluación de todas las actividades de la ciencia y la tecnología, es decir, un conjunto de sistemas de información cuyo principal objetivo es racionalizar la gestión y planificación de la ciencia y tecnología, aplicada a instituciones públicas y privadas con actividades de ese tipo (SiUV, 2014).

Mediante GREC, se puede obtener:

- La gestión informatizada del currículum de sus investigadores, con informes e indicadores.
- El directorio de los recursos humanos de investigación, líneas de actividad, especialidades, sectores de aplicación y producción científico-tecnológica, e informes varios sobre estos.
- El directorio de los colectivos implicados (instituciones, centros, departamentos y grupos) podrá realizar la gestión informática de convocatorias, solicitudes y actividades, planificación y distribución de recursos. Así mismo, puede identificar la competencia y excelencia, los puntos fuertes y débiles del sistema.
- Un sistema gerencial de financiación que permite una gestión estratégica de los recursos, la simulación y prospectiva de acciones de política científica, y la gestión y seguimiento de contratos programa.
- Un sistema de difusión e intercambio de informaciones, elaboración de informes de actividad y memorias, la búsqueda de especialistas.

- Un sistema de fomento de colaboración científico-tecnológica que permita el aumento de las relaciones entre el sistema público de ciencia y tecnología y el sector privado (empresas).
- Un sistema de evaluación cualitativa y cuantitativa de toda la producción científica y tecnológica, mediante indicadores, aplicación de baremos y varios procesos metodológicos.

### 1.2.4 Sistema de Información Científica de Andalucía

El Sistema de Información Científica de Andalucía (SICA) es un sistema global de Información científica certificada en tiempo real. Un sistema de ámbito regional que agrupa la producción científica y que, a medida que ésta se genera, es introducida y validada en poco tiempo, facilitando un análisis fiable y evolutivo de las políticas científicas (Cabrera, 2008).

El Sistema de Información Científica de Andalucía recoge los resultados de la investigación generada en las Universidades y centros de investigación andaluces y que pronto también considerará la actividad científica y tecnológica del resto de agentes del Sistema Andaluz del Conocimiento.

El sistema está compuesto por las siguientes secciones (SiCA, 2014):

- Investig-AN: gestor curricular de la producción y la actividad científica de los agentes del Sistema Andaluz del Conocimiento estructurado según el estándar CVN.
- Conect-AN: gestor de redes sociales basadas en el conocimiento para establecer espacios de colaboración entre los agentes del Sistema Andaluz del Conocimiento.
- Export-AN: extractor de actividad y producción científica asociada a dominios institucionales.
- Agrup-AN: gestor de grupos del Plan Andaluz de Investigación Desarrollo e Innovación (PAIDI).

### 1.2.5 Sistema de Información para la Gestión de Programas y Proyectos de Ciencia e Innovación Tecnológica

El Sistema de Información para la Gestión de Programas y Proyectos de Ciencia e Innovación Tecnológica (SIPROCIT) constituye un sistema de apoyo a la planificación, control, evaluación y proyección de los Programas y Proyectos de Ciencia e Innovación, acorde a las prioridades establecidas para un período determinado en Cuba y que contribuye incrementar el nivel de integración del Sistema de Programas y Proyectos (SPP) (Sánchez, y otros, 2008).

La aplicación cuenta con 3 secciones: catálogos, programas y proyectos e información estadística.

### 1.2.6 Sistema de Gestión de la Nueva Universidad

El Sistema de Gestión de la Nueva Universidad (SIGENU) surge en junio de 2004 a solicitud de la dirección del Ministerio de Educación Superior de Cuba (MES), como una respuesta a las necesidades de automatización de los procesos fundamentales de la gestión académica de una Institución de Educación Superior (IES) en todas sus modalidades de estudio. El sistema está compuesto por varios módulos que gestionan la información de un estudiante desde que se matrícula hasta que se gradúa o causa baja definitiva.

Funciones principales:

- Inscripción de estudiantes
- Registro de asignaturas
- Registro de evaluaciones
- Control de bajas
- Emisión de reportes oficiales

Este proyecto está siendo utilizado en las 17 Instituciones de Educación Superior que pertenecen al MES y en sus 169 sedes universitarias municipales. En el curso 2008-2009 fue implantando en el Instituto Superior de Relaciones Internacionales (ISRI) y el Instituto Superior de Diseño Industrial (ISDI) (SIGENU-CUJAE, 2014).

### 1.2.7 Experiencias en la Universidad de las Ciencias Informáticas

En la UCI aunque no se tiene implantado un sistema que gestione toda la información científica que se genera, se han desarrollado un conjunto de aplicaciones que satisfacen algunas de las necesidades referentes a las actividades científicas e investigativas. A continuación se describen varias soluciones que por su contexto y utilidad pueden realizar un mejor aporte a la solución.

**Open Journal System**

El Open Journal System (OJS) es un sistema para la gestión editorial de la Serie Científica UCI (SC-UCI) que es una publicación seriada interna de la universidad, en la cual profesores y estudiantes involucrados en trabajos investigativos de corte científico tienen la oportunidad de hacer públicos los resultados de sus investigaciones. Este sistema manipula información como los datos de autor: nombre, apellidos, correo electrónico, filiación y otros. Además de información de los artículos como: título, resumen y el texto completo. OJS es un producto de código abierto y liberado bajo la licencia pública general GNU de la (General Public License GNU GPL) (Ortiz Batista, Yunaysy, y otros, 2013).

**Sistema de indicadores cienciométricos**

Para la confección del Sistema de indicadores cienciométricos (SIndiCIT) se adaptó el Sistema de Indicadores de Ciencia, Tecnología e Innovación (CTI) vigente en las instituciones y universidades del Ministerio de Educación Superior (MES) y en el Ministerio de Ciencia Tecnología y Medio Ambiente de Cuba (CITMA) a las condiciones existentes en la Universidad.

Este sistema depende, para su procesamiento, de una serie de entradas referente a los indicadores cienciométricos que son introducidos por las diferentes estructuras del sistema de CTI en la universidad.

El actual sistema brinda la posibilidad de llevar un control de los indicadores relacionados con la ciencia, tecnología e innovación que se realiza en la UCI, aunque también puede ser utilizado con el mismo propósito en otros centros. Brinda la posibilidad de calcular una serie de parámetros preestablecidos, los cuales pueden dar una idea de cómo se encuentra la institución. Es importante tener en cuenta que el uso está determinado por los permisos

que tenga asignado el usuario después de autenticado en el sistema (Ortiz Batista, Yunaysy, y otros, 2013).

**Dirección de Investigaciones**

En la universidad, la Dirección de Investigaciones (DI) se apoya tanto en los sitios web internos como en el portal de la universidad para la divulgación de información. Entre los temas principales están:

- Publicar documentación útil para realizar una publicación o documentación de proyectos y temas afines, normas de publicación para la serie interna de la UCI y notas informativas.
- Publicar todos los premios obtenidos en el ámbito universitario o eventos en lo nacional e internacionalmente.
- Publicar informes de balance de ciencia y técnica y las convocatorias de los diversos eventos que se realizan en la universidad, así como sus resultados.
- Brindar toda la información referente a la política científica de la universidad, líneas de investigación, doctorados y procedimientos.

### 1.2.8 Análisis de los sistemas presentados anteriormente

La tabla muestra las principales funcionalidades de la aplicación a desarrollar, y de estas cuáles están contenidas de una forma u otra en los sistemas analizados.

| | GREC | SICA | SGI | PGIL | SIPROCIT | SIGENU |
|---|---|---|---|---|---|---|
| ***GP*** | X | X | X | X | X | |
| ***EP*** | X | | X | X | X | |
| ***GLI*** | X | | | X | | |
| ***GN*** | | | | | | |
| ***GS*** | | | | | | |
| ***RS*** | X | | X | | | |
| ***Ind*** | X | | | | | |
| ***GU*** | X | X | X | X | | X |
| ***R*** | X | X | | X | X | X |

Tabla 1. Comparación de los sistemas estudiados

**Leyenda**

- Gestionar problemática: **GP**, evaluación de problemáticas: **EP**, gestión de líneas de investigación: **GLI**, generación de notificaciones: **GN,** gestión de solicitudes: **GS**, registro de salidas: **RS**, uso de indicadores: **Ind**, gestión de usuarios: **GU**, reportes: **R**.

A pesar de la existencia de los sistemas informáticos que gestionan la actividad científica y que brindan un punto de encuentro para los investigadores, estos han sido desarrollados para darle solución a problemas específicos de las instituciones, lo que dificulta su adopción por cualquier otra.

En el ámbito nacional, la mayoría de los sistemas constituyen portales web informativos, los cuales no están concebidos para la gestión de la información relacionada con las problemáticas de carácter científico de una institución, por lo que no cumplen con las necesidades identificadas.

Teniendo en cuenta los criterios antes expuestos no ha sido posible la adopción de ningún sistema como solución al problema planteado, debido a que no se ajustan a las políticas definidas por la DI de la universidad o no están acorde con la soberanía tecnológica a la que aspira el país.

Por tanto se propone el desarrollo de una aplicación web basada en tecnologías de código abierto para la gestión de las necesidades de investigación en la red de investigadores de la UCI, teniendo como premisa los intereses y estrategias de trabajo de la Dirección de Investigaciones de la Universidad (UCI, 2014).

### 1.3 Estudio de la metodología, lenguajes, patrones, tecnologías y herramientas

Para que el desarrollo de un producto cumpla los indicadores de confiabilidad y calidad necesarios en la actualidad se deben tener en cuenta la tendencia de tecnologías, lenguajes, metodologías y herramientas necesarias para garantizar un software competitivo y que cumpla con todas las expectativas del cliente.

Durante el desarrollo del trabajo de diploma se realizó un estudio de metodologías, tecnologías y herramientas con el fin de determinar las más apropiadas para el sistema propuesto, garantizando de esta manera que la solución informática utilice las tecnologías

definidas en el marco tecnológico de la Dirección de Investigaciones de la Universidad previsto para facilitar la integración con otras aplicaciones (UCI, 2014).

### 1.3.1 Metodología de desarrollo de software

**Exteme Programming (XP)**

La programación extrema, o Extreme Programming (XP), es una metodología de desarrollo ágil, una de las más exitosas en tiempo reciente. Su autor principal es Kent Beck, quien eligió algunas características de otras metodologías y las relacionó de forma que cada una complementara a la otra. Así, XP se puede definir como un conjunto de pasos de diversas metodologías, acopladas de manera que sean pasos flexibles a seguir utilizadas con el uso común, para realizar un desarrollo más agradable y sencillo. Consiste en una programación rápida o extrema, es una disciplina de desarrollo de software basado en la sencillez, la comunicación, la retroalimentación, el coraje y el respeto. Todo el equipo está junto cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto. Las entregas son frecuentes, y existe una refactorización continua, lo que permite mejorar el diseño cada vez que se le añada una nueva funcionalidad. (Beck, 2004).

Se selecciona XP sobre otras metodologías ya que es una metodología ágil, recomendada para proyectos pequeños de corta duración. Define un conjunto de buenas prácticas muy útiles para el equipo de desarrollo durante todo el proceso. Es la metodología ágil más popular y con más bibliografía disponible en internet. Está definida dentro del marco tecnológico establecido por la Dirección de Investigaciones de la Universidad (UCI, 2014).

### 1.3.2 Lenguajes de modelado

Para el desarrollo de una solución con calidad y competitividad el uso de lenguajes de modelado es realmente necesario ya que ayuda a visualizar y controlar la arquitectura del sistema, a comprender qué se está construyendo, muchas veces descubriendo oportunidades para la simplificación y reutilización. Además permite comunicar a otros la estructura deseada y el comportamiento del sistema.

**Notación de modelado de procesos de negocio**

La notación para gestión de procesos de negocio (BPMN) es el estándar más reciente para el modelado de procesos de negocio y servicio web. Permite modelar gráficamente las operaciones de los procesos del negocio, de forma que los usuarios no técnicos del negocio puedan leer y comprender hasta los procesos más complejos. Permite hacer un mejor uso de la gestión de procesos de negocio (BPM), ya que normaliza el método de notación que sirve como ayuda en la automatización de procesos (Ugarte Fajardo, 2008).

Para el modelado de procesos de negocio se selecciona BPMN por presentar una visión sistémica de la organización y sus procesos. Además facilita y propicia la medición, evaluación y control de los procesos lo que permite identificar puntos críticos y soluciones que se traducen en mejoramiento continuo. Crea procesos independientes de las personas que los manejan, proporcionando objetividad, solidez y continuidad. También permite a las personas ver con claridad los procesos y lograr un verdadero entendimiento de los mismos.

**Lenguaje de modelado unificado**

El Lenguaje Unificado de Modelado (UML 2.0) es un leguaje de modelado estandarizado de propósito general en el campo de la ingeniería de software orientada a objetos. La norma fue creada por el Object Management Group (OMG) en 1997, y desde entonces se ha convertido en el estándar del sector para el modelado de sistemas de software. UML incluye un conjunto de técnicas de notación gráfica para crear modelos visuales de programación orientada a objetos (Pressman, 2004).

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema, para documentarlo y para construirlo.

Para el modelado de los artefactos del sistema se seleccionó UML permitiendo mejorar los tiempos de desarrollo, modela utilizando conceptos orientados a objetos, consolida muchas de las notaciones y conceptos más usados orientados a objetos, además es fácilmente entendible.

### 1.3.3 Herramienta de ingeniería de software

**Visual Paradigm 8.0**

Visual Paradigm es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo de software a través de la representación de todo tipo de diagramas. Aunque es una herramienta privada, tiene una alternativa libre y gratuita. Fue diseñado para satisfacer una amplia gama de usuarios que desarrollan software de una forma fiable través de la utilización de un enfoque orientado a objetos (Visual Paradigm, 2014).

Se selecciona Visual Paradigm por ser una herramienta muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelente facilidades de interoperabilidad con otras aplicaciones.

### 1.3.4 Patrones

Los patrones son una disciplina de resolución de problemas reciente en la ingeniería del software que ha emergido en mayor medida de la comunidad de orientación a objetos, aunque pueden ser aplicados en cualquier ámbito de la informática y las ciencias en general. Los patrones de software facilitan la reutilización del diseño y de la arquitectura, capturando las estructuras estáticas y dinámicas de colaboración de soluciones exitosas a problemas que surgen al construir aplicaciones (Welicki, 2014).

Los patrones le ayudan a construir sobre la experiencia colectiva de ingenieros de software experimentados. Capturan la experiencia existente y que ha demostrado ser exitosa en el desarrollo de software, y ayudan a promover las buenas prácticas de diseño. Cada patrón aborda un problema específico y recurrente en el diseño o implementación de un sistema de software. (Buschmann, 1996)

Es muy importante conocer dos principios postulados por Martin Fowler en *"Analysis Patterns*" (Fowler, 1997) y que se deben tener presente en todo momento: los patrones son

un punto de partida, no un destino. Y los modelos no están bien o mal, sino que son más o menos útiles.

**Patrones de diseño**

Un patrón de diseño describe una estructura recurrente de componentes que se comunican para resolver un problema general de diseño en un contexto particular. Nomina, abstrae e identifica los aspectos clave de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. Identifica las clases e instancias participantes, sus roles y colaboraciones y la distribución de responsabilidades (Welicki, 2014).

**Patrones GRASP**

Los Patrones de Software para la Asignación General de Responsabilidad (GRASP), representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es el acrónimo para *General Responsibility Assignment Software Patterns* (Tabares, 2011).

GRASP destaca 5 patrones principales: Bajo acoplamiento, Experto, Alta cohesión, Creador y Controlador.

**Patrones GoF**

Los patrones GoF se descubren como una forma indispensable de enfrentarse a la programación a raíz del libro “Design Patterns—Elements of Reusable Software” de Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides, a partir de entonces estos patrones son conocidos como los patrones de la pandilla de los cuatro (GoF, gang of four) (Carlos A. Guerrero, 2013).

Los patrones de diseño GoF se clasifican en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

**Inyección de dependencias**

Inyección de dependencias, es un patrón de diseño, que permite "inyectar" objetos a una clase, en vez de que la clase cree el objeto por sí solo. Esto permite que la clase que utilice

el objeto no requiera especificar como crearlo, sino que otra clase se ocupará en crear este objeto según sus especificaciones y simplemente será agregado en la clase. (Griffin Caprio, 2005).

Típicamente este contenedor es implementado por un marco de trabajo externo a la aplicación por lo cual en la aplicación también se utilizará inversión de control al ser el contenedor quien invoque el código de la aplicación. Esta es la razón por la que los términos de inversión de control e inyección de dependencias se confunden habitualmente entre sí (Fowler, 2004).

**Unit of work**

Cuando se está introduciendo y sacando datos de una base de datos, es importante no perder de vista lo que ha cambiado para no perder información. Del mismo modo hay que insertar los nuevos objetos y retirar los que hayan sido eliminados. Se puede cambiar la base de datos con cada cambio en su modelo de objetos, pero esto puede dar lugar a un conjunto de llamadas a las base de datos que aunque son pequeñas termina siendo muy lento. Además se requiere que usted tenga una transacción abierta para interacción, lo que es poco práctico y la situación empeora si se necesita hacer un seguimiento de los objetos que tiene en la base de datos (Fowler, 2002).

Unit of work o unidad de trabajo realiza un seguimiento de todo lo que haces durante una transacción comercial y coordina la redacción de los cambios y la resolución de problemas de concurrencia que puede afectar a la base de datos. Cuando haya terminado, se da cuenta de todo lo que hay que hacer para modificar la base de datos como resultado de su trabajo (Miller, 2009).

**Patrones arquitectónicos**

Los patrones arquitectónicos se utilizan para expresar una estructura de organización base o esquema para un software. Proporcionando un conjunto de sub-sistemas predefinidos, especificando sus responsabilidades, reglas, directrices que determinan la organización, comunicación, interacción y relaciones entre ellos.

Los patrones arquitectónicos se centran en proporcionar modelos y métodos reutilizables específicamente para la arquitectura general de los sistemas de información.

En el desarrollo de la aplicación web propuesta se utilizará el marco de trabajo Symfony2, el cual está basado en el patrón Modelo–Vista–Controlador (MVC) (Potencier., 2011).

El marco de **MVC** incluye los componentes siguientes (Microsoft, 2014):

**Modelos:** los objetos de modelo son las partes de la aplicación que implementan la lógica del dominio de datos de la aplicación. A menudo, los objetos de modelo recuperan y almacenan el estado del modelo en una base de datos. En las aplicaciones pequeñas, el modelo es a menudo una separación conceptual en lugar de física. Por ejemplo, si la aplicación solo lee un conjunto de datos y lo envía a la vista, la aplicación no tiene un nivel de modelo físico ni las clases asociadas. En ese caso, el conjunto de datos asume el rol de un objeto de modelo.

**Vistas:** las vistas son los componentes que muestra la interfaz de usuario de la aplicación. Normalmente, esta interfaz de usuario se crea a partir de los datos de modelo.

**Controladores:** los controladores son los componentes que controlan la interacción del usuario, trabajan con el modelo y por último seleccionan una vista para representar la interfaz de usuario. En una aplicación MVC, la vista solo muestra información; el controlador administra y responde a los datos proporcionados por el usuario y su interacción. Por ejemplo, el controlador administra los valores de la cadena de consulta y pasa estos valores al modelo, que a su vez podría usarlos para consultar la base de datos.

### 1.3.5 Lenguajes de programación

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

**PHP 5.5**

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo. Se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy. Lo que ha atraído el interés de múltiples sitios con gran demanda de tráfico como Facebook, para optar por PHP como tecnología de servidor (PHP Group, 2014).

**JavaScript**

JavaScript es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Es el lenguaje script más popular en Internet, y funciona en todos los principales navegadores como Internet Explorer, Firefox, Chrome, Opera y Safari (W3Schools, 2014).

Entre las principales características del lenguaje se pueden encontrar:

- Es simple, no hace falta tener conocimientos avanzados de programación para poder hacer un programa en JavaScript.
- Maneja objetos dentro de la página Web y sobre ese objeto se pueden definir diferentes eventos. Dichos objetos facilitan la programación de páginas interactivas.
- Es dinámico, responde a eventos en tiempo real.
- El auge de las tecnologías web y el nacimiento de la web 2.0 a inicios del siglo XXI provocaron el incremento del uso de Javascript en sitios webs cada vez más dinámicos. Esta situación motivó al mismo tiempo el surgimiento de numerosas librerías de Javascript que facilitan considerablemente el trabajo con los objetos.

### 1.3.6 Librerías y framework de desarrollo

El uso de un framework puede simplificar considerablemente el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Proporciona una estructura fuerte al código, garantizando la generación de código más legible y fácil de mantener, además encapsula operaciones complejas en instrucciones sencillas con lo que facilita el desarrollo de aplicaciones.

Las librerías persiguen un objetivo similar, con la diferencia de que están orientadas a un uso más pasivo y comúnmente no generan código fuente, sino que el programador hace uso de las funcionalidades que esta brinda.

#### Symfony 2

Symfony2 es la versión más reciente de Symfony, el popular framework para desarrollar aplicaciones PHP. Se anunció por primera vez a principios de 2009 y supone un cambio radical tanto en arquitectura interna como en filosofía de trabajo respecto a sus versiones anteriores. Symfony2 ha sido ideado para exprimir al límite todas las nuevas características de PHP 5.3 y por eso es uno de los frameworks PHP con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en tu proyecto. Symfony2 también es el framework que más ideas incorpora del resto de frameworks, incluso de aquellos que no están programados con PHP (Eguiluz, 2014).

Se puede definir Symfony2 con diez términos: versátil, útil, flexible, visionario, popular, con buen soporte y uso de buenas prácticas, con excelente rendimiento, con bastante documentación y con más de 200 programadores de todo el mundo.

### 1.3.7 Herramienta IDE

#### Netbeans 8.0

NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. *Sun*

*MicroSystems* fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

NetBeans es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. Es un producto libre y gratuito sin restricciones de uso (Netbeans, 2014).

Es seleccionado Netbeans porque es un entorno de desarrollo integrado con una excelente interfaz gráfica, gratis y de código libre. Con soporte para HTML5 y otros lenguajes de programación, además tiene una comunidad muy activa de desarrolladores.

### 1.3.8 Gestor de base de datos

**PostgreSQL 9.4**

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS). Este está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. El proyecto PostgreSQL sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto. Este proporciona un gran número de características que normalmente solo se encontraban en las bases de datos comerciales tales como DB2 u Oracle (Gonzáles, 2015)

Se selecciona PostgreSQL por ser un ORDBMS multiplataforma de código abierto ampliamente popular e ideal para tecnologías web. Es fácil de administrar, presenta una sintaxis SQL estándar y fácil de aprender, con capacidad de replicación de datos y soporte empresarial disponible.

## 1.4 Técnicas para la captura y validación de requisitos

La identificación de los requisitos que debe cumplir un software es una actividad que se desarrolla al iniciarse cada sistema. En este proceso los analistas extraen de diferentes fuentes de información los datos que son necesarios para conocer las funcionalidades que implementará el sistema, haciéndose necesario por parte de los analistas el empleo de

técnicas que permitan establecer una buena comunicación con los interesados del producto y así lograr la satisfacción del cliente. Las técnicas a utilizar para la captura de requisitos son:

**Entrevistas:** es una de las técnicas más usadas en la captura de requisitos. Consiste en establecer una conversación entre personas de ambas partes. Estas son aplicadas a los especialistas funcionales.

**Tormenta de ideas:** reunión de varios interesados en la que todos expresan sus ideas sobre el problema y su solución. La forma de llevarla a cabo es que cada participante diga su idea sin ser interrumpido por otro. Al finalizar la sesión de lluvia de ideas se puede hacer una recolección de ideas sin duplicidad.

**Sistemas existentes:** se utiliza para el análisis de varios sistemas existentes que estén relacionados con el que va hacer construido. Analizándolos en cuanto a un grupo de características necesarias que debe llevar el que se va a elaborar.

También existen técnicas para la validación de los requisitos las cuales tienen como objetivo general demostrar que su definición es la que el usuario final necesita (Soto, 2012).

Muchas son las técnicas para validar requisitos, pero las que serán utilizadas durante el desarrollo del trabajo son:

**Revisión Técnica Formal:** son las revisiones realizadas por el usuario final del sistema y especialistas con el objetivo de validar la especificación de requisitos permitiendo detectar deficiencias, ambigüedades, omisiones y errores, tanto de formato como de contenido (Pérez, 2008).

**Prototipos de Interfaz de Usuario:** ayudan a identificar, comunicar y probar un producto antes de crearlo. La realización de los mismos, logra un entendimiento común entre el cliente y los desarrolladores, solucionando así la mayoría de los cambios posteriores en los proyectos de desarrollo de software (Machado Machado , 2012).

## 1.5 Pruebas de software

Las pruebas de software son básicamente un conjunto de actividades dentro del desarrollo del software que involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados donde el único instrumento adecuado para determinar el status de la calidad de un producto software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos (PruebaSoft, 2014).

Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso en desarrollo asegurando la calidad del sistema con el objetivo principal de presentar la información sobre la calidad del producto a las personas responsables de estas. Todo este proceso de pruebas, sus objetivos y los métodos y técnicas usados se describen en el plan de prueba.

**Pruebas de aceptación:** en esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique. Una prueba de aceptación es un escenario de utilización del sistema y el comportamiento que de él se espera, visto desde la perspectiva del cliente, usuario o sistema externo que interactúa con el programa.

**Pruebas unitarias:** una prueba unitaria es una forma de comprobar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

### Conclusiones parciales

Se realizó un análisis y estudio de los principales conceptos asociados a la comunicación y colaboración en el ámbito científico e investigativo, lo cual permitió establecer el marco conceptual en el contexto de la problemática formulada y orientar la concepción y desarrollo de la solución, tomando además como referencia los principales artículos sobre la importancia de la creación de redes de colaboración para la generación de conocimiento.

# Capítulo 1: Fundamentación Teórica

El estudio del estado del arte de las soluciones informáticas que brindan soporte a las investigaciones y mejoran la colaboración entre investigadores, evidenció las limitaciones de estos sistemas para satisfacer las necesidades existentes hoy en la Dirección de Investigaciones de la UCI.

Las herramientas y tecnologías antes seleccionadas son las más idóneas dentro del marco tecnológico establecido para facilitar la integración con otras aplicaciones que se están desarrollando por la Dirección de Investigaciones de la Universidad.

Las técnicas para la captura de requisitos utilizadas en desarrollo del trabajo fueron entrevistas, tormentas de ideas y sistemas existentes, que fueron validadas a través de las RTF y los prototipos de interfaz de usuario.

Las pruebas realizas al sistemas fueron las unitarias y las de aceptación cumpliendo con las expectativas del cliente.

**CAPÍTULO 2. ANÁLISIS Y DISEÑO**

## Introducción

El siguiente capítulo tiene como objetivo fundamental la presentación de la propuesta de solución del sistema a desarrollar. Además se presentan los artefactos generados en las dos primeras fases de la metodología de desarrollo utilizada, tales como Historias de Usuario (HU), plan de iteraciones, estimación de esfuerzo y tarjetas CRC (Clase, Responsabilidad y Colaboración). También se detallan los requisitos no funcionales, el modelo de datos, los patrones arquitectónicos y de diseño utilizados y la arquitectura del sistema.

## 2.1 Propuesta de solución

Tomando en cuenta los objetivos que se propone el presente trabajo de diploma y utilizando el análisis realizado en el capítulo anterior, se propone como solución desarrollar una aplicación web para la gestión de las necesidades de innovación tecnológica que eleve el trabajo colaborativo entre especialistas, académicos e investigadores en la UCI.

Para el uso del sistema los usuarios se autenticarán por el dominio uci.cu, permitiendo mostrar un perfil personalizado para cada persona. Todas las funcionalidades giran en torno al ciclo de vida de una problemática, desde su creación hasta su solución. Solo los usuarios con privilegios pueden crear las problemáticas y una vez creadas, si deciden hacerse cargo de ellas se publican directamente, en caso contrario quedarán disponibles a otros usuarios con privilegios para su adopción y posterior publicación.

Teniendo en cuenta que no existe una persona con conocimiento profundo en todas las líneas de investigación que definen las problemáticas, estas no se revisarán por expertos sino que para esta función se utilizará el conocimiento combinado de los usuarios del sistema de inteligencia colectiva, permitiendo que cada uno evalúe la problemática a través

de un voto efectuado, el cual sumará algebraicamente su valor de IDEC[2] directamente al valor de la problemática.

Un usuario interesado en ser colaborador de una problemática podrá enviar una solicitud de colaboración a la persona encargada y esta decidirá si es aceptado o no. El sistema también permite a los usuarios interesados conocer el desarrollo de una problemática y/o seguirla para de esta manera sean notificados de todos los cambios y eventos relacionados con esa problemática.

Al finalizar una problemática su responsable introducirá en el sistema las vías de realización que esta generó y la evaluación obtenida por cada uno de los indicadores que se le aplicaron para evaluar su alcance. Atendiendo a las políticas científicas aprobadas por la universidad se considera para el estudio una vía de realización a la descripción de los caminos posibles de ejecución de una solución según el alcance de la problemática.

## 2.2 Requisitos Funcionales (RF)

A través de los años se ha podido constatar que los requerimientos o requisitos son la pieza fundamental en un proyecto de desarrollo de software, ya que marcan el punto de partida para actividades como la planeación, básicamente en lo que se refiere a las estimaciones de tiempos y costos, así como la definición de recursos necesarios y la elaboración de cronogramas que será uno de los principales mecanismos de control con los que se contará durante la etapa de desarrollo.

Además la especificación de los requerimientos es la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto ya que estos son un reflejo detallado de las necesidades de los clientes o usuarios del sistema y es contra lo que se va a estar verificando si se están cumpliendo las metas trazadas (ARIAS CHAVES, 2011).

[2] Índice de desarrollo científico

Se identificaron un total de 50 requisitos funcionales:

RF01 Autenticar usuario

RF02 Cerrar sesión

**Gestionar problemática**

RF03 Adicionar problemática

RF04 Modificar problemática

RF05 Asumir problemática

RF06 Evaluar problemática

RF07 Cerrar problemática

RF08 Eliminar problemática

RF09 Mostrar detalles de la problemática

RF10 Mostrar listado de problemáticas

RF11 Mostrar problemáticas por membresía de usuario

RF12 Mostrar problemáticas asociadas a la detallada

RF13 Mostrar problemáticas sin responsables

RF14 Seguir problemática

RF15 Cancelar seguimiento de la problemática

RF16 Registrar vía de realización de la problemática

RF17 Evaluar vía de realización de la problemática

**Gestionar comentario**

RF18 Adicionar comentario

RF19 Modificar comentario

RF20 Eliminar comentario

RF21 Mostrar comentarios

**Gestionar acuerdo**

RF22 Adicionar acuerdo

RF23 Eliminar acuerdo

RF24 Adicionar responsabilidad de cada colaborador

RF25 Mostar responsabilidad de cada colaborador

**Gestionar solicitud de colaboración**

RF26 Enviar solicitud de colaboración

RF27 Mostrar solicitudes de colaboración recientes

RF28 Mostrar todas las solicitudes de colaboración

**Gestionar notificación**

RF29 Adicionar notificación

RF30 Mostrar notificaciones recientes

RF31 Mostrar todas las notificaciones

**Gestionar rol**

RF32 Adicionar rol

RF33 Modificar rol

RF34 Eliminar rol

RF35 Mostrar roles

**Gestionar línea de investigación**

RF36 Adicionar línea de investigación

RF37 Modificar línea de investigación

RF38 Eliminar línea de investigación

RF39 Mostrar líneas de investigación

**Gestionar indicador**

RF40 Adicionar indicador

RF41 Modificar indicador

RF42 Eliminar indicador

RF43 Mostrar indicadores

**Gestionar vía de realización**

RF44 Adicionar vía de realización

RF45 Modificar vía de realización

RF46 Eliminar vía de realización

RF47 Mostrar vías de realización

**Gestionar evaluación**

RF48 Adicionar evaluación

RF49 Modificar evaluación

RF50 Eliminar evaluación

RF51 Mostrar evaluaciones

## 2.3 Requisitos No Funcionales (RNF)

Los RNF son los requisitos que imponen restricciones al diseño o funcionamiento del sistema software (tal como requisitos de funcionamiento, estándares de calidad, o requisitos del diseño). Son propiedades o cualidades que el producto debe tener, que lo hacen un producto atractivo, usable, rápido o confiable.

1. **Usabilidad**
   - RNF1: el acceso al sistema debe ser fácil y rápido.
   - RNF2: las etiquetas de cada funcionalidad y los campos de cada interfaz tendrán títulos asociados a su función de negocio.
2. **Seguridad**
   - RNF3: proteger la información manejada por el sistema de accesos no autorizados.
   - RNF4: garantizar que las funcionalidades del sistema se muestren de acuerdo al rol del usuario que esté activo.
   - RNF5: permitir autenticarse a través del servicio de autenticación LDAP proporcionado por la universidad.
3. **Rendimiento**
   - RNF6: el sistema debe ser capaz de funcionar al ser instalados en una misma PC todos sus componentes tales como gestor de bases de datos y aplicación web.
4. **Portabilidad**
   - RNF7: la solución debe poder ser ejecutada desde las plataformas Windows y Linux.
5. **Validación de información**
   - RNF8: el sistema debe validar automáticamente la información contenida en los formularios de ingreso. En el proceso de validación de la información, se deben

tener en cuenta aspectos tales como obligatoriedad de campos, longitud de caracteres permitida por campo y manejo de tipos de datos.

6. **Hardware**

   **Cliente:**

   - RNF9: requerimientos mínimos: procesador Pentium IV a 800 MHz, 128mb de memoria RAM y un navegador web.
   - RNF10: las computadoras clientes deben estar conectadas en red.

   **Servidor:**

   - RNF11: requerimientos mínimos: procesador Dual Core a 3.00 GHz, 1gb de memoria RAM y una capacidad de 40gb de disco duro.
   - RNF12: el servidor debe tener al menos 1 tarjeta de red para establecer la conexión.

7. **Software**

   **Cliente:**

   - RNF13: sistema operativo con interfaz gráfica y soporte para red.
   - RNF14: las interfaces deben ser compatibles con Mozilla Firefox 30.0 o superior.

   **Servidor:**

   - RNF15: servidor web Apache 2.4 o superior.
   - RNF16: gestor de base de datos PostgreSQL 9.0 o superior.

## 2.4 Planificación

En esta primera fase de la metodología, el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. Los clientes establecen la prioridad de las historias de usuario en correspondencia con las necesidades más inmediatas para luego asignarlas, por el orden de relevancia, a las iteraciones planificadas. A partir de las historias se realiza la estimación del esfuerzo que se requiere por parte del equipo para su posterior implementación. Las

estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto.

### 2.4.1 Historias de usuario

Entre los artefactos definidos en la metodología seleccionada están las historias de usuario que son utilizadas para especificar las funcionalidades que brindará el sistema. Cada historia de usuario es la representación de un requerimiento de software escrito utilizando el lenguaje común del usuario. Representan una forma rápida de gestionar los requerimientos de los usuarios sin tener que elaborar una gran cantidad de artefactos formales y sin requerir mucho tiempo. Constituyen la base para realizar las pruebas de aceptación, la estimación y la planificación del proyecto.

**Las historias de usuario serán representadas mediante tablas con la siguiente información:**

1. **Número:** número de la historia de usuario.
2. **Nombre de Historia de Usuario:** nombre de la historia de usuario a desarrollar.
3. **Usuario:** involucrados en la funcionalidad de la HU.
4. **Iteración Asignada:** número de la iteración en la cual se desarrollará.
5. **Prioridad en negocio:**
   - **Alta:** HU que resultan fundamentales en el desarrollo y control integral del sistema.
   - **Media:** HU que resultan para el cliente funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.
   - **Baja:** HU que sirven de ayuda al control de elementos asociados al desarrollo del sistema.
6. **Riesgo en desarrollo:**
   - **Alta:** cuando en su implementación se considera la posible existencia de errores que lleven la inoperatividad del código.
   - **Media:** cuando en su implementación pueden aparecer errores que puedan retrasar la entrega del sistema.

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

7. **Estimación técnica:** tiempo estimado que se demorará el desarrollo de la HU.
8. **Descripción:** descripción de la HU.

Como resultado del trabajo realizado se elaboraron un total de 50 HU, se muestra la descripción de algunas de las principales y las restantes se pueden consultar en los anexos.

| HISTORIA DE USUARIO | |
|---|---|
| **Número:** HU-01 | **Usuario:** Investigadores |
| **Nombre de Historia:** Adicionar problemática | |
| **Prioridad:** Alta | **Riesgo en desarrollo:** Alta |
| **Estimación Técnica:** 1 | **Iteración asignada:** 1 |
| **Descripción:** El usuario podrá crear una nueva problemática llenando los datos pertinentes del formulario para su creación y a continuación se le preguntará si desea hacerse responsable de ella o no. | |

Tabla 2. HU Crear problemática

### 2.4.2 Estimación de esfuerzo

Para el desarrollo satisfactorio de la solución propuesta, se realizó una estimación de esfuerzo para cada una de las historias de usuario, arrojando los siguientes resultados:

| No. | HISTORIAS DE USUARIO | ESTIMACIÓN TÉCNICA |
|---|---|---|
| **1** | Autenticar usuario | 2 |
| **2** | Cerrar sesión | 1 |
| **3** | Adicionar problemática | 1 |
| **4** | Modificar problemática | 1 |
| **5** | Asumir problemática | 2 |
| **6** | Evaluar problemática | 2 |

| | | |
|---|---|---|
| **7** | Cerrar problemática | 1 |
| **8** | Eliminar problemática | 1 |
| **9** | Mostrar detalles de la problemática | 1 |
| **10** | Mostar listado de problemáticas | 2 |
| **11** | Mostrar problemáticas por membresía de usuario | 2 |
| **12** | Mostrar problemáticas asociadas a la detallada | 2 |
| **13** | Mostrar problemáticas sin responsables | 1 |
| **14** | Seguir problemática | 2 |
| **15** | Cancelar seguimiento de la problemática | 1 |
| **16** | Registrar vía de realización de la problemática | 1 |
| **17** | Evaluar vía de realización de la problemática | 1 |
| **18** | Adicionar comentario | 1 |
| **19** | Modificar comentario | 1 |
| **20** | Eliminar comentario | 1 |
| **21** | Mostrar comentarios | 1 |
| **22** | Adicionar acuerdo | 1 |
| **23** | Eliminar acuerdo | 1 |
| **24** | Adicionar responsabilidad de cada colaborador | 1 |
| **25** | Mostar responsabilidad de cada colaborador | 1 |
| **26** | Enviar solicitud de colaboración | 1 |
| **27** | Mostrar solicitudes de colaboración recientes | 2 |
| **28** | Mostrar todas las solicitudes de colaboración | 1 |
| **29** | Adicionar notificación | 1 |
| **30** | Mostrar notificaciones recientes | 2 |
| **31** | Mostrar todas las notificaciones | 1 |
| **32** | Adicionar rol | 1 |
| **33** | Modificar rol | 1 |
| **34** | Eliminar rol | 1 |
| **35** | Mostrar roles | 1 |
| **36** | Adicionar línea de investigación | 1 |

| | | |
|---|---|---|
| **37** | Modificar línea de investigación | 1 |
| **38** | Eliminar línea de investigación | 1 |
| **39** | Mostrar líneas de investigación | 1 |
| **40** | Adicionar indicador | 1 |
| **41** | Modificar indicador | 1 |
| **42** | Eliminar indicador | 1 |
| **43** | Mostrar indicadores | 1 |
| **44** | Adicionar vía de realización | 1 |
| **45** | Modificar vía de realización | 1 |
| **46** | Eliminar vía de realización | 1 |
| **47** | Mostrar vías de realización | 1 |
| **48** | Adicionar evaluación | 1 |
| **49** | Modificar evaluación | 1 |
| **50** | Eliminar evaluación | 1 |
| **51** | Mostrar evaluaciones | 1 |

Tabla 3. Estimación de esfuerzo

### 2.4.3 Plan de iteraciones

En el plan de iteraciones se establece cuántas serán necesarias realizar sobre el sistema hasta completarlo. Sin embargo, se precisa establecer el contenido de trabajo para todas y cada una de ellas regulando la cantidad de historias de usuario a implementar dentro del rango establecido por la estimación efectuada. Una iteración no es más que un pequeño proyecto, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración.

Tomando como referencia los aspectos antes tratados la aplicación que se pretende construir se desarrollará en tres iteraciones, explicadas más detalladamente a continuación:

**Iteración 1**

La iteración tiene como finalidad implementar las historias de usuario que se consideraron más necesarias atendiendo a su relevancia e impacto para el negocio. Se da respuesta a

las funcionalidades de gestión problemáticas, gestión de acuerdos, gestión de usuario y gestión de solicitudes.

**Iteración 2**

En esta iteración se implementan las historias de usuario de prioridad media para el cliente: gestión de comentarios, gestión de seguimiento, gestión de notificaciones y gestión de vías de realización.

**Iteración 3**

Con la tercera y última iteración se implementan todas historias de usuario restantes, entre ellas los cuatro nomencladores que define el sistema.

A modo de resumen se presenta la siguiente tabla que muestra las tres iteraciones analizadas previamente con las historias de usuario que incluyen y su duración:

| ITERACIONES | HISTORIAS DE USUARIO | DURACIÓN |
|---|---|---|
| **1** | Gestionar problemática | 3 semanas |
| | Gestionar acuerdo | |
| | Gestionar solicitud | |
| | Gestionar usuario | |
| **2** | Gestionar comentario | 2 semanas |
| | Gestionar seguimiento | |
| | Gestionar notificación | |
| | Gestionar vías de realización | |
| **3** | Gestionar rol | 1 semana |
| | Gestionar línea de investigación | |
| | Gestionar indicador | |
| | Gestionar evaluación | |

Tabla 4. Plan de iteraciones

### 2.4.4 Plan de entregas del sistema

El plan de entregas es el compromiso final del equipo de desarrollo con los clientes donde se realiza un cronograma de entregas para establecer qué historias de usuario serán

agrupadas para conformar una entrega, y el orden de las mismas. El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por el equipo y el tiempo con que dispone el cliente para tener en sus manos la solución. Es una actividad muy importante entre el equipo de desarrollo y el cliente, ya que la entrega tardía o temprana de la solución, repercute notablemente en todos los involucrados. Tras algunas iteraciones es recomendable realizar nuevamente una reunión con los actores del proyecto, para evaluar nuevamente el plan de entregas y ajustarlo si es necesario.

| No. ENTREGA | HITO | FECHA DE ENTREGA |
|---|---|---|
| **1** | Fin de la primera iteración | 18/04/2015 |
| **2** | Fin de la segunda iteración | 02/05/2015 |
| **3** | Fin de la tercera iteración | 09/05/2015 |

Tabla 5. Plan de entregas

## 2.5 Diseño

XP establece prácticas especializadas que inciden directamente en la realización del diseño para lograr un sistema robusto y reutilizable tratando de mantener su simplicidad, es decir, crear un diseño evolutivo que se va mejorando incrementalmente y que permite hacer entregas pequeñas y frecuentes de valor para el cliente. A la hora de darle cumplimiento a la actividad de diseñar, XP no especifica ninguna técnica de modelado. Pueden utilizarse indistintamente sencillos esquemas en una pizarra, diagramas de clases utilizando UML o tarjetas CRC siempre que sean útiles, tributen a la comprensión y no requieran mucho tiempo en su creación.

### 2.5.1 Tarjetas CRC

La metodología XP en lugar de utilizar diagramas para desarrollar modelos representa las clases mediante tarjetas. Las tarjetas CRC (Clase, Responsabilidad y Colaboración) ayudan al equipo a definir actividades durante el diseño del sistema. Cada tarjeta representa una clase en la programación orientada a objetos y define sus responsabilidades y las colaboraciones con las otras clases.

Como resultado del trabajo realizado durante la fase de diseño se elaboraron un total de 17 tarjetas CRC, a continuación se muestran algunas de ellas:

| Clase: ProblemáticaController | |
|---|---|
| **Responsabilidad** | **Colaboración** |
| • Crear problemática<br>• Modificar problemática<br>• Concluir problemática<br>• Adoptar problemática<br>• Evaluar problemática<br>• Mostrar problemáticas<br>• Mostrar detalles de la problemática<br>• Mostrar problemáticas huérfanas (sin responsable definido)<br>• Mostrar problemáticas relacionadas con la detallada | • Problemática<br>• ProblemáticaType |

Tabla 6. Tarjeta CRC Problemática

### 2.5.2 Modelo de datos

Un modelo de datos es la representación abstracta de los datos en un sistema gestor de base de datos. Básicamente el modelo de datos está formado por tres elementos fundamentales que son: objetos, atributos y relaciones.

El modelo de datos correspondiente al sistema consta de un total de catorce tablas, de ellas cinco nomencladores y las nueve restantes para el almacenamiento de la información manipulada por este. Para su construcción se tuvo en cuenta todos los datos que deben persistir en el sistema y en los nomencladores se agruparon los estándares predeterminados del negocio del proceso.

El modelo siguiente refleja los conceptos, las relaciones y las reglas del dominio estudiado, con el objetivo de darle respuesta al problema afrontado por el equipo.

# Capítulo 2: Análisis y diseño

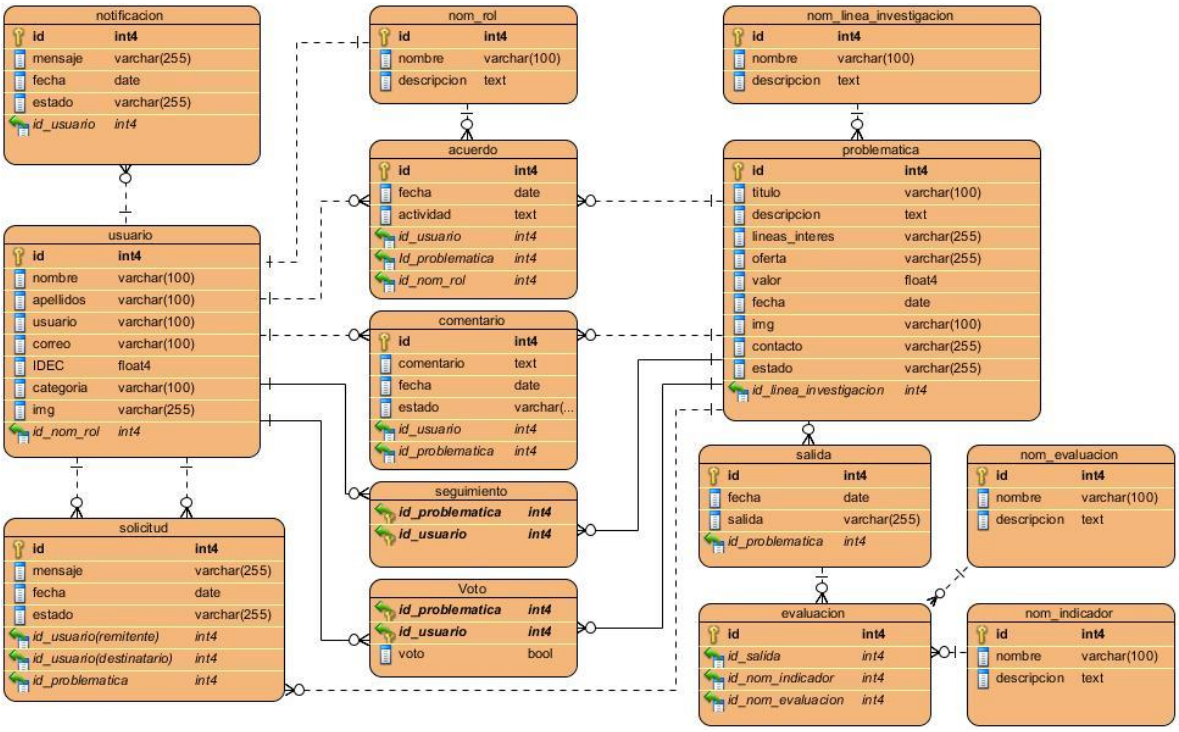


Figura 1. Modelo de datos

## 2.6 Patrones arquitectónicos y de diseño

Si bien la elaboración de un buen diseño del software contribuye directamente a la calidad del producto final, gran parte de esta yace en la utilización adecuada de los patrones de diseño y arquitectónicos existentes en la actualidad. Los patrones son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan.

Con empleo del framework Symfony2 se garantiza la utilización de varios de estos patrones, permitiendo a los desarrolladores utilizar buenas prácticas de programación y ahorrar tiempo y recursos. A continuación se describen los principales patrones empleados en el diseño e implementación de la solución.

### 2.6.1 Inyección de dependencias

Dentro del marco de trabajo Symfony2 cada *bundle* proporciona un fichero de configuración llamado *services.yml.* Este contiene dos secciones (*services y parameters*). En el primero

se declaran todos los parámetros de configuración y en el segundo todos los servicios que se utilizarán.

Los controladores heredan de la clase *Controller*, la cual proporciona un atributo público llamado *container*, una instancia del contenedor de dependencias. Este permite que desde cualquier clase derivada de la anteriormente mencionada se pueda obtener una instancia del contenedor de dependencias por lo que se puede instanciar cualquier servicio existente haciendo uso de la función get().

Para un fácil entendimiento de lo planteado se ilustra un ejemplo muy simple, en el cual se configura el servicio be_simple_soap en el contenedor de symfony2. Este será el encargado de obtener los datos de los usuarios a través del servicio de autenticación en su versión seis que ofrece la UCI.

```
1  be_simple_soap:
2      clients:
3          uci:
4              # required
5              wsdl: https://autenticacion2.uci.cu/v6/PasarelaAutenticacionWS.wsdl
```

Figura 2. Servicio Be_simple_soap

Una vez creado el servicio se podrá utilizar a través del método get() en cualquier controlador de la aplicación.

```
49      private function RegistroNuevoUsuario($user)
50      {
51          $cliente = $this->container->get('besimple.soap.client.uci');
52
53          $usuarioUDDI = $cliente->ObtenerPersonaDadoUsuario($user);
54          $usuario = new Usuario();
55          $usuario->setNombre($usuarioUDDI->Nombres);
56          $usuario->setApellidos($usuarioUDDI->Apellidos);
```

Figura 3. Acceso al servicio be_simple_soap brindado por el contenedor de dependencias

### 2.6.2 Modelo-Vista-Controlador (MVC)

El patrón MVC obliga a dividir y organizar el código de acuerdo a las convenciones establecidas por el framework, la interfaz de usuario se guarda en la vista, la manipulación de datos se guarda en el modelo y el procesamiento de las peticiones constituye el controlador. El empleo del patrón MVC en un sistema resulta bastante útil además de restrictivo. Symfony2 está basado en el patrón de arquitectura conocido MVC, formado por tres niveles: el Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio, la Vista es la encargada de originar las páginas que son mostradas como resultado de las acciones y el Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. Symfony2 toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo.

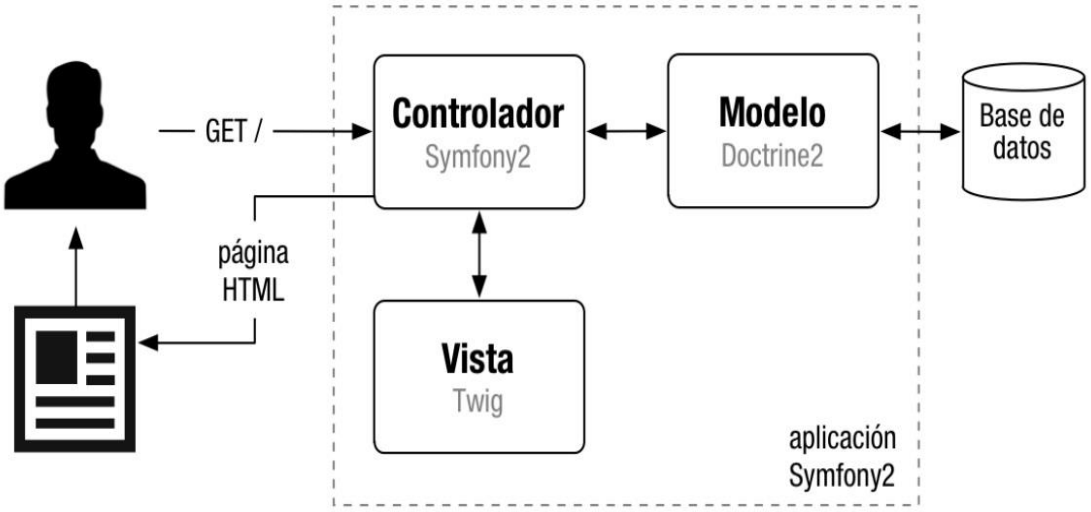


Figura 4. Patrón MVC en Symfony2

### 2.6.3 Unit of Work

Doctrine 2 utiliza el *Unit Of Work* para encolar todas las consultas SQL que tiene que ejecutar cuando se invoca el método **flush().** Una entidad puede ser persistida pasándosela al método **persist()** del **EntityManager**. Si aplicamos esta operación sobre una entidad, su

estado cambia a ***MANAGED***, que significa que su persistencia ahora estará manejada por el EntityManager. Hay que tener en cuenta que el método persist no ejecuta una consulta SQL en la base de datos.

```
35      if ($form->isValid()) {
36          $em = $this->getDoctrine()->getManager();
37          $em->persist($entity);
38          $em->flush();
```

Figura 5. Método persist() en Doctrine 2

Una entidad puede ser eliminada de la base de datos pasándosela al método ***remove*** del **EntityManager**. Aplicando este método en alguna entidad, cambia su estado a ***REMOVED*** lo que significa que será borrada cuando el método **flush()** sea invocado (Doctrine 2 ORM, 2015).

```
228
229             $em->remove($entity);
230             $em->flush();
231         }
```

Figura 6. Método remove() en Doctrine 2

### 2.6.4 Patrones GRASP

**Controlador:** este patrón se puede evidenciar dentro del sistema desarrollado en el controlador frontal que implementa Symfony2 para atender todas las peticiones web que se realizan por el usuario, es el único punto de entrada de toda la aplicación. Cuando recibe una petición, utiliza el sistema de enrutamiento para enviar la acción al controlador responsable de la solución y se encarga de manejar la seguridad y ejecución de los filtros. Además se crean clases controladoras para cada caso de uso del sistema, ubicadas en src\AppBundle\Controller.

**Experto:** en el sistema desarrollado Symfony2 utiliza el mapeador de objetos y relaciones por sus siglas en inglés (ORM) Doctrine2 para:

- Realizar su capa de abstracción en el modelo
- Encapsular toda la lógica de los datos
- Generar las clases con todas las funcionalidades comunes de las entidades

Las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria. Las clases *repository* constituyen el mejor ejemplo.

**Alta cohesión:** En el caso del sistema propuesto utilizando el patrón alta cohesión se le asigna a cada clase las responsabilidades que le corresponde y se establecen las condiciones para que cada clase colabore con las demás en la resolución de tareas que las implican a todas y que no son capaces de resolver por sí solas. Un ejemplo de ello son los métodos *actions* de las clases, los cuales son responsables de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades.

**Bajo acoplamiento:** Las clases que implementan la lógica de negocio y de acceso a datos se encuentran en el modelo, estas clases no tienen asociaciones con las de la vista o el controlador por lo que la dependencia en este caso es baja.

### 2.6.5 Patrones GOF

**Decorador:** la plantilla base base.html.twig, padre de todas las vistas, conocida como plantilla global, contiene el código HTML que es tradicional en todas las páginas del sistema, para no tener que repetirlo. El contenido de la plantilla se integra en las demás vistas del sistema, o si se mira desde el otro punto de vista, decora el resto de las plantillas twig.

## 2.7 Arquitectura

El desarrollo de la aplicación responde a una arquitectura basada en capas la cual está compuesta por la capa de presentación, capa de negocio y capa acceso a datos. Los sistemas o arquitecturas en capas brindan entre otros beneficios un cierto aislamiento ya que en caso de ocurrir un error o de que sea necesario actualizar algún elemento solo se realizaría el cambio en la capa correspondiente y el resto se mantiene intacto. En la

siguiente imagen se puede apreciar la estructura en capas y la ubicación de los elementos del patrón arquitectónico MVC.

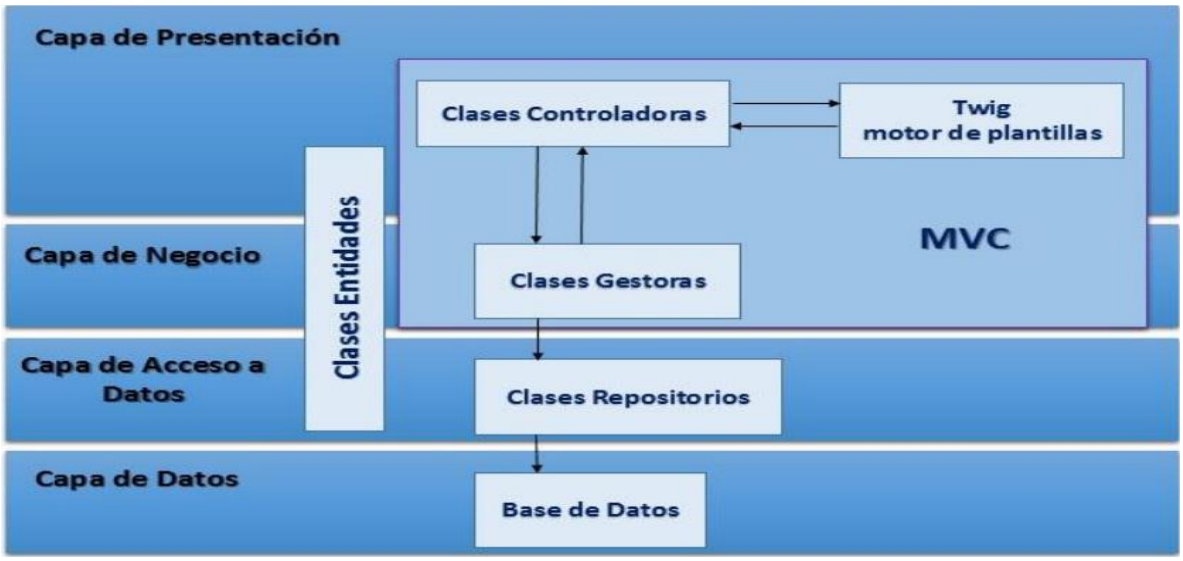


Figura 7. Arquitectura del sistema

## 2.8 Métricas de diseño

Las métricas de diseño permiten medir de forma cuantitativa la calidad de los atributos internos del software. Son varios los puntos de vista relacionados con la calidad del software. Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software con medidas que pueden ayudar al desarrollador a juzgar la calidad de un diseño a nivel de componente. Las métricas se centran en cuantificar tanto la complejidad, como la responsabilidad, reutilización, acoplamiento y cantidad de pruebas.

### 2.8.1 Tamaño Operacional de Clases (TOC)

Esta métrica se encuentra entre las planteadas por Lorenz y Kidd y pertenece al grupo de las métricas de tamaño de clase. La misma se centra en la cantidad de las operaciones para cada clase individual y los valores promedio para el sistema como un todo. El tamaño operacional está dado por el número de métodos asignados a una clase y evalúa la responsabilidad, la complejidad de implementación y la reutilización.

Para evaluar las métricas son necesarios los valores de los umbrales para los parámetros de calidad. Los umbrales para esta métrica se basan en el promedio de operaciones por clases obtenidos, estos valores fueron los aplicados en el diseño del sistema y se muestran en la siguiente tabla.

| | Categoría | Criterio |
|---|---|---|
| **Responsabilidad** | Baja | < =Prom. |
| | Media | Entre Prom. y 2* Pom. |
| | Alta | > 2* Prom. |
| **Complejidad implementación** | Baja | < =Prom. |
| | Media | Entre Prom. y 2* Pom. |
| | Alta | > 2* Prom. |
| **Reutilización** | Baja | > 2*Prom. |
| | Media | Entre Prom. y 2* Pom. |
| | Alta | <= Prom. |

Tabla 7. Criterios para evaluar los atributos según el umbral de TOC

A partir de la evaluación de cada una de las clases según los atributos anteriormente mencionados se obtuvo el siguiente resultado:

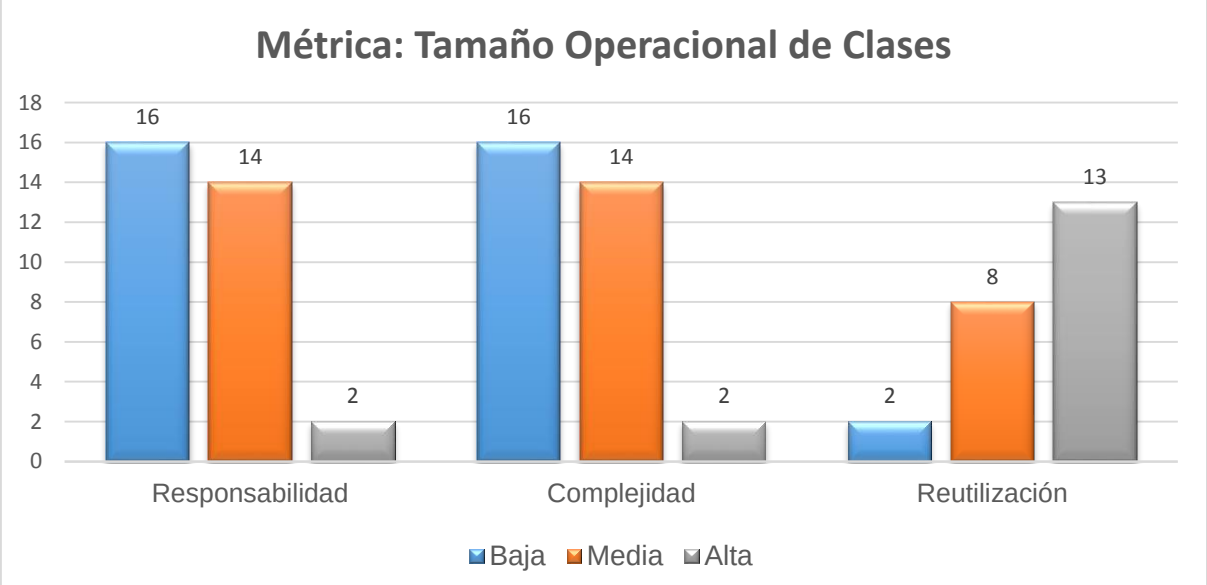


Figura 8. Resultados de la evaluación del diseño según TOC

En el análisis de los resultados que se obtuvieron al realizar la evaluación del diseño propuesto mediante la métrica TOC, se demuestra que el mismo presenta un bajo nivel de responsabilidad y complejidad entre sus clases y una alta reutilización.

### 2.8.2 Relaciones entre Clases (RC)

Esta métrica se basa en la cantidad de relaciones de uso que presenta una clase determinada con las demás clases. Utiliza los atributos acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas para medir la calidad del diseño de la clase.

Los umbrales para evaluar el diseño según esta métrica se basan en el promedio de relaciones por clases y el caso del atributo Acoplamiento, utiliza la propia cantidad de relaciones, estos valores fueron los aplicados en el diseño del sistema y los mismos son reflejados en la siguiente tabla.

| | Categoría | Criterio |
|---|---|---|
| **Acoplamiento** | Bajo | 1 |
| | Medio | 2 |
| | Alto | >2 |
| **Complejidad de Mantenimiento** | Baja | <= Prom. |
| | Media | Entre Prom. y 2*Prom. |
| | Alta | > 2*Prom. |
| **Reutilización** | Baja | >2* Prom. |
| | Media | Entre Prom. y 2*Prom. |
| | Alta | <= Prom. |
| **Cantidad de Pruebas** | Baja | <= Prom. |
| | Media | Entre Prom. y 2*Prom. |
| | Alta | > 2*Prom. |

Tabla 8. Criterios para evaluar los atributos según el umbral de RC

A partir de la evaluación de cada una de las clases según los atributos anteriormente mencionados se obtuvo el siguiente resultado:

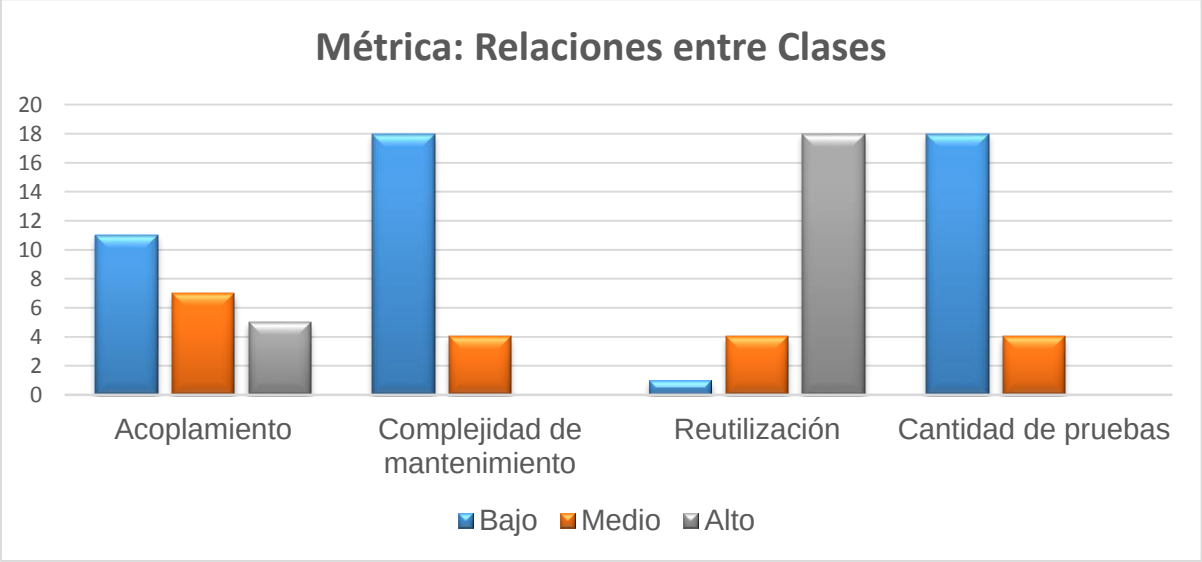


Figura 9. Resultados de la evaluación del diseño según RC

En el análisis de los resultados que se obtuvieron al realizar la evaluación del diseño propuesto mediante la métrica RC, se puede concluir que las clases poseen bajos niveles de acoplamiento, complejidad de mantenimiento y cantidad de pruebas y altos niveles de reutilización.

## Conclusiones parciales

En este capítulo quedaron definidas las características fundamentales de la propuesta de solución para el desarrollo del sistema.

Fueron descritos 51 RF en las historias de usuario y se identificaron 16 RNF.

La estimación de esfuerzo, el plan de entregas y el plan de iteraciones generados durante la primera fase permiten planificar correctamente el desarrollo de la aplicación.

En la fase de diseño se elaboraron las tarjetas CRC y el modelo de datos.

Se muestra como se aplican cada uno de los patrones arquitectónicos y de diseño en la solución para lograr una correcta estructuración.

Se obtuvieron resultados satisfactorios en la evaluación del diseño mediante las métricas TOC y RC.

## CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS

### Introducción

En el capítulo se muestra la organización y las dependencias entre los componentes de la aplicación. Se lleva a cabo la implementación de las historias de usuario descritas en las fases anteriores, planificadas en tres iteraciones. Y finalmente se desarrollan las pruebas unitarias y de aceptación realizadas según propone XP.

### 3.1 Diagrama de despliegue

Un diagrama de despliegue es la forma de mostrar la configuración de nodos de procesamientos en tiempo de ejecución y los componentes que en ellos residen. Estos nodos forman la topología de hardware sobre el que se ejecuta el sistema. Este diagrama se preocupa principalmente de la distribución, entrega e instalación de las partes que constituye el sistema físico. Se utilizan en el diseño y la implementación. Un diagrama de despliegue consta de la interconexión de nodos a partir de relaciones de asociación. Resulta más amplio que el diagrama de componentes en el sentido de que puede tener más clases de elementos (Campderrich Falgueras, 2003).

El diagrama de despliegue de la aplicación quedaría de la siguiente manera:

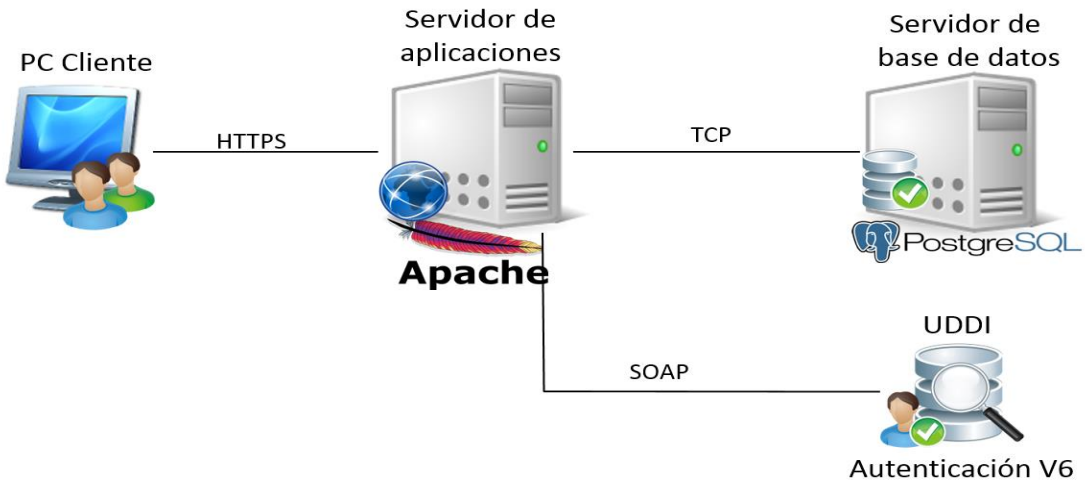


Figura 10. Diagrama de despliegue

# Capítulo 3. Implementación y pruebas

## 3.2 Estándares de codificación

Es importante utilizar buenas prácticas a la hora de programar para lograr un código más consistente, mejoramos la lectura del mismo y facilitamos la colaboración con otros programadores. Una de las mejores maneras de lograr esto, es mediante el uso de estándares.

El uso de estándares establecidos permitirá ser más objetivos a la hora de revisar código de otros, ayudará a mejorar la reusabilidad del código, existirá código compatible con otros proyectos y se hará más fácil el mantenimiento del sistema.

En la implementación del sistema se utilizarán los estándares de codificación en PHP (PSR-0, PSR-1, PSR-2) los cuales son aceptados por la gran comunidad de PHP.

**PSR-0. Estándar de carga automática (PHP, 2015)**

- Los nombres de espacio y las clases deben tener la siguiente estructura <Vendor name> (<Namespace>)*<Class Name>.
- Cada nombre de espacio debe tener un nombre de espacio superior ("Vendor name").
- Cada nombre de espacio puede tener tantos sub-nombres de espacio como se quiera.
- Los nombres de los nombres de espacio o clases deben ser separados por un guion bajo (_).
- Todos los archivos deben tener la extensión .php.

**PSR-1. Estándar de codificación básica (PHP, 2015)**

- Los archivos deben utilizar solamente <?php y las short tags <?=
- Los archivos sólo deben utilizar una codificación UTF-8.
- Las funciones sólo deben retornar un sólo valor.
- Los nombres de espacio y las clases deben seguir las reglas del PSR-0.
- Los nombres de clases deben ser escritas utilizando la técnica StudlyCaps.

- Las constantes deben ser definidas en MAYÚSCULAS y utilizando guion bajo (_) cómo separador.
- Métodos y funciones deben ser escritos utilizando la técnica camelCase.

**PSR-2. Guía de estilo de codificación (PHP, 2015)**

- El código debe seguir el estándar PSR-1.
- Debemos de validar que la función que vamos a crear no exista utilizando la función function_exists().
- Las llaves deben de estar abajo solamente en las clases y métodos.
- La identación debe ser con un tabulador establecido a 4 espacios.
- Las constantes true, false y null deben ser escritos en minúsculas.
- El número de caracteres por línea deben ser de 80 columnas aunque también esta aceptado que sean hasta 120.
- Ya no debes utilizar la palabra reservada var para declarar una propiedad, debes utilizar public, private, static o protected.
- Debe haber un espacio después de cada estructura de control (if, for, foreach, while, switch, try...catch, etc.).

## 3.3 Pruebas

Uno de los pilares de la Programación Extrema es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones. XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final (Gutierrez, y otros, 2007).

### 3.4.1 Pruebas unitarias

Las pruebas unitarias prueban que un pequeña parte del código de la aplicación funciona tal y como debería hacerlo. Un test que no pasa satisfactoriamente es la mejor señal de que algo no funciona bien en la aplicación. Esta es la gran ventaja de las pruebas unitarias, que avisan cada que vez que falla la aplicación, por lo que se recomienda ejecutar las pruebas cada vez que se complete una funcionalidad.

Por convención, cada prueba unitaria y funcional de Symfony2 se define en una clase cuyo nombre acaba en Test y se encuentra dentro del directorio Tests/ del bundle. Además, se recomienda utilizar dentro de Tests/ la misma estructura de directorios del elemento que se quiere probar.

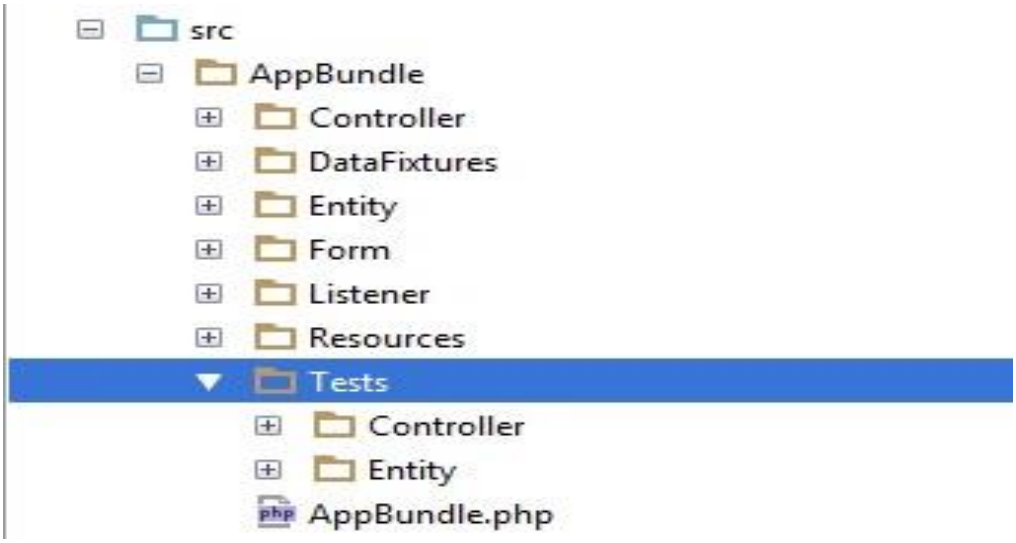


Figura 11. Estructura de directorios para las pruebas en Symfony2

Las clases de las pruebas de PHPUnit siempre heredan de PHPUnit_Framework_TestCase. Su código puede contener tantas propiedades y métodos como se necesite, pero los métodos que se ejecutan en el test siempre se llaman testXXX(). Los test unitarios se basan en comprobar que el código cumple una serie de condiciones, también llamadas aserciones. Además de probar su funcionamiento normal, es muy importante que el test incluya pruebas para todos los casos extremos (números negativos, valores null, etc.).

```php
namespace AppBundle\Tests\Entity;
use AppBundle\Entity\Usuario;
use Symfony\Component\Validator\Validation;

class UsuarioTest extends \PHPUnit_Framework_TestCase
{
    public function setUp()
    {
        $this->validator = Validation::createValidatorBuilder()
            ->enableAnnotationMapping()
            ->getValidator();
    }
    public function testValidarNombre()
    {
        $user = new Usuario();
        $listaErrores = $this->validator->validate($user);
        $this->assertGreaterThan(0, $listaErrores->count(), 'El nombre no debe dejarse en blanco');
        $error = $listaErrores[0];
        $this->assertEquals('This value should not be blank.',$error->getMessage());
    }
```

Figura 12. Fragmento de código de la clase UsuarioTest

Asegurar que la información creada por la aplicación sea correcta es crítico para su buen funcionamiento. Para ello, no basta con añadir reglas de validación a las entidades, sino que es imprescindible comprobar que todas se están cumpliendo correctamente. Las pruebas unitarias automatizan esta tarea.

```
PHPUnit 4.6.6 by Sebastian Bergmann and contributors.

Configuration read from D:\wamp\www\GIT\banpro\app\phpunit.xml.dist

........................................

Time: 2 seconds, Memory: 23.50Mb

OK (40 tests, 80 assertions)
```

Figura 13. Resultado de las puebas unitarias

Se realizaron un total de 40 pruebas a las entidades y a los controladores del sistema, las cuales fueron ejecutadas correctamente en su totalidad.

### 3.4.2 Pruebas de aceptación

Las pruebas de aceptación son una parte integral del desarrollo incremental practicado por XP. Todas las historias de usuario son apoyadas por pruebas de aceptación, que se definen por el cliente. Hacen frente al temor de malinterpretar el negocio. Las pruebas de aceptación son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado. El cliente es la persona adecuada para diseñar las pruebas de aceptación. Sin embargo esto supone el grave problema de que el cliente no tiene que tener, y en general no tiene, la formación adecuada para desarrollar buenas pruebas de aceptación. Por lo cual se debería apoyar al cliente en esta actividad para lograr que cubran, si no todo, si la mayor cantidad posible de las funcionalidades registradas en las historias de usuario ( Gutiérrez, y otros, 2010).

A continuación se muestran algunos ejemplos de las pruebas de aceptación construidas para validar el sistema con el cliente.

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** PA-03 | **Número de HU:** 03 |
| **Nombre:** adicionar problemática | |
| **Descripción:** prueba para la funcionalidad de adicionar problemática | |
| **Entrada/ Pasos de ejecución:** se intenta registrar una nueva problemática con sus datos correspondientes | |
| **Resultado esperado:** se publica la nueva problemática | |
| **Evaluación de la prueba:** prueba satisfactoria | |

Tabla 9. Caso de prueba de aceptación 03

El desarrollo del sistema estuvo dividido en 3 iteraciones, para la primera iteración se realizaron 19 pruebas de aceptación de las cuales 13 se realizaron de forma satisfactoria y

en 6 se detectaron no conformidades, solucionadas posteriormente. Para la segunda iteración se realizaron un total de 11 pruebas de aceptación en las cuales 9 fueron satisfactorias y se detectaron no conformidades en 2 de ellas. En la tercera iteración se realizaron 20 pruebas de aceptación siendo en su totalidad satisfactorias, para un total de 51 pruebas de aceptación.

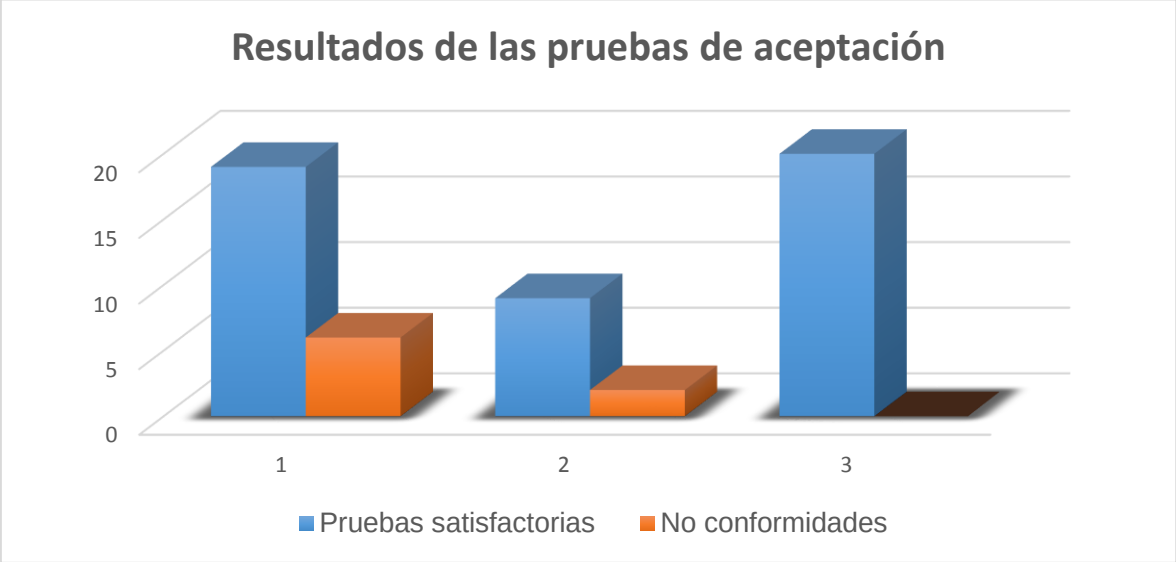


Figura 14. Resultados de las pruebas de aceptación

### 3.4 Experimento

Para la validez de los resultados se precisa disponer de una muestra confiable y representativa de usuarios reales o potenciales de la solución. El conjunto que reúne tales propiedades asciende a una población con tamaño superior a 5000. La determinación de una muestra descarta criterios estadísticos para basarse en el estudio "Number of Participants in User Testing" de Nielsen (2012a), en que se proponen la selección de grupos entre 5 y 15 usuarios para la aplicación de las técnicas. Nielsen pudo demostrar con 83 casos de estudio que una cifra mayor a 15 usuarios en la muestra obtendría similares resultados cuando se aplican técnicas de usabilidad, y que por tanto, en ese rango se reportaban la mayoría de los hallazgos (Figura 15). Partiendo de este aporte y tomando en consideración la naturaleza de las técnicas propuestas se decide que los grupos de prueba de la muestra no tengan un tamaño mayor a 15.

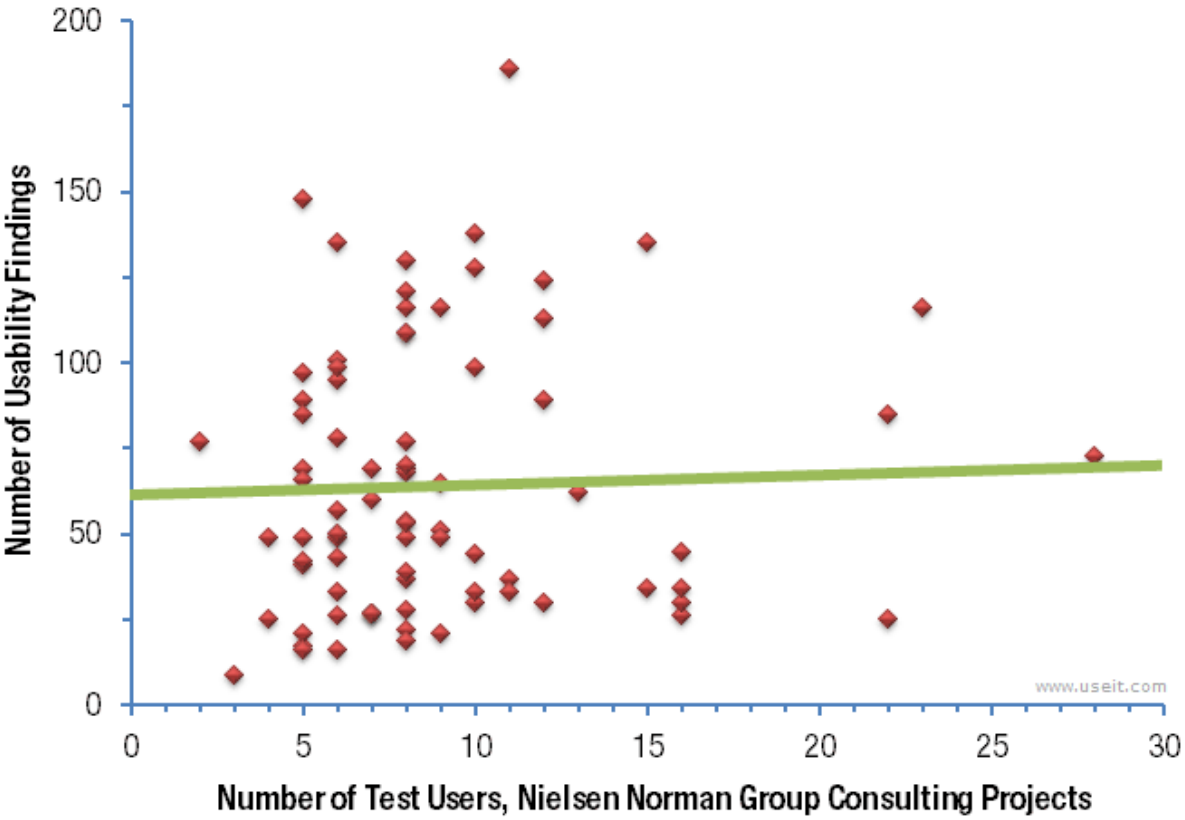


Figura 15. Estudio de Nielsen sobre la cantidad de usuarios a seleccionar para realizar pruebas de usabilidad

### 3.5.1 Caracterización de la población

En el estudio el valor fundamental de la muestra no descansa en su tamaño, sino en su representatividad. Para la eventual generalización de los resultados, la muestra deberá representar la mayoría de las propiedades de la población. El método de selección recurrido fue el aleatorio simple.

Las características de la muestra de la población para la realización de las pruebas fueron las siguientes:

# Capítulo 3. Implementación y pruebas

**Características generales de la población:**

| Variable | Cantidad |
|---|---|
| Estudiantes | 2877 |
| Profesores | 502 |
| Especialistas | 787 |
| Recién graduados | 962 |

Tabla 10. Características de la población.

- 2 grupos de muestra los cuales se dividieron de la siguiente forma:

**Grupo A:**

a) 5 profesores

b) 6 estudiantes

c) 2 especialistas

d) 2 recién graduados

**Criterios de selección:**

**Profesores**

1 dpto. Ciencias básicas

2 dpto. Técnicas de programación

2 dpto. Ciencias sociales y humanidades

**Recién graduados**

1 dpto. Técnicas de programación

1 Centro de desarrollo CEGEL

**Estudiantes**

1 de primero, región occidente

1 de segundo, región occidente

1 de tercero, región central

3 de cuarto, región central

2 de quinto, región oriental

**Especialistas**

1 Occidente, centro de desarrollo CEIGE

1 Centro, centro de desarrollo CEIGE

**Grupo B:**

a) 5 profesores

b) 4 estudiantes

c) 5 especialistas

d) 1 recién graduados

**Criterios de selección:**

**Profesores**

3 dpto. Ingeniería de software

2 dpto. Técnicas de programación

**Estudiantes**

1 de primero, región oriente

1 de tercero, región oriente

1 de cuarto, región central

1 de quinto, región occidente

**Recién graduados**

1 Centro de desarrollo CEIGE

**Especialistas**

2 dpto. Desarrollo de Componentes

2 dpto. Desarrollo de Aplicaciones de Gestión Empresarial

1 dpto. Desarrollo de Aplicaciones de Gestión Financiera y Aduanales

### 3.5.2 Resultados

Teniendo en cuenta la variable del estudio se determinaron los indicadores:

- Membresía a proyectos
- Actividad en la red de investigadores
- Nuevos proyectos

Para evaluar estos indicadores se definen 5 tareas para medir su comportamiento en los grupos seleccionados.

- Creación de problemáticas
- Solicitudes de membresía
- Seguimiento
- Comentarios
- Votos

Se define el **Grupo A** como activo y el **Grupo B** de control, a ambos se le aplicarán las mismas tareas para poder controlar los resultados del experimento. Todas estas tareas tributan al trabajo colaborativo.

**Resultados del estudio en el Grupo A**

Los valores mostrados fueron recopilados durante los 15 días en que se realizó el experimento.

| Días\Tareas | Creación de problemáticas | Solicitudes de membresía | Seguimiento | Comentarios | Votos |
|---|---|---|---|---|---|
| **1** | 1 | 1 | 0 | 3 | 3 |
| **2** | 0 | 2 | 2 | 5 | 2 |
| **3** | 0 | 1 | 3 | 4 | 6 |
| **4** | 0 | 0 | 1 | 1 | 5 |
| **5** | 1 | 2 | 2 | 6 | 2 |
| **6** | 0 | 1 | 5 | 8 | 8 |
| **7** | 1 | 1 | 3 | 5 | 6 |
| **8** | 1 | 2 | 5 | 4 | 6 |
| **9** | 2 | 5 | 4 | 8 | 8 |
| **10** | 0 | 2 | 7 | 9 | 6 |
| **11** | 1 | 2 | 6 | 11 | 7 |
| **12** | 1 | 3 | 10 | 10 | 8 |
| **13** | 2 | 4 | 8 | 11 | 10 |
| **14** | 1 | 5 | 13 | 9 | 12 |
| **15** | 2 | 7 | 16 | 13 | 12 |
| **Total** | **13** | **38** | **85** | **107** | **101** |

Tabla 11. Valores del estudio. Grupo A

El siguiente gráfico ilustra de una mejor manera los resultados de la tabla anterior, mostrando un visible incremento en el tiempo.

# Capítulo 3. Implementación y pruebas

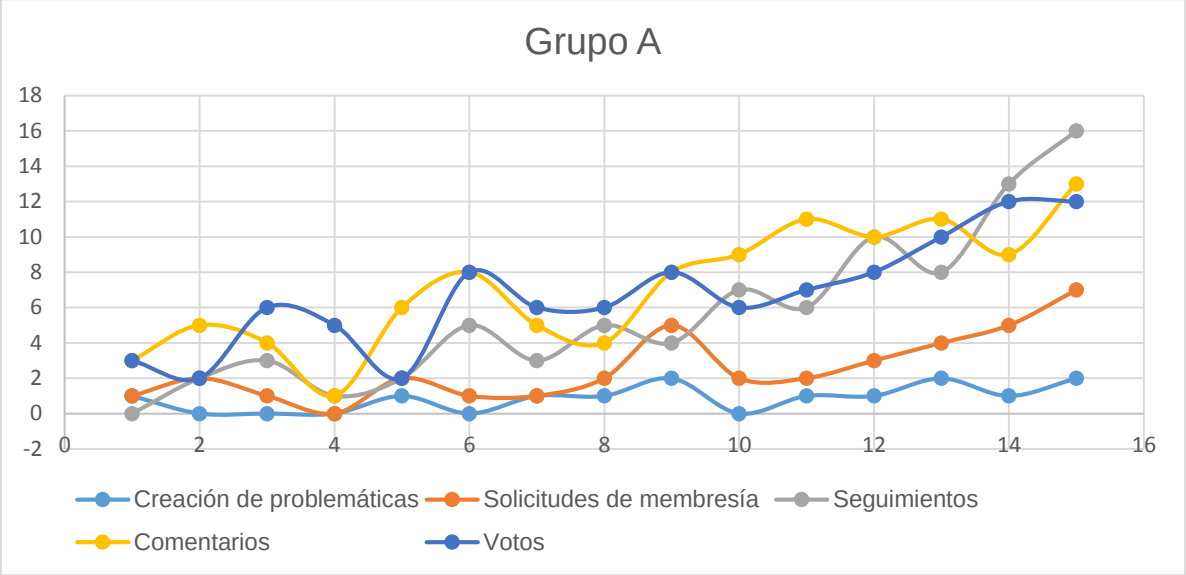


Figura 16. Comportamiento del estudio. Grupo A

**Resultados del estudio en el Grupo B**

Los valores mostrados fueron recopilados durante los 15 días en que se realizó el experimento.

| Días\Tareas | Creación de problemáticas | Solicitudes de membresía | Seguimiento | Comentarios | Votos |
|---|---|---|---|---|---|
| **1** | 0 | 0 | 2 | 1 | 4 |
| **2** | 1 | 1 | 1 | 3 | 1 |
| **3** | 1 | 2 | 2 | 6 | 7 |
| **4** | 2 | 1 | 1 | 2 | 6 |
| **5** | 0 | 2 | 3 | 5 | 3 |
| **6** | 0 | 1 | 4 | 7 | 9 |
| **7** | 1 | 1 | 2 | 6 | 4 |
| **8** | 1 | 2 | 3 | 3 | 4 |
| **9** | 1 | 3 | 5 | 7 | 7 |
| **10** | 1 | 1 | 7 | 10 | 7 |
| **11** | 0 | 2 | 6 | 12 | 4 |

| | | | | | |
|---|---|---|---|---|---|
| **12** | 0 | 5 | 9 | 10 | 6 |
| **13** | 2 | 4 | 9 | 12 | 9 |
| **14** | 2 | 7 | 13 | 11 | 10 |
| **15** | 2 | 8 | 15 | 15 | 15 |
| **Total** | **14** | **40** | **82** | **110** | **96** |

Tabla 12. Valores del estudio. Grupo B

El siguiente gráfico ilustra de una mejor manera los resultados de la tabla anterior, mostrando un visible incremento en el tiempo.

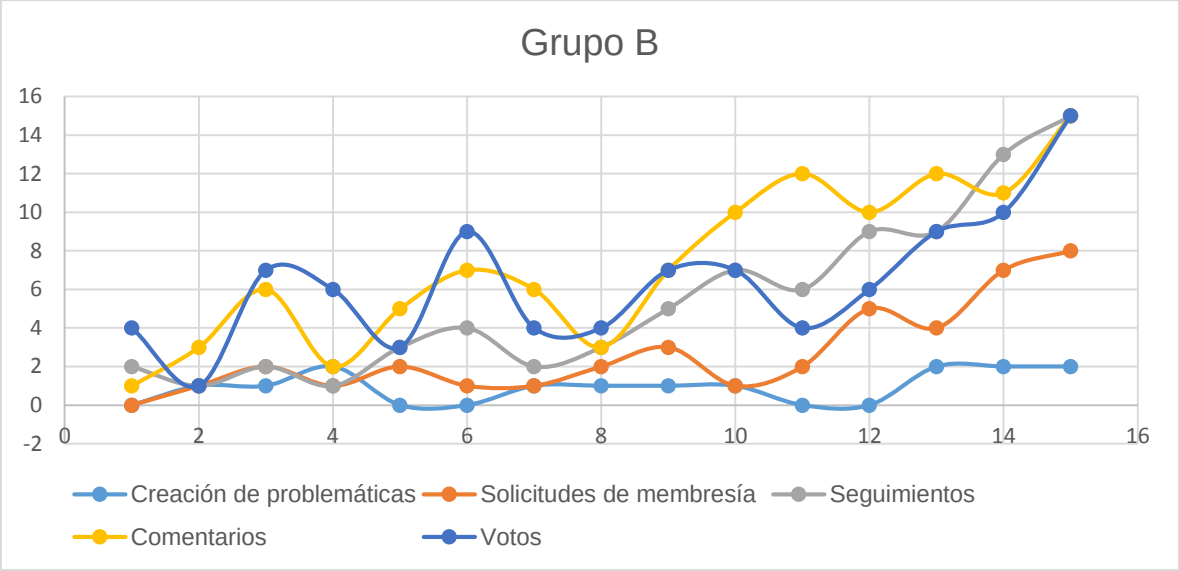


Figura 17. Comportamiento del estudio. Grupo B

**Muestra combinada**

Para comprobar la similitud entre los resultados obtenidos se decide hacer una muestra combinada con los resultados en ambos grupos.

| | **Creación de problemáticas** | **Solicitudes de membresía** | **Seguimiento** | **Comentarios** | **Votos** |
|---|---|---|---|---|---|
| **Grupo A** | 13 | 38 | 85 | 107 | 101 |
| **Grupo B** | 14 | 40 | 82 | 110 | 96 |

Tabla 13. Resumen de los valores del estudio en ambos grupos.

Los valores de la tabla se ilustran en el siguiente gráfico.

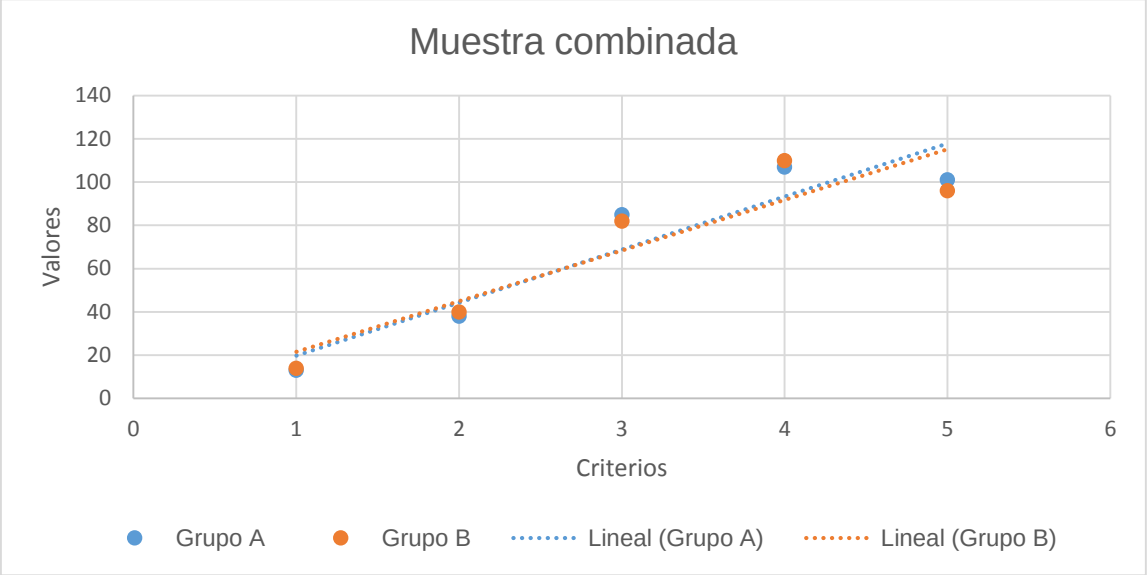


Figura 18. Tendencia de los grupos estudiados

**Conclusiones del experimento**

Teniendo en cuenta que los resultados obtenidos en el Grupo A son similares a los obtenidos en el Grupo B y que en ambos grupos se observa una tendencia ascendente se puede concluir que dadas las tareas que debieron ejecutar los usuarios en cada uno de los grupos la solución está promoviendo el trabajo colaborativo entre los usuarios de la aplicación.

## Conclusiones parciales

En este capítulo se realizaron las actividades correspondientes a las fases de implementación y prueba que propone la metodología seleccionada.

Se elaboró el diagrama de despliegue de la aplicación y se definieron los estándares de implementación a utilizar.

Se realizaron pruebas unitarias y de aceptación al sistema para comprobar su correcto funcionamiento, arrojando resultados satisfactorios.

La validación de la variable trabajo colaborativo demuestra que la solución promueve el trabajo colaborativo entre los usuarios.

# Conclusiones

## CONCLUSIONES

Al finalizar el trabajo de diploma se logró cumplir de manera satisfactoria los objetivos planteados al inicio del mismo, obteniéndose como resultado directo una aplicación web para la gestión de las necesidades de innovación tecnológica de la UCI.

Se realizó un profundo estudio del estado del arte de las principales soluciones informáticas existentes para la gestión de información científica, la cual permitió establecer diferencias y características en común entre ellas.

El estudio indicó como resultado que no es factible adoptar ninguno de los sistemas analizados debido a que no se ajustan a las políticas definidas por la DI de la universidad o no están acorde con la soberanía tecnológica a la que aspira el país.

Se seleccionaron las herramientas y tecnologías más idóneas dentro del marco tecnológico establecido para facilitar la integración con otras aplicaciones que se están desarrollando por la Dirección de Investigaciones de la Universidad.

Se identificaron los RF y RNF del sistema.

Se aplicaron patrones arquitectónicos y de diseño a la solución y se diseñaron los artefactos que dieron paso a la implementación del sistema.

La solución desarrollada fue validada mediante la realización de pruebas unitarias y de aceptación comprobando el cumplimiento de los requerimientos planteados por el cliente.

El desarrollo de la aplicación web para la gestión de las necesidades de innovación tecnológica propuesta elevará el trabajo colaborativo entre especialistas, académicos e investigadores de la UCI.

# Recomendaciones

## RECOMENDACIONES

Utilizar la aplicación web en la Dirección de Investigaciones de la Universidad.

Integrar la aplicación web con el agente inteligente desarrollado para la Dirección de Investigaciones de la Universidad.

Integrar la aplicación web con el sistema para la gestión curricular de los trabajadores de la universidad.

Integrar la aplicación obtenida con el portal web de la Dirección de Investigaciones de la Universidad.

# Referencias Bibliográficas

## REFERENCIAS BIBLIOGRÁFICAS


**Gutiérrez J. J. [y otros]** PRUEBAS DEL SISTEMA EN PROGRAMACIÓN [En línea] // Departamento de lenguajes y sistemas informáticos. - Universidad de Sevilla, 2010. - www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.

**Palomo Iván F., Veloso Carlos G. y Rodolfo** Sistema de Gestión de la Investigación en la Universidad de Talca, Chile [Libro]. - [s.l.] : Revista Información Tecnológica, 2007. - Vol. 18. - 0718-0764.

**ARIAS CHAVES Michael** La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. [Publicación periódica]. - [s.l.] : InterSedes, 2011. - 10 : Vol. VI.

**Barranco Jesús Esteban y Fundación Cotec para la Innovación Tecnológica** Innovación tecnológica. Ideas básicas [Libro]. - Madrid : Gráficas Arias Montano, S.A, 2001.

**Barrere Rodolfo [y otros]** [En línea]. - marzo de 2012. - http://www.observatoriocts.org/files/Archivo%20Documental/Papeles%20del%20Observatorio/InvestigacionyDesarrollo_EDITADO_FINAL_DEFINITIVO.pdf.

**Beck Kent** Extreme Programming Explained [Libro]. - [s.l.] : Addison Wesley., 2004.

**Buschmann Frank** Pattern Oriented Software Architecture, Volume 1: A System of Patterns [Sección de libro]. - [s.l.] : Willey & Sons, 1996.

**Cabrera Francisco Manuel Solís** Las Comunidades Autónomas frente a la I+D+i [Libro]. - [s.l.] : Revista madri+d, 2008.

**Campderrich Falgueras Benet** Ingeniería del software [Libro]. - Barcelona : Universidad Oberta de Catalunya (UOC), 2003.

**Carlos A. Guerrero Johanna M. Suárez* y Luz E. Gutiérrez** Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web [Publicación periódica]. - 2013. - 3 : Vol. 24. - 0718-0764.

**Castro Díaz-Balart Fidel y Delgado Fernández Mercedes** Ciencia, innovación y futuro. [Libro]. - La Habana : Ediciones Huracán, 2001.

# Referencias Bibliográficas

**Curto J.** [En línea] = ¿qué es la gestión de la información?. - 2006. - http://informationmanagement.wordpress.com/2006/09/21/%C2%BFque-es-la-gestion-de-la-informacion-1-de-4/.

**DGP-CIUP** Universidad Pedagógica Nacional de Colombia [En línea]. - 2014. - http://www.pedagogica.edu.co/pgil/pgil.php.

**Dirección General de Producción** Informe de la Infraestructura Productiva [Informe] / Dirección de la Infraestructura Productiva ; Universidad de las Ciencias Informáticas. - Habana : [s.n.], 2011.

**Doctrine 2 ORM** Doctrine 2 ORM documentation [En línea]. - 2015. - http://doctrine-orm.readthedocs.org/en/latest/reference/working-with-objects.html.

**Dra. Escobar Yéndez Nilia Victoria** La innovación tecnológica [Publicación periódica]. - [s.l.] : BVS. Revistas médicas cubanas, 2000. - 4 : Vol. IV.

**Eguiluz Javier** Symfony.es [En línea]. - 2014. - http://symfony.es.

**Fowler Martin** Patterns of Enterprise Application Architecture. [Libro]. - [s.l.] : Addison-Wesley, 2002.

**Fowler Martin** Analysis Patterns: Reusable Object Models [Sección de libro]. - [s.l.] : Adisson Wesley, 1997.

**Fowler Martin** Inversion of Control Containers and the Dependency Injection pattern [En línea]. - 2004. - http://martinfowler.com/articles/injection.html.

**Fowler Michael** Galileo and Einstein [En línea]. - 2009. - 2014. - http://galileoandeinstein.physics.virginia.edu/lectures/LaPrimeraCienciaGriega.htm.

**Gonzáles Carlos D.** [En línea]. - febrero de 2015. - http://www.usabilidadweb.com.ar/postgre.php.

**Graells Dr. Pere Marquès** [En línea]. - 2008. - https://docs.google.com/document/d/1rKWgUcP2MkUfrYAQm1j6pWeuSfan3xCPvEUt4vfxQJE/edit?hl=es.

# Referencias Bibliográficas

**Griffin Caprio** Design Patterns: Dependency Injection. [En línea]. - MSDN Magazine, 2005. - https://msdn.microsoft.com/en-us/magazine/cc163739.aspx.

**Gutierrez Javier J. [y otros]** Pruebas del Sistema en Programación Extrema [En línea]. - 2007. - http://www.researchgate.net/publication/228410687_Pruebas_del_Sistema_en_Programacin_Extrema.

**Historia de la ciencia** Historia de la ciencia [En línea]. - 2014. - 2014. - http://historiadelaciencia.idoneos.com/366706/.

**J. M. Russell [y otros]** Colaboración científica entre países de la región [Publicación periódica]. - [s.l.] : REVISTA ESPAÑOLA DE DOCUMENTACIÓN CIENTÍFICA, 2007. - 178-204. - 0210-0614.

**Machado Machado Armando** Análisis y Diseño de los procesos Administración Compra y Venta para el Sistema de Administración de Relaciones con el Cliente [Informe]. - La habana : [s.n.], 2012.

**Mesa Lic. Yuniet Rojas** De la gestión de información a la gestión del conocimiento [Libro]. - [s.l.] : Acimed: Revista cubana de los profesionales de la información y de la comunicación en salud, 2006. - Vol. 14.

**Microsoft** Microsoft Developer Network [En línea]. - 2014. - https://msdn.microsoft.com/es-es/library/dd381412%28v=vs.108%29.aspx.

**Miller Jeremy** The Unit Of Work Pattern And Persistence Ignorance [En línea]. - MSDN Magazine, 2009. - https://msdn.microsoft.com/en-us/magazine/dd882510.aspx.

**Netbeans** NetBeans [En línea]. - 2014. - https://netbeans.org/index_es.html.

**Ortiz Batista, Yunaysy y Gulín González, Jorge** Repositorio Intitucional UCI [En línea]. - 2013. - http://repositorio_institucional.uci.cu/jspui/handle/ident/8016.

**Pérez María de Lourdes** Ingeniería de Requerimientos. [En línea]. - 2008. - http://dgsa.uaeh.edu.mx:8080/bibliotecadigital/bitstream/231104/415/1/Ingenieria%20de%20requerimientos.pdf.

# Referencias Bibliográficas

**PHP Group** PHP [En línea]. - 2014. - http://www.php.net.

**PHP** PHP Framework Interop Group [En línea]. - 2015. - http://www.php-fig.org/psr/psr-2/es/.

**PHP** PHP Framework Interop Group [En línea]. - 2015. - http://www.php-fig.org/psr/psr-1/.

**PHP** PHP Framework Interop Group [En línea]. - 2015. - http://www.php-fig.org/psr/psr-0/.

**Potencier. Fabien** [En línea]. - otubre de 2011. - http://fabien.potencier.org/article/49/what-is-symfony2.

**Pressman Roger S.** Ingeniería del software: Un enfoque prático [Libro]. - [s.l.] : McGraw-Hill, 2004. - 978-0-07-337597-7.

**PruebaSoft** Gestión de la Calidad y Pruebas de Sotfware. [En línea]. - 2014. - http://pruebasdesoftware.com/laspruebasdesoftware.htm.

**Ramírez Dr. Jorge Ruiz** [En línea]. - marzo de 2010. - enero de 2015. - http://www.scielo.org.ve/scielo.php?script=sci_arttext&pid=S0798-22592010000200001&lng=es&nrm=iso. - 0798-2259.

**San Juan Rosabal Antonio y Rodríguez González Rolando** Sistema para el control de actividades científicas [Libro]. - 2011. - Vol. 3. - 1989-4155.

**Sánchez O.E., Garea B. y Lantigua S.** Sistema de información para la gestión de programas y proyectos de ciencia e innovación tecnológica. [Sección de libro]. - 2008.

**Serradell López Enric y Juan Pérez Ángel A.** Universidad Abierta de Cataluña [En línea]. - 2003. - http://www.uoc.edu/dt/20133/.

**SiCA** [En línea]. - 2014. - https://sica2.cica.es.

**SIGENU-CUJAE** Sistema de Gestión de la Nueva Universidad [En línea]. - 2014. - http://sigenu.mes.edu.cu:8080/dmmes.

**SiUV** Universidad de Valencia [En línea]. - 2014. - http://www.uv.es/siuv/cas/anuncios/aplicacions/gestioninvenstigacion.wiki.

# Referencias Bibliográficas

**Soto Lauro Prof** Mitecnologico.com [En línea]. - Especificaciones de Requerimientos, Ensenada BC,México, 2012. - http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos.

**Sparks Geoffrey y Sparx Systems** Craftware [En línea]. - http://www.craftware.net/es/descargas/modelo_de_componentes.pdf.

**Tabares Ricardo Botero.** Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación. [En línea] // Entre Ciencia e Ingeniería 8 (2011): 161-173.. - 2011. - http://biblioteca.ucp.edu.co.

**UCI** Estrategia de Investigación. [Informe]. - La Habana : UCI, 2014.

**Ugarte Fajardo Ing. Jorge** BPMN: estandar para modelamiento de procesos [Libro]. - 2008.

**UNESCO** UNESCO SCIENCE REPORT [Publicación periódica]. - [s.l.] : United Nations Educational, Scientific and Cultural Organization, 2010. - 978-92-3-104132-7.

**Visual Paradigm** Visual Paradigm [En línea]. - 2014. - http://www.visual-paradigm.com/.

**W3Schools** www.w3schools.com [En línea]. - 2014. - http://www.w3schools.com/js/default.asp.

**Welicki León** Microsoft Developer Network [En línea]. - 2014. - https://msdn.microsoft.com/es-es/library/bb972242.aspx.

## ANEXOS

### Anexo 1. Criterios de selección de los sistemas estudiados

Para la selección de los sistemas informáticos que se analizan se tuvieron en cuenta varios criterios para garantizar su similitud con los intereses de la investigación:

1. Gestión de innovación tecnológica

Cada uno de los sistemas debe gestionar de alguna manera problemas, problemáticas o proyectos en dependencia de cómo se defina en su negocio las necesidades de innovación tecnológica.

2. Contexto universitario

Los sistemas deben haber sido concebidos para el entorno universitario y/o desplegados en este.

3. Ambiente científico

Los sistemas internacionales fueron seleccionados de España y países iberoamericanos como Chille y Colombia, observando una cultura científica cercana a Cuba.

4. Actualidad

Estas aplicaciones informáticas deben ser las más recientes en este tema (no mayor a 10 años de su publicación) y mantenerse en explotación al momento del estudio.