Universidad de las Ciencias Informáticas

Facultad 3

Sistema para la gestión de la información de Investigación y Posgrado del Centro de Gobierno Electrónico

Autor:

Vladimir Medina Miguel

Tutor:

MSc. Isabel González Flores

Cotutor

Ing. Cesar Manuel Cruzata

La Habana, junio del 2016

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos 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 _____.

Vladimir Medina Miguel

_____

Firma del autor

MSc. Isabel González Flores

_____

Firma de la tutora

Ing. Cesar Manuel Cruzata

_____

Firma del cotutor

Agradecimientos

Vladimir Medina Miguel

Para llevar a cabo mi sueño fueron muchas las personas que influyeron a lo largo de la carrera.

Quiero agradecer en primer lugar a mi querida mamá, que es lo más grande y lindo que me ha dado la vida, gracias por ayudarme a llegar hasta aquí y convertirme en el hombre que soy hoy.

A mi familia.

A mi novia.

Mil gracias a mi tutora por su paciencia.

Gracias a Fidel y a la Revolución por brindarme esta oportunidad.

Dedicatoria

Vladimir Medina Miguel

Dedico el presente trabajo de culminación de la carrera:

A mí.

A toda mi familia y amigos.

A mi novia.

A mi otra familia.

RESUMEN

Cada año en la Universidad de las Ciencias Informáticas se identifican los objetivos de trabajo que guían los procesos sustantivos, siendo la investigación uno de ellos. En el Centro de Gobierno Electrónico información del área de Investigación y Posgrado no se encuentra accesible para ser consultada y actualizada por los trabajadores. La forma en la que se gestiona y almacena provoca demoras en el proceso de entrega del reporte o pérdida de calidad del mismo y no permiten la realización de análisis de tendencias. El presente trabajo tiene como propósito desarrollar un sistema informático para la gestión de la información asociada al área de Investigación y Posgrado, a partir de los planes de resultados anuales, que contribuya a elevar el control de la información y la toma de decisiones en el Centro de Gobierno Electrónico. Se utilizó un enfoque de desarrollo ágil, seleccionándose la programación extrema como metodología de desarrollo de software. Las tecnologías y herramientas utilizadas seleccionadas facilitan el desarrollo y garantizan la soberanía tecnológica. Las pruebas realizadas permitieron comprobar el adecuado diseño e implementación de la solución, así como la satisfacción de los usuarios del sistema.

Palabras clave: gestión de información, investigación, plan de resultados.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Cada año en la Universidad de las Ciencias Informáticas (UCI) se identifican los objetivos de trabajo que guían los procesos sustantivos y se desagregan jerárquicamente para su cumplimiento por facultades, departamentos, centros productivos y demás áreas de la universidad. Actualmente se contemplan cinco áreas de resultados clave (ARC) que engloban la educación basada en el sistema de valores de la revolución cubana. Las ARC de la UCI son: profesional competente comprometido con la revolución, colectivo revolucionario de excelencia, impacto económico y social, gestión de la educación superior y desarrollo de aplicaciones y servicios informáticos de calidad.

La investigación es uno de los procesos sustantivos de la Universidad de las Ciencias Informáticas (UCI), donde existe un sistema de ciencia e innovación tecnológica que integra todos los factores, recursos y acciones de la institución en función de dar cumplimiento a la política científica aprobada por su consejo universitario. Estas investigaciones potencian los resultados en la formación, la investigación, la extensión universitaria y la producción de aplicaciones y servicios informáticos. Se ofrecen entrenamientos, cursos de posgrado, diplomados, maestrías y un programa especial de formación de doctores, se promueve la participación en eventos científicos y la publicación en revistas referenciadas. La gestión de la actividad científica de la universidad se refleja en el ARC número tres, está vinculada estrechamente a las demandas de los Organismos de la Administración Central del Estado, generando impactos en lo económico y lo social. Esta área tiene entre sus objetivos incrementar la visibilidad de los resultados de la investigación, alineado a la implementación de los Lineamientos de la Política Económica y Social del Partido y la Revolución.

Los objetivos de trabajo de las facultades de la UCI se desagregan en sus centros productivos y departamentos, y estos últimos en los planes de resultados de sus trabajadores. Las contribuciones que se obtienen a partir de estos planes de resultados conforman la planificación de Ciencia, Tecnología e Innovación (CTI), que posteriormente se medirá mediante el sistema de CTI.

Los resultados de CTI de cada centro productivo y departamento tributan a la facultad a la que pertenecen, los resultados de la universidad se obtienen uniendo los de cada facultad y de las demás áreas. Son revisados y analizados de manera trimestral, con un corte semestral y un cierre anual, para finalmente generar un informe llamado balance de los indicadores de CTI y Posgrado, donde se especifica el estado de cumplimiento de los objetivos de trabajo del año.

El Centro de Gobierno Electrónico presenta dificultades para llevar a cabo el seguimiento y control del cumplimiento del plan de CTI, el plan anual de resultados de los trabajadores y los objetivos estratégicos. Algunas de las deficiencias identificadas son las siguientes:

- La información de Investigación y Posgrado no se encuentra accesible para ser consultada y actualizada por los trabajadores del centro, en la medida en que se van obteniendo los resultados.
- Algunas evidencias enviadas por correo electrónico tienen un tamaño superior a un megabyte y en ocasiones no son recibidas cuando el buzón del asesor de Investigación y Posgrado está lleno o próximo a llenarse.
- Los dispositivos de almacenamiento externo en los que se entregan las evidencias deben ser revisados para detectar la existencia de virus informáticos, la duración de esta tarea varía en función de la capacidad y cantidad de archivos de cada dispositivo.
- Los repositorios utilizados con anterioridad han presentado problemas de seguridad, pues cada persona podía manipular la información de cualquier individuo, así como colocar la evidencia en un espacio inadecuado. Además, las evidencias no se revisaban antes de ser almacenadas.
- El asesor de Investigación y Posgrado utiliza un sistema de carpetas para organizar los tipos de evidencias, pudiendo guardar una evidencia en un espacio que no le corresponde, haciendo difícil su localización posteriormente.
- En el nombre de cada evidencia que se guarda se debe consignar, al menos, el departamento del que procede y el nombre del primer autor de forma estandarizada, con el objetivo de facilitar su identificación. Actualmente estos datos son registrados manualmente por el asesor de Investigación y Posgrado.
- Los reportes de cumplimiento de los objetivos de trabajo del centro y de los planes de resultados, por departamento e individual, no se dan con inmediatez, los datos no se encuentran estandarizados y su procesamiento de forma manual provoca demoras en el proceso de entrega del reporte o pérdida de calidad del mismo por estar sujeto a errores.
- La forma de almacenar las evidencias de CTI del centro no permite la realización de un análisis de tendencias de años anteriores para la toma de decisiones.

Las deficiencias anteriormente mencionadas hacen que este proceso se torne engorroso, lento y poco confiable, teniéndose que verificar manualmente las evidencias de cada persona para poder comprobar el cumplimiento de los planes de resultados y de los objetivos de trabajo del centro. Esta situación afecta la obtención de información con rapidez y precisión, como es el caso de identificar el momento en el que un trabajador cumple o sobre cumple su plan de resultados, las revistas en las que más se han publicado artículos científicos, el evento en el que más se participa, así como el de menor participación.

A raíz de lo expuesto se plantea el siguiente **problema a resolver**: ¿cómo mejorar la gestión de la información asociada al área de Investigación y Posgrado para incrementar el control de la información y contribuir a la toma de decisiones?

El **objeto de estudio** va dirigido al proceso de gestión de la información en el área de Investigación y Posgrado.

Para el desarrollo de la investigación se trazó el siguiente **objetivo general**: desarrollar un sistema informático para la gestión de la información asociada al área de Investigación y Posgrado, a partir de los planes de resultados anuales, que incremente el control de la información y contribuya a la toma de decisiones en el Centro de Gobierno Electrónico.

Estableciéndose como **campo de acción:** informatización del proceso de gestión de la información en el área de Investigación y Posgrado del Centro de Gobierno Electrónico.

Como **objetivos específicos** se identificaron los siguientes:

1. Elaborar el marco teórico referencial relacionado con proceso de gestión de la información en el área de Investigación y Posgrado.
2. Realizar el diseño de la solución para incrementar la gestión de la información en el área de Investigación y Posgrado del Centro de Gobierno Electrónico.
3. Implementar un sistema informático que permita incrementar el control de la información y contribuir a la toma de decisiones en el Centro de Gobierno Electrónico.
4. Verificar la solución propuesta a partir de pruebas y métricas.

Como **posible resultado** de la investigación se espera obtener la primera versión del sistema para la gestión de la información asociada al área de Investigación y Posgrado del Centro de Gobierno Electrónico.

A partir de lo antes expuesto se formula la siguiente **idea a defender**: el desarrollo de un sistema informático para la gestión de la información asociada al área de Investigación y Posgrado, a partir de los planes de resultados anuales, permitirá elevar el control de la información y contribuir a la toma de decisiones en el Centro de Gobierno Electrónico.

Los métodos científicos utilizados en la investigación son los que se presentan a continuación.

Métodos teóricos

Análisis y síntesis: para extraer los elementos más importantes de los documentos analizados, realizando el análisis de las tecnologías, metodologías y herramientas a utilizar en el desarrollo del sistema, de esta manera se eligieron las que contribuyen con el desarrollo de la investigación.

Histórico-lógico: para realizar el estudio de las principales aplicaciones que se han desarrollado para la gestión de la información de Investigación y Posgrado.

Métodos empíricos

Entrevistas: se aplica este método con el objetivo de obtener información sobre la gestión de Investigación y Posgrado en el Centro de Gobierno Electrónico, intercambiando directamente con el personal involucrado en esta actividad.

El presente trabajo de investigación se desglosa en los siguientes capítulos:

Capítulo 1: Fundamentación teórica, se reflejan los conceptos, teorías, métodos y sistemas homólogos existentes necesarios para la comprensión del problema de investigación propuesto. Se realiza el análisis y selección de la metodología de desarrollo de software, herramientas y lenguajes de programación para dar solución al problema de la investigación.

Capítulo 2: Propuesta de solución, se describen los aspectos fundamentales de la solución propuesta, de acuerdo a la metodología de software seleccionada, la arquitectura, los patrones de diseño y el modelo de datos generado.

Capítulo 3: Pruebas de la solución, se presenta la verificación del diseño y la validación de la solución propuesta.

CAPÍTULO 1: Fundamentación teórica

1.1 Introducción

En el presente capítulo se hace referencia a los elementos teóricos resultantes del análisis de la literatura consultada. Se realiza el estudio de los sistemas de gestión de la información del área de Investigación y Posgrado existentes, así como la metodología de desarrollo de software, los lenguajes de programación, tecnologías y herramientas para dale solución al problema de la investigación.

1.2 Sistemas informáticos existentes

En la actualidad existen aplicaciones informáticas dedicadas a la gestión de todo tipo de información. Para lograr una mejor comprensión de las características del sistema a desarrollar se realizó el estudio de sistemas que gestionan información relacionada con los indicadores de CTI.

Sistema de indicadores UCI

Brinda la posibilidad de llevar el control de los indicadores de CTI de las facultades de la UCI, sin embargo, solo pueden consultar la información las personas con determinados niveles de acceso, fundamentalmente los Vicedecanos de Investigación y Postgrado de cada facultad y directivos de investigación de la universidad. El sistema dificulta el balance real de los indicadores de CTI, ya que solo permite el registro del número total de resultados de CTI.

Sistema Informático para la gestión de la formación de postgrado en los profesionales del municipio Mayarí, SICOP

Es un sistema para la gestión de la formación de postgrado en los profesionales del municipio Mayarí (Otero Méndez, 2008), no incluye el control de la participación en eventos científicos, la publicación de artículos científicos, así como otros indicadores de CTI que requiere el Centro de Gobierno Electrónico (CEGEL). Se desarrolló con la herramienta privativa FoxProx versión 8.0 y solo se ejecuta sobre la plataforma de Windows, elementos que no garantizan la soberanía tecnológica por la que apuesta nuestro país.

Sistema de Gestión Universitaria

Es un sistema de planificación empresarial de recursos universitarios desarrollado en la UCI, incluye la concepción de varias líneas de desarrollo agrupadas en las áreas de proceso: pregrado, postgrado, producción, investigación, ingreso, ubicación laboral, laboratorio, residencia, extensión universitaria, cooperación internacional, biblioteca y teleformación. Actualmente el subsistema de Postgrado no cuenta con las funcionalidades que permitan la gestión de todos los indicadores de CTI.

En la tabla 1 se presenta la comparación de los sistemas estudiados teniendo en cuenta algunos indicadores.

Indicadores	Nacional	UCI	
	SICOP	SIndiCIT	Akademos
Gestiona evidencias de CTI	No	No	Si
Gestiona plan de resultados	No	No	No
Guarda evidencias por año	No	No	No
Gestiona indicadores de CTI	No	Si	Si

Tabla 1. Resumen de características de los sistemas existentes a nivel nacional y en la UCI según los indicadores de comparación.

Teniendo en cuenta los indicadores y términos expuestos, ya que son de vital importancia para el desarrollo de la investigación; se concluye que los sistemas estudiados no contribuyen a elevar el control de la información ni ayudar la toma de decisiones en el CEGEL, la presente investigación se opta por el desarrollo de un nuevo sistema informático que dé solución a la situación problemática existente.

1.3 Metodología de desarrollo de software

El desarrollo de software no es una tarea sencilla, prueba de ello son las investigaciones que se realizan con el fin de obtener el éxito en este campo y lograr la calidad del producto que es el principal objetivo de las organizaciones. De ahí que los desarrolladores prestaran más atención a este tema y surgieran nuevas propuestas metodológicas con diferentes resultados en el proceso de desarrollo del software (Torres, 2003).

Las metodologías para el desarrollo de software consisten en un conjunto de procedimientos, técnicas, herramientas y soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Una metodología durante su ciclo de vida indica qué es lo que hay que obtener durante todo el proceso de desarrollo, es decir, cómo se obtienen los productos parciales y finales, pero no cómo hacerlos (Garcia, Felix Oscar Rubio, 2010). Estas metodologías se clasifican en dos grandes grupos: tradicionales y metodologías ágiles.

Las metodologías tradicionales centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto. Presentan altos costes al implementar un cambio y tienen una falta de flexibilidad en proyectos donde el entorno es cambiante. Las metodologías tradicionales se focalizan en la planificación, control del proyecto, especificación de requisitos y en el

modelado (INTECO, 2009). Son eficaces y necesarias para proyectos de gran envergadura que requieren de mucho tiempo y recursos, donde la organización es de vital importancia para su desarrollo. **Las metodologías ágiles** son flexibles, pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto. La comunicación con el cliente es constante al punto de requerir un representante de él durante el desarrollo. Los proyectos son altamente colaborativos y se adaptan mejor a los cambios, al igual que las entregas constantes al cliente y la retroalimentación por parte de él. Tanto el producto como el proceso son mejorados frecuentemente (Navarro Cadavid, y otros, 2013). Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan según se describe en (INTECO, 2009).

A continuación se presenta una tabla comparativa entre estas dos clasificaciones de metodologías desarrollada por (INTECO, 2009)**.**

Criterios	Metodologías ágiles	Metodologías tradicionales
Cambios en los requisitos	Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Control del proyecto	Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
Interacción con el cliente	El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Integrantes	Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Generar artefactos	Pocos artefactos.	Muchos artefactos.
Roles de proyecto	Pocos roles.	Muchos roles.
Arquitectura de software	Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 2. Comparación entre las metodologías agiles y tradicionales.

A partir del estudio realizado y teniendo en cuenta que el sistema a desarrollar no posee un gran número de funcionalidades, que el equipo está compuesto por un desarrollador en constante colaboración con el cliente, se opta por el uso de una metodología ágil.

SCRUM

Es una metodología ágil basada en la teoría de control de procesos empíricos. Emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo. No es un proceso o una técnica para construir productos; en lugar de eso, es un marco de trabajo dentro del cual se pueden emplear varias técnicas y procesos (Schwaber, y otros, 2013). Según (Proyectos agiles, 2015) está

especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Programación extrema

Programación extrema (XP, por sus siglas en inglés) es una metodología ágil centrada en potenciar las relaciones interpersonales como punto clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo (Ingenería de Software, 2016). La misma se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, así como una comunicación fluida entre todos los participantes; también presenta simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (Ingenería de Software, 2016).

El ciclo de vida consta de 6 fases: exploración, planificación de la entrega, iteraciones, producción, mantenimiento y muerte (Calabria, y otros, 2003)**.**

A continuación se presenta una tabla comparativa entre estas dos metodologías ágiles, desarrollada por (Mendes Calo, y otros, 2009).

Criterios	XP	SCRUM
Fechas de entrega	Las iteraciones de entrega son de una a tres semanas.	Las iteraciones de entrega son de dos a cuatro semanas.
Valorar la interacción del equipo por sobre el proceso	Los miembros de XP trabajan en dúos.	Los miembros de SCRUM trabajan de forma independiente.
Valorar la respuesta al cambio	Permite introducir cambios en la iteración en curso.	Permite la evolución y el cambio, pero no es recomendable en la iteración en curso.
Valorar el seguimiento de un plan	Define un plan detallado para cada iteración, que puede ser modificado.	Define un plan detallado de iteraciones, no acepta cambios durante una iteración.
Valorar desarrollo de software que funcione	Generar entregable con prueba satisfactoria e integrada con el resto de las funciones al finalizar cada iteración.	Generar entregable con prueba satisfactoria al finalizar cada iteración.

Tabla 3. Comparación entre las metodologías agiles XP y SCRUM.

Después del análisis de ambas metodologías se decide utilizar la metodología XP para la realización del trabajo por ser la que más se adapta a las características del proyecto. Es una metodología que consiste en una programación rápida o extrema que se identifica por tener como parte del equipo de desarrollo al usuario final, pues es la clave para llegar al éxito en el proyecto. Es capaz de adaptarse a los cambios que puedan surgir en cualquier punto del ciclo de vida del proyecto. XP es ante todo una metodología que busca la satisfacción del cliente para ello le ofrece la posibilidad de mantener un constante intercambio con el equipo de desarrollo, de forma tal que pueda intervenir cuando el producto que se obtenga en cada iteración no sea el deseado.

1.4 Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar instrucciones que pueden ser llevadas a cabo por un ordenador. Puede usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión o como modo de comunicación humana. Permiten especificar de manera precisa sobre qué datos debe operar una computadora, cómo deben ser almacenados o transmitidos y qué acciones debe tomar bajo una gran cantidad de opciones posibles. Todo esto, a través de un lenguaje que intenta ser relativamente próximo al lenguaje humano o natural (Suárez, 2015).

Según (Roque Cuevas, 2007) se denomina lenguaje fuente a las órdenes que escribe el programador, las cuales son traducidas al lenguaje máquina de la computadora. A continuación se presentan los lenguajes de programación que se utilizarán para dar solución al problema de la investigación.

Pre-procesador de hipertexto (PHP)

Es un lenguaje de código abierto muy popular, especialmente adecuado para al desarrollo de aplicaciones web dinámicas y que puede ser incrustado en HTML. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP. Presenta una extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales (PHP, 2015). Según (Suárez, 2015) presenta una mayor capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad (MySQL y PostgreSQL).

Tiene la capacidad de expandir su potencial utilizando una enorme cantidad de módulos. Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda. Es libre, por lo que se presenta como una alternativa de fácil acceso para todos. Se opta por el uso de este lenguaje, en su versión 5.5.9, debido a su potencial para el trabajo en el entorno web.

Lenguaje de marcado de hipertexto (HTML5)

Provee básicamente tres características: estructura, estilo y funcionalidad, es considerado el producto de la combinación de lenguaje de marcado de hipertexto (HTML), hoja de estilo en cascada (CSS) y Javascript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML 5: HTML provee los elementos estructurales, CSS se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista, y Javascript tiene todo el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales (Gauchat, 2012).

En (Miguel Ángel Álvarez, 2009) se explica que incluyen novedades significativas en diversos ámbitos. No sólo se trata de incorporar nuevas etiquetas o eliminar otras, sino que supone mejoras en áreas que hasta ahora quedaban fuera del lenguaje y para las que se necesitaba utilizar otras tecnologías, entre ellas se encuentran:

- Estructura del cuerpo: la mayoría de las webs tienen un formato común, formado por elementos como cabecera, pie y navegadores. HTML 5 permite agrupar todas estas partes de una web en nuevas etiquetas que representarán cada uno de las partes típicas de una página.
- Bases de datos locales: el navegador permite el uso de una base de datos local, con la que se puede trabajar en una página web por medio del cliente y a través de una interfaz de programación de aplicaciones (API), lo que permitirá la creación de aplicaciones web que funcionen sin necesidad de estar conectados a internet.
- Fin de las etiquetas de presentación: todas las etiquetas que tienen que ver con la presentación del documento, es decir, que modifican estilos de la página, serán eliminadas. La responsabilidad de definir el aspecto de una web correrá a cargo únicamente de CSS.

JavaScript

Es un lenguaje de programación que se utiliza principalmente del lado del cliente, implementado como parte de un navegador web. Técnicamente, Java Script es un lenguaje de programación interpretado por lo que no es necesario compilar el código para ejecutarlos. En otras palabras, los programas escritos con Java Script se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Además, está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, estaciones de trabajo y sobre arquitecturas distintas o con sistemas operativos diversos (Eguiluz, 2015).

Según (MDN, 2015) es un lenguaje de script multiplataforma que amplía las ventajas de HTML al trabajar en entornos web, aporta dinamismo y velocidad, permitiendo realizar mejoras en la interfaces de usuario.

Es pequeño y ligero, está diseñado para una fácil incrustación en otros productos y aplicaciones. Es mucho más liberado que el Java ya que no tiene que declarar todas las variables, clases y métodos. No debe preocuparse si sus métodos son públicos, privados o protegidos y no tiene que implementar sus interfaces. Los tipos de variables, parámetros y funciones de retorno no son explícitamente definidos.
Se opta la utilización de este lenguaje debido a las características antes mencionadas, además que es eficiente para la realización de validaciones y en el desarrollo de páginas web dinámicas.

Hojas de estilo en cascada (CSS3)

Es un lenguaje que trabaja junto a HTML para proveer estilos visuales a los elementos del documento como tamaño, color, fondo y borde (Gauchat, 2012). Ofrece nuevas posibilidades para crear impacto con sus diseños. Permite el uso de hojas de estilo más diversas para variedades de ocasiones, entre otras características que harán la aplicación más amigable para los usuarios. La novedad más importante que aporta CSS 3, de cara a los desarrolladores de webs, consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos, que a menudo complicaban el código de los sitios web.
Entre las principales características de CSS 3 se encuentran: bordes redondeados, con degradados, imágenes, cajas con sombra, múltiples columnas, texto con sombra. También incluye mejoras en el uso de la opacidad (pudiéndose fijar a una caja, imagen o texto), selectores y personalizado de fuentes (CSS3, 2015).

1.5 Sistema gestor de base de datos

Un Sistema gestor de base de datos (SGBD), consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos. El objetivo primordial de un SGBD es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer y almacenar información de la base de datos. De igual forma es una aplicación que permite a los usuarios definir, crear y mantener la base de datos, y proporciona acceso controlado a la misma (Fernandez, 2009).

MySQL

Utiliza el Lenguaje de Consulta Estructurado (SQL, dado por sus siglas en inglés), es uno de los SGBD más usado y estandarizados para acceder a bases de datos relacionales. Facilita la integración con programas desarrollados en C y C++ pues fue desarrollado en este lenguaje. Es multiplataforma, totalmente funcional en sistemas operativos como Linux, Mac X, UNIX y Microsoft Windows. Permite la

creación de bases de datos con acceso desde páginas web dinámicas, dando la posibilidad de realizar múltiples y rápidas consultas.

PostgreSQL

Es un sistema de gestión de bases de datos relacional orientada a objetos de software libre, publicado bajo la licencia de distribución de software berkeley (BSD). Según (Stinson, 2001) permite que mientras un proceso escriba en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último que se ejecutó. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos. En (PostgreSQL, 2013) se plantea que utiliza un modelo cliente-servidor y multiproceso en vez de multihilo para garantizar la estabilidad del sistema. Las características más importantes a destacar en la versión 9.3, según (PostgreSQL, 2013):

- Conectores de datos foráneos modificables.
- Sumas de verificación de páginas de datos.
- Failover rápido.
- Transmisión de sólo remasterización.
- Métodos constructores y extractores adicionales para JSON.
- Vistas actualizables automáticas.
- Pg_dump en paralelo para acelerar el respaldo de bases de datos muy grandes.
- Lateral JOINs.
- Procesos en segundo plano definidos por el usuario.

Después de analizar las características y prestaciones de los SGBD mencionados anteriormente, se opta por el uso de PostgreSQL en su versión 9.3, ya que provee de gran capacidad de almacenamiento, consistencia, escalabilidad y rendimiento bajo grandes cargas de trabajo. Es un SGBD objeto-relacional, distribuido bajo la licencia BSD. Además, este presenta una comunidad en la universidad en la cual el desarrollador puede apoyarse, no siendo así para MySQL.

PgAdmin

Se selecciona pgAdmin III como aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y Windows (ArPUG, 2016).

Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características

de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor y un agente para lanzar scripts programados. La conexión al servidor puede hacerse mediante conexión TCP/IP o Unix Domain Sockets (en plataformas Unix), y puede encriptarse mediante SSL para mayor seguridad (Guía Ubuntu, 2016).

Apache

Ha sido seleccionado Apache dado que es un servidor HTTP de código abierto para sistemas operativos modernos. Proporciona un servidor seguro, eficiente y extensible que proporciona servicios HTTP en sincronización con los estándares HTTP actuales. Mejora de almacenamiento en caché y soporta grandes cantidades de información (Apache, 2015). Presenta, entre otras características, mensajes de error altamente configurables, es flexible y de diseño modular, presenta bases de datos de autenticación y negociado de contenido.

1.6 Marco de trabajo

En el desarrollo de software, un marco de trabajo es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado para ayudar a desarrollar y unir los diferentes componentes de un proyecto (Gutiérrez, 2016). A continuación se presentan algunas de las características que se tienen en cuenta para elegir un marco de trabajo.

Persistencia de datos: la mayoría de los marcos de trabajo brindan soporte para el trabajo con múltiples SGBD, principalmente MySQL, PostgreSQL y Oracle. Una condición que potencia el uso del marco de trabajo es precisamente el hecho de que soporte un mayor número de SGBDs y que permita al desarrollador abstraerse del gestor dotándolo de una API consistente y unificada, donde sin modificar el código fuente pueda estar interactuando con cualquier gestor de los que el marco de trabajo soporta. Además, es habitual que los marcos de trabajo contengan herramientas que posibiliten el mapeo de tuplas a objetos, lo que se conoce como ORM *(Object-Relational Mapping).*

Motores de plantillas: un marco de trabajo puede brindar uno o más motores o sistemas de plantilla, que puede ser tan simple como el reemplazo de palabras claves, o un poco más robusto como para generar automáticamente contenido como campos de un formulario. El contenido puede provenir directamente de una base de datos. La sintaxis de la plantilla varía según el marco de trabajo, pero debe diferenciarse del HTML y de las variables a expandir. Los sistemas de plantillas ayudan a separar la lógica de negocio de la de presentación, lo que es siempre considerado una buena práctica.

Gestión de usuarios: a diferencia de los sitios estáticos, en los que cualquier usuario es considerado por igual como anónimo, en las aplicaciones web se requiere restringir el acceso a sus contenidos, es por ello que se considera una característica prominente de los marcos de trabajo que posean cuentas de usuario genéricas y extensibles, de manera que estos puedan registrarse, acceder a los contenidos acorde con su nivel de privilegio y cambiar sus contraseñas de acceso.

Trabajo con caché: para mejorar el rendimiento de un sistema web, los desarrolladores a menudo se auxilian de la caché para almacenar cierto contenido de uso muy frecuente, de modo que este no tenga que generarse ante cada petición de la página. Los marcos de trabajo brindan un mecanismo para almacenar dicho contenido, ya sea en la base de datos o en el sistema de ficheros. El uso de la caché reduce considerablemente el consumo de ancho de banda y la recarga del servidor.

Seguridad: en ocasiones las páginas web se conciben para estar accesibles a usuarios autenticados, de ahí que los entornos de trabajo permitan verificar y solicitar autenticación antes de conceder el acceso a un determinado recurso o contenido, característica que combinan con la gestión de usuarios.

En la actualidad existen gran variedad de marcos de trabajo enfocados en el desarrollo de aplicaciones web. Algunos tienen características similares y ventajas sobre el resto, por ejemplo: Symfony, CodeIgniter, Boson, entre otros. A continuación se exponen las características de estos marcos de trabajo.

CodeIgniter

Entorno de desarrollo abierto que permite crear webs dinámicas con PHP, ayudando a realizar proyectos de forma mucho más rápida, sin tener que crear toda la estructura desde cero. Esto se debe a que dispone de un conjunto bastante amplio de librerías para realizar tareas comunes, así como una interface simple y una estructura lógica sencilla. Es un entorno muy simple. El núcleo del sistema requiere muy pocas librerías para funcionar adecuadamente. Las librerías adicionales que se necesiten se cargan de forma dinámica, con lo cual el sistema es muy sencillo y muy rápido (Codeigniter, 2011). Sin embargo la autenticación se realiza con bibliotecas externas y no tiene ORM.

Boson

En la UCI se desarrolla una arquitectura de referencia nombrada BOSON, la cual garantiza que todas las soluciones que se desarrollen cumplan con la versatilidad necesaria para construir sistemas de distintos dominios y proporcione la capacidad de integración con un mínimo esfuerzo. El marco de trabajo Boson materializa los requisitos comunes de las arquitecturas base, para sistemas de gestión web a partir del desarrollo de varios componentes. Utiliza varios componentes como son:

- Componente caché.
- Componente estructuras de datos.

- Componente excepciones.
- Componente trazas.
- Componente seguridad.
- Componente IUX.
- Componente portal.

Este marco de trabajo se encuentra en desarrollo por lo que, hasta que se libere una versión estable no es factible su utilización.

Symfony

Symfony es un marco de trabajo PHP para el desarrollo de aplicaciones y sitios web rápidos y seguros. Su código, y el de todos los componentes y librerías que incluye se publican bajo licencia MIT de software libre. Cuenta con un sitio online en el cual se puede encontrar una gran cantidad de documentación, también de forma gratuita. Está diseñado para explotar las potencialidades de PHP 5.3, lo cual supone mejoras en el rendimiento del marco de trabajo. Su arquitectura está totalmente desacoplada, esto permite eliminar aquellos componentes que no necesitamos para el desarrollo de nuestro proyecto (symfony.es, 2014). Está diseñado para que se ajuste a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con las librerías de otros fabricantes.

Se elige Symfony 2.6 como marco de trabajo de desarrollo porque reúne las mejores prácticas de desarrollo web e integra muchas librerías y herramientas que aportan fortaleza al producto final. Usa Doctrine para el mapeo de objetos relacionales. Las herramientas de generación de código, la interfaz de línea de comando para la instalación del sistema desarrollado y otras tareas comunes automatizadas, convierten a Symfony en un potente marco de trabajo para PHP. Se tuvo en cuenta también que el autor

de esta investigación se encuentra familiarizado con el marco de trabajo, ya que se utiliza para el desarrollo de soluciones informáticas del centro productivo al que pertenece.

1.7 Mapeo de objetos relacionales

Mapeo de objetos relacionales (ORM) un componente de software que permite trabajar con los datos persistidos como si ellos fueran parte de una base de datos orientada a objetos (tuProgramacion, 2015). Los ORM son elegidos en dependencia del marco de trabajo y del lenguaje de programación que se utilice. En este caso se seleccionó como marco de trabajo Symfony 2.6 y como lenguaje de programación PHP, por lo que se consideraron Doctrine y Propel como ORM a utilizar. Ambos tienen muchas características básicas similares, ya que apoyan todas las operaciones habituales de CRUD (Crear, Modificar, Eliminar y Actualizar), de crear un nuevo registro para actualizar las existentes. Además, ambos pueden generar las clases PHP, Propel basado en XML, mientras Doctrine se basa en YAML. Otra característica que tienen en común es que apoyan diferentes motores de bases de datos, contienen la validación de datos en los modelos, relaciones entre ellos y apoyan a la herencia simple (Doctrine, 2015).

A pesar de ser tan similares Doctrine posee el hecho de ser portable, lo que le permite integrarse fácilmente a cualquier proyecto. También tiene un lenguaje de consulta propio para ayudar a la extracción de los objetos de la base de datos que entiende las relaciones entre clases, por lo que no es necesario escribir los joins (operador empleado en bases de datos para unir información que se encuentra distribuida por varias tablas); además es el ORM que los creadores de Symfony 2 aconsejan a los desarrolladores usar (Eguiluz, 2014) y está definido en la propuesta de desarrollo que tiene CEGEL para la implementación de aplicaciones web.

Por los elementos anteriormente expuestos se decide utilizar el ORM Doctrine en su versión 2.0, para el mapeo de la base de datos de la aplicación web para CEGEL.

1.8 Motor de plantillas

El uso de un motor de plantillas les permite a los desarrolladores separar el código PHP del código HTML, lo cual hace más sencillo el desarrollo de la interfaz del sistema.

Para el desarrollo de la aplicación web, será utilizado Twig pues junto al hecho de ser rápido y eficiente, se le atribuye la condición de haber sido creado por el propio Fabien Potencier, autor y máximo responsable de Symfony, quien recomienda a Twig para el uso del marco de trabajo. Twig se caracteriza fundamentalmente por ser:

- **Rápido:** compila las plantillas hasta código PHP regular optimizado. El costo general en comparación con código PHP regular se ha reducido al mínimo.
- **Seguro:** tiene un modo de recinto de seguridad para evaluar el código de plantilla que no es confiable. Esto permite utilizar Twig como un lenguaje de plantillas para aplicaciones donde los usuarios pueden modificar el diseño de la plantilla.
- **Flexible:** es alimentado por flexibles analizadores léxico y sintáctico. Esto permite al desarrollador definir sus propias etiquetas y filtros personalizados, y crear su propio DSL (Lenguaje Específico del Dominio) (Potencier, 2011).

Para la generación de las vistas, se decidió el uso de Twig en su versión 1.0, puesto que los creadores de Symfony 2 recomiendan su utilización para la creación de las plantillas de la aplicación, además ofrece la herencia entre plantillas, que permite crear una estructura base que contenga todos los elementos comunes del sitio y define los bloques que las plantillas descendientes pueden sustituir, garantizando así organización en el código y facilidad de mantenimiento.

1.9 Marco de trabajo de JavaScript

JQuery es una biblioteca JavaScript rápida, pequeña y rica en funciones. Hace manipulación y recorrido de documentos HTML, control de eventos, efectos y animaciones personalizadas, selección de elementos del Modelo de Objetos del Dominio (por sus siglas en inglés DOM) y manipulación de la hoja de estilos CSS, con una API que funciona a través de muchos navegadores. Brinda varias utilidades como obtener información del navegador, operar con vectores y funciones para rutinas comunes (Foundation, 2013).

Por tanto, el uso de JQuery en su versión 1.8 como librería Javascript, es una aceptada selección pues ofrece una infraestructura que facilita la creación de aplicaciones complejas del lado del cliente. Ayuda en la creación de interfaces de usuario, efectos dinámicos y en el uso de Ajax, permite la abstracción casi total del uso de este. Por otra parte, garantiza la creación de todo tipo de funcionalidades en el lado cliente de manera sencilla, y favorece en el rendimiento por el manejo rápido de propiedades y CSS.

1.10 Marco de trabajo de CSS

Boostrap

Es un marco de trabajo desarrollado para simplificar la creación de diseños web. Combina CCS y JavaScript para lograr una interfaz agradable y vistosa. La mayor ventaja es que se puede crear interfaces que se adapten a los distintos navegadores con el apoyo de un marco de trabajo potente con

numerosos componentes web que ahorrarán mucho esfuerzo y tiempo. Algunas de sus características son:

- Ofrece una serie de plantillas CSS y ficheros JavaScript que facilitan la integración del marco de trabajo de forma sencilla y potente en los proyectos webs.
- Permite crear interfaces que se adapten a los diferentes navegadores, tanto de escritorio como tabletas y móviles a distintas escalas y resoluciones.
- Se integra perfectamente con las principales bibliotecas de JavaScript, por ejemplo, JQuery.
- Ofrece un diseño sólido usando estándares como CSS3/HTML5.
- Es un marco de trabajo ligero que se integra de forma sencilla con el proyecto actual.
- Funciona con todos los navegadores, incluido Internet Explorer usando HTML para que reconozca las etiquetas HTML5.
- Dispone de distintas capas redefinidos con estructuras fijas a 940 píxeles de distintas columnas o diseños fluidos (Borillo, 2012).

Se opta por el uso de Bootstrap en su versión 3.0 como librería CSS, ya que ofrece una elegante tipografía, formas, botones, cuadros y navegación. Además, permite abstraerse de la definición de estilos, contribuyendo de esta manera a la reducción del tiempo de desarrollo.

1.11 Entorno de desarrollo integrado

Un entorno de desarrollo integrado (IDE) es un programa informático compuesto por un conjunto de herramientas para programar en un lenguaje de programación o varios, donde podemos encontrar como mínimo un editor, compilador, interprete y depurador de uno o varios lenguajes de programación. Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones (Regalut, 2012).

NetBeans

Netbeans es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Es un producto libre y gratuito sin restricciones de uso. Es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento.

Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse

para desarrollar sus propias herramientas y soluciones. Netbeans ofrece la posibilidad de desarrollo en otros lenguajes como C/C++ y PHP. Este IDE incorpora además herramientas potentes para el trabajo con XML, HTML, PHP, Groovy, Javadoc, JavaScript, HTML 5, CSS y JSP.

PhpStorm

Cuenta con un editor de código rico para PHP que entiende su código y estructura profundamente, soporta PHP 5.3, 5.4, 5.5 y 5.6 para los proyectos modernos y antiguos. El entorno de desarrollo integrado proporciona finalización de código inteligente, resaltado de sintaxis, la configuración de formato de código extendido, sobre la marcha de la comprobación de errores, plegado de código, apoya las mezclas de idiomas y más.

Posee gran disponibilidad de complementos, transfiriendo diversas características a marcos específicos, como finalización de código, la navegación, tipos de inferencia y otras mejoras prácticas para varios marcos de PHP. Utiliza el mismo entorno de desarrollo integrado en Windows, Mac OS X y Linux con su clave única (JetBrains, 2013). Además, es el IDE preferido por la mayoría de programadores de Symfony. Al margen de todas las características habituales en los entornos de desarrollo, PhPStorm es el único que ofrece una integración casi perfecta con Symfony gracias a su plugin para Symfony 2 (Symfony, 2014).

Después de analizar las características y prestaciones de los IDE mencionados anteriormente, se opta por el uso de PhpStorm en su versión 8.0.2 por las características antes mencionadas, su buen rendimiento en múltiples plataformas y consumo racional del hardware de la computadora. Además, permite el completamiento de código y una mejor integración con depuradores que NetBeans.

1.12 Visual Paradigm

Se opta por utilizar la herramienta Visual Paradigm para la realización del modelo de datos, debido a su alta capacidad de integración con lenguajes de programación. Es una herramienta muy completa, fácil de usar y con soporte multiplataforma.

Su uso es sencillo para la creación de todo tipo de diagramas UML, para los que dispone de un número considerable de estereotipos, permitiendo mayor entendimiento de los mismos (INEI, 2015). Según lo especificado en (Visual Paradigm, 2000) permite visualizar el flujo central detallado de cada proceso mediante diagramas, posibilitando la obtención de los mismos definidos por la metodología escogida. Algunas características que presenta son:

- Navegación intuitiva entre la escritura del código y su visualización.
- Generador de informes en formato PDF/HTML.

- Documentación automática Ad-hoc.
- Generación de bases de datos: transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Interoperabilidad con modelos UML.
- Ingeniería inversa de bases de datos: desde sistemas gestores de bases de datos (DBMS) existentes a diagramas de Entidad-Relación.

1.13 Conclusiones parciales

Del estudio de los sistemas que gestionan información de Investigación y Postgrado se concluye que ninguno satisface las necesidades de CEGEL, es por ello que se propone implementar un sistema que contribuya a elevar el control de la información y la toma de decisiones. Se opta por un enfoque de desarrollo ágil, siendo XP la metodología de desarrollo de software a utilizar porque se ajusta a las características de la investigación.

Las herramientas y tecnologías para el desarrollo de software seleccionadas facilitan el desarrollo del sistema, además son libres y gratuitas. Las métricas y los tipos de pruebas seleccionados permiten identificar deficiencias en el diseño y en la implementación de las funcionalidades del sistema, con el objetivo de solventarlas y obtener un sistema que contribuya a solucionar el problema de la investigación.

CAPÍTULO 2: Propuesta de solución

2.1 Introducción

En este capítulo se describen los aspectos fundamentales de la solución propuesta, de acuerdo a la metodología de software seleccionada, la arquitectura, los patrones de diseño y el modelo de datos. Se presentan las fases de Exploración, Iteraciones y Planificación, propias de la metodología de desarrollo utilizada.

2.2 Fase I: Exploración

Según (Letelier Torres, 2003) los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo 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.

2.2.1 Historia de usuario

Entre los artefactos que define la metodología seleccionada se encuentran las historias de usuario (HU) que son utilizadas para registrar los requerimientos de los clientes, crear las pruebas de aceptación y realizar la estimación de cada una de las iteraciones durante la fase de planificación como se plantea en (Calabria, y otros, 2003). Describen brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales en base a lo que se estima necesario para el sistema. El tratamiento de las HU es muy dinámico y flexible. Cada una es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas, así se asume en (Grupo ISSI, 2003). A continuación se describe la HU encargada de registrar el plan de resultados CTI de CEGEL, las restantes se encuentran en el **¡Error! No se encuentra el origen de la referencia.**.

Historia de usuario	
Número: 1	**Nombre:** Gestionar el plan de CTI del centro
Usuario: administrador	**Iteración asignada:** 1
Prioridad en negocio: alta	**Puntos estimados:** 2
Riesgo en desarrollo: alto	**Puntos reales:** 2
Descripción: se podrán registrar los datos de los indicadores de CTI que debe cumplir el centro en un año determinado. Se debe especificar el año al que se le desea realizar el plan (ejemplo: 2016). Se debe especificar la cantidad de evidencias de CTI por:	

• Resultados introducidos, de tipo: empresa, territorial o ramal, nacional e internacional, especificando el total de resultados que se deben tener en el año. • Publicaciones científicas, del grupo 1, 2, 3, 4, en libros u otras fuentes, especificando el total de publicaciones que se deben tener en el año. • Participaciones en eventos de tipo: municipal, provincial, nacional e internacional, especificando el total de participaciones que se deben tener en el año. • Premios de tipo: UCI, Ministerio de educación superior (MES) u otro Órgano de la administración central del estado (OACE), Academia de ciencias de Cuba (ACC), CITMA, internacional, especificando el total de participaciones que se deben tener en el año. • Registros, se debe especificar la cantidad de registros que se deben tener en el año. • Doctorado, se debe especificar la cantidad de defensas de doctorado que se deben tener en el año. • Maestría, se debe especificar la cantidad de defensas de maestría que se deben tener en el año. • Se debe especificar que el año indicado esté activo. Los valores introducidos del plan del centro se deben mostrar en la pantalla inicial de todos los usuarios, además se deberá identificar, para cada tipo de indicador de CTI, cuál es el valor del plan y cuál es el estado real de cumplimiento.
Observaciones: El plan puede ser modificado en cualquier momento. Los usuarios que no sean administradores del sistema no pueden modificar el plan del centro, solamente pueden visualizar los datos. Solamente puede estar activo el plan del año en curso.

Tabla 4. Gestionar el plan de CTI del centro.

A continuación se explica cada uno de los datos que deben ser llenados en una HU:

- **Número:** identificador de la HU.
- **Nombre:** nombre que identifica a la HU.
- **Usuario:** involucrados en la ejecución de la HU.
- **Iteración asignada:** iteración en que se implementará la HU.
- **Prioridad en el negocio:** prioridad de la HU con respecto al resto de las HU (alta, media o baja)
- **Riesgo en desarrollo:** riesgo en la implementación de la HU (alto, medio o bajo)
- **Puntos estimados:** estima el esfuerzo asociado a la implementación de la HU. Un punto equivale a una semana ideal de programación, generalmente de uno a tres puntos.
- **Puntos reales:** resultado del esfuerzo asociado a la implementación de la HU. Un punto equivale a una semana ideal de programación, generalmente de uno a tres puntos.
- **Descripción:** descripción sintetizada de la HU.

- **Observaciones:** información adicional.

2.2.2 Requisitos no funcionales del sistema

Los requisitos no funcionales detallan las propiedades o cualidades que el producto debe tener, aumentándole funcionalidad al sistema, pues hacen al producto atractivo, fácil de usar, rápido y confiable. A continuación se presentan los requisitos no funcionales definidos:

Requisito de usabilidad: el sistema podrá ser utilizado por usuarios que tengan experiencias básicas en informática y en la gestión de actividades de Investigación y Posgrado. La aplicación web debe poseer un diseño *Responsive*, a fin de garantizar la adecuada visualización en múltiples computadores personales, dispositivos tableta y teléfonos inteligentes.

Requisito de soporte: se mantendrá un sistema de codificación estándar siguiendo las normas establecidas.

Requisitos de interfaz: el sistema debe ofrecer una interfaz amigable y fácil de operar, la tipografía debe ser uniforme, de un tamaño adecuado y con un contraste que resalte los textos. Todas las interfaces deben estar en idioma español.

Requisitos de portabilidad: el sistema debe desarrollarse sin basarse en características propias de algún sistema operativo, para lograr un producto multiplataforma.

Requisitos de disponibilidad: el sistema debe estar disponible las 24 horas del día, los 365 días del año desde todos los puntos donde exista una máquina conectada a la red universitaria.

Requisitos de seguridad:

- **Autenticación:** los usuarios se autenticarán haciendo uso de servicios.
- **Confiabilidad:** la información manejada por el sistema estará protegida de accesos no autorizados.
- **Integridad**: la información que es manejada por el sistema permanecerá inalterada a menos que sea modificada por el personal autorizado.
- **Disponibilidad:** solo tendrá acceso al sistema el personal autorizado.

Ambiente de ejecución:

Estación de trabajo

Software: navegador con soporte para HTML 5 y CSS 3.

Hardware**:**

- 512 MB de memoria RAM.
- Como mínimo un microprocesador a 1.0 GHz.

- Conexión de red.

Servidor

Hardware**:**

- 2 GB de memoria RAM.
- Un procesador Intel Core 2 Duo.
- Disco duro de 500GB.

2.3 Fase II: planificación de la entrega

En esta fase los clientes establecen la prioridad de las HU en correspondencia con las necesidades inmediatas para luego asignarlas, por el orden de relevancia, a las iteraciones planificadas. A partir de las HU 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 HU la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. El tiempo de implementación de una HU generalmente es de uno a tres puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la velocidad de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las HU que fueron terminadas en la última iteración.

Estimación de esfuerzos por HU

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

No	HU	Estimación (semana)
1.	Autenticar usuario	1
2.	Gestionar nomenclador	1
3.	Gestionar el plan de CTI del centro	2
4.	Gestionar el plan de resultados de los trabajadores de cada departamento	2
5.	Gestionar indicador de CTI: evento	1
6.	Gestionar indicador de CTI: publicación	1
7.	Gestionar indicador de CTI: registro	1/2
8.	Gestionar indicador de CTI: resultado introducido	1/2
9.	Gestionar indicador de CTI: maestría	1/2
10.	Gestionar indicador de CTI: doctorado	1/2

11.	Gestionar indicador de CTI: premio	1/2
12.	Visualizar estado de cumplimiento del plan de resultados de cada trabajador	1/2
13.	Visualizar estado de cumplimiento del plan de CTI de cada departamento	1/2
14.	Visualizar estado de cumplimiento del plan de CTI del centro	1/2
15.	Notificar evidencia de CTI registrada	1/2
16.	Revisar evidencia de CTI	1/2
17.	Notificar evidencia de CTI rechazada	1/2
18.	Notificar evidencia de CTI aceptada con modificaciones	1/2
19.	Listar las revistas en las que más se ha publicado en el centro	1/2
20.	Listar las revistas en las que más se ha publicado en el departamento	1/2
21.	Listar los eventos en los que más se ha participado en el centro	1/2
22.	Listar los eventos en los que más se ha participado en el departamento	1/2

Tabla 5. Puntos de estimación por HU.

2.4 Fase III: Iteraciones

Esta fase incluye varias iteraciones del sistema antes de la primera entrega. El calendario es dividido en un número de iteraciones de tal manera que cada iteración tome de una a cuatro semanas de implementación. En la primera iteración se crea un sistema que abarca los aspectos más importantes de la arquitectura global, esto se logra seleccionando las HU que hagan referencia a la construcción de la estructura de todo el sistema. El cliente decide qué HU van a ser implementadas para cada iteración. Además, se realizan las pruebas funcionales, realizados por el cliente, al final de cada iteración. Al final de la última iteración el sistema está listo para ser puesto en producción (Calabria, y otros, 2003).

2.4.1 Plan de iteraciones

En el plan de iteraciones se especifican las HU a implementar en cada iteración del sistema, estableciéndose cuatro iteraciones para la realización del sistema en coordinación con el cliente.

Iteración 1: la iteración tiene como finalidad implementar las HU relacionadas con la autenticación en el sistema y la creación de los nomencladores configurables. Se gestionan el plan de CTI de CEGEL, para un año determinado, así como los planes de resultados de los trabajadores de los departamentos. Se da respuesta a las HU 1, 2, 3 y 4.

Iteración 2: en esta iteración se realizan las HU relacionadas con el registro de las evidencias de CTI y su gestión dentro del sistema. De acuerdo a las evidencias registradas, se actualizan los planes de

resultados individuales, de los departamentos y el del centro. Se da respuesta a las HU 5, 6, 7, 8, 9,10 y 11.

Iteración 3: en esta iteración se realizan las HU usuario relacionadas con la revisión y aprobación de las evidencias de CTI registradas, además se realizan las notificaciones pertinentes. Se da respuesta a las HU 14, 15, 16, 17 y 18.

Iteración 4: en esta iteración se realizan las HU usuario relacionadas con la generación de reportes del sistema. Se da respuesta a las HU 19, 20, 21 y 22.

A modo de resumen se presenta la siguiente tabla que muestra las cuatro iteraciones analizadas previamente con las HU que incluyen y su duración estimada.

Iteraciones	Historias de usuario	Duración
1	Autenticar usuario	Seis semanas
	Gestionar nomenclador	
	Gestionar el plan de CTI del centro	
	Gestionar el plan de resultados de los trabajadores de cada departamento	
2	Gestionar indicador de CTI: evento	Cuatro semanas y media
	Gestionar indicador de CTI: publicación	
	Gestionar indicador de CTI: registro	
	Gestionar indicador de CTI: resultado introducido	
	Gestionar indicador de CTI: maestría	
	Gestionar indicador de CTI: doctorado	
	Gestionar indicador de CTI: premio	
3	Visualizar estado de cumplimiento del plan de resultados de cada trabajador	Tres semanas y media
	Visualizar estado de cumplimiento del plan de CTI de cada departamento	
	Visualizar estado de cumplimiento del plan de CTI del centro	
	Notificar evidencia de CTI registrada	
	Revisar evidencia de CTI	

	Notificar evidencia de CTI rechazada	
	Notificar evidencia de CTI aceptada con modificaciones	
4	Listar las revistas en las que más se ha publicado en el centro	Dos semanas
	Listar las revistas en las que más se ha publicado en el departamento	
	Listar los eventos en los que más se ha participado en el centro	
	Listar los eventos en los que más se ha participado en el departamento	

Tabla 6. Plan de duración de las iteraciones.

2.4.2 Plan de entregas

A continuación se muestra el plan de entregas desarrollado para dar solución al problema planteado, para su desarrollo se tuvieron en cuenta los puntos de estimación.

No	Historia de usuario	Fecha inicio	Fecha fin
1	Autenticar usuario Gestionar nomenclador Gestionar el plan de CTI del centro Gestionar el plan de resultados de los trabajadores de cada departamento	16/02/2016	29/03/2016
2	Gestionar indicador de CTI: evento Gestionar indicador de CTI: publicación Gestionar indicador de CTI: registro Gestionar indicador de CTI: resultado introducido Gestionar indicador de CTI: maestría Gestionar indicador de CTI: doctorado Gestionar indicador de CTI: premio	29/03/2016	29/04/2016
3	Visualizar estado de cumplimiento del plan de resultados de cada trabajador Visualizar estado de cumplimiento del plan de CTI de cada departamento	29/04/2016	22/05/2016

	Visualizar estado de cumplimiento del plan de CTI del centro Notificar evidencia de CTI registrada Revisar evidencia de CTI Notificar evidencia de CTI rechazada Notificar evidencia de CTI aceptada con modificaciones		
4	Listar las revistas en las que más se ha publicado en el centro Listar las revistas en las que más se ha publicado en el departamento Listar los eventos en los que más se ha participado en el centro Listar los eventos en los que más se ha participado en el departamento	22/05/2016	06/06/2016

Tabla 7. Plan de entregas.

2.4.3 Modelo de datos

El modelo de datos es un conjunto de conceptos, reglas y convenciones que permiten describir y manipular los datos de la parcela de un cierto mundo real que deseamos almacenar en la base de datos (Piattini, y otros, 1999). Básicamente el modelo de datos está formado por tres elementos fundamentales que son:

- Objetos (entidades que existen y se manipulan)
- Atributos (características básicas de estos objetos)
- Relaciones (forma en que se enlazan los distintos objetos entre sí)

El modelo de datos correspondiente al sistema consta de 26 tablas, 15 para los nomencladores y 11 para el almacenamiento de la información. Para su construcción se tuvieron en cuenta todos los datos que deben persistir en el sistema.

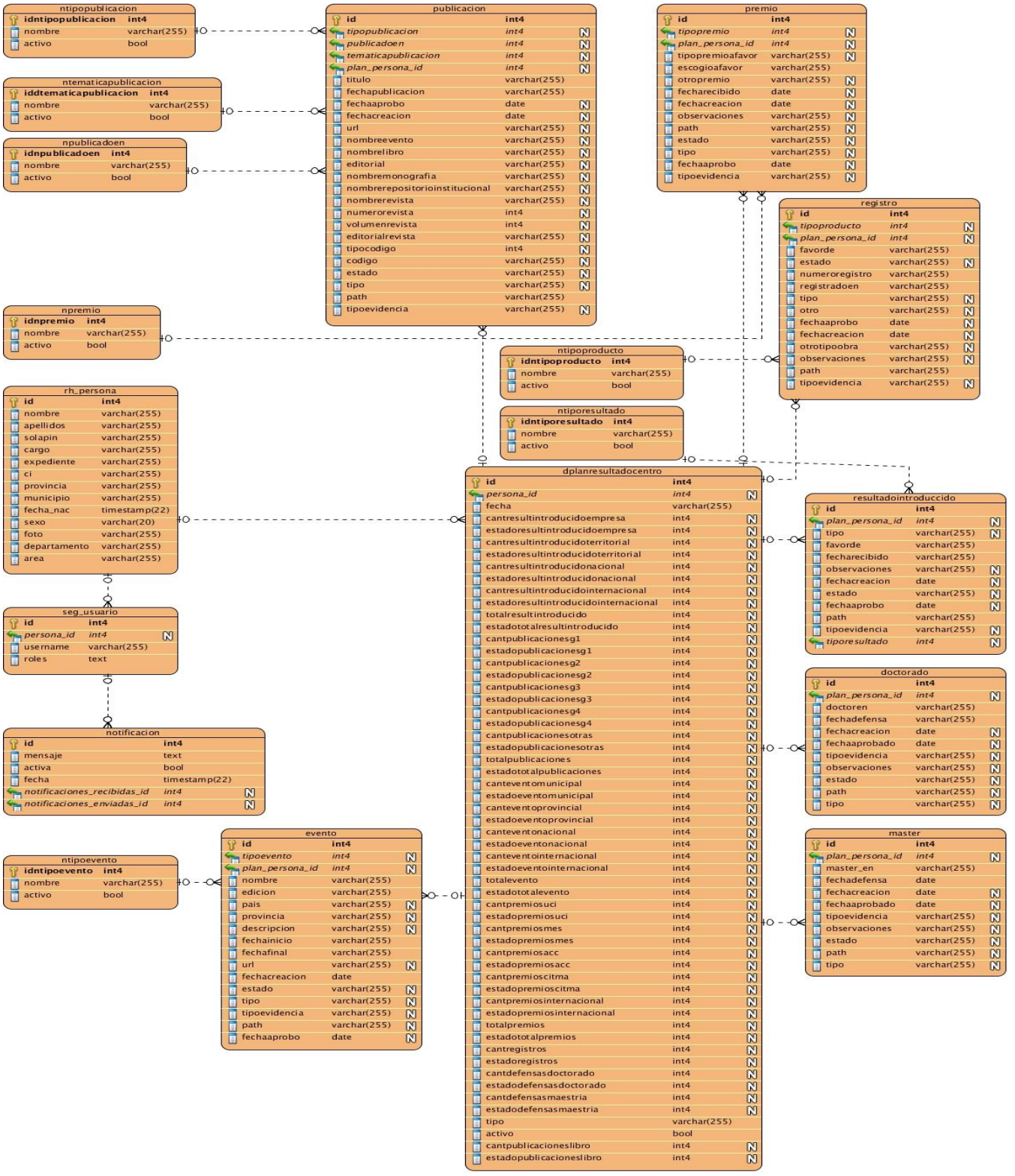

Figura 1. Diagrama Entidad Relación.

[elaboración propia]

2.4.4 Tarjetas Clase - Responsabilidad - Colaborador

Las tarjetas Clase - Responsabilidad - Colaborador (CRC) representan una entidad del sistema, a la cual asignar responsabilidades y colaboraciones. El formato físico de las tarjetas CRC facilita la interacción entre los participantes del proyecto, en sesiones en las que se aplican técnicas de grupos como tormenta de ideas o juego de roles, y se ejecutan escenarios a partir de la especificación de requisitos, HU o casos de uso. De esta forma, van surgiendo las entidades del sistema junto con sus responsabilidades y colaboraciones (Casas, y otros, 2009).

En la metodología XP el proceso de diseño es iterativo. Las tarjetas no se crean en un mismo tiempo, se van desarrollando en la medida que se ejecuten las iteraciones del sistema y se le añaden responsabilidades y colaboradores según sea necesario.

Las tarjetas CRC están compuestas por tres campos:

- **Clase:** nombre de la clase que se está modelando.
- **Responsabilidades:** es una descripción del propósito de la clase.
- **Colaboradores:** los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

A continuación se muestra la tarjeta CRC de la clase DplanresultadoController que se encarga de gestionar los planes de CTI del centro, las restantes se encuentran en el **¡Error! No se encuentra el origen de la referencia.** al final del documento.

Clase DplanresultadoController	
Responsabilidad	**Colaboración**
Gestionar los planes de CTI del centro	Persona Dplanresultado DplanresultadoRepository

Tabla 8. Tarjeta CRC: DplanresultadoController.

2.4.5 Arquitectura de software

Según Pressman, la arquitectura de software es la representación que capacita al ingeniero del Software para: analizar la efectividad del diseño para la consecución de los requisitos fijados, considerar las alternativas arquitectónicas en una etapa en la cual hacer cambios en el diseño es relativamente fácil, y reducir los riesgos asociados a la construcción del software (Pressman, 2010).

Arquitectura en capas

En la arquitectura en capas como organización jerárquica, cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Las

capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas (Garlan, y otros, 1993). El uso de arquitecturas en capas, explícitas o implícitas, es frecuente en la actualidad y entre las principales ventajas que brinda se encuentran:

- Soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
- Admite naturalmente optimizaciones y refinamientos.
- Proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas (Pérez Ávila, 2010).

Para el desarrollo del sistema se define una arquitectura en capas, la cual está compuesta por las capas Vista, Controlador, Modelo y Datos. El uso de esta arquitectura permite dividir el problema a resolver y que cada capa contenga solo las funcionalidades relacionadas con sus tareas, proporcionando la alta reutilización del código.

El marco de trabajo Symfony 2 propone una estructura de paquetes que deja bien definido en qué lugar se encuentran las clases correspondientes a la vista, al modelo y al controlador, como se muestra en la figura 2.

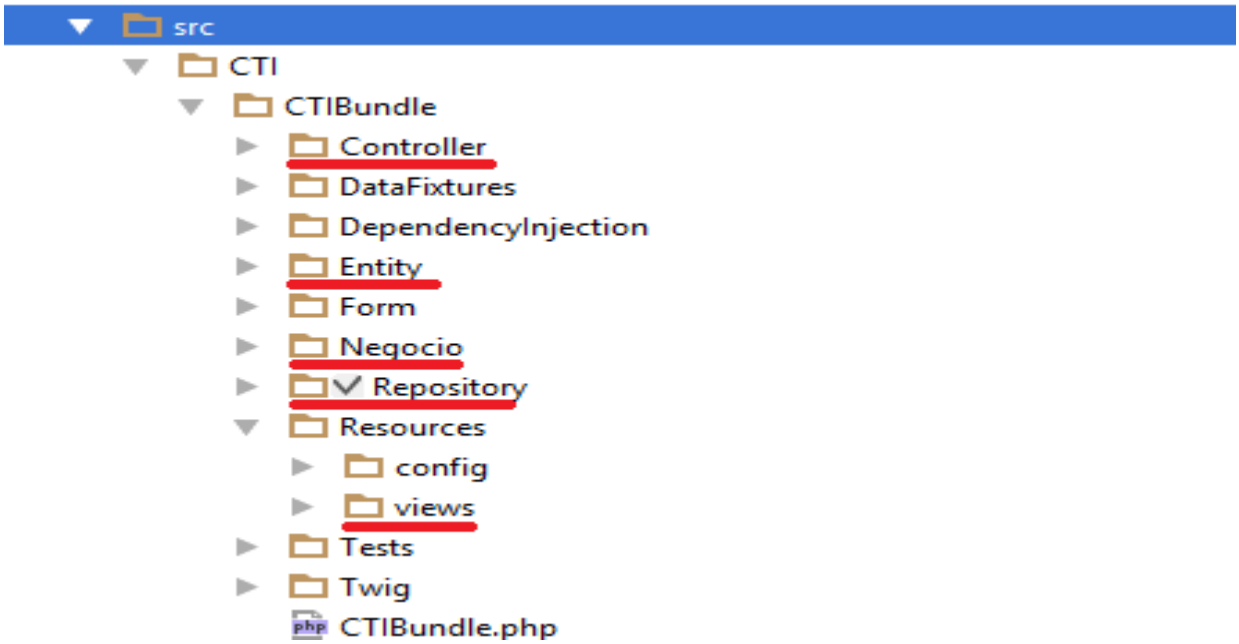

Figura 2. Estructura propuesta por Symfony 2.

[elaboración propia]

En la figura 3 se muestra la arquitectura del sistema y la ubicación de los componentes en cada capa.

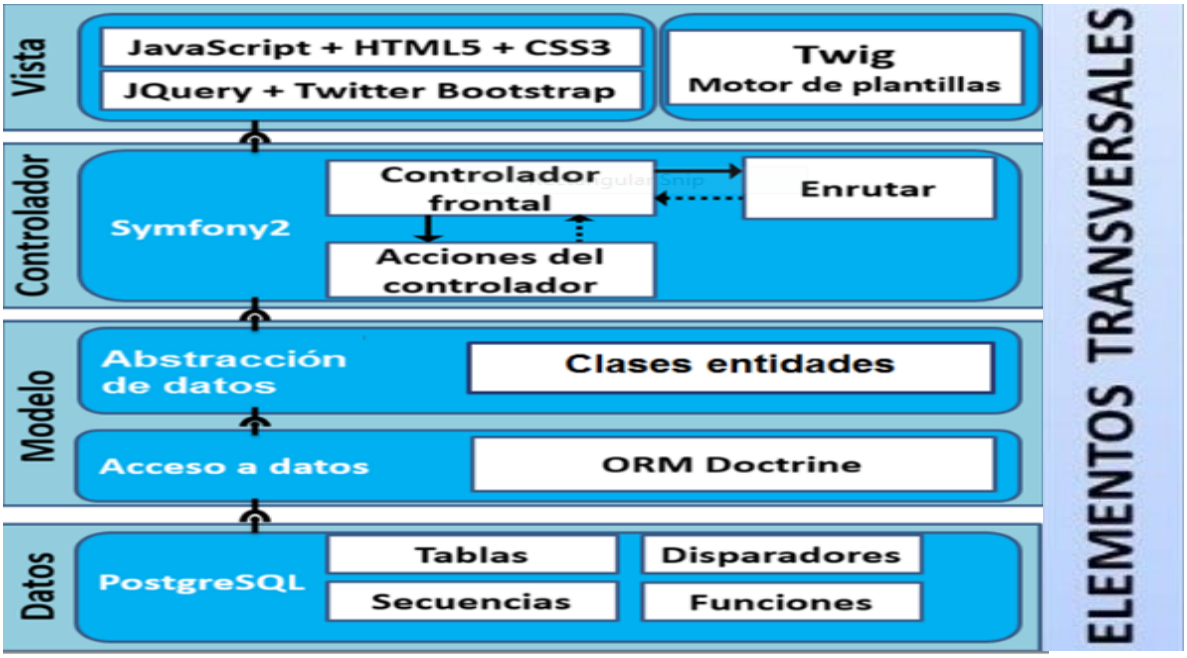

Figura 3. Arquitectura del sistema.

[elaboración propia]

Vista

Se encarga de gestionar los datos de entrada y salida mediante las interfaces de usuario, esta capa contiene los componentes o páginas con los que el usuario interactúa. En ella se visualizan los datos procesados mediante el navegador web utilizando tecnologías como JavaScript, HTML5, CSS3, JQuery y Twitter Bootstrap, así como las plantillas generadas por el motor de plantillas Twig. En la estructura propuesta por Symfony2 las vistas se encuentran en el directorio src\CTI\CTIBundle\Resources\views.

Controlador

Es la capa intermedia entre la vista y el modelo, es la encargada de responder a las acciones del usuario e invocar cambios en el modelo o generar la vista apropiada, dependiendo de las peticiones del usuario. En Symfony 2 esta capa queda evidenciada a través del controlador frontal, mediante los archivos app.php para entornos de producción y app_dev.php para entornos de desarrollo, que se conciben para la ejecución de la aplicación en el servidor de producción y para el uso de los programadores respectivamente, pues les provee de información útil en todas las páginas.

Los archivos de enrutamiento también forman parte de esta capa intermedia los cuales permite determinar qué controlador está asociado con la página que se solicita. En la estructura propuesta por Symfony las clases controladores se encuentran en el directorio src\CTI\CTBundle\Controller.

Modelo

Esta capa está compuesta por las subcapas acceso a datos y abstracción de datos. La subcapa de acceso a datos se encarga del manejo de la lógica de negocio, las clases que componen esta capa son las gestoras y los repositorios, ubicadas en src\CTI\CTBundle\Negocio y src\CTI\CTBundle\ Repository respectivamente. Las clases gestoras reciben la información obtenida de la vista por medio de las controladoras y posteriormente se encargan de tratar dicha información en conjunto con las clases repositorios. Las clases repositorios contienen un conjunto de consultas para acceder y realizar acciones sobre los datos de tablas específicas de la base de datos.

En esta capa también se encuentra el ORM Doctrine que posibilita la separación de la aplicación respecto al sistema gestor de base de datos mediante su lenguaje propio de consultas DQL. La subcapa de abstracción de datos contiene las entidades las cuales son una representación de las tablas de la base de datos mapeadas por el ORM Doctrine, de ahí que constituyen una abstracción de los datos reales almacenados en la base de datos. Las clases entidades se encuentran en src\CTI\CTBundle\Entity.

Datos

En esta capa se encuentra el gestor de base de datos PostgreSQL y en este un conjunto de tablas, secuencias y funciones que permiten persistir la información con la que trabaja la aplicación y gestionar su almacenamiento.

Elementos transversales

Son un conjunto de facilidades y mejoras que permiten que la arquitectura sea más flexible y adaptable. Entre estas mejoras se encuentran el contenedor de servicios de Symfony 2, la gestión de seguridad, tratamiento de excepciones, caché, configuración y validaciones.

2.4.6 Patrones arquitectónicos

Los patrones arquitectónicos son imprescindibles para la construcción y diseño de un sistema de software pues ofrecen la organización estructural de la aplicación.

Modelo-Vista-Controlador

El patrón arquitectónico Modelo Vista Controlador (MVC) divide una aplicación interactiva en tres componentes: el modelo, las vistas y los controladores. Estos dos últimos componentes, en conjunto, forman la interfaz del usuario, asegurando la consistencia entre esta y el modelo.

La figura 4 muestra como Symfony está basado en dicho patrón arquitectónico:

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

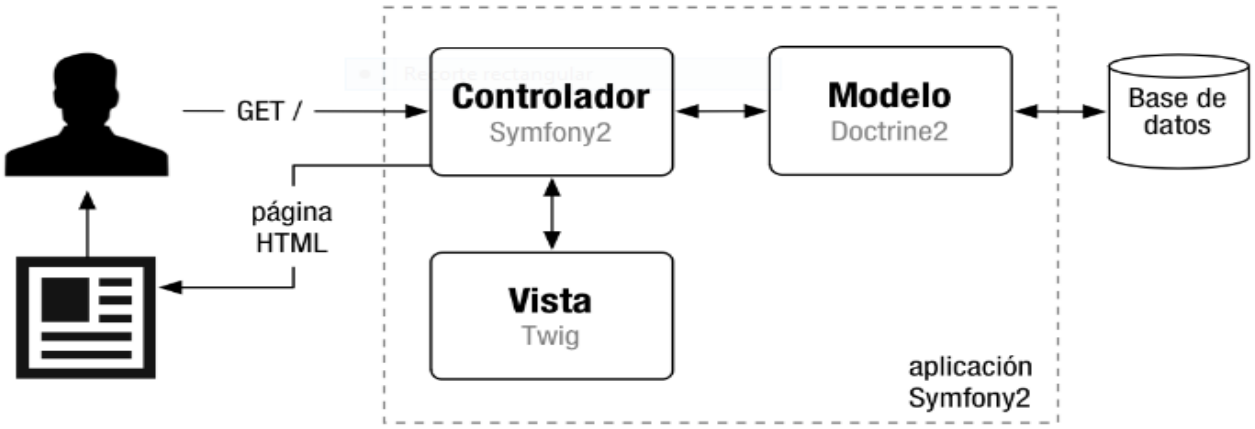

Figura 4. Patrón MVC.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como en un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones. El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación (Eguiluz, 2012).

2.4.7 Patrones de diseño

Los patrones de diseño son herramientas que proveen facilidades para crear un software reutilizable de buena calidad. Cada patrón describe un problema que ocurre repetidamente en nuestro entorno, y describe el núcleo de la solución a ese problema, de tal forma que ésta pueda ser usada un millón de

veces, sin hacer el mismo trabajo dos veces (Asenjo González, y otros, 2003). Los patrones de diseño se encuentran separados en dos grupos: patrones generales de software para asignación de responsabilidades (GRASP) y la banda de los cuatro (GoF). Para el desarrollo de la herramienta se hace uso de patrones de diseño pertenecientes a ambos grupos.

2.4.7.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos. El nombre se eligió para indicar la importancia de captar (grasping) estos principios, si se quiere diseñar eficazmente el software orientado a objetos (Larman, 1999). A continuación se mencionan y describen los que se utilizaron durante el desarrollo de la investigación.

Experto

El patrón experto en información es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento).

Es empleado en todas las clases entidades que contiene el sistema, en la figura 5 se muestra la clase evento, experta en la información de cada evidencia de tipo evento.

Figura 5. Evidencia del patrón experto.

[elaboración propia]

Creador

El patrón creador nos ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto, o usa directamente las instancias creadas del objeto.
Este patrón se evidencia en las clases Controller, que son las responsables de la construcción de los formularios que se visualizarán en las vistas de la aplicación. En la figura 6 se muestra un ejemplo que refleja el patrón creador.

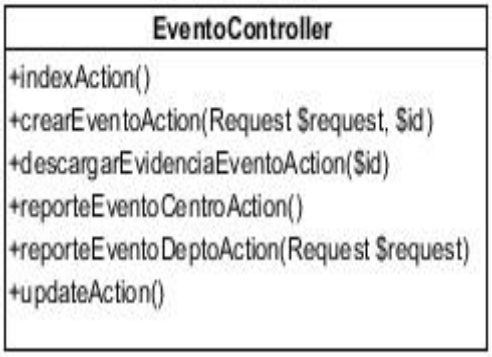

Figura 6. Evidencia del patrón creador.
[elaboración propia]

Controlador

El patrón Controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento.
Se evidencia su uso en las clases controladoras y en el controlador frontal de Symfony 2. Las peticiones realizadas por el usuario son manejadas por el controlador frontal, único punto de entrada a la aplicación en un entorno determinado, que se localiza en el directorio CTI/web/app.php. Además, en todas las clases controladoras está presente y sirve de intermediario entre las interfaces y los algoritmos que las implementan. Las clases controladoras se encuentran en el directorio src\CTI\CTIBundle\Controller.

Alta cohesión

Este patrón determina que la información almacenada en una clase debe ser coherente y estar relacionada con esta, en mayor medida y enfocada en sus responsabilidades. Al realizar un diseño donde las clases del componente mantengan una alta cohesión, como por ejemplo las clases

controladoras, es posible ganar en claridad y facilidad a la hora de entender el diseño, además de simplificar el mantenimiento y soportar mayor capacidad de reutilización.

Se evidencia cuando se desea mostrar el plan de CTI del centro. La acción de mostrar el plan es responsabilidad de DplanresultadocentroController, pero la clase que tiene los datos es Dplanresultadocentro. De esta manera se evidencia la relación que debe existir entre ambas clases, ya que la primera solo se encarga de procesar el plan, utilizando los datos que devuelve la clase Dplanresultadocentro.

Bajo acoplamiento

Consiste en tener las clases lo menos relacionadas entre sí, para que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión en las demás, potenciando la reutilización y disminuyendo la dependencia entre las clases.

Este patrón se evidencia dentro del marco de trabajo Symfony 2 en las clases que implementan la lógica del negocio y de acceso a datos que se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista, lo que proporciona que la dependencia en este caso sea baja. Como existe poca dependencia entre esas clases se permite una mayor reutilización.

2.4.7.2 Patrones GoF

Los patrones GoF describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación clases y la formación de estructuras de mayor complejidad. Nos permiten crear grupos de objetos para ayudarnos a realizar tareas complejas.

Decorador

Patrón de tipo estructura, a nivel de objetos. Añade responsabilidades adicionales a un objeto dinámicamente. Se utiliza para la vista y el layout o plantilla global que decora el contenido de la misma. Este patrón se utiliza de manera implícita en el marco de trabajo Symfony2 con el motor de plantillas Twig, que proporciona la herencia entre plantillas para la creación de una plantilla base que contiene los elementos comunes del sitio web y define los bloques que se van a rediseñar en las plantillas que hereden de esta. Esto proporciona una forma flexible de introducir o eliminar funcionalidades a las plantillas del sistema. En la figura 7 se aprecia una parte de la plantilla crearevento.html.twig que hereda de base.html.twig y a la vez redefine los bloques título y ruta.

```
{% extends '@Seguridad/Default/index.html.twig' %}
    {% block titulo %}
        Evidencias de evento
```

```
    {% endblock %}
    {% block ruta %}
        <div class="breadcrumbs" id="breadcrumbs">
            <ul class="breadcrumb">
                <li>
                    <i class="ace-icon fa fa-home home-icon"></i>
                    <a href="{{ path('index') }}">Inicio</a>
                </li>
                {% if idevento != null %}
                    <li>
                        <a href="{{ path('listadoevidenciasporrevisar')
}}">Listado de evidencias por revisar </a>
                    </li>
                {% endif %}
                <li class="active">
                    Evidencias de evento
                </li>
            </ul>
        </div>
    {% endblock %}
```

Figura 7. Uso del patrón Decorador en la clase crearevento.html.twig.

[elaboración propia]

2.4.7.3 Inyección de dependencias

La Inyección de dependencias consiste en pasar (inyectar) a las clases todos los objetos que necesitan (dependencias) ya creados y configurados (Eguiluz, 2014). Dicho patrón garantiza que se suministren objetos a una clase en lugar de ser la propia clase quien cree el objeto, según las relaciones que fueron establecidas entre las clases.

Este patrón es la clave para la compresión del funcionamiento del marco de trabajo, permitiendo que este sea rápido y flexible, se evidencia en los servicios, que no son más que un objeto PHP que se crea cuando se es necesario utilizar la función a la que hace referencia dicho servicio y en los contenedores de servicios los cuales son objetos PHP que gestionan la creación de instancias de servicios, es decir, objetos. Sin el contenedor de servicios sería necesario crear cada servicio manualmente. El contenedor de servicio se encuentra accesible desde cualquier controlador de symfony2 y a través de estos controladores es posible acceder a cualquier servicio mediante el método get() o a través del objeto container. Como ejemplo de este patrón se tiene la clase WebserviceUserProvider a la cual se asocia la clase GestUsuario que permite registrar un usuario.

```
class WebserviceUserProvider implements UserProviderInterface
{
    public function loadUserByUsername($username)
```

```
    if ($user->Autenticado && $user->Area->IdArea == '196') {
  $usuario = $this->em-getRepository('SeguridadBundle:Usuario')-
>findOneBy(array('username' => $user->Usuario));
 if (is_null($usuario)) {

    return $this->container->get('gestionar_usuario')-
>registrarUsuario($user, $password);
}
```

Figura 8. Uso del patrón Inyección de dependencias en la clase WebserviceUserProvider.

[elaboración propia]

Las ventajas de utilizar un contenedor de servicios es que un servicio solo es creado cuando se necesita y de igual forma es eliminado una vez que se deja de usar, esto trae un ahorro de memoria y un aumento de velocidad de procesamiento.

2.4.8 Tareas de ingeniería

Una vez identificadas las HU los programadores proceden a descomponer cada una en tareas específicas, las denominadas tareas de programación que están escritas técnicamente y que darán solución a la historia correspondiente. En las siguientes tablas se muestran las tareas de ingeniería correspondientes a la HU Gestionar nomenclador.

Tarea de ingeniería	
Número tarea: 3	**Número HU:** 2
Nombre tarea: crear interfaz de gestionar nomenclador	
Tipo tarea: desarrollo	**Puntos estimados:** 6 días
Fecha inicio: 23/02/2016	**Fecha fin:** 29/02/2016
Programador responsable: Vladimir Medina Miguel	
Descripción: esta tarea facilita la creación de una interfaz para Gestionar nomenclador.	

Tabla 9 . Tarea de ingeniería: crear interfaz de gestionar nomenclador.

Tarea de ingeniería	
Número tarea: 4	**Número HU:** 2
Nombre tarea: validar campos de la interfaz gestionar nomenclador	
Tipo Tarea: Desarrollo	**Puntos Estimados:** 1 día
Fecha Inicio: 29/02/2016	**Fecha Fin:** 1/03/2016
Programador Responsable: Vladimir Medina Miguel	
Descripción: Verificar que se introduzcan los datos correspondientes a los nomencladores correctamente.	

Tabla 10. Tarea de ingeniería: validar campos de la interfaz gestionar nomenclador.

2.5 Verificación del diseño

La medición es un aspecto fundamental para cualquier disciplina de la ingeniería. Permite tener una visión de la eficacia del software. Se utilizan medidas para entender mejor los atributos de los modelos que se crean y para valorar la calidad de los productos de ingeniería o de los sistemas que se construyen. En la Ingeniería de Software se obtiene un conjunto de medidas indirectas que dan lugar a métricas que proporcionan una indicación de la calidad de algún producto de software. Las métricas constituyen un método de evaluación de los productos y procesos de software, las cuales suelen ser aplicadas a muchas organizaciones, procesos y productos (Edith, 2008).

Para evaluar la calidad del diseño del sistema se opta por el uso de las métricas orientada a clases: tamaño operacional de clase (TOC) y relaciones entre clases (RC.

TOC

La métrica TOC está dada por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad (CUFM, 2012):

- **Responsabilidad:** un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
- **Complejidad de implementación:** un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- **Reutilización:** un aumento del TOC implica una disminución del grado de reutilización de la clase.

En la siguiente tabla se muestra la aplicación de la métrica TOC, donde CP se refiere a la cantidad de procedimientos.

Atributo de calidad	Categoría	Criterio
Responsabilidad	Baja	CP< =Promedio
	Media	Promedio<=CP<=2* Promedio
	Alta	CP> 2* Promedio
Complejidad de implementación	Baja	CP< =Promedio
	Media	Promedio<=CP<=2* Promedio
	Alta	CP> 2* Promedio
Reutilización	Baja	CP> 2*Promedio
	Media	Promedio<=CP<=2* Promedio
	Alta	CP<= Promedio

Tabla 11. Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC

A continuación se muestran las clases del sistema evaluadas mediante la métrica TOC.

Clase	Cantidad de procedimientos	Responsabilidad	Complejidad de implementación	Reutilización
Dplanresultadocentro	13	Alta	Alta	Baja
Doctorado	5	Media	Media	Media
Evento	6	Media	Media	Media
Master	5	Media	Media	Media
Persona	9	Media	Media	Media
Premio	5	Media	Media	Media
Publicacion	6	Media	Media	Media
Registro	5	Media	Media	Media
Resultadointroducido	5	Media	Media	Media
Usuario	4	Baja	Baja	Alta
Ncategoria	3	Baja	Baja	Alta
Ncursocapacitacion	3	Baja	Baja	Alta
Ncursoentrenamiento	3	Baja	Baja	Alta
Ncursoposgrado	3	Baja	Baja	Alta
Ndiplomado	3	Baja	Baja	Alta
Nfrecuencia	3	Baja	Baja	Alta
Ngradocientifico	3	Baja	Baja	Alta
Nomenclador	6	Media	Media	Media
Notificacion	2	Baja	Baja	Alta
Npremio	3	Baja	Baja	Alta
Npublicadoen	3	Baja	Baja	Alta
Ntematicapublicacion	3	Baja	Baja	Alta
Ntipoevento	3	Baja	Baja	Alta
DplanresultadocentroRepository	3	Baja	Baja	Alta
NomencladorRepository	2	Baja	Baja	Alta
PersonaRepository	5	Media	Media	Media
PublicacionRepository	1	Baja	Baja	Alta
EventoRepository	1	Baja	Baja	Alta

GestUsuario	4	Baja	Baja	Alta

Tabla 12. Evaluación de las clases del sistema mediante la métrica TOC.

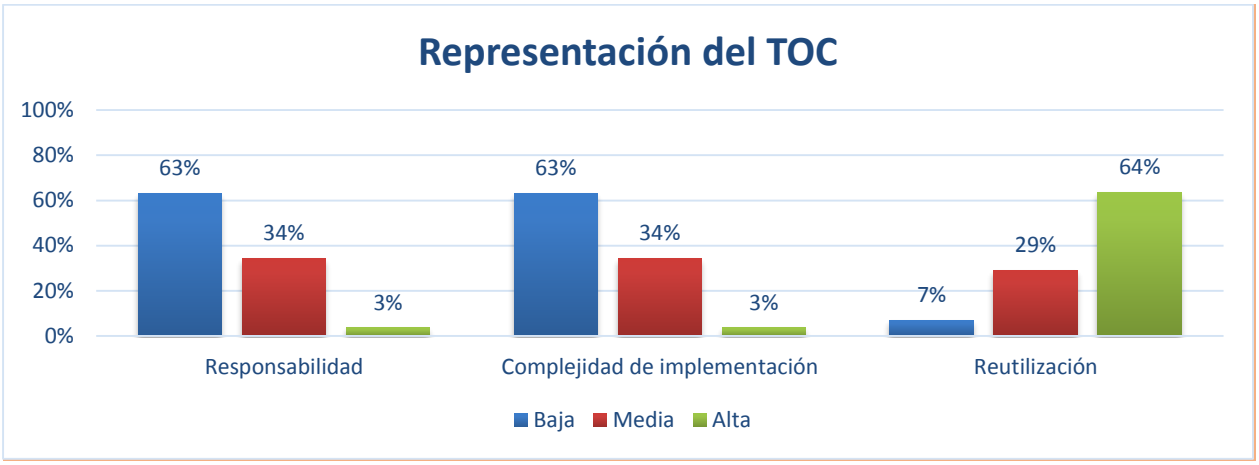

Figura 9. Representación de los resultados de la métrica TOC.

[elaboración propia]

Como resultado de la aplicación de la métrica TOC se evidencia que las clases del sistema poseen una baja responsabilidad, baja complejidad de implementación y alta reutilización, por lo que el diseño de las clases en cuanto a cantidad de funcionalidades es satisfactorio.

RC

La métrica RC está dada por el número de relaciones de uso que tenga una clase con otras, o sea, el número de dependencias que una clase tiene con otra clase y evalúa los siguientes atributos de calidad (CUFM, 2012):

- **Acoplamiento:** un aumento del RC implica un aumento del Acoplamiento de la clase.
- **Complejidad de mantenimiento:** un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
- **Reutilización:** Un aumento del RC implica una disminución en el grado de reutilización de la clase
- **Cantidad de pruebas:** un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

En la siguiente tabla se muestra la aplicación de la métrica RC, donde CRU se refiere a la cantidad de relaciones de uso.

Atributo de calidad	Categoría	Criterio
Acoplamiento	Baja	1
	Media	2

	Alta	>2
Complejidad de mantenimiento	Baja	CRU<= Promedio
	Media	Promedio<=CRU<=2* Promedio
	Alta	CRU> 2* Promedio
Reutilización	Baja	CRU>2* Promedio
	Media	Promedio<=CRU<= 2* Promedio
	Alta	CRU<= Promedio
Cantidad de pruebas	Baja	CRU<= Promedio
	Media	Promedio<=CRU<=2* Promedio
	Alta	CRU> 2* Promedio

Tabla 13. Rangos de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica.

A continuación se muestran las clases del sistema evaluadas mediante la métrica RC.

Clase	**Cantidad de relaciones de uso**	**Acoplamiento**	**Complejidad de mantenimiento**	**Reutilización**	**Cantidad de pruebas**
Dplanresultadocentro	6	Alto	Alta	Baja	Alta
Doctorado	1	Bajo	Baja	Alta	Baja
Evento	2	Medio	Media	Media	Media
Master	1	Bajo	Baja	Alta	Baja
Persona	2	Medio	Media	Media	Media
Premio	2	Medio	Media	Media	Media
Publicacion	4	Alto	Alta	Baja	Alta
Registro	2	Medio	Media	Alta	Media
Resultadointroducido	2	Medio	Media	Media	Media
Usuario	3	Alto	Alta	Baja	Alta
Ncategoria	1	Ninguno	Baja	Alta	Baja
Ncursocapacitacion	1	Ninguno	Baja	Alta	Baja
Ncursoentrenamiento	1	Ninguno	Baja	Alta	Baja
Ncursoposgrado	1	Ninguno	Baja	Alta	Baja
Ndiplomado	1	Ninguno	Baja	Alta	Baja
Nfrecuencia	1	Ninguno	Baja	Alta	Baja

Ngradocientifico	1	Ninguno	Baja	Alta	Baja
Nomenclador	1	Ninguno	Baja	Alta	Baja
Notificacion	2	Medio	Media	Media	Media
Npremio	1	Ninguno	Baja	Alta	Baja
Npublicadoen	1	Ninguno	Baja	Alta	Baja
Ntematicapublicacion	1	Ninguno	Baja	Alta	Baja
Ntipoevento	1	Ninguno	Baja	Alta	Baja
DplanresultadocentroRepository	1	Ninguno	Baja	Alta	Baja
NomencladorRepository	1	Ninguno	Baja	Alta	Baja
PersonaRepository	2	Medio	Media	Media	Media
PublicacionRepository	1	Ninguno	Baja	Alta	Baja
EventoRepository	1	Ninguno	Baja	Alta	Baja
GestUsuario	1	Bajo	Baja	Alta	Baja

Tabla 14. Representación de los resultados de la métrica RC.

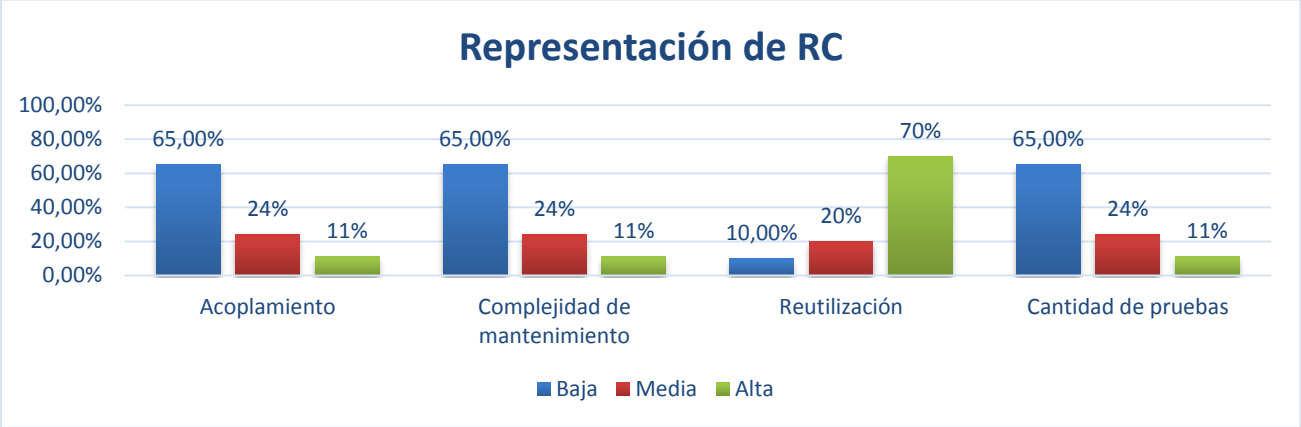

Figura 10. Representación de los resultados de la métrica RC.

[elaboración propia]

Como resultado de la aplicación de la métrica RC se evidencia que las clases del sistema poseen un bajo acoplamiento, baja complejidad de mantenimiento, alta reutilización y una baja cantidad de pruebas, por lo que, de forma general, se valoran como buenos los resultados.

2.6 Estándar de codificación

Los estándares de codificación son pautas que se utilizan en la escritura del código fuente, las mismas aseguran de que todos los programadores del proyecto mantengan un vocabulario común, además permiten tener un código entendible y organizado. Son empleados para asegurar la unificación en el código. Le provee una guía para el encargado del mantenimiento o actualización del sistema, con código claro y bien documentado.

En el desarrollo de la aplicación se decide usar el estándar de codificación CamelCase, específicamente la variante UpperCamelCase para los nombres de las clases y lowerCamelCase para los nombres de los métodos y variables. A continuación se describen las convencciones de nomenclatura.

General

- ✓ Se exceptúan el uso de las tildes y la letra ñ, la que será sustituida por nn.
- ✓ Se usarán nombres claros y libres de ambigüedades.

Identación

- ✓ El código del sistema será identado por tabulaciones en lugar de hacer uso de espacios en blanco.

Clases

- ✓ El nombre de las clases siempre comenzará con mayúscula. En caso de ser una palabra compuesta, cada una de las palabras comenzarán también de la misma forma.
- ✓ Intentar mantener los nombres de las clases descriptivos y simples. Usar palabras completas, evitar acrónimos y abreviaturas.

Nombre de variables y métodos

- ✓ No se utilizarán nombres de variables que puedan ser ambiguos.
- ✓ Los nombres de las variables booleanas deben ser positivos.
- ✓ La primera letra usada para los nombres de las variables y métodos es minúscula, en caso de que el nombre sea compuesto empieza con minúscula y el comienzo de la otra palabra es mayúscula.

2.7 Descripción de la solución

El sistema propuesto constituye una herramienta para la gestión de la información asociada al área de Investigación y Postgrado, a partir de los planes de resultados anuales. Contempla varios tipos de usuarios: administrador, director, subdirector, jefe de departamento y trabajador. El administrador tiene

acceso a todas las funcionalidades del sistema, el director y subdirector pueden ver la información de todos los trabajadores del centro, los jefes de departamento pueden gestionar la información de sus trabajadores, y los trabajadores, al igual que los restantes roles pueden gestionar sus propias evidencias de CTI.

Solamente tendrán acceso al sistema los trabajadores que pertenecen a CEGEL, después de registrarse con el usuario y contraseña que utilizan para acceder a los servicios telemáticos de la UCI, podrán observar la siguiente estructura del sistema.

En la barra superior se muestra a todos los usuarios:

- Logo y nombre del sistema, desde el que se puede acceder a la pantalla inicial.
- El nombre del usuario registrado y la opción de salir del sistema.

El administrador del sistema podrá tener acceso a las opciones:

- Cantidad de evidencias de CTI por revisar.
- Notificaciones del sistema: cuando un trabajador sube una evidencia de CTI el sistema le notifica al administrador del sistema (asesor de Investigación y Posgrado), vía correo electrónico, para que la revise y determine si la acepta, si la acepta con modificaciones o si la rechaza. De acuerdo a la selección realizada se notifica a los trabajadores.

En el menú izquierdo se muestran las opciones de gestión del sistema:

- Planes: se gestiona el plan del centro, de los departamentos y el personal.
- Usuarios: se gestionan los permisos de los usuarios del sistema.
- Nomencladores: se gestionan los nomencladores del sistema.
- Evidencias: permite subir las evidencias de CTI de los trabajadores, así como descargarlas fuera del sistema.
- Revisar evidencia: permite que el administrador del sistema revise los datos de la evidencia subida al sistema, realice las modificaciones que considere necesarias para que sea aprobada.

En el área de trabajo se muestran los formularios asociados a las funcionalidades del menú izquierdo. En la figura 11 se muestra la pantalla inicial del administrador del sistema y en la figura 12 la interfaz relacionada con la gestión del plan de CTI del centro.

Figura 11. Pantalla inicial del administrador del sistema

[elaboración propia]

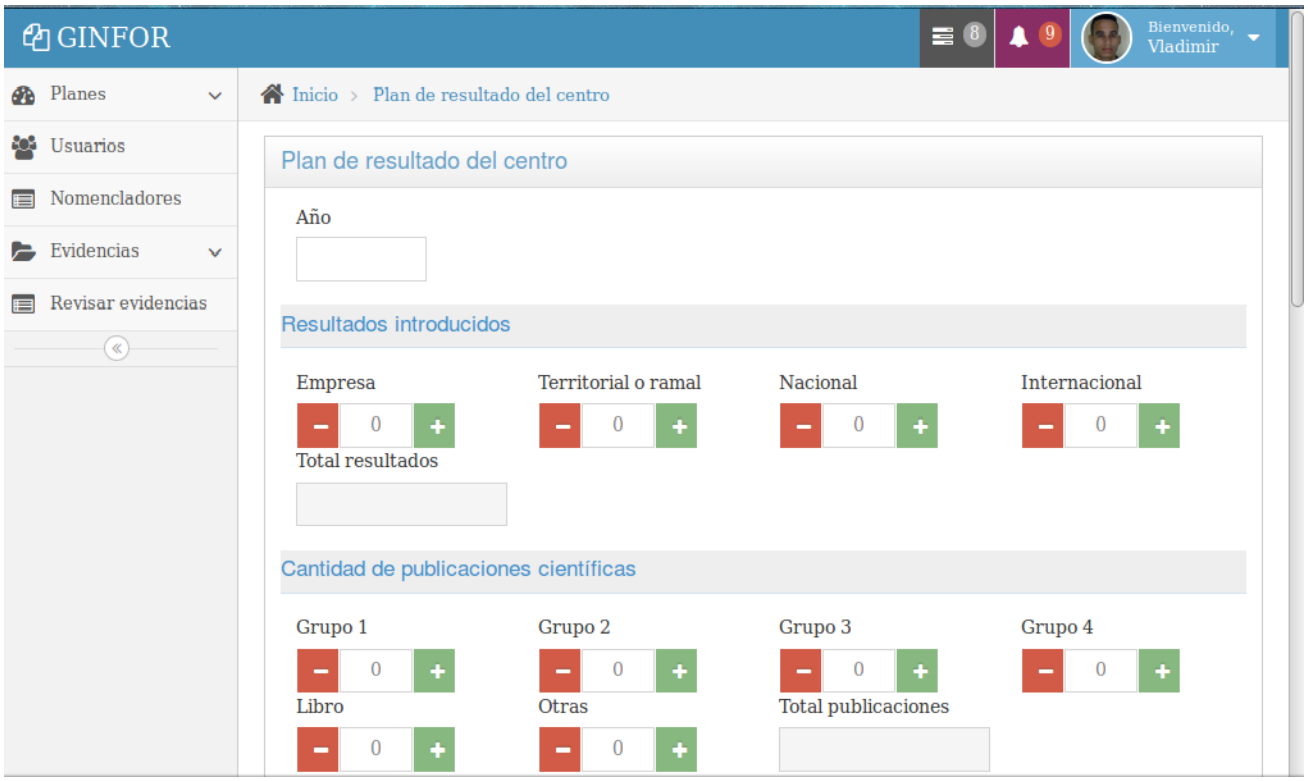

Figura 12. Gestionar el plan de CTI del centro.

[elaboración propia]

2.8 Conclusiones parciales

En este capítulo se describió la propuesta de solución del problema de la investigación. Las HU establecidas por el cliente permitieron identificar las funcionalidades del sistema a desarrollar. La prioridad para el negocio de cada una permitió realizar la estimación del esfuerzo para su desarrollo en cuatro iteraciones. La creación de tarjetas CRC permitieron añadir responsabilidades y colaboración a las clases. La aplicación de los patrones y el uso de un mismo estándar de codificación facilitaron la reutilización del código y una mejora en el nivel de complejidad de las clases. El uso del modelo arquitectónico definido permitió analizar el sistema desde distintos puntos de vista y la aplicación de la metodología XP posibilitó reducir el tiempo de desarrollo estimado ante cambios de último momento.

CAPÍTULO 3: Pruebas de la solución

3.1 Introducción

En este capítulo se realiza la validación de la solución propuesta mediante las pruebas de caja blanca, caja negra y de aceptación, así como el cumplimiento del problema de la investigación.

3.2 Fase IV: Producción

En esta fase el producto se pone en producción y se realizan todas las pruebas al sistema. Este momento es donde se afinan los detalles del sistema debido a que se tiene un gran conocimiento del diseño y, además, se dispone del hardware donde se va a correr el sistema. Durante esta fase se debe ir más despacio a medida que es desarrollado el software. Esto no significa que el desarrollo se detenga pero si es de considerar, que el riesgo se vuelve más importante a medida que los cambios afectan la planificación de entrega del proyecto (Pressman, 2005).

3.2.1 Pruebas de caja blanca

En (Pressman, 2005) se plantea que la prueba de caja blanca del software se basa en un examen cercano al detalle procedimental. Se prueban las rutas lógicas del software y la colaboración entre componentes, al proporcionar casos de pruebas que ejerciten conjuntos específicos de condiciones, bucles o ambos. Al emplear los métodos de prueba de caja blanca el ingeniero del software podrá derivar casos de prueba que:

- Garanticen que todas las rutas independientes dentro del módulo se han ejercido al menos una vez.
- Ejerciten los lados verdadero y falso de todas las decisiones lógicas.
- Ejecuten todos los bucles en sus límites y dentro de sus límites operacionales.
- Ejerciten estructuras de datos internos para asegurar su validez.

Para ejecutar este tipo de pruebas se utilizará la técnica de flujo básico que, según (Pressman, 2005), permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Para aplicar esta prueba se han definido en una serie de pasos a seguir que a continuación se describen:

- **Notación del grafo de flujo:** usando el código como base, se realiza la representación del grafo de flujo (grafo del programa) mediante una sencilla notación. Cada construcción estructurada tiene su correspondiente símbolo: los nodos que representa una o más sentencias

procedimentales, las aristas representan flujo de control y son análogas a las flechas del diagrama de flujo y las regiones, áreas delimitadas por aristas y nodos.

- **Complejidad ciclomática:** es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa y propone que se realice al menos un caso de prueba por cada camino independiente, de manera que se asegure la ejecución de cada sentencia como mínimo una vez. La complejidad ciclomática V (G), se calcula de tres formas:
  - ✓ Número de regiones del grafo de flujo.
  - ✓ V (G) = A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
  - ✓ V (G) = P+ 1, donde P es el número de nodos predicado contenidos en el grafo de flujo G.
- **Determinar un conjunto básico de caminos linealmente independientes:** el valor de V (G) coincide con el número de caminos linealmente independientes de la estructura de control del programa.
- **Obtención de casos de prueba:** se realizan los casos de pruebas que forzarán la ejecución de cada camino del conjunto básico.

A continuación se muestra el resultado de aplicar la técnica del camino básico al método *estado_plan_personalAction*, que se encarga de mostrar el estado de cumplimiento del plan de resultados activo de un trabajador dado.

```
public function estadoPlanPersonalAction($id)
{
    $em = $this->getDoctrine()->getManager();//1
    $cumplimiento_plan = null; //1
    $plan_persona = $em->getRepository('CTIBundle:Dplanresultadocentro')-
>find($id); //1
    $cant_evento_nacional = $plan_persona->getCanteventonacional();
    $estado_evento_nacional = $plan_persona->getEstadoeventonacional();//1
    $cant_evento_internacional = $plan_persona-
>getCanteventointernacional();//1
    $estado_evento_internacional = $plan_persona-
>getEstadoeventointernacional();//1
    $cant_evento_provincial = $plan_persona->getCanteventoprovincial();//1
    $estado_evento_provincial = $plan_persona-
>getEstadoeventoprovincial();//1
    $cant_evento_municipal = $plan_persona->getCanteventomunicipal();//1
    $estado_evento_municipal = $plan_persona->getEstadoeventomunicipal();//1
    if ($estado_evento_nacional >= $cant_evento_nacional &&
      $estado_evento_internacional >= $cant_evento_internacional &&
```

```
        $estado_evento_provincial >= $cant_evento_provincial &&
        $estado_evento_municipal >= $cant_evento_municipal) //2
    {
  if($estado_evento_nacional-1 >= $cant_evento_nacional
||$estado_evento_internacional-1 >= $cant_evento_internacional
||$estado_evento_provincial-1 >= $cant_evento_provincial ||
  $estado_evento_municipal-1 >= $cant_evento_municipal) //3
        {
            $cumplimiento_plan = '2'; //4
        }else {//5
            $cumplimiento_plan = '1'; //6
            return $this-
>render('CTIBundle:Persona:estado_evidencia.html.twig', array(
                'cumplimiento' => $cumplimiento_plan
            )); //6
        }//7
    }//8
    return $this->render('CTIBundle:Persona:estado_evidencia.html.twig',
array(
        'cumplimiento' => $cumplimiento_plan
    )); //9
}

}
```

Figura 13 . Método estadoPlanPersonalAction.

[elaboración propia]

En la figura 14 se muestra el grafo dirigido del flujo asociado al algoritmo *estadoPlanPersonalAction.*

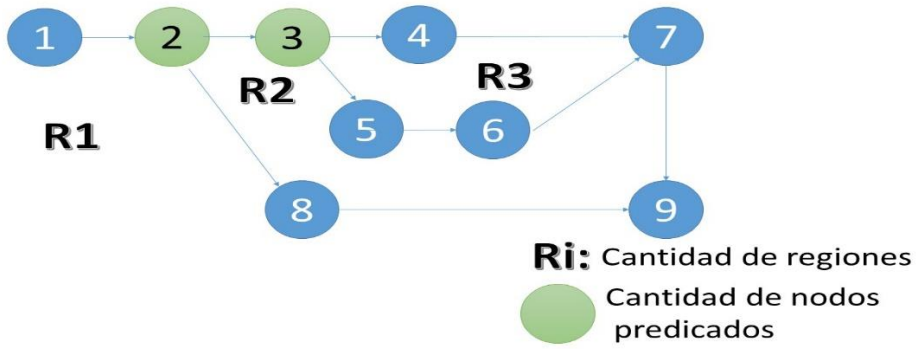

Figura 14. Grafo del flujo.

[elaboración propia]

Tomando como referencia al grafo dirigido del flujo se procede a calcular la complejidad ciclomática empleando las tres variantes propuestas por (Pressman, 2005).

Existen tres regiones.

V (G) = 10 aristas - 9 nodos + 2= 3

V (G) = 2 predicado + 1 = 3

La complejidad ciclomática es igual a tres, lo que significa que existen tres posibles caminos linealmente independientes y representa el mínimo número de casos de prueba para el algoritmo, a continuación se muestran los caminos existentes.

Número	Camino
1	1-2-3-4-7-9
2	1-2-3-5-6-7-9
3	1-2-8-9

Tabla 15. Caminos por donde el flujo puede circular.

El próximo paso es ejecutar los casos de pruebas para cada camino y se compara con los resultados esperados, verificando que las instrucciones se ejecuten por lo menos una vez. A continuación se muestran los casos de prueba para los caminos básicos identificados.

Caso de prueba para el camino básico 1	
Descripción	Comprueba el estado en que se encuentra el plan de resultados de un trabajador.
Condiciones de ejecución	Para esta prueba es necesario que el trabajador tenga cumplido los indicadores del plan y que uno de esos indicadores sea mayor que la cantidad que se le planificó.
Entrada	Id del trabajador.
Resultado	Debe devolver el estado: sobre cumplido.
Resultado de la prueba	Prueba satisfactoria.

Tabla 16. Caso de prueba para el camino básico 1.

Caso de prueba para el camino básico 2	
Descripción	Comprueba el estado en que se encuentra el plan de resultados de un trabajador.
Condiciones de ejecución	Para esta prueba es necesario que el trabajador tenga cumplido los indicadores del plan asignado.
Entrada	Id del trabajador.
Resultado	Debe devolver el estado: cumplido.
Resultado de la prueba	Prueba satisfactoria

Tabla 17. Caso de prueba para el camino básico 2.

Caso de prueba para el camino básico 3	
Descripción	Comprueba el estado en que se encuentra el plan de resultados de un trabajador.

Condiciones de ejecución	Para esta prueba es necesario que el trabajador no tenga cumplido alguno de los indicadores del plan asignado.
Entrada	Id del trabajador.
Resultado	Debe devolver el estado: por cumplir.
Resultado de la prueba	Prueba satisfactoria

Tabla 18. Caso de prueba para el camino básico 3.

Después de ejecutar los casos de prueba diseñados se comprobó que cada sentencia del código se ejecuta al menos una vez, teniendo en cuenta todas las condiciones lógicas en sus variantes verdaderas y falsas, además los resultados fueron satisfactorios.

3.2.2 Prueba de caja negra

Las pruebas de caja negra se concentran en las HU del sistema, permiten al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitarán por completo todas las HU de un programa. Según (Pressman, 2005) tratan de encontrar errores en las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en la estructura de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Según (Pressman, 2005) para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

- **Técnica de la partición de equivalencia:** divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **Técnica del análisis de valores límites:** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **Técnica de grafos de causa-efecto:** permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Según se establece en (Pressman, 2005) dentro del método de caja negra la técnica de la partición de equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. Se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de clases de prueba

que hay que desarrollar. A continuación se presentan algunos ejemplos de casos de prueba desarrollados utilizando la técnica de partición equivalente.

Escenario	Descripción	Doctor en	Fecha	Observaciones	Respuesta del sistema	Flujo central
EC 1.1 Insertar datos del indicador correctamente	Se introduce los datos correctamente para crear la evidencia de doctorado.	V	V	NA	Se inserta una nueva evidencia de doctorado en el sistema.	Se selecciona la opción Doctorado en la página principal. Se llenan y seleccionan todos los campos. Se selecciona la opción Aceptar.
EC 1.2 Dejar campos vacíos	Se intenta insertar una evidencia dejando campos vacíos.	NA	NA	NA	Muestra un mensaje de error sobre los campos que se dejaron vacíos.	Se selecciona la opción Doctorado en la página principal. Se llenan y seleccionan todos los campos. Se selecciona la opción Aceptar.

						Se muestran mensajes de error.

Tabla 19. Caso de prueba para la HU Gestionar indicador de CTI: doctorado.

A continuación se presentan los casos de prueba correspondientes a la HU Gestionar indicador de CTI: maestría.

Escenario	Descripción	Máster en	Fecha	Observaciones	Respuesta del sistema	Flujo central
EC 1.1 Insertar datos del indicador correctamente	Se introduce los datos correctamente para crear la evidencia de doctorado.	V	V	NA	Se inserta una nueva evidencia de maestría en el sistema.	Se selecciona la opción Máster en la página principal. Se llenan y seleccionan todos los campos. Se selecciona la opción Aceptar.
EC 1.2 Dejar campos vacíos	Se intenta insertar una evidencia dejando campos vacíos.	NA	NA	NA	Muestra un mensaje de error sobre los campos que se dejaron vacíos.	Se selecciona la opción Máster en la página principal. Se llenan y seleccionan todos los campos.

						Se selecciona la opción Aceptar. Se muestran mensajes de error.

Tabla 20. Caso de prueba para la HU Gestionar indicador de CTI: doctorado.

Las pruebas de caja negra fueron ejecutadas por un especialista del grupo de calidad de CEGEL en dos iteraciones. En la primera iteración se detectaron un total de 16 no conformidades que se corresponden a errores de idioma, validación de los datos de entradas y funcionalidades ausentes e incorrectas, como se muestran gráficamente en la figura 16. El 33 % representa los errores de idioma, el 45 % los de validación de los datos de entrada, el 22 % los de funcionalidades ausentes e incorrectas.

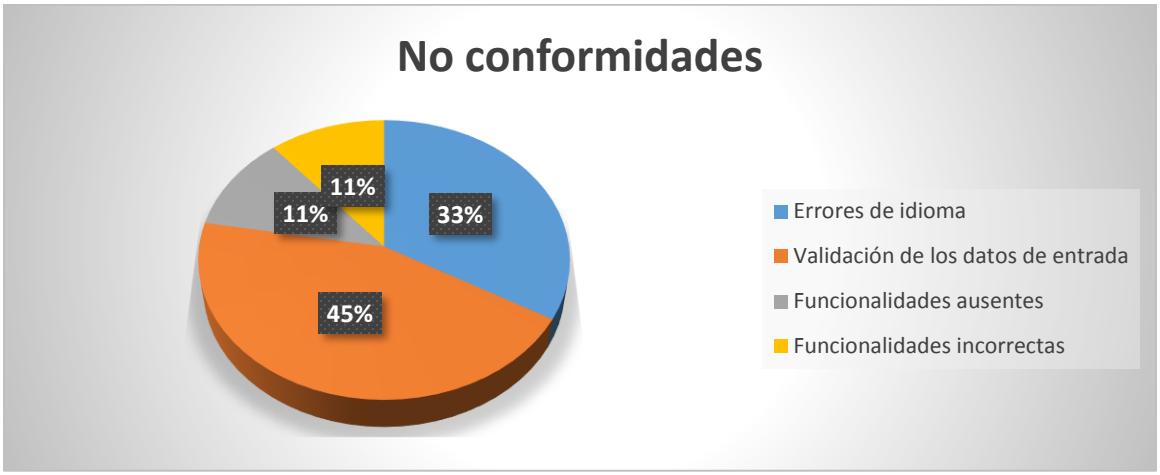

Figura 15. Gráfica de no conformidades de las pruebas funcionales.

[elaboración propia]

Las no conformidades fueron corregidas y en la segunda iteración no se detectaron nuevas no conformidades, emitiéndose el acta de liberación que se muestra en el **¡Error! No se encuentra el origen de la referencia.**.

3.2.3 Pruebas de aceptación

Este tipo de pruebas se desprenden directamente de las especificaciones de los requerimientos del usuario, verificando que el software realiza lo especificado. Se describe un escenario de ejecución o uso

del sistema desde la perspectiva del usuario, teniendo en cuenta requisitos funcionales o no funcionales. Cada uno de los requisitos puede tener una o más pruebas de aceptación (Díaz, y otros, 2009).

A continuación se muestran los casos de pruebas de aceptación diseñados para las HU de Gestionar el plan de CTI del centro y Gestionar el plan de resultados de los trabajadores de cada departamento debido a que son dos funcionalidades básicas del sistema.

Caso de prueba de aceptación	
Código: CPA-02	**No. Historia de usuario: 3**
Nombre: Gestionar el plan de CTI del centro.	
Descripción: el usuario debe crear un plan de CTI para el centro para un año determinado.	
Condiciones de ejecución: el usuario debe estar autenticado en el sistema y ser director del centro o administrador del sistema.	
Entrada/Pasos de ejecución: el usuario debe seleccionar el año del plan de CTI a crear y registrar los datos de los indicadores a cumplir.	
Resultado esperado: se registra el plan de CTI del centro y se muestra en la pantalla inicial el plan de CTI creado.	
Evaluación de la prueba: prueba satisfactoria.	

Tabla 21. Caso de prueba de aceptación para la HU Gestionar el plan de CTI del centro

Caso de prueba de aceptación	
Código: CPA-02	**No. Historia de usuario: 4**
Nombre: Gestionar el plan de resultados de los trabajadores de cada departamento.	
Descripción: el usuario debe crear el plan de resultados de todos los trabajadores de su departamento.	
Condiciones de ejecución: el usuario debe estar autenticado en el sistema y ser jefe de departamento o administrador del sistema.	
Entrada/Pasos de ejecución: el usuario selecciona los trabajadores y llenará los datos del plan que será asignado a los trabajadores escogidos.	
Resultado esperado: se registra el plan de CTI de cada trabajador del departamento, este plan puede ser modificado, visualizado y eliminado.	
Evaluación de la prueba: prueba satisfactoria.	

Tabla 22. Caso de prueba de aceptación para la HU Gestionar el plan de resultados de los trabajadores de cada departamento.

El director de CEGEL y la asesora de Investigación y Posgrado realizaron las pruebas de aceptación del sistema, teniendo en cuenta los casos de prueba definidos. Los resultados obtenidos fueron satisfactorios.

3.2.4 Validación de los resultados

La satisfacción del cliente dentro del proceso de desarrollo de software se ha convertido en un punto central en la validación de cualquier investigación científica. Para medir la satisfacción del cliente existen diversas técnicas, dentro de las que se encuentran: encuesta, panel, buzón de sugerencia, la técnica Iadov, entre otras.

Se utilizó la técnica de Iadov para conocer el grado de satisfacción de los usuarios en la interacción con el sistema desarrollado. Según (López Rodriguez, y otros, 2002) los criterios que se utilizan se fundamentan en las relaciones que se establecen entre tres preguntas cerradas que se intercalan dentro de un cuestionario y cuya relación el sujeto desconoce. Estas tres preguntas se relacionan a través de lo que se denomina el "Cuadro lógico de Iadov", la encuesta confeccionada se aplicó a 15 trabajadores de CEGEL (**¡Error! No se encuentra el origen de la referencia.**).

La tabla 22 representa el cuadro lógico modificado con las preguntas cerradas para la valoración por parte de los usuarios.

	1. ¿Considera que el sistema es complejo y difícil de entender?								
	No			No sé			Sí		
	2. Si pudiera elegir entre usar o no el software desarrollado para realizar su trabajo ¿lo usaría?								
5. ¿Le satisface el sistema desarrollado para acceder a la información que necesita y agilizar su trabajo?	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Me satisface mucho	1	2	6	2	2	6	6	6	6
No me satisface tanto	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me satisface	6	3	6	3	4	4	3	4	4
No me satisface nada	6	6	6	6	4	4	6	4	5

No sé qué decir	2	3	6	3	3	3	6	3	4

Tabla 23. Cuadro lógico de Iadov para medir la satisfacción de los usuarios.

El número resultante de la interrelación de las tres preguntas indica la posición individual de cada usuario en la siguiente escala de satisfacción: clara satisfacción (A), más satisfecho que insatisfecho (B), no definida (C), más insatisfecho que satisfecho (D), clara insatisfacción (E) y contradictoria (C). La cantidad de respuestas por cada categoría es utilizada para calcular el Índice de Satisfacción Grupal (ISG) mediante la fórmula (1):

$$ISG = \frac{A(+1) + B(0{,}5) + C(0) + D(-0{,}5) + E(-1)}{N} \quad (1)$$

Donde N representa el número total de usuarios encuestados. El valor del ISG permite reconocer las siguientes categorías grupales:

- Insatisfacción: desde (- 1) hasta (- 0,5).
- Contradicción: desde (- 0,49) hasta (+ 0,49).
- Satisfacción: desde (0,5) hasta (1).

El valor obtenido del ISG fue de 0,90, lo que refleja la aceptación de la propuesta, un reconocimiento a su utilidad, en tanto los usuarios han emitido criterios donde evidencian su satisfacción por la contribución del sistema.

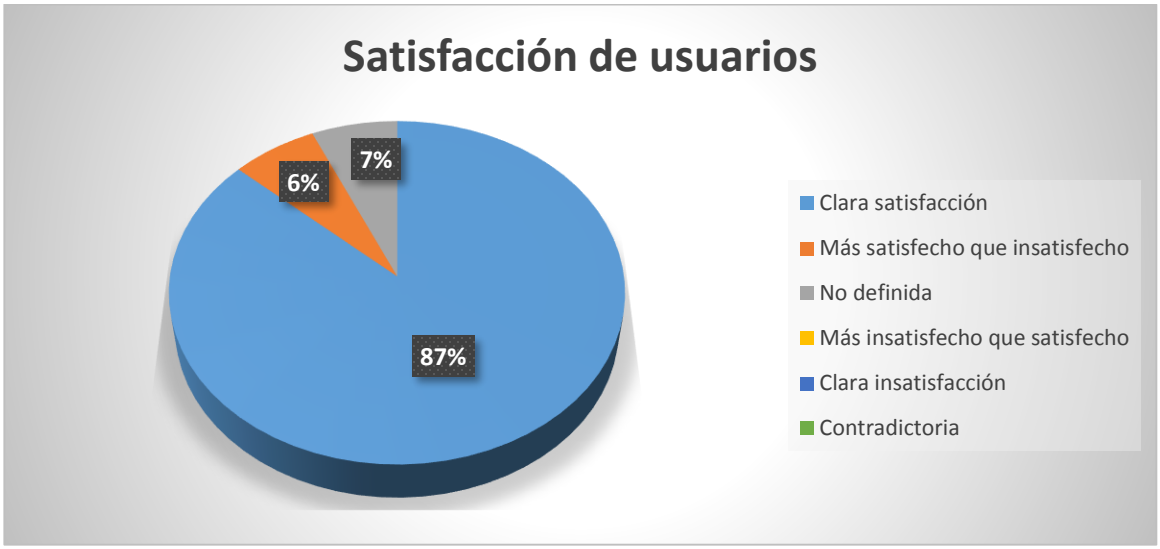

Figura 16. Nivel de satisfacción de usuarios.

[elaboración propia]

La técnica de Iadov contempla además dos preguntas complementarias de carácter abierto. Estas permiten profundizar en las causas que originan los diferentes niveles de satisfacción. Las respuestas dadas plantearon sugerencias de utilidad para la presente y futuras investigaciones, una de las

mencionadas es la incluir en el sistema la gestión de las actividades de Posgrado. Además, reflejaron la satisfacción en cuando a la posibilidad de visualizar el estado de cumplimiento de los planes de resultados de los trabajadores y del plan de CTI del centro. El análisis de estas opiniones arrojó resultados satisfactorios por la preponderancia de aspectos positivos planteados, que sirve como fundamento del alto valor obtenido en el ISG.

3.3 Conclusiones parciales

En este capítulo se describieron las pruebas efectuadas a las funcionalidades del sistema. Las pruebas de caja blanca efectuadas a los métodos principales del sistema mediante la técnica “camino básico”, permitieron comprobar la operatividad del sistema. Las pruebas de caja negra aplicadas a las HU permitieron detectar un conjunto de no conformidades que fueron resueltas en dos iteraciones, obteniéndose la solución esperada por el cliente y el acta de liberación emitida por un especialista del grupo de calidad de CEGEL.

La prueba de aceptación permitió verificar que el software realiza lo especificado por el cliente, obteniéndose finalmente la carta de aceptación. La técnica de Iadov permitió conocer el alto grado de satisfacción de los usuarios, evidenciados en por la preponderancia de aspectos positivos planteados, que sirve como fundamento del alto valor obtenido en el ISG.

CONCLUSIONES GENERALES

La investigación evidenció la necesidad de desarrollar un nuevo sistema para la gestión de la información de Investigación y Posgrado de CEGEL, ya que los sistemas estudiados no contribuyen a elevar el control de la información ni ayudar la toma de decisiones. La metodología de desarrollo de software, lenguajes, tecnologías y herramientas seleccionadas contribuyen al desarrollo del sistema, asegurando la soberanía tecnológica por la que apuesta nuestro país.

Mediante el diseño e implementación de las funcionalidades identificadas se obtuvo un sistema que permite la gestión de la información asociada al área de Investigación y Posgrado, a partir de los planes de resultados anuales, que contribuye permite incrementar el control de la información y contribuye a la toma de decisiones en el CEGEL. Las pruebas realizadas permitieron comprobar el adecuado diseño del sistema, el cumplimento de los requisitos acordados con el cliente y el alto grado de satisfacción de los usuarios.

BIBLIOGRAFÍA

Apache. 2015. The Apache Software Foundation. *The Apache Software Foundation.* [Online] The Apache Software Foundation, 2015. [Cited: Febrero 3, 2016.] www.apache.org.

ArPUG. 2016. Grupo de usuarios PostgreSQL de Argentina. [Online] 2016. [Cited: Mayo 17, 2016.] http://www.postgresql.org.ar/trac/wiki/PgAdmin.

Asenjo González, Diego Andrés and Ríos Peña, Alejandro. 2003. *Patrones de Diseño.* Habana. 2003.

Borillo, Ricardo. 2012. genbetadev. [Online] 2012. http://www.genbetadev.com/frameworks/bootstrap.

Calabria, Luis and Píriz, Pablo. 2003. Universidad de Uruguay. *Metodología XP.* [Online] 2003. [Cited: Marzo 13, 2016.] http://fi.ort.edu.uy/innovaportal/file/2021/1/metodologia_xp.pdf.

Casas, Sandra and Reinaga, Héctor. 2009. *Aspectos Tempranos: un enfoque basado en Tarjetas CRC.* Argentina : Sandra Casas y Héctor Reinaga, 2009.

Codeigniter, Inc. 2011. Codeigniter. [Online] 2011. http://codeigniter.com/..

CSS3. 2015. CSS3.com. *CSS3.com.* [Online] CSS3.com, Mayo 2015. [Cited: Febrero 12, 2016.] http://www.css3.com/.

CUFM. 2012. Fundamento del Diseño de Software. *Unidad 5. Estándares de Diseño.* [Online] IngSoftwareII-CUFM, 2012. [Cited: Abril 2, 2016.] https://cufmingsoftware.wordpress.com/estandares-de-diseno/.

Díaz, Javier, et al. 2009. *Herramientas open source para testing de aplicaciones Web. Evaluación y usos.* Universidad de La Plata. Buenos Aires. Argentina. 2009.

Doctrine. 2015. doctrine. [Online] 2015. [Cited: Marzo 2, 2016.] http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/.

Edith, Leydis. 2008. *Análisis y Diseño de un Nodo Virtual de Procesos.* Habana. 2008.

Eguiluz, Javier . 2015. LIBROSWEB. *Introducción a JavaScript.* [Online] 2015. [Cited: Febrero 10, 2016.] http://librosweb.es/libro/javascript/..

Eguiluz, Javier. 2012. Libros web. *Symfony, la guia definitiva.* [Online] 2012. [Cited: Mayo 14, 2016.] http://librosweb.es/libro/symfony_1_2/.

—. 2014. PHPCuba. *Desarrollo Web Ágil con Symfony 2.* [Online] 2014. [Cited: Abril 28, 2016.] ftp://ftp.prod.uci.cu/PHP/Documentacion/Symfony2/2.4/.

Fernandez, H. 2009. Slideshare. [Online] 2009. [Cited: Febrero 18, 2016.] http://www.slideshare.net/hugofern/sistema-gestin-de-bases-de-datos.

Garcia, Felix Oscar Rubio. 2010. Escuela de Ingeneria Civil Informatica. *Escuela de Ingeneria Civil Informatica.* [Online] 2010. [Cited: febrero 6, 2016.] http://www.eici.ucm.cl/Academicos/R_Villarroel/descargas/ing_sw_1/Metodologias.pdf.

Garlan, David and Shaw, Mary. 1993. *Advances in software eingineering and knowledge engineering.* New Jersey : World Scientific Publishing Company, 1993. ISBN: 981-02-1594-0.

Gauchat, Juan Diego. 2012. *El gran libro de HTML5, CSS3 y Javascript.* Barcelona, España : Marcombo, 2012. ISBN: 978-84-267-1770-2.

Grupo ISSI. 2003. *Metodologías Ágiles en el Desarrollo de Software.* Alicante : Grupo ISSI, 2003.

Guía Ubuntu. 2016. Guía documentada de Ubuntu. [Online] Mayo 14, 2016. http://www.guia-ubuntu.com/index.php/PgAdmin_III.

Gutiérrez, Javier. 2016. Lenguajes y Sistemas Infórmaticos,Universidad de Sevilla. [Online] 2016. [Cited: Febrero 20, 2016.] www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

INEI. 2015. academia. *Herramientas Case.El mejor soporte para el proceso de desarrollo de software.* [Online] 2015. [Cited: Febrero 17, 2016.] http://www.academia.edu/4513393/Libro_HERRAMIENTAS_CASE.

Ingenería de Software. 2016. Ingenería de Software. *Ingenería de Software.* [Online] 2016. http://ingenieriadesoftware.mex.tl/images/18149/PROGRAMACI%C3%93N%20EXTREMA.pdf.

INTECO. 2009. *INGENIERÍA DEL SOFTWARE:METODOLOGÍAS Y CICLOS DE VIDA.* España : INTECO, 2009. ISBN: 9788478290963.

JetBrains. 2013. JetBrains s.r.o. *JetBrains.* [Online] 2013. [Cited: Febrero 16, 2016.] referencia: https://www.jetbrains.com/phpstorm/features/.

Larman, Craig. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Mexico : PRENTICE-HALL, 1999.

Letelier Torres, Patricio y Otros. 2003. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Universidad Politécnica de Valencia (UPV) : Grupo ISSI, 2003.

López Rodriguez, Alejandro and Gonazález Maura, Viviana. 2002. *La técnica de Iadov. Una aplicación para el estudio de la satisfacción de los alumnos por las clases de educación física.* [Online] 2002. [Cited: Junio 17, 2016.] http://www.efdeportes.com/efd47/iadov.htm.

MDN. 2015. Mozilla Developer Network. *Mozilla Developer Network.* [Online] © 2005-2015 Mozilla Developer Network and individual contributors, 2015. [Cited: Febrero 10, 2016.] https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Obsolete_Pages/Guía_JavaScript_1.5/Concepto_de_JavaScript#toc.

Mendes Calo, Karla, Estevez, Elsa and Fillottrani, Pablo. 2009. *Un Framework para Evaluación de Metodologías Ágiles.* Argentina.Universidad Nacional del Sur : EdiUNS, 2009.

Miguel Ángel Álvarez. 2009. desarrolloweb.com. *desarrolloweb.com.* [Online] Octubre 14, 2009. [Cited: Febrero 10, 2016.] http://www.desarrolloweb.com/articulos/que-es-html5.html.

Navarro Cadavid, Andrés and Fernández Martínez, Juan Daniel & Morales Vélez, Jonathan. 2013. *Revisión de metodologías ágiles para el desarrollo de software.* Colombia : Editorial:"Universidad ICESI", 2013. Vol. 11, No. 2.

Otero Méndez, Ángel Juan. 2008. *SISTEMA INFORMÁTICO PARA LA GESTIÓN DE LA FORMACION DE POSTGRADO EN LOS PROFESIONALES DEL MUNICIPIO MAYARI,SICOP.* Holguín : Instituto Superior Minero Metalúrgico Dr. Antonio Nuñez, 2008.

Pérez Ávila, Rodolfo. 2010. *Arquitectura de Software Vista de Integración.* La Habana. 2010.

PHP. 2015. PHP. *PHP.* [Online] The PHP Group, 2015. [Cited: Febrero 8, 2016.] http://php.net/manual/es/security.intro.php.

Piattini, Miguel A and E, Marcos. 1999. *Diseño de base de Datos Relacionales. Ra-Ma.* 1999.

PostgreSQL. 2013. The PostgreSQL Global Development Group. *PostgreSQL.* [Online] 9 9, 2013. [Cited: Febrero 16, 2016.] http://www.postgresql.org/docs/current/static/supported-platforms.html.

Potencier, Fabien. 2011. *Manual de Twig.* 2011.

Pressman, Roger. 2010. *Ingeniería de Software. Un enfoque práctico.* New York : Mac Graw Hill. Higher Education, 2010. ISBN.

Pressman, Roger S. 2005. *Ingeniería del Software. Un enfoque práctico 6ta edición.* Nueva York : McGraw-Hill, 2005. 0072853182.

Proyectos agiles. 2015. proyectosagiles.org. *proyectosagiles.org.* [Online] Creative Commons by-sa, 2015. [Cited: Febrero 5, 2016.] http://www.proyectosagiles.org/que-es-scrum.

Regalut. 2012. Desarrollo Móvil Multiplataforma. *Desarrollo Móvil Multiplataforma.* [Online] 2012. [Cited: Febrero 10, 2016.] http://desarrollomovilmultiplataforma.blogspot.com/2012/08/aspectos-teoricos-entorno-de-desarrollo.html.

Roque Cuevas, Elena García. 2007. *Principios básicos de Informática.* Madrid, España : Dykinson, 2007, 2007. ISBN: 978-84-9849-098-5.

Schwaber, Ken and Sutherland, Jeff. 2013. *La Guía de Scrum.* PLANETA, 2013. ISBN: 9788408135326.

Stinson, Barry. 2001. *PostgreSQL: Essential Reference.* s.l. : s.l. : Sams Publishing, 2001. ISBN: 0752064711216.

Suárez, María Lorena. 2015. Competencias en TIC: Colección Fascículos Digitales. *Cuaderno 4: Lenguajes del lado del servidor y lenguajes del lado del usuario.* [Online] 2015. [Cited: Febrero 8, 2016.] http://escritorioalumnos.educ.ar/datos/recursos/lenguajes_de_programacion_4.pdf.

Symfony. 2014. Symfony.es. [Online] 25 de agosto de 2014, 8 25, 2014. [Cited: Febrero.] http://symfony.es/noticias/2014/08/25/conoce-el-nuevo-plugin-de-symfony2-para-phpstorm/.

symfony.es. 2014. Symfony.es. *Symfony.es.* [Online] 2014. http://symfony.es/que-es-symfony .

Torres, Patricio Letelier. 2003. *Metodologías Ágiles en el Desarrollo del Software.* Alicante : Ingeniería del Software y Sistemas de Información, 2003.

tuProgramacion. 2015. tuprogramacion. *tuprogramacion.* [Online] 2015. [Cited: Marzo 1, 2016.] http://www.tuprogramacion.com/glosario/que-es-un-orm/.

Visual Paradigm. 2000. Visual Paradigm International. *Visual Paradigm.* [Online] 2000. [Cited: Febrero 16, 2016.] http://www.visual-paradigm.com.