Universidad de las Ciencias Informáticas

Facultad 1

Integración del Sistema de Indicadores Cienciométricos con la plataforma de gestión de Ciencia, Tecnología e Innovación de la UCI.

Trabajo de diploma para optar por el título de ingeniero en ciencias informáticas.

Autor: Orlando Pérez Nobrega

Tutores: Ing. Yunaysy Ortiz Batista

Ing. Yordanis Medina León

Ciudad de La Habana, Cuba

Junio del 2010

“Año 52 de la Revolución”

DECLARACIÓN DE AUTORÌA

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

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

_____

Firma del Autor

Orlando Pérez Nobrega

_____ _____

Firma del Tutor **Firma del Tutor**

Ing. Yunaysy Ortiz Batista Ing. Yordanis Medina León

DATOS DE CONTACTO

Tutor: Ing. Yunaysy Ortiz Batista

E-mail: yunaysy@uci.cu

Tutor: Ing. Yordanis Medina León

E-mail: ymleon@uci.cu

Agradecimientos

Muchas han sido las personas que han hecho de este sueño, una realidad. Por lo que hoy, cuando se cumple por fin mi primera meta en la vida quiero agradecerles por todo lo conseguido, y por formar parte de esos momentos que nunca saldrán de mi mente:

A mis padres y hermanas por apoyarme en todo momento, ese apoyo moral, que vale más que cualquier riqueza.

A mis amigos por todos esos momentos que hemos vivido como una gran familia, y en especial a Ramón Ortega, Luis O. Cuza y Danay Guara.

A mis tutores, tribunal y oponente que han contribuido en mi formación como profesional y me han guiado para el desarrollo de esta investigación, gracias por haber manejado mi carrera profesional.

A todos aquellos que han brindado su ayuda desinteresada para la creación de esta investigación.

Dedicatoria

A mi padre Orlando Pérez Peña, por ser mi gran ejemplo y enseñarme a ser fuerte y seguir adelante.

A mi madre Enelis Nobrega Vargas por ser la mejor madre del mundo y siempre estar a mi lado.

A mis hermanas Yanexis y Nityanis por ser lo que más aprecio en la vida y por todo el apoyo que me han brindado.

A toda mi familia en general por su apoyo incondicional.

RESUMEN

La Dirección de Investigaciones de la Universidad de la Ciencias Informáticas (UCI) cuenta con el Sistema de Indicadores Cienciométricos (SIndiCIT) para medir los avances científicos alcanzados por esta institución. Este sistema presenta una gran dificultad y es la no integración con la plataforma de gestión de Ciencia Teología e Innovación. La presente investigación tiene como objetivo desarrollar una nueva versión del SIndiCIT, integrada completamente a dicha plataforma, lo cual permitirá eliminar inconsistencias en la información que se gestiona. Para ello se utilizó *Symfony* y *Rational Unified Process* como plataforma y metodología de desarrollo respectivamente, y *Doctrine* como *Object Relational Mapper* para el acceso a los datos.

ÌNDICE

AGRADECIMIENTOS I

DEDICATORIA II

RESUMEN III

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 6

1.1. Conceptos esenciales 6
1.1.1. Indicadores 6
1.1.2. Cienciometría 6
1.1.3. Indicadores de Ciencia, Tecnología e Innovación 7
1.1.4. Sistema de indicadores para medir la Ciencia, Tecnología e Innovación 7
1.2. Sistemas de gestión de indicadores para medir la Ciencia y Tecnología, vinculados al campo de acción. 7
1.2.1. Sistema Integrado de Información sobre Investigación Científica y Tecnológica de México. 7
1.2.2. CYTCES 8
1.2.3. Sistema de Indicadores para medir la Ciencia en la Universidad de las Ciencias Informáticas 9
1.3. Tendencias y tecnologías actuales. 10
1.3.1. Arquitectura de *Software* (AS) 10
1.3.1.1. Arquitectura Cliente–Servidor 11
1.3.2. Patrones de diseño 12
1.3.2.1. Patrón Modelo–Vista–Controlador (MVC) 12
1.3.3. Metodologías de desarrollo de s*oftware* 14
1.3.3.1. *Rational Unified Process* o Proceso Unificado de Desarrollo de Software (RUP) 15
1.3.4. Lenguaje de modelado 16
1.3.4.1. UML 16
1.3.5. Tecnologías y lenguajes de programación del lado del cliente 18
1.3.5.1. JavaScript 18
1.3.5.2. XHTML (Lenguaje de Marcado de Hipertexto Extensible) 19
1.3.5.3. Ajax 19
1.3.6. Tecnologías y lenguajes de programación del lado del servidor 20
1.3.6.1. *Personal Home Page* (PHP) 21
1.3.7. Sistema Gestor de Bases de Datos 22
1.3.7.1. PostgreSQL 22
1.3.8. Frameworks 23
1.3.8.1. Symfony 24
1.3.9. *Object Relation Mapper* o Mapeador de Objeto Relacional (ORM) 25
1.3.9.1. Doctrine 25
1.4. Herramientas 25
1.4.1. NetBeans 25
1.4.2. Visual Paradigm 26
1.4.3. Servidor Apache 27
1.5. Conclusiones 27
CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA 29

2.1 Introducción 29
2.2 Descripción del negocio actual 29
2.3 Propuesta de solución 30
2.4 Modelo del negocio 30
2.4.1 Reglas del negocio a considerar 31
2.4.2 Actores del negocio 31
2.4.3 Trabajadores del negocio 32
2.4.4 Diagrama de casos de usos del negocio 32
2.4.5 Realización de casos de uso 32
1.4.6 Modelo de objetos del negocio. 34
2.5 Definición de los requisitos 35
2.5.1 Requerimientos funcionales 35
2.5.2 Requerimientos no funcionales 37
2.6 Modelado del sistema 38
2.6.1 Definición de los actores del sistema. 39
2.6.2 Modelo de casos de uso del sistema 39
2.6.4 Descripción textual de los casos de uso 41
2.7 Conclusiones 60
CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA 61

3.1 Introducción 61
3.2 Diseño del sistema 61
3.2.1 Diagrama de clases de diseño *web* 62
3.2.2 Diagramas de interacción 63
3.3 Diseño de la base de datos 64
3.3.1 Diagrama de clases persistentes 64
3.3.2 Descripción de las tablas 65
3.5 Conclusiones 68
CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA 69

4.1 Introducción 69
4.2. Modelo de implementación 69
4.2.1. Diagrama de despliegue 69
4.2.2 Diagrama de componentes 70
4.3 Casos de prueba 71
4.3.1 Prueba de Caja Negra 71
4.4 Conclusiones 74
CONCLUSIONES 75

RECOMENDACIONES 76

BIBLIOGRAFÍA 77

GLOSARIO 78

ANEXOS 81

ÌNDICE DE FIGURAS

Figura 1 Arquitectura Cliente-Servidor 11
Figura 2 Patrón MVC 13
Figura 3 Flujos de trabajo de RUP 16
Figura 4 Diagrama de Casos de Uso del Negocio 32
Figura 5 DA CUN Recopilar Información para el Balance 34
Figura 6 Modelo de Objetos del Negocio 35
Figura 7 Modelo de Casos de Uso del sistema 40
Figura 8 Funcionamiento de Symfony basado en el patrón MVC 62
Figura 9 Diagrama de Clases del diseño Web Gestionar Área 63
Figura 10 Diagrama de Clases Persistentes 65
Figura 11 Diagrama de Despliegue 69
Figura 12 Diagrama de Componentes 71

ÌNDICE DE TABLAS

Tabla 1 Actores del negocio 31
Tabla 2 Descripción de los trabajadores del negocio 32
Tabla 3 Descripción textual del CUN Recopilar Información para el Balance 34
Tabla 4 Actores del sistema 39
Tabla 5 Listado de casos de uso del sistema 41
Tabla 6 Descripción textual del CU Visualizar Datos del Personal. 42
Tabla 7 Descripción textual del CU Visualizar Capacitación. 42
Tabla 8 Descripción textual del CU Visualizar Patentes y Registros. 43
Tabla 9 Descripción textual del CU Visualizar Resultados Introducidos. 44
Tabla 10 Descripción textual del CU Visualizar Publicaciones. 45
Tabla 11 Descripción textual del CU Gestionar Área. 47
Tabla 12 Descripción textual del CU Gestionar Año. 49
Tabla 13 Descripción textual del CU Gestionar Capacitación Recibida. 50
Tabla 14 Descripción textual del CU Gestionar Capacitación Ofertada. 52
Tabla 15 Descripción textual del CU Gestionar Patentes y Registros. 54
Tabla 16 Descripción textual del CU Gestionar Módulos Independientes. 56
Tabla 17 Descripción textual del CU Gestionar Productos Terminados. 57
Tabla 18 Descripción textual del CU Gestionar Departamento. 59
Tabla 19 Descripción de la tabla: tb_dcapacitación_recibida. 66
Tabla 20 Descripción de la tabla: tb_dcapacitación_ofertada. 66
Tabla 21 Descripción de la tabla: tb_danno. 66
Tabla 22 Descripción de la tabla: tb_dmodulo_independiente. 66
Tabla 23 Descripción de la tabla: tb_dpatente_registro. 67
Tabla 24 Descripción de la tabla: tb_dproducto_terminado. 67
Tabla 25 Descripción de la tabla: tb_darea. 67
Tabla 26 Descripción de la tabla: tb_darea_anno. 67
Tabla 27 Pasos para obtener los casos de pruebas. 72
Tabla 28 Escenarios del CU. 73
Tabla 29 Casos de pruebas para el CU. 73
Tabla 30 Casos de prueba con los valores de los datos. 73

INTRODUCCIÓN

El hombre desde la antigüedad siempre se ha preocupado por medir los resultados obtenidos en diferentes aspectos de su vida. Este interés de conocer el impacto de lo que hace, ha ido en aumento a medida que la humanidad ha evolucionado. Proponiéndose cada vez metas más ambiciosas, siendo necesaria la utilización de nuevas técnicas y la mejora de las ya existentes.

La Dirección de Investigaciones (DI) en la Universidad de las Ciencias Informáticas (UCI) como parte de esa humanidad deseosa de medir su trabajo, se ha dado a la difícil tarea de medir de forma eficiente sus avances científicos. Lo que permitirá evaluar la producción científica de los profesores, investigadores y estudiantes de la universidad. Logrando así, que se potencien los resultados científicos y de innovación de una forma más confiable.

La DI está formada por las diferentes líneas de investigación de la UCI, teniendo en cuenta el carácter creador del modo de producción de la misma, que implica generar conocimiento para poder desarrollar mejores productos de *software* y profesionales más capaces. Desde sus inicios, trabaja incansablemente por la organización y el buen desempeño de la actividad científica del centro. Sus esfuerzos están encaminados a evaluar, guiar y orientar metodológicamente la actividad de Ciencia, Tecnología e Innovación (CTI) en la universidad, así como, estimular la participación de estudiantes y trabajadores, establecer alianzas con diferentes instituciones científicas e introducir el centro dentro del sistema científico nacional.

Pero para poder planificar, ejecutar y evaluar la actividad científica, se requiere necesariamente un trabajo estadístico previo de toma de datos básicos y posterior análisis de los mismos. Para de esta forma llegar a construir los indicadores que permiten evaluar la actividad científica alcanzada por la UCI.

Debido a que el sistema de indicadores de CTI vigente en las instituciones y universidades del Ministerio de Educación Superior y el Ministerio de Ciencia Tecnología y Medio Ambiente de Cuba, no se adapta a las condiciones existentes en la UCI, pues esta presenta características que le diferencian del tipo de universidad clásica del país, la más importante de ellas consiste en que la formación curricular y el núcleo fundamental de las investigaciones científicas que se desarrollan se centran en las ramas de las ciencias informáticas y de la computación, fue necesaria la creación de un sistema de indicadores que permitiera evaluar la producción científica de los profesores, investigadores y estudiantes de la universidad, adaptado a las exigencias de la misma. Estos indicadores generales de medición de la actividad de CTI para los centros de educación superior adaptados a la UCI son:

1) Premios obtenidos.

2) Publicaciones científicas.

3) Patentes y registros.

4) Participación en proyectos.

5) Ingresos por la actividad científica.

6) Resultados introducidos.

7) Trabajos presentados en eventos.

8) Capacitación.

9) Empleo de estudiantes.

Para automatizar este proceso, hace alrededor de tres años se desarrolló el Sistema de Indicadores Cienciométricos (SIndiCIT), herramienta utilizada actualmente para evaluar el desarrollo científico de la universidad.

Este sistema depende para su procesamiento de una serie de entradas referente a los indicadores cienciométricos que son introducidos por las diferentes áreas. A partir de estas entradas y a través de un mecanismo de ponderación, el sistema permite un conjunto de salidas que posibilitan la evaluación de la producción científica de la universidad.

Por otro lado, se encuentra la plataforma de gestión de Ciencia, Tecnología e Innovación, la cual tiene como misión gestionar de forma informatizada los procesos fundamentales de CTI en la universidad, los cuales constituyen algunos de los indicadores mencionados anteriormente.

Teniendo en cuenta que la información que se introduce en el SIndiCIT es información cuantitativa, y es la misma que se gestiona en cada uno de los procesos –en la plataforma de gestión de CTI- pero de manera cualitativa, existe la posibilidad de que se introduzcan datos inconsistentes en cada una de las aplicaciones, perdiéndose confiabilidad en los mismos.

Lo que da paso a la **situación problémica** de que: el actualmente el Sistema de Indicadores de Ciencia, Tecnología e Innovación de la Universidad de las Ciencias Informáticas, no se encuentra integrado con la plataforma de gestión de la Dirección de Investigaciones. Lo cual trae consigo que exista inconsistencia en los datos y que la información que maneja el sistema no sea la más confiable.

Teniendo en cuenta lo anteriormente expuesto, el **problema científico** queda formulado de la siguiente forma: ¿Cómo eliminar la inconsistencia de la información entre el actual Sistema de Indicadores Cienciométricos y la plataforma de gestión de Ciencia, Tecnología e Innovación en la Universidad de las Ciencias Informáticas?

El **objeto de estudio** de la presente investigación lo constituye el sistema de Ciencia, Tecnología e Innovación en la Universidad de las Ciencias Informáticas, derivándose como **campo de acción** el Sistema de Indicadores Cienciométricos (SIndiCIT) en la Universidad de las Ciencias Informáticas y la plataforma de gestión de Ciencia, Tecnología e Innovación.

Para guiar la presente investigación se define como **hipótesis:** Si se desarrolla una nueva versión del actual Sistemas de Indicadores Cienciométricos, integrada a la plataforma de gestión de Ciencia, Tecnología e Innovación en la Universidad de las Ciencias Informáticas, entonces se eliminará la inconsistencia de la información entre ambas aplicaciones.

Identificándose como **variable independiente:** Nueva versión del actual sistema de indicadores cienciométricos y como v**ariable dependiente:** Inconsistencia de la información entre ambas aplicaciones. Quedando reflejada la operacionalización de las mismas en el anexo 1.

El **objetivo general** para solucionar el problema anterior es desarrollar una nueva versión del actual sistema de indicadores integrada a la plataforma de gestión de Ciencia, Tecnología e Innovación en la Universidad de las Ciencias Informáticas, derivándose como **objetivos específicos**:

- Analizar el actual Sistema de Indicadores Cienciométricos de la Universidad de las Ciencias Informáticas
- Implementar una nueva aplicación que integre el Sistema de Indicadores Cienciométricos con el resto de los procesos que se desarrollan como parte de la actividad de Ciencia, Tecnología e Innovación en la Universidad de las Ciencias Informáticas.

Para lograr el cumplimiento de los objetivos específicos se proponen las siguientes **tareas investigativas**:

1. Valorar las técnicas de programación, lenguaje y *framework* para el desarrollo de la aplicación.
2. Analizar el actual Sistema de Indicadores Cienciométricos.
3. Analizar la integración con la plataforma de gestión de Ciencia, Tecnología e Innovación.

4. Obtener los artefactos de Negocio, Sistema, Diseño e Implementación, como resultado de la utilización de la metodología de desarrollo RUP.
5. Implementar el sistema de indicadores.

Métodos de investigación utilizados: a nivel teórico.

Histórico-lógico: En esta investigación dicho método se usa con el objetivo de estudiar todo lo referente a los sistemas de indicadores relacionados con la Ciencia, Tecnología e Innovación en el ámbito, internacional, nacional y específicamente en la Universidad de las Ciencias Informáticas, para de esta forma tener mayor conocimiento del tema, sus características e importancia. Este método ayuda a dar cumplimiento al estudio del estado del arte relacionado con sistemas de indicadores cienciométricos.

Analítico-Sintético: Este método es usado para analizar el problema en partes más pequeñas y luego como un todo integrado. La utilización de este método permite comprender mejor toda la información anterior, la cual será de gran utilidad para lograr un adecuado diseño del sistema.

Métodos de investigación utilizados: a nivel empírico.

Método de la Modelación: Partiendo de todo lo investigado y aprendido se realizan los modelos correspondientes al ciclo de vida del *software* que ayudan a dar cumplimiento a las tareas de diseño de los procesos involucrados en la solución.

Método de observación: Este método sirve de apoyo para obtener información sobre el tema a tratar, y ayuda a dar solución a prácticamente todas las tareas. Haciendo uso de él, se puede ver cómo funciona el sistema que actualmente se utiliza en la universidad para medir el desarrollo de la actividad de Ciencia, Tecnología e Innovación y de qué manera fue desarrollado.

Entrevista: Se entrevistó al desarrollador del sistema que actualmente se utiliza, donde se obtuvo la mayor cantidad de información sobre los problemas que presenta dicho sistema, las posibles soluciones a estos problemas, los procesos que realizarán y todo lo nuevo que se le desea incorporar.

El documento se encuentra estructurado como se muestra a continuación:

Capítulo 1: Fundamentación Teórica. Este capítulo incluye el estudio del estado del arte sobre el tema tratado, a nivel UCI, nacional e internacional. Se describen las tecnologías, metodologías y herramientas que sirven de apoyo para la propuesta de solución del problema en cuestión.

Capítulo 2: Características del Sistema. Este capítulo incluye la descripción del negocio a automatizar y la información que se maneja, además de los requisitos y casos de uso del sistema.

Capítulo 3: Diseño del Sistema. Este capítulo incluye la descripción y análisis de la solución que se propone para darle respuesta a la problemática planteada.

Capítulo 4: Implementación y Prueba. Este capítulo incluye las estrategias de codificación, descripción del flujo de implementación, realización de los diagramas de componente, despliegue y diagrama de clases persistentes, así como, una descripción de las tablas representadas en este último.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Conceptos esenciales.

El uso de indicadores cienciométricos es cada vez más frecuente en las distintas instituciones de un país, para definir sus políticas de apoyo a las actividades científicas y tecnológicas. En el caso particular de la UCI cada vez es más evidente la necesidad de cuantificar el impacto de sus programas de ciencia y tecnología de una forma eficiente. En este contexto, esta institución ha consolidado la cultura de la evaluación y del monitoreo de la investigación que desarrollan sus grupos de profesores, estudiantes e investigadores en general. A continuación se relacionan un conjunto de términos, que sería conveniente especificar su significado para una mejor comprensión del tema.

1.1.1. Indicadores

Según la definición de indicadores que dieran Mario Albornoz y Eduardo Martínez: "Los indicadores representan una medición agregada y compleja que permite describir o evaluar un fenómeno, su naturaleza, estado y evolución, articula o correlaciona variables y su unidad de medida es compuesta o relativa. Los indicadores suelen presentar las características siguientes: generalidad, correlación entre variables distintas o de distintos contextos, cuantificabilidad, temporalidad, y posibilidad de constituirse en componentes básicos de desarrollos teóricos". [1]

Los mismos entonces, tienen que pasar por un proceso de selección en el cual deben ser evaluados a la luz de una serie de criterios de calidad, entre ellos, los siguientes:

En general, se distinguen tres funciones de un indicador:

- La simplificación. Trata de describir un fenómeno complejo en una forma sencilla.
- La cuantificación. Expresa (hasta donde es posible) un fenómeno en forma cuantitativa.
- La comunicación. Provee información sobre un fenómeno.

[1]

1.1.2. Cienciometría

La cienciometría pretende abarcar el análisis de factores que pueden ser determinantes para el desarrollo de la actividad científica: número de investigadores, su especialidad, productividad, repercusión, así como el estudio de los productos de investigación publicados en la literatura

científica, principalmente artículos en revistas. Es en este sentido es utilizado este término en la presente investigación. [21]

1.1.3. Indicadores de Ciencia, Tecnología e Innovación

Mario Albornoz y Hernán Jaramillo expresan: "¿Qué es un indicador en ciencia y tecnología? Es algo tan simple como un valor numérico que expresa un rasgo, el nivel de desarrollo de una dimensión del sistema de ciencia y tecnología de un país." [2]

Al respecto completa esta formulación, la definición de indicadores de ciencia y tecnología realizada por Rémi Barré, en otra publicación: "Definimos los indicadores de la ciencia y de la tecnología como siendo los conocimientos cuantitativos sobre las actividades científicas, tecnológicas y de innovaciones útiles para establecer, ejecutar y seguir las políticas de investigación". [2]

1.1.4. Sistema de indicadores para medir la Ciencia, Tecnología e Innovación

Se define como un mecanismo sistemático de monitoreo, compuesto por un conjunto de elementos organizados y relacionados entre sí. Para evaluar el potencial innovador de una empresa, a través de las principales características que dan una idea más completa y objetiva del esfuerzo innovador. Combinando los aspectos técnicos, sociales, cualitativos y cuantitativos de la evaluación del desempeño de las actividades de ciencia y tecnología. [23]

1.2. Sistemas de gestión de indicadores para medir la Ciencia y Tecnología, vinculados al campo de acción.

En el mundo existen diferentes sistemas de gestión de indicadores que permiten medir los resultados científicos alcanzados por las instituciones, estos logran tal tarea a través de un conjunto de indicadores, los cuales pueden variar en dependencia de las necesidades específicas de la institución o el ámbito donde se aplique el mismo. A continuación de describen algunos de estos sistemas:

En el ámbito internacional:

1.2.1. Sistema Integrado de Información sobre Investigación Científica y Tecnológica de México

El Sistema Integrado de Información sobre Investigación Científica y Tecnológica (SIICYT) es un instrumento que refuerza la integración y solidez del Sistema Nacional de Ciencia y Tecnología. El SIICYT integrará los esfuerzos de diferentes instituciones educativas, centros de investigación, organismos públicos, empresas y personas físicas y morales del sector público y privado. A fin de promover el desarrollo y la vinculación de la ciencia básica y la innovación tecnológica, así como convertir la ciencia y la tecnología en un elemento fundamental de la cultura general de la sociedad.

Este sistema es un instrumento para articular la información sobre las políticas, programas, áreas estratégicas, proyectos y participantes en el sistema nacional de ciencia - tecnología - empresa. Las empresas y dependencias comparten los resultados de apoyos a proyectos de investigación y desarrollo tecnológico sobre su estrategia de negocios, competitividad y crecimiento. Los artículos científicos y tecnológicos con arbitraje se encuentran disponibles en el sistema.

Se generan cálculos de información alrededor de las empresas y dependencias que generan tecnologías propias. El registro de investigadores crece a 115,000 miembros, información que se encuentra en las bases de datos de las instituciones de educación superior, centros de Investigación y empresas incorporadas al Sistema. El registro anual de proyectos de ciencia y tecnología aumenta exponencialmente respecto al de 2001. [16]

En el ámbito nacional:

1.2.2. CYTCES

CYTCES Sistema de Indicadores de Ciencia, Tecnologías e Innovación que se encuentra actualmente vigente en las instituciones y universidades del ministerio de Educación Superior y el Ministerio de Ciencias, Tecnologías y Medio Ambiente de Cuba. El mismo consta de una serie de indicadores ponderados. Ver Anexo 2

- **Relevancia:** Aquellos que miden premios y reconocimientos obtenidos como resultado de la investigación científica, otorgados por instituciones de prestigio nacional o internacional.
- **Visibilidad:** Indicadores bibliométricos relacionados con publicaciones científicas. Incluye participación en congresos y conferencias científicas nacionales e internacionales.
- **Tecnología:** Incluye patentes y registros como resultado del desarrollo tecnológico.

- **Pertinencia:** Indicadores evaluativos del monto de recursos financieros ingresados por la universidad como resultado de la comercialización de productos de diferentes clases
- **Impacto:** Incluye la evaluación de los aportes económicos de los productos universitarios en la economía y en la sociedad cubana.
- **Capacitación del Capital Humano:** Expresión de la calidad del proceso de formación del capital humano (profesores, especialistas y estudiantes) de la universidad.

[10]

En la Universidad de las Ciencias Informáticas:

1.2.3. Sistema de Indicadores para medir la Ciencia en la Universidad de las Ciencias Informáticas

El Sistema de Indicadores de Ciencia, Tecnologías e Innovación (SindiCIT) que se encuentra actualmente vigente en la Universidad de las Ciencias Informáticas, brinda la posibilidad de llevar un control de los indicadores relacionados con las actividades de Ciencia, Tecnología e Innovación que se realizan en el centro universitario. Este sistema brinda la posibilidad de calcular una serie de parámetros preestablecidos, los cuales pueden dar una idea de cómo se encuentra la institución en análisis del centro en general. Siendo diseñado para que lo usen personas con determinados conocimientos y no por todos los usuarios, ya que para poder ingresar datos, es necesario estar previamente registrado en la aplicación y tener los permisos necesarios.

Pero, debido a que el sistema de Indicadores de Ciencia, Tecnología e Innovación vigente en las instituciones y universidades del Ministerio de Educación Superior y el Ministerio de Ciencia Tecnología y Medio Ambiente de Cuba, no se adapta a las condiciones existentes en la Universidad de las Ciencias Informáticas, pues esta presenta características que le diferencian del tipo de universidad clásica del país, la más importante de ellas consiste en que la formación curricular y el núcleo fundamental de las investigaciones científicas que se desarrollan, se centran en las ramas de las Ciencias Informáticas y de la Computación, fue necesaria la creación de un sistema de indicadores que permitiera evaluar la producción científica de los profesores, investigadores y estudiantes de la universidad, adaptado a las exigencias de la universidad.

Conclusiones sobre los sistemas similares estudiados:

Después de haber realizado un análisis de los sistemas anteriores, se puede constatar de que para toda institución vinculada a la actividad de CTI, es de interés medir su desarrollo científico.

Existe a nivel nacional un Sistema de Indicadores de Ciencia, Tecnología e Innovación, pero “sistema“ como conjunto de reglas o principios para medir el avance científico en la Educación Superior y el Ministerio de Ciencias, Tecnologías y Medio Ambiente de Cuba, y no “sistema” como herramienta informática para realizar dicha medición.

En cuanto al SIndiCIT, es un *software* para el cálculo de indicadores cienciométricos que se adapta a las características propias de la UCI, pero se encuentra totalmente aislado de la plataforma de gestión de Ciencia, Tecnología e Innovación del centro, provocando inconsistencias en la información que se manipula.

Teniendo en cuenta lo anterior, se propone entonces el desarrollo de una nueva versión del SIndiCIT, que integrada a la mencionada plataforma de gestión, evite inconsistencias en los datos gestionados por ambas aplicaciones y aumente la confiablidad en los mismos.

1.3. Tendencias y tecnologías actuales.

Es importante destacar que la selección de las tecnologías a utilizar se hace en base a las necesidades específicas de cada situación. De forma general está incorrecto decir que una tecnología es mejor que otra, sin antes hacer un análisis para escoger la que cumpla de forma satisfactoria con los requisitos y necesidades. A continuación se describen cada una de las tecnologías seleccionadas para el desarrollo de la propuesta de solución.

1.3.1. Arquitectura de *Software* (AS)

La Arquitectura de *Software* podría definirse de forma general como el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de definir los módulos principales y las responsabilidades que tendrá cada uno de estos módulos. Define también la interacción que existirá entre dichos módulos, el control y flujo de datos, la secuenciación de la información, los protocolos de comunicación y la ubicación en el *hardware*. Sin embargo, la definición oficial de arquitectura del *software* es la de *IEEE Std 1471-2000* que reza así: “La Arquitectura del

Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución". La AS aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño. [5]

1.3.1.1. Arquitectura Cliente–Servidor

La arquitectura cliente-servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. En términos generales, esta arquitectura se forma a partir de la agrupación de conjuntos de elementos que efectúan procesos distribuidos y cómputo cooperativo. [5]

¿Qué es un cliente?

Se denomina cliente al proceso que inicia el diálogo, requerimiento o solicitud de recursos. Esta solicitud puede convertirse en múltiples solicitudes de trabajo a través de redes *LAN* o *WAN*. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente. [5]

¿Qué es un servidor?

Se denomina servidor a cualquier recurso de cómputo dedicado a responder las solicitudes del cliente. Los servidores pueden estar conectados a los clientes a través de redes *LANs* o *WANs*, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes. [5]

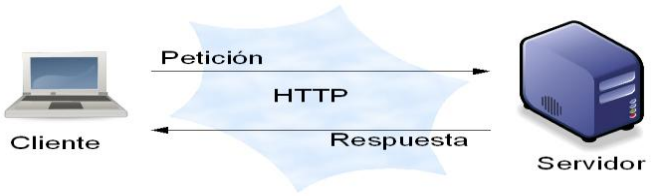

Figura 1 Arquitectura Cliente-Servidor

Ventajas de la arquitectura cliente-servidor:

- El servidor no necesita tanta potencia de procesamiento, parte del proceso se reparte con los clientes.
- Se reduce el tráfico de red considerablemente.

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor, de forma que un programa cliente defectuoso o no autorizado, no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos.
- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).

- Fácil mantenimiento: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulamiento.

[5]

1.3.2. Patrones de diseño

Los patrones son soluciones de sentido común, que deberían formar parte del conocimiento de un diseñador experto. Además, facilitan la comunicación entre diseñadores, pues establecen un marco de referencia. Por otro lado, los patrones de diseño, facilitan el aprendizaje al programador inexperto, pudiendo establecer parejas problema-solución. En la programación orientada a objetos (POO), resulta complicado descomponer el sistema en objetos (encapsulación, granularidad, dependencias, flexibilidad y reusabilidad), los patrones de diseño permiten identificar a los objetos apropiados de una manera mucho más sencilla. También posibilitan determinar la granularidad de los objetos. Además, los patrones de diseño, ayudan a especificar las interfaces, identificando los elementos clave y las relaciones existentes entre estas. De igual modo permite más fácilmente la especificación de la implementación. También, de forma casi automática, ayudan a reutilizar código facilitando la decisión entre “herencia o composición”. [6]

1.3.2.1. Patrón Modelo–Vista–Controlador (MVC)

El MVC es un patrón de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en

aplicaciones *Web*, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. [20]

Modelo (*Model*): El modelo representa información persistente del mundo real, con la cual trabajará la aplicación. Encapsula los datos, las funcionalidades y debe preservar la integridad de los datos.

Vista (*View*): Muestra la información al usuario. Generalmente consiste de toda la información obtenida del modelo que se le muestra al usuario de la aplicación con elementos de diseño que la hacen amigable e interactiva. Pueden existir múltiples vistas del modelo y cada vista tiene asociado un componente controlador.

Controlador (*Controller*): Responde a eventos e interpreta las operaciones del usuario; codificando los movimientos, pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio (*service requests*) para el modelo o la vista. Es el que debe controlar los eventos.

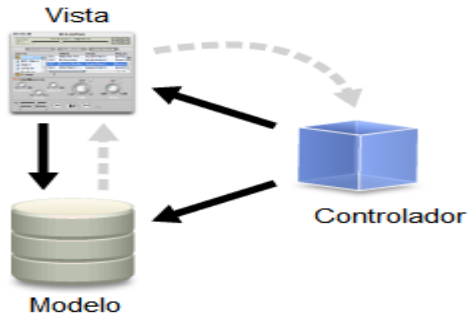

Figura 2 Patrón MVC

Definidos los conceptos de Modelo, Vista y Controlador, a continuación se definirán las **responsabilidades de cada parte:**

El modelo deberá:

- Acceder a los datos persistentes, a la capa de almacenamiento de datos. Algo ideal es que éste sea independiente de la base de datos utilizada.
- Definir las reglas del Negocio.
- Notificar los cambios que se hayan realizado.

El controlador deberá:

- Recibir los eventos de entrada.
- Realizar la acción correspondiente al evento que se ha disparado.

Las vistas deberán:

- Conseguir los datos que aporta el modelo y mostrárselos al usuario.
- Saber cuál es su controlador asociado, el cual puede pedirle algunos servicios típicos como el de actualizar.

[20]

1.3.3. Metodologías de desarrollo de s*oftware*

Cada vez que un grupo de personas enfrenta la tarea de desarrollar un *software* llega una pregunta que es de vital importancia para todo el proceso que se avecina, ¿Qué metodología usar para el desarrollo de un *software*? El desarrollo de todo producto informático es difícil de controlar, pero si no se trabaja por la guía de una metodología de desarrollo, solo obtendremos al final la insatisfacción del usuario y con ello también la de los desarrolladores. Tampoco resulta fácil tener en cuenta qué metodología utilizar para el desarrollo de un producto ya que estas varían teniendo en cuenta la envergadura de lo que se va a desarrollar.

El **Proceso de Desarrollo de *Software*** siempre incluye ciertos riesgos y dificultades, sin importar el tamaño del *software* a realizar. Sin el apoyo de una metodología, nunca se llegaría a satisfacer las necesidades de los clientes para los cuales se trabaja.

Las metodologías son el conjunto de procedimientos, técnicas, herramientas y un soporte documental, que ayuda a los desarrolladores a realizar un nuevo *software*.

Actualmente existen diversas Metodologías de Desarrollo de *Software*, entre las más conocidas se encuentran: metodologías tradicionales o robustas como *Rational Unified Process (RUP),* Técnica de Modelado en Objeto (OMT) y metodologías ágiles como *Extreme Programming (XP).* Cada una de las metodologías antes mencionadas, tienen sus particularidades. No obstante, las más aceptadas y utilizadas actualmente son RUP y XP, por la rigurosa disciplina de trabajo que imponen sobre el proceso de desarrollo del *software*, haciendo énfasis en la previa y total planificación del proyecto a desarrollar. Dando cumplimiento al objetivo de conseguir un *software* más eficiente y predecible.

1.3.3.1. *Rational Unified Process* o Proceso Unificado de Desarrollo de Software (RUP)

El Proceso Unificado de Desarrollo de *Software*, es un proceso que captura las mejores prácticas del conocimiento de líderes en ingeniería de *software* y proporciona a los equipos de desarrollo guías, estándares y recomendaciones para la construcción de *software* de alta calidad. Es una forma disciplinada de asignar tareas y responsabilidades en una empresa de desarrollo (quién hace qué, cuándo y cómo). RUP es una guía de cómo usar el Lenguaje de Modelado Unificado (UML) de la forma más efectiva. RUP sigue un modelo iterativo que aborda primero las tareas más riesgosas logrando reducir los riesgos del proyecto y tener un subsistema ejecutable tempranamente. Tiene como principal objetivo asegurar la producción de *software* de calidad dentro de plazos y presupuestos predecibles. Dirigido por casos de uso, centrado en la arquitectura, iterativo (mini-proyectos) e incremental (versiones). [14]

Dirigido por casos de uso: Basándose en los casos de uso, los desarrolladores crean una serie de modelos de diseño e implementación que los llevan a cabo. Además, estos modelos se validan para que sean conformes a los casos de uso. Finalmente, los casos de uso también sirven para realizar las pruebas sobre los componentes desarrollados. [14]

Centrado en la arquitectura: En la arquitectura de la construcción, antes de construir un edificio éste se contempla desde varios puntos de vista: estructura, conducciones eléctricas, fontanería. Cada uno de estos aspectos está representado por un gráfico con su notación correspondiente. Siguiendo este ejemplo, el concepto de arquitectura de *software* incluye los aspectos estáticos y dinámicos más significativos del sistema. [14]

Iterativo e incremental: Todo sistema informático complejo supone un gran esfuerzo que puede durar desde varios meses hasta años. Por lo tanto, lo más práctico es dividir un proyecto en varias fases. Actualmente se suele hablar de ciclos de vida en los que se realizan varios recorridos por todas las fases. Cada recorrido por las fases se denomina iteración, en la que se realizan varios flujos de trabajo. Además, cada iteración parte de la anterior incrementado o revisando la funcionalidad implementada. Se suele denominar proceso. [14]

Este es uno de los procesos más generales que existe, está enfocado a cualquier tipo de proyecto así no sea de *software*, se basa en la documentación generada en cada uno de sus cuatro fases:

1. Inicio (puesta en marcha).

2. Elaboración (definición, análisis y diseño).
3. Construcción (implementación).
4. Transición (fin del proyecto) en las cuales se ejecutarán varias iteraciones (según el tamaño del proyecto).

[14]

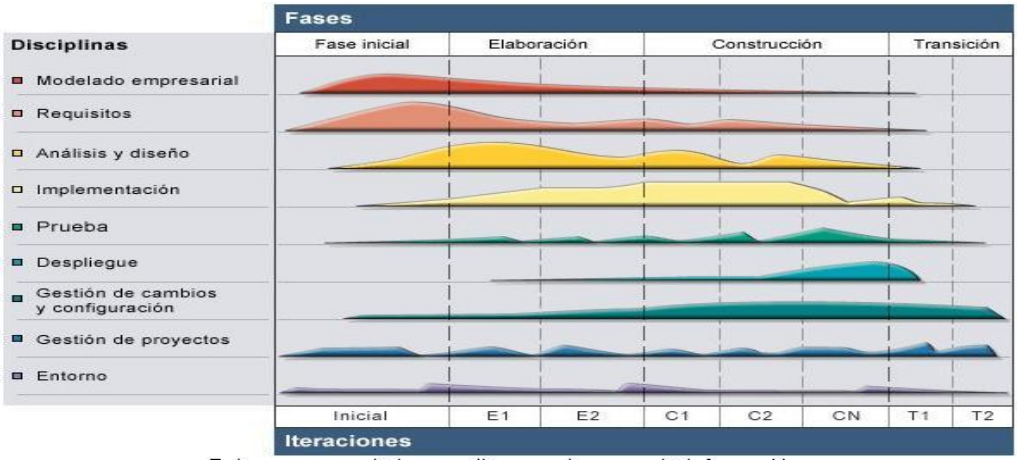

Figura 3 Flujos de trabajo de RUP

1.3.4. Lenguaje de modelado

1.3.4.1. UML

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*. Aun cuando se reconoce que UML no cuenta con un nivel de abstracción ideal para describir una arquitectura, es hoy el lenguaje más utilizado para hacerlo. [9]

Es sumamente importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

El modelo más aceptado a la hora de establecer las vistas necesarias para describir una arquitectura de *software* es el modelo 4+1 vistas arquitectónicas. Aunque no existe de forma explícita una vista arquitectónica, estas 4+1 vistas pretenden describir en su conjunto la arquitectura del sistema.

Las cuatro vistas principales definidas en este modelo son:

- **Vista Lógica:** modelo de objetos, clases, entidad – relación, etc.
- **Vista de Proceso:** modelo de concurrencia y sincronización.
- **Vista de Desarrollo:** organización estática del *software* en su entorno de desarrollo.
- **Vista Física:** modelo de correspondencia *software* – *hardware*.

[9]

Además de estas cuatro vistas principales, existe otra vista, la "+1". La misma se muestra y traza en cada una de las cuatro principales y está formada por las necesidades funcionales que cubre el sistema, a partir de lo cual es identificada como vista de "casos de uso" en ocasiones. A partir de ello se deduce que, según este modelo, la arquitectura es en realidad evolucionada desde los escenarios de los casos de uso. Según la metodología RUP, estas vistas se definen:

- **Vista Lógica:** Muestra elementos de diseño arquitectónicamente significativos del sistema. Soporta los requerimientos funcionales, identifica mecanismos y diseña elementos comunes a través del sistema.
- **Vista de Procesos:** Muestra la estructura de procesos del sistema. Especifica las líneas de mando que ejecutan cada operación en cada una de las clases señaladas en la vista lógica.
- **Vista de Implementación:** Agrupa una colección de componentes y subsistemas de implementación que proveen las funcionalidades del sistema.
- **Vista de Despliegue:** Cuenta con una descripción de los nodos físicos de los cuales se constituye el sistema.
- **Vista de Casos de Uso:** Modela los casos de uso arquitectónicamente significativos del sistema. Aquellos necesarios para el usuario final. Además, actúa como indicador que ayuda al diseñador a descubrir los elementos de la arquitectura durante su diseño y valida e ilustra el diseño de la misma.

[9]

Algunas de las características que propician que con el uso de UML se pueda desarrollar un modelado eficiente con simplicidad de la comunicación entre desarrolladores de *software*, la facilidad de

entendimiento y aprendizaje de sus principales principios, permite y viabiliza la comunicación entre trabajadores del proyecto y usuarios, la estandarización de los elementos del diseño de sistemas y que constituye el estándar más utilizado mundialmente, entre otras.

Por todo lo anteriormente mencionado y teniendo en cuenta además, que para el desarrollo de la aplicación se va a seguir el paradigma orientado a objeto, se ha decidido que el lenguaje de modelado a utilizar sea el UML.

1.3.5. Tecnologías y lenguajes de programación del lado del cliente

Los lenguajes del lado del cliente son aquellos que pueden ser directamente "digeridos" por el navegador y no necesitan un tratamiento previo. Son totalmente independientes del servidor. [15]

1.3.5.1. JavaScript

JavaScript es un lenguaje del lado del cliente basado en objetos, ya que admite la creación de éstos, similares a una clase Java, donde podremos, instanciar un objeto, crear propiedades y métodos. Se utiliza principalmente integrado en un navegador *web*, permitiendo el desarrollo de interfaces de usuario mejoradas y páginas *web* dinámicas. Y se caracteriza por tener entradas dinámicas y funciones de primera clase. Ha tenido influencias de múltiples lenguajes y se diseñó con una sintaxis similar al lenguaje de programación Java, aunque más fácil de utilizar para personas que no son especialistas. [12]

Todos los navegadores modernos interpretan el código *JavaScript* integrado dentro de las páginas *web*. Para interactuar y mejorar la calidad de una página *web*, *JavaScript* contiene embebida una implementación del Modelo de Objetos del Documento (DOM, por sus siglas en inglés). Al utilizarlo podemos crear diferentes efectos e interactuar con nuestros usuarios de una forma mucho más acabada y moderna. [12]

Las principales características de este lenguaje son:

- No necesita compilación, ya que es un lenguaje de programación interpretado y no es necesario compilar los programas para ejecutarlos.

- Multiplataforma.
- Lenguaje de alto nivel que se aproxima más al lenguaje natural humano que al lenguaje binario de las computadoras, el que se conoce como lenguaje de bajo nivel.
- Admite programación estructurada lo que permite hacer correcciones o modificaciones después de haber concluido un programa o aplicación.
- Maneja la mayoría de los eventos que se pueden producir sobre la página *web*.
- No se necesita ningún Entorno de Desarrollo del *Software* (SDK).

[12]

1.3.5.2. XHTML (Lenguaje de Marcado de Hipertexto Extensible)

XHTML (Lenguaje de Marcado de Hipertexto Extensible) es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de remplazar a HTML ante su limitación de uso con las cada vez más abundantes herramientas basadas en XML. XHTML extiende HTML 4.0 combinando la sintaxis del mismo y diseñado para mostrar datos, mientras que con la de XML fue diseñado para describir los datos. Su objetivo fundamental es avanzar en el desarrollo del proyecto del *World Wide Web Consortium*, más conocido como W3C, con la idea de lograr una *web* semántica, donde la información, y la forma de presentarla estén claramente separadas. [13]

Este lenguaje presenta ventajas claras sobre el HTML, entre ellas:

- Un navegador no necesita implementar heurísticas para detectar qué quiso poner el autor, por lo que la conversión de lenguajes puede ser mucho más sencillo.
- Como es también una implementación de XML se puede utilizar fácilmente herramientas creadas para procesamiento de documentos XML genéricas (editores y XSLT).

[13]

1.3.5.3. Ajax

Acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML), es una tecnología de desarrollo *web* para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. [11]

JavaScript es el lenguaje en el que normalmente se efectúan las funciones de llamada de Ajax, mientras que el acceso a los datos se realiza mediante una petición en segundo plano que viene con formato XML. El objeto encargado de realiza la misma está disponible en los navegadores actuales.

Ajax está compuesto realmente por muchas tecnologías, cada una floreciendo por su propio mérito, uniéndose en poderosas nuevas formas, incorporando:

- Presentación basada en estándares usando XHTML y CSS.
- Exhibición e interacción dinámicas usando el DOM (*Document Object Model*).
- Intercambio y manipulación de datos usando XML y Transformaciones XSL[1] (XLST).
- *JavaScript*: une todas las partes.

[11]

Una aplicación AJAX elimina la naturaleza "arrancar-frenar-arrancar-frenar" de la interacción en la *web*, introduciendo un intermediario, un motor AJAX, entre el usuario y el servidor. Parecería que sumar una capa a la aplicación la haría menos reactiva, pero la verdad es lo contrario.

En vez de cargar una página *web* al inicio de la sesión, el navegador carga al motor AJAX (escrito en *JavaScript*). Este motor es el responsable de renderizar la interfaz que el usuario ve y comunicarse con el servidor en nombre del usuario. El motor AJAX permite que la interacción del usuario con la aplicación suceda asincrónicamente (independientemente de la comunicación con el servidor).

1.3.6. Tecnologías y lenguajes de programación del lado del servidor

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor *web*, justo antes de que se envíe la página, a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que

[1] Estándar de la organización W3C que presenta una forma de transformar documentos XML e incluso a formatos que no son XML.

verá el cliente. El cliente solamente recibe una página con el código XHTML resultante de la ejecución de la página del lado del servidor. [15]

1.3.6.1. *Personal Home Page* (PHP)

PHP es un lenguaje de *script*[2], interpretado en el lado del servidor, utilizado para la generación de páginas *web* dinámicas, similar al *Active Server Pages* (ASP) de *Microsoft*, embebido en páginas XHTML y ejecutado en el servidor. A diferencia de *Java* o *JavaScript*, que se ejecutan en el navegador, PHP se ejecuta en el servidor, por eso permite acceder a los recursos que tenga el servidor, como por ejemplo podría ser una base de datos. El programa PHP es ejecutado en el servidor y el resultado es enviado al navegador. El modo de operación de PHP es el siguiente:

1. El navegador realiza una petición al servidor (se escribe la URL).
2. Después el servidor ejecuta el código PHP solicitado y retorna el código XHTML generado al navegador.
3. Por último, el navegador muestra la respuesta del servidor.

[3]

Este tipo de iteración permite algunas operaciones complejas, como conexiones a bases de datos o ejecución de complejos programas. PHP, además de soportar un número masivo de bases de datos, incluyendo *INFORMIX, ORACLE, Sybase, Solid, MySQL y PostgreSQL*, también ofrece una gran variedad de funciones, que permiten desarrollar múltiples acciones que van desde enviar un correo electrónico o un archivo, crear una imagen en tiempo de ejecución, interactuar con diversos protocolos de comunicación, interactuar con documentos XML, autenticación, creación dinámica de documentos PDF (documentos de Acrobat Reader), entre muchos elementos. [3]

Las principales características de PHP son:

- Es un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones *webs* dinámicas con acceso a información almacenada en una base de datos.

[2] *Un script, cuya traducción literal es guión: es un archivo de órdenes o archivo de procesamiento por lotes.*

- El código fuente escrito en PHP es invisible al navegador y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los Sistemas Gestores de Base de Datos que se utilizan en la actualidad, destacando su conectividad con *MySQL y PostgreSQL*.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados extensiones).
- 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.
- Permite aplicar técnicas de programación orientada a objetos.

[3]

1.3.7. Sistema Gestor de Bases de Datos

¿Qué es una base de datos?

Conjunto de datos persistentes, interrelacionados entre sí de forma lógica, que representa una situación del mundo real. [7]

¿Qué es un Sistema Gestor de Base de Datos?

Se le denomina Sistema Gestor de Base de Datos (SGBD) al tipo de *software* específico que se dedica a servir de interfaz entre la base de dato, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición y manipulación de datos, además de un lenguaje de consulta. Su propósito general es el de manejar de manera clara, sencilla y ordenada un conjunto de información. [7]

Existen diversas formas de manejar dichas bases de datos, con gestores como: Oracle, SQL Sever, MySQL, entre otros. [7]

1.3.7.1. PostgreSQL

El sistema gestor de bases de datos que almacena los datos de la aplicación es PostgreSQL.

PostgreSQL es un sistema de base de datos relacional perteneciente al ámbito del *software* libre que destaca por su robustez, escalabilidad y cumplimiento de los estándares SQL.

Cuenta con versiones para una amplia gama de sistemas operativos, entre ellos: Linux, Windows, Mac SX, Solaris, BSD, Tru64 y otros más. **PostgreSQL** soporta la realización de transacciones seguras; también, vistas, uniones, claves extranjeras, procedimientos almacenados y *triggers*. Incluye la mayor parte de los tipos de datos especificados en los estándares SQL92 y SQL99, como: entero, numérico, booleano, *char*, *varchar*, fecha, *interval* o *timestamp*. [18]

Otras características interesantes de PostgreSQL son las siguientes:

- Alta concurrencia, que evita tener que bloquear una tabla cuando se está escribiendo en ella.
- Copias de seguridad en línea.
- Replicación asíncrona.
- Transacciones anidadas.
- Optimizar consultas.

[18]

Si de cifras se trata, es importante saber que en PostgreSQL el tamaño máximo de la base de datos es ilimitado; el de una tabla asciende a 32 TB, el de una fila a 1.6 TB y el de un campo de datos a 1 GB; el número de filas en una tabla es ilimitado, pero no el de columnas, que aunque oscila entre 250 y 1600 columnas por tabla [18], no afecta al sistema que se pretende desarrollar, ya que la información relacionada con los indicadores de CTI, que se almacenará en las tablas de la base de datos no ascenderá a las 250 columnas por cada tabla, y el proceso de almacenado de la información gestionada se realizará de forma anual, no afectando en este sentido las capacidades que provee PostgreSQL para almacenar información.

1.3.8. Frameworks

El concepto *framework* se emplea en el desarrollo de *software*. En general, con el término *framework*, se hace referencia a una estructura de *software* compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En términos informales, un *framework* se puede

considerar como funcionalidades generales a las cuales se les puede agregar las específicas para resolver un problema determinado y obtener una solución informática. [19]

1.3.8.1. Symfony

Symfony es un completo *framework* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones *web*. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación *web*. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación *web* compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación *web*. [19]

Symfony está desarrollado completamente con PHP5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios *web* de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (*Unix, Linux, etc.)* como en plataformas Windows. [19]

Características de Symfony:

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- Independiente del sistema gestor de bases de datos. Su capa de abstracción y el uso de Doctrine, permiten cambiar con facilidad de SGBD en cualquier fase del proyecto.
- 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, adaptable 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[3] y que permite un mantenimiento muy sencillo.
- Aunque utiliza MVC (Modelo-Vista-Controlador), tiene su propia forma de trabajo en este punto, con variantes del MVC clásico como la capa de abstracción de base de datos y el controlador frontal.

[19]

1.3.9. *Object Relation Mapper* o Mapeador de Objeto Relacional (ORM)

¿Qué es un ORM?

Un *Object Relation Mapper* (ORM) es una técnica de programación que permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, es decir, las tablas de nuestra base de datos pasan a ser clases y los registros objetos que podemos manejar con facilidad. [19]

1.3.9.1. Doctrine

Doctrine es un ORM para PHP que se sitúa arriba de la poderosa PHP DBAL (Capa de Abstracción de la Base de Datos). Una de sus características claves es la facultad de, opcionalmente, escribir consultas a la base de datos en un lenguaje OO (Orientado a Objetos). Este brinda a los desarrolladores una poderosa alternativa al SQL manteniendo una máxima flexibilidad sin necesidad de duplicar código. [19]

1.4. Herramientas

1.4.1. NetBeans

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de *software* llamados módulos. Un módulo es un archivo que contiene clases escritas, en este caso para interactuar con las APIs[4] de NetBeans, y un archivo especial que lo identifica como

[3] Es un generador de documentación de código abierto escrito en PHP
[4] Abreviatura de **A**plication **P**rogramming **I**nterface: servicios o funciones que el Sistema Operativo ofrece al programador

módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de *software*. [3]

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. *Sun Microsystems* fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos. [3]

La nueva versión que trae NetBeans IDE es la 6.8 ME, la primera que soporta Java EE 6. Pero lo más interesante a juicio de esta investigación, es la integración total que trae con el lenguaje de programación PHP y más específicamente con el *framework* Symfony. Esta característica permite el desarrollo ágil debido a un gran autocompletamiento de código por parte del entorno de trabajo, de forma que ahorra tiempo a la hora de programar y evita errores comunes de tipografía a los desarrolladores. Por otra parte, esta nueva versión de NetBeans permite integrarse con el servidor *web* Apache durante el *debug* una aplicación *web* alojada en este servidor.

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas).
- Administración de las configuraciones del usuario.
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato).
- Administración de ventanas.
- Framework basado en asistentes (diálogo paso a paso).

[3]

1.4.2. Visual Paradigm

Visual Paradigm 6.1 utiliza UML como Lenguaje de Modelado, es una herramienta CASE poderosa y fácil de usar. Permite representar todo tipo de diagramas UML para las distintas fases como la captura de requisitos, análisis, diseño e implementación, permite además, generar códigos, aplicar ingeniería inversa en los lenguajes Java, C++, PHP, XML, Ada y Python. Además, la generación de código apoya *C#, Visual Basic .NET, Object Definition Language (ODL), Flash, Delphi y Perl*. Mediante esta herramienta se puede desarrollar un producto de calidad teniendo el diseño centrado en casos de usos

y enfocado al negocio. Facilita la comunicación ya que utiliza un lenguaje estándar común a todo el equipo de desarrollo. También posee disponibilidad de múltiples versiones, en múltiples plataformas y es fácil de instalar y actualizar. Presenta la posibilidad de la interoperabilidad con otras aplicaciones como es el *Rational Rose.* Tiene disponible distintas versiones: *Enterprise, Professional, Standard, Modeler, Personal y Community*. [22]

Entre sus características principales se destacanr:

- Soporte de UML.
- Diagramas de Procesos del Negocio.
- Modelado colaborativo con Sistemas Concurrentes de Versiones (CVS) y Subversion.
- Interoperabilidad con modelos UML2.
- Ingeniería inversa, código a modelo y código a diagrama.
- Editor de detalles de casos de uso.
- Generación de bases de datos y transformación de diagramas de Entidad-Relación en tablas de base de datos.

[22]

1.4.3. Servidor Apache

Es un servidor *web* de código abierto multiplataforma, que implementa el protocolo HTTP/1.1 e incorpora la noción de sitios virtuales. Inicialmente consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, a *patchy server* (un servidor "parcheado"). [4]

Apache es altamente configurable y permite bases de datos de autenticación y negociado de contenido. Entre sus más interesantes ventajas podemos mencionar su carácter modular, lo cual permite incrementar o quitar los módulos que posee en función de las necesidades específicas del usuario. Es un sistema de código abierto, lo que significa que se puede modificar para adaptarlo a los requerimientos. Por último, es el servidor *web* más popular, ya que es muy fácil de conseguir y con una gran comunidad que provee ayuda y soporte. [4]

1.5. Conclusiones

Teniendo en cuenta la situación problémica existente y el estudio del estado del arte, se decidió:

- Desarrollar una nueva versión del actual sistema de indicadores cienciométricos SIndiCIT, que se integre a la plataforma de gestión de Ciencia, Tecnología e Innovación en la Universidad de las Ciencias Informáticas, que elimine la inconsistencia de la información entre ambas aplicaciones.

Para el desarrollo de la aplicación se utilizará un conjunto de tecnologías y herramientas definidas por el proyecto y el Centro de Informatización.

- Como metodología de desarrollo de software: RUP (*Rational Unified Process*), una de sus mejores prácticas es la de desarrollar iterativamente, lo que ayuda a mitigar los riesgos en forma temprana y continua.
- Como lenguaje de modelado: UML (*Unified Modeling Language*), este ofrece una amplia variedad de diagramas para visualizar el sistema desde varias perspectivas, conservando un control eficaz.
- Como lenguaje de programación: PHP 5.2.8, lenguaje diseñado para trabajar sobre la *web* y de código abierto que presenta soporte para varios servidores *web* y para conectarse a una gran variedad de base de datos.
- Como Sistema Gestor de Base de Datos: PostgreSql 8.3, sistema escalable que tiene la capacidad de almacenar procedimientos en la propia base de datos.
- Apache 2.2.6 como Servidor Web.
- Como Framework o Marco de Trabajo: Symfony 1.4.
- NetBeans 6.8 como IDE (Entorno Integrado de Desarrollo), este soporta Symfony como Framework.
- Como herramienta CASE Visual Paradigm 6.4, que soporta UML.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En el presente capítulo se ofrece una descripción de los procesos del negocio relacionados con la gestión de indicadores para medir la actividad de Ciencia, Tecnología e Innovación que tiene lugar en la Universidad de las Ciencias Informáticas (UCI). Se especifican los actores y trabajadores del negocio, los casos de uso, los diagramas de actividades y el modelo de objetos del negocio. Para el buen desarrollo del nuevo sistema es necesario centrarse en los requerimientos funcionales y no funcionales, a partir de los cuales se identifican las opciones del sistema, que se representan mediante los casos de uso del sistema y la descripción de los mismos.

2.2 Descripción del negocio actual

La Dirección de Investigaciones (DI) en conjunto con los Vicedecanos de Investigación y Postgrado (VDIP) quienes se encargan de organizar, promover y controlar la actividad científica y postgraduada de las distintas aéreas de la universidad, es la estructura encargada de medir los resultados de Ciencia, Tecnología e Innovación alcanzados en la misma.

Todos estos resultados generados de la actividad científica, son recopilados y sintetizados en los balances de CTI, en los cuales se mide el desarrollo científico que ha alcanzado la universidad en el período de un año.

Para conocer el comportamiento de los indicadores en cada área, es necesaria la **obtención de la información para la realización del balance.** En este proceso los vicedecanos, una vez que le llegan los documentos que avalan la participación de los integrantes de su área, elaboran un informe (documento *Word*) con dicha información y lo envían a la DI unido a estos certificados que avalan dichos resultados.

Para medir el desarrollo científico alcanzado por el centro en el período de un año, se utiliza el ya mencionado SIndiCIT como herramienta de apoyo en estos balances de CTI. Pues este sistema permite –a partir de la introducción de todos los datos relacionados con la actividad científica- un conjunto de salidas que posibilita comparar la Universidad con otras instituciones en el país, así como establecer un ranking entre las diferentes áreas del centro. La introducción de estos datos, la realizan los asesores de investigación en las diferentes áreas de la universidad.

Una vez elaborado el informe para el balance –a nivel de área- e introducidos los datos en el SIndiCIT, los asesores de investigación de la DI, deben verificar la autenticidad de la información recibida y la introducida en el SIndiCIT. En caso de encontrar algún error en la información, se le comunica a los Vicedecanos por vía telefónica o mediante un correo electrónico.

Actualmente se desarrolla una plataforma de gestión de Ciencia, Tecnología e Innovación que se encargará de registrar la información relacionada con premios, publicaciones, trabajos presentados, líneas de investigación, entre otros indicadores. Esta es la misma información que es manejada en el SIndiCIT, lo que pueda traer como consecuencia la ocurrencia de inconsistencias en la información que se manipula.

2.3 Propuesta de solución

Para darle solución a la situación problemática, se propone desarrollar una nueva versión del actual sistema de indicadores SIndiCIT, que integrada a la plataforma de gestión de Ciencia, Tecnología e Innovación de la UCI, logre una centralización de la información que se maneja y de esta forma mejorar la calidad con que se mide el desarrollo científico alcanzado por la UCI y la realización del balance anual. La nueva aplicación será desarrollada como un *plugin*[5] más dentro de la plataforma de gestión de CTI.

Tendrá un adecuado control de acceso que se manejará a través de la asignación de permisos a los diferentes roles. Esta aplicación será accedida desde las diferentes estructuras (de investigación) donde se introducirá determinada información básica, permitiendo a la DI validar dicha información, así como visualizar otro conjunto de indicadores que se obtienen del resto de los procesos que se manejan en la plataforma de gestión.

2.4 Modelo del negocio

Modelar un negocio es una labor de los analistas de procesos de negocio, quienes tienen la misión de entender cómo funciona el negocio; modelando el mismo a través de diagramas de actividades donde se refleja la secuencia de pasos que se llevan a cabo, las personas beneficiadas con las acciones realizadas y las que realizan las actividades. El Modelo del Negocio da la posibilidad de alcanzar cierto

[5] Módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

nivel de conocimientos sobre el problema en cuestión, y permite obtener una visión de la organización que posibilita definir y comprender los roles, responsabilidades y procesos de la misma. [14]

2.4.1 Reglas del negocio a considerar

Las reglas de negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, por lo que regulan algún aspecto del negocio. [14]

En el sistema se han definido las reglas del negocio que se mencionan a continuación:

- Los vicedecanos de investigación y postgrado son los encargados de actualizar los datos de su área.
- El administrador es el encargado de asignar los permisos a todos los usuarios.
- Se establecerá un tiempo límite para introducir la información en el sistema.

2.4.2 Actores del negocio

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externo, con los que el negocio interactúa. Lo que se modela como actor es el rol que se desempeña cuando se interactúa con el negocio para beneficiarse de sus resultados. [14]

Para tener una visión sobre cómo se desarrolla el proceso de medición de los avances científicos en la Dirección de Investigación de la Universidad de las Ciencias Informáticas, fue necesario realizar un estudio, con el objetivo fundamental de lograr la modelación del negocio, definiéndose como actor del negocio el se muestra en la tabla 1:

Actor	Justificación
Reloj	Indica que comience con la recopilación de la información necesaria de CTI para el balance

Tabla 1 Actores del negocio

2.4.3 Trabajadores del negocio

Un trabajador del negocio es una abstracción de una persona (o grupo de personas), una máquina o un sistema automatizado; que actúa en el negocio realizando una o varias actividades, interactuando con otros trabajadores del negocio y manipulando entidades. Representa un rol. Ver tabla 2. [14]

Trabajador	Justificación
Vicedecano	Es aquel rol encargado de elaborar el informe de su área con los resultados producto de la actividad de CTI para el balance y enviarlo junto con los certificados a la DI de la universidad.
Asesor de la DI	Es aquel rol encargado de verificar toda la información que llega a la DI.

Tabla 2 Descripción de los trabajadores del negocio

2.4.4 Diagrama de casos de usos del negocio

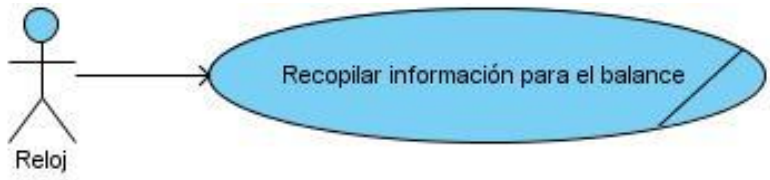

Figura 4 Diagrama de Casos de Uso del Negocio

2.4.5 Realización de casos de uso

Con el propósito de lograr una mejor comprensión del proceso, a continuación se especifica el casos de uso del negocio mediante la descripción textual y el Diagrama de Actividades (DA) correspondiente.

- **CUN Recopilar Información para el Balance**

Caso de uso	Recopilar Información para el balance
Actores	Reloj. (Inicia)

Trabajadores	Vicedecanos y Asesores de la DI
Propósito	Proporcionar a l a DI la información necesaria de CTI para su posterior análisis y realización del balance anual.
Resumen: El CUN se inicia cuando llega el momento en que se debe comenzar a recopilar la información necesaria para realizar el balance anual. Los vicedecanos confeccionan un informe con dicha información y lo envían a la DI, donde los asesores de dicha dirección verifican la autenticidad de la información recibida.	
Precondiciones	
Casos de Uso Asociados	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del negocio**
1. El tiempo indica que se debe comenzar con la recopilación de la información.	Los vicedecanos de cada área elaboran un informe, donde se refleja la información necesaria para la realización del balance anual. 1.2 Los vicedecanos envían el informe junto con los certificados que lo avalan a la DI y registran en el SIndiCIT la información relacionada con los indicadores cienciométricos. 1.3 El asesor de investigación de la DI revisa la autenticidad de la información.
Flujos Alternos	
	1.3 Si el asesor detecta algún error en la información enviada por el vicedecano, se lo informa inmediatamente. 1.3.1 El vicedecano rectifica la información y la envía nuevamente.

Poscondiciones	Quedan archivados y registrados en el SIndiCIT los datos necesarios para la realización del balance.
Prioridad	Crítico
Mejoras	

Tabla 3 Descripción textual del CUN Recopilar Información para el Balance

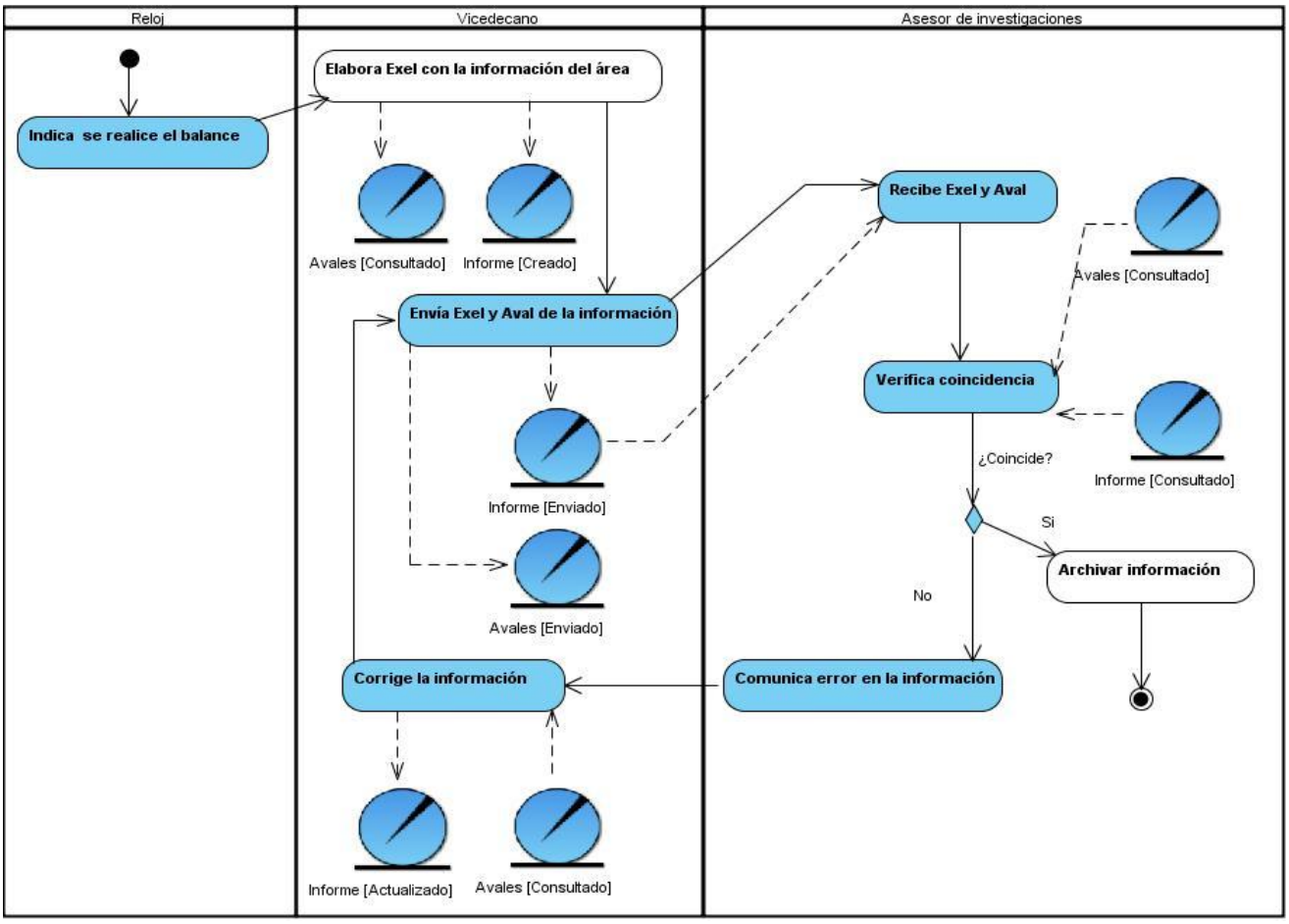

Figura 5 DA CUN Recopilar Información para el Balance

1.4.6 Modelo de objetos del negocio.

Los modelos de objetos son apropiados para representar la estructura de los objetos, sus asociaciones, y como interactúa dinámicamente. Muestran la relación entre los trabajadores y entidades dentro del flujo de trabajo modelamiento del negocio. [9]

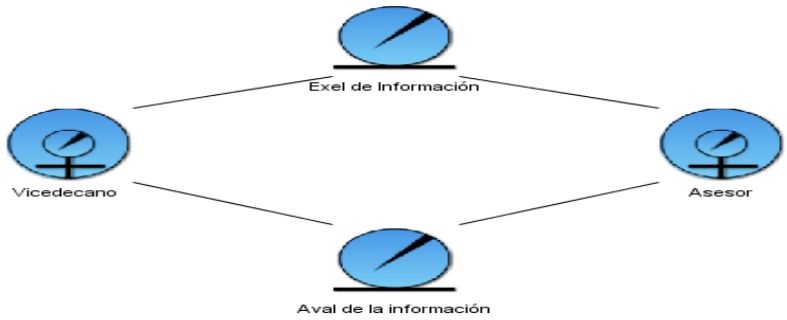

Figura 6 Modelo de Objetos del Negocio

2.5 Definición de los requisitos

La especificación de los requisitos de *software* o captura de requisitos, es el proceso de indagar sobre lo que se debe construir, y de esta forma guiar el desarrollo hacia el sistema correcto. Con una descripción suficientemente buena de los mismos podrá llegarse a un acuerdo entre el cliente y los desarrolladores sobre qué debe y qué no debe hacer el sistema. [14]

Como resultado del proceso de captura de requisito se obtienen los siguientes requerimientos.

2.5.1 Requerimientos funcionales

Los Requerimientos Funcionales son capacidades o condiciones que el sistema debe cumplir, o sea, indican qué es lo que el sistema debe hacer. [14]

R 1 Gestionar capacitación recibida.

R 1.1 Registrar capacitación recibida.

R 1.2 Modificar capacitación recibida.

R 1.3 Eliminar capacitación recibida.

R 2 Gestionar capacitación ofertada.

R 2.1 Registrar capacitación ofertada.

R 2.2 Modificar capacitación ofertada.

R 2.3 Eliminar capacitación ofertada.

R 3 Gestionar datos del personal.

R 3.1 Registrar datos del personal.

R 3.2 Modificar datos del personal.

R 3.3 Eliminar datos del personal.

R 4 Gestionar patentes y registros.

R 4.1 Registrar patentes y registros.

R 4.2 Modificar patentes y registros.

R 4.3 Eliminar patentes y registros

R 5 Gestionar módulos independientes.

R 5.1 Registrar módulos independientes.

R 5.2 Modificar módulos independientes.

R 5.3 Eliminar módulos independientes.

R 6 Gestionar productos terminados.

R 6.1 Registrar productos terminados.

R 6.2 Modificar productos terminados.

R 6.3 Eliminar productos terminados.

R 7 Gestionar año.

R 7.1 Registrar año.

R 7.2 Modificar año

R 7.3 Eliminar año

R 8 Gestionar área.

R 8.1 Registrar área.

R 8.2 Modificar área.

R 8.3 Eliminar área.

R 9 Gestionar departamento.

R 9.1 Registrar departamento.

R 9.2 Modificar departamento.

R 9.3 Eliminar departamento.

R 10 Visualizar datos del personal.

R 11 Visualizar capacitaciones

R 12 Visualizar patentes y registros

R 13 Visualizar resultados introducidos

R 14 Visualizar publicaciones

2.5.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. [14]

Hardware:

Para el desarrollo: PC Intel Pentium 4 o superior, CPU 3GHZ o superior, 512 MB RAM o superior, 40 GB HDD o superior.

Para explotación del cliente: PC Pentium 3 o superior, CPU 133 MHZ o superior, 256 RAM mínimo, 512 RAM recomendada o superior.

Para explotación del servidor: CPU Dual Core 2.0 GHZ o superior, memoria RAM de 1 GB recomendada o superior, 80 GB HDD.

Usabilidad:

Facilidad de uso por parte de los usuarios: el sistema debe presentar una interfaz que permita la fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada. Debe, además, ser una interfaz de fácil navegación que posibilite a los usuarios sin experiencia una rápida adaptación.

Emplear perfiles de usuario: diferenciar las interfaces y opciones para los usuarios que accedan al sistema según los diferentes roles que estos tengan dentro del sistema (asesores de la DI, asesores de investigación de las áreas y administradores).

Menús: el sistema debe presentar una serie de menús, tanto laterales como en barra de iconos flotantes que permitan el acceso rápido a la información por parte de los usuarios, aprovechando así las potencialidades de estas estructuras.

Fiabilidad:

Seguridad de la base de datos: la base de datos deberá estar fraccionada en esquemas que permitan un mejor uso de la información y la división de forma lógica de las funcionalidades del sistema, trayendo consigo además, la protección de la información al ocurrir un incidente sobre una parte de la base de datos. El SGBD escogido debe presentar facilidades de administración de roles y usuarios restringiendo el acceso a los datos.

Políticas de seguridad por usuarios y roles: el sistema debe contar con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo en dependencia del nivel de autorización que presente un usuario determinado.

Seguridad:

- Autenticar al usuario antes de que pueda realizar cualquier acción sobre el portal.
- Garantizar diferentes niveles de acceso al Sistema, brindado por la asignación de roles.

2.6 Modelado del sistema

El Modelado del sistema se centra fundamentalmente en estructurar los requisitos funcionales y no funcionales mediante casos de uso.

2.6.1 Definición de los actores del sistema.

Cada trabajador del negocio (inclusive si fuera un sistema ya existente) que tiene actividades a automatizar es un candidato a actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema. [14]

En la Tabla 4 se describen los actores del sistema que debe ser desarrollado.

Actor	Descripción
Usuario simple	Es aquella persona que solo tiene acceso a visualizar información de los indicadores.
Administrador	Es aquel usuario del sistema que podrá modificar la información y darles permiso a los usuarios. Puede gestionar áreas, años y departamentos.
Vicedecano de investigación	Es aquella persona encargada de gestionar los datos de su área.
Asesor de la DI	Es la persona encargada de verificar la información y modificarla en caso necesario.

Tabla 4 Actores del sistema

2.6.2 Modelo de casos de uso del sistema

El modelo de Casos de Uso del sistema representa las funcionalidades deseadas y el entorno del sistema a través de actores y casos de uso; además, sirve como un contrato entre los clientes y los desarrolladores. Este sienta las bases necesarias para el desarrollo del análisis y el diseño del sistema. Ver figura 6. [6]

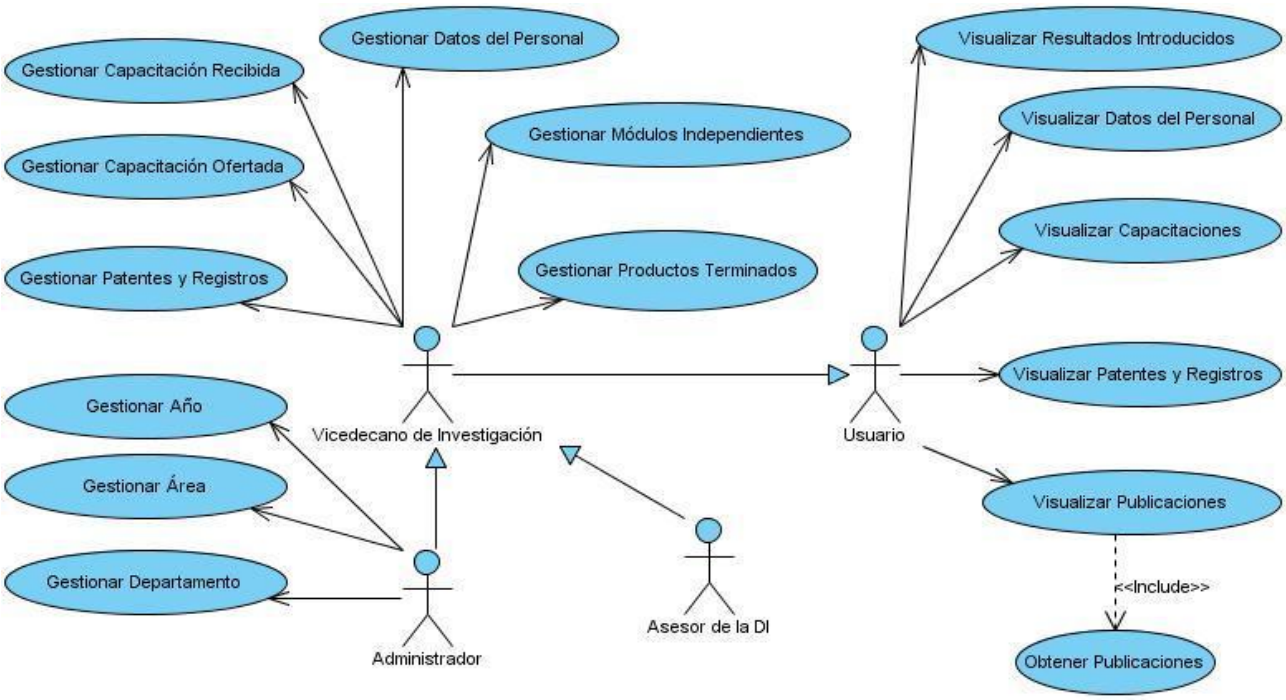

Figura 7 Modelo de Casos de Uso del sistema

2.6.3 Listado de los casos de uso del sistema.

Referencia	Caso de uso	Prioridad
CUS 1	CU_ Gestionar Capacitación Recibida.	Crítico
CUS 2	CU_ Gestionar Capacitación Ofertada.	Crítico
CUS 3	CU_ Gestionar Datos Del Personal.	Crítico
CUS 4	CU_ Gestionar Patentes y Registros.	Crítico
CUS 5	CU_ Gestionar Módulos Independientes.	Crítico
CUS 6	CU_ Gestionar Productos Terminados.	Crítico
CUS 7	CU_ Gestionar Año.	Crítico
CUS 8	CU_ Gestionar Área.	Crítico
CUS 9	CU_ Gestionar Departamentos	Crítico

CUS 10	CU_ Visualizar Datos del Personal.	Crítico
CUS 11	CU_ Visualizar Capacitaciones	Crítico
CUS 12	CU_ Visualizar Patentes y Registros	Crítico
CUS 13	CU_ Visualizar Resultados Introducidos	Crítico
CUS 14	CU_ Visualizar Publicaciones	Crítico

Tabla 5 Listado de casos de uso del sistema

2.6.4 Descripción textual de los casos de uso

Con el propósito de lograr una mejor comprensión de los procesos a automatizar, se especifican los casos de uso del sistema mediante la descripción textual de los mismos.

Caso de Uso:	**Visualizar Datos del Personal.**
Actores:	Usuario simple.
Resumen:	El caso de uso se inicia cuando el usuario accede a las fuentes de indicadores para visualizar los datos del personal que se hayan registrado. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	
Referencias	**R 12**
Prioridad	**Crítico**
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario accede a la opción “Área de análisis”.	2. El sistema muestra un buscador para visualizar los datos generales e indicadores correspondientes a las los datos del personal que se hayan registrado.
3. El usuario selecciona los criterios de búsqueda y ordena buscar.	4. El sistema muestra los datos en detalle del Personal que corresponde con los criterios de la búsqueda y finaliza al caso de uso.

Tabla 6 Descripción textual del CU Visualizar Datos del Personal.

Caso de Uso:	**Visualizar Capacitaciones.**
Actores:	Usuario simple.
Resumen:	El caso de uso se inicia cuando el usuario accede a las fuentes de indicadores para visualizar los datos de las capacitaciones que hayan registradas. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	
Referencias	**R 16**
Prioridad	**Crítico**

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario accede a la opción “Área de análisis”.	2. El sistema muestra un buscador para visualizar los datos generales e indicadores correspondientes a las capacitaciones que hayan registradas.
3. El usuario selecciona los criterios de búsqueda y ordena buscar.	4. El sistema muestra el resultado de la búsqueda.
5. El usuario accede a “*Capacitaciones”.*	6. El sistema muestra los datos en detalle de las capacitaciones (recibidas y ofertadas) correspondientes a los criterios de búsqueda y finaliza el caso de uso.

Tabla 7 Descripción textual del CU Visualizar Capacitación.

Caso de Uso:	**Visualizar Patentes y Registros**
Actores:	Usuario simple.
Resumen:	El caso de uso se inicia cuando el usuario accede a las fuentes de indicadores para visualizar los datos de las patentes y registros que hayan

	almacenadas. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	
Referencias	**R 14**
Prioridad	**Crítico**
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario accede a la opción “Área de análisis”.	2. El sistema muestra un buscador para visualizar los datos generales e indicadores correspondientes a las patentes y registros que hayan registradas.
3. El usuario selecciona los criterios de búsqueda y ordena buscar.	4. El sistema muestra el resultado de la búsqueda.
5. El usuario accede a “Patentes y Registros”.	6. El sistema muestra los datos en detalle de las patentes y registros correspondiente a los criterios de búsqueda y finaliza el caso de uso.

Tabla 8 Descripción textual del CU Visualizar Patentes y Registros.

Caso de Uso:	**Visualizar Resultados Introducidos**
Actores:	Usuario simple.
Resumen:	El caso de uso se inicia cuando el usuario accede a las fuentes de indicadores para visualizar los datos de los resultados introducidos que se hayan registrados. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	
Referencias	**R 15**
Prioridad	**Crítico**
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario	2. El sistema muestra un buscador para visualizar los

accede a la opción” Área de análisis”.	datos generales e indicadores correspondientes a los resultados introducidos que hayan registrados.
3. El usuario selecciona los criterios de búsqueda y ordena buscar.	4. El sistema muestra el resultado de la búsqueda.
5. El usuario accede a “Resultados Introducidos”.	6. El sistema muestra los datos en detalle de los resultados introducidos (Módulos Independientes y Productos Terminados) correspondiente a los criterios de búsqueda y finaliza el caso de so.

Tabla 9 Descripción textual del CU Visualizar Resultados Introducidos.

Caso de Uso:	**Visualizar Publicaciones**
Actores:	Usuario simple.
Resumen:	El caso de uso se inicia cuando el usuario accede a las fuentes de indicadores para visualizar los datos de las publicaciones que hayan registradas. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	
Referencias	**R 16**
Prioridad	**Crítico**
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario accede a la opción “Área de análisis”.	2. El sistema muestra un buscador para visualizar los datos generales e indicadores correspondientes a las publicaciones que hayan registradas.
3. El usuario selecciona los criterios de búsqueda y ordena buscar.	4. El sistema muestra el resultado de la búsqueda.
5. El usuario accede a “Publicaciones”.	6. El sistema muestra los datos en detalle de las publicaciones correspondientes a los criterios de búsqueda y finaliza el caso de uso.

Tabla 10 Descripción textual del CU Visualizar Publicaciones.

Caso de Uso:	Gestionar área.
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el usuario accede a las fuentes de indicadores para registrar, modificar o eliminar los datos de un área determinada. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	Para registrar, eliminar y modificar, debe primeramente haber listado las áreas y haber accedido a los detalles de la misma. Solo es posible eliminar las áreas que no tengan algún indicador asignado.
Referencias	**R 10**
Prioridad	**Crítico**.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario accede a la opción "Gestionar área".	2. Si el usuario accede a la opción *Registrar*, ir al escenario "Registrar área". Si el usuario accede a la opción *Modificar,* ir al escenario "Modificar área". Si el usuario accede a la opción *Eliminar,* ir al escenario "Eliminar área".
Escenario "Registrar área"	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario para registrar los datos del área.
4. El usuario inserta los datos y pincha el botón "Guardar"	5. El sistema valida los datos.

	6. El sistema registra los datos y los muestra en pantalla, finalizando así el caso de uso.
Prototipo de Interfaz	
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema muestra un mensaje de alerta y finaliza el caso de uso.
Escenario "Modificar área"	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario con los campos del área habilitados para que sean modificados.
4. El usuario modifica los datos y pincha el botón "Guardar"	5. El sistema valida los datos.
	6. El sistema registra los cambios y muestra los cambios recién registrados, finalizando así el caso de uso.
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema un mensaje de alerta y finaliza el caso de uso.
Escenario "Eliminar área"	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un mensaje de alerta.
4. El usuario pincha el botón "Aceptar"	5. El sistema elimina el área seleccionada y finaliza el caso de uso.

Poscondiciones	Se registran, modifican o eliminan los datos de un área.

Tabla 11 Descripción textual del CU Gestionar Área.

Caso de Uso:	Gestionar año.
Actores:	Administrador.
Resumen:	El caso de uso se inicia cuando el usuario accede a las fuentes de indicadores para registrar, modificar o eliminar los datos de un año determinado. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	Para registrar, eliminar, debe primeramente haber listado los años y haber accedido a los detalles del mismo. Solo es posible eliminar los años que no tengan algún indicador asignado.
Referencias	**R 9**
Prioridad	**Crítico**.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario accede a gestionar año.	2. Si el usuario accede a la opción *Registrar*, ir al escenario “Registrar año”. Si el usuario accede a la opción *Modificar,* ir al escenario “Modificar año”. Si el usuario accede a la opción *Eliminar,* ir al escenario “Eliminar año”.
Escenario “Registrar año**”**	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario para registrar los datos del año.
4. El usuario inserta los datos y pincha el botón “Guardar”	5. El sistema valida los datos.

	6. El sistema registra los datos y los muestra en pantalla, finalizando así el caso de uso.
Prototipo de Interfaz	
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema muestra un mensaje de alerta y finaliza el caso de uso.
Escenario "Modificar año"	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario con los campos del año habilitados para que sean modificados.
4. El usuario modifica los datos y pincha el botón "Guardar"	5. El sistema valida los datos.
	6. El sistema registra los cambios y muestra los cambios recién registrados, finalizando así el caso de uso.
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema un mensaje de alerta y finaliza el caso de uso.
Escenario "Eliminar año"	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un mensaje de alerta.
4. El usuario pincha el botón "Aceptar"	5. El sistema elimina el año seleccionado y finaliza el caso de uso.

Poscondiciones	Se registran, modifican o eliminan los datos de un año.

Tabla 12 Descripción textual del CU Gestionar Año.

Caso de Uso:	Gestionar Capacitación Recibida.
Actores:	Vicedecanos de investigación.
Resumen:	El caso de uso se inicia cuando el usuario accede a las fuentes de indicadores para registrar o modificar los datos de una capacitación recibida determinada. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	Para registrar o modificar, debe primeramente haber realizado una búsqueda por los criterios área y año de este indicador, donde el año debe estar en estado ACTIVO.
Referencias	**R 1**
Prioridad	**Crítico**.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario accede a la opción “Capacitaciones”.	2. Si el usuario accede a la opción *Registrar*, ir al escenario “Registrar capacitación recibida”. Si el usuario accede a la opción *Modificar,* ir al escenario “Modificar capacitación recibida”.
Escenario “Registrar capacitación recibida”	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario para registrar los datos de la capacitación recibida.
4. El usuario inserta los datos y pincha el botón “Guardar”	5. El sistema valida los datos.
	6. El sistema registra los datos y los muestra en pantalla, finalizando así el caso de uso.

Prototipo de Interfaz	
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema muestra un mensaje de alerta y finaliza el caso de uso.
Escenario "Modificar capacitación recibida"	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario con los campos de la capacitación recibida habilitados para que sean modificados.
4. El usuario modifica los datos y pincha el botón "Guardar"	5. El sistema valida los datos.
	6. El sistema registra los cambios y muestra los cambios recién registrados, finalizando así el caso de uso.
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema un mensaje de alerta y finaliza el caso de uso.
Poscondiciones	Se registran o modifican los datos de una capacitación recibida.

Tabla 13 Descripción textual del CU Gestionar Capacitación Recibida.

Caso de Uso:	Gestionar Capacitación Ofertada.
Actores:	Vicedecano de investigación.
Resumen:	El caso de uso se inicia cuando el usuario accede a las fuentes de indicadores para registrar o modificar los datos de una capacitación ofertada determinada. El caso de uso finaliza cuando se accede a otra

	opción.
Precondiciones:	Para registrar o modificar, debe primeramente haber realizado una búsqueda por los criterios área y año de este indicador, donde el año debe estar en estado ACTIVO.
Referencias	**R 2**
Prioridad	**Crítico**.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario accede a la opción “Capacitaciones.	2. Si el usuario accede a la opción *Registrar*, ir al escenario “Registrar capacitación ofertada”. Si el usuario accede a la opción *Modificar,* ir al escenario “Modificar capacitación ofertada”.
Escenario “Registrar capacitación ofertada”	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario para registrar los datos de la capacitación ofertada.
4. El usuario inserta los datos y pincha el botón “Guardar”	5. El sistema valida los datos.
	6. El sistema registra los datos y los muestra en pantalla, finalizando así el caso de uso.
Prototipo de Interfaz	
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema muestra un mensaje de alerta y finaliza el caso de uso.
Escenario “Modificar capacitación ofertada”	

Acción del Actor	Respuesta del Sistema
	3. El sistema muestra un formulario con los campos de la capacitación ofertada habilitados para que sean modificados.
4. El usuario modifica los datos y pincha el botón “Guardar”	5. El sistema valida los datos.
	6. El sistema registra los cambios y muestra los cambios recién registrados, finalizando así el caso de uso.
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema un mensaje de alerta y finaliza el caso de uso.
Poscondiciones	Se registran o modifican los datos de una capacitación ofertada.

Tabla 14 Descripción textual del CU Gestionar Capacitación Ofertada.

Caso de Uso:	Gestionar Patentes y Registros
Actores:	Vicedecano de investigación.
Resumen:	El caso de uso se inicia cuando el usuario accede a las fuentes de indicadores para registrar o modificar los datos de una patente y registro determinada. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	Para registrar o modificar, debe primeramente haber realizado una búsqueda por los criterios área y año de este indicador, donde el año debe estar en estado ACTIVO.
Referencias	**R 6**
Prioridad	**Crítico**.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. El caso de uso se inicia cuando el usuario accede a la opción “Patentes y registros”.	2. Si el usuario accede a la opción *Registrar*, ir al escenario “Registrar patentes y registros”. Si el usuario accede a la opción *Modificar,* ir al escenario “Modificar patentes y registros”.
Escenario “Registrar patentes y registros”	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario para registrar los datos de la patente y registro.
4. El usuario inserta los datos y pincha el botón “Guardar”	5. El sistema valida los datos.
	6. El sistema registra los datos y los muestra en pantalla, finalizando así el caso de uso.
Prototipo de Interfaz	
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema muestra un mensaje de alerta y finaliza el caso de uso.
Escenario “Modificar patentes y registros”	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario con los campos de la patente y registro habilitados para que sean modificados.
4. El usuario modifica los datos y pincha el botón “Guardar”	5. El sistema valida los datos.
	6. El sistema registra los cambios y muestra los cambios recién registrados, finalizando así el caso de

	uso.
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema un mensaje de alerta y finaliza el caso de uso.
Poscondiciones	Se registran o modifican los datos de una patente y registro.

Tabla 15 Descripción textual del CU Gestionar Patentes y Registros.

Caso de Uso:	Gestionar Módulos Independientes
Actores:	Vicedecano de investigación.
Resumen:	El caso de uso se inicia cuando el usuario accede a las fuentes de indicadores para registrar, modificar o eliminar los datos de un módulo independiente determinado. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	Para registrar o modificar, debe primeramente haber realizado una búsqueda por los criterios área y año de este indicador, donde el año debe estar en estado ACTIVO.
Referencias	**R 7**
Prioridad	**Crítico**.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario accede a la opción "Módulos independientes".	2. Si el usuario accede a la opción *Registrar*, ir al escenario "Registrar módulos independientes". Si el usuario accede a la opción *Modificar,* ir al escenario "Modificar módulos independientes".
Escenario "Registrar patentes y registros"	
Acción del Actor	**Respuesta del Sistema**

	3. El sistema muestra un formulario para registrar los datos de un módulo independiente.
4. El usuario inserta los datos y pincha el botón “Guardar”	5. El sistema valida los datos.
	6. El sistema registra los datos y los muestra en pantalla, finalizando así el caso de uso.
Prototipo de Interfaz	
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema muestra un mensaje de alerta y finaliza el caso de uso.
Escenario “Modificar módulos independientes”	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario con los campos del módulo independiente habilitados para que sean modificados.
4. El usuario modifica los datos y pincha el botón “Guardar”	5. El sistema valida los datos.
	6. El sistema registra los cambios y muestra los cambios recién registrados, finalizando así el caso de uso.
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema un mensaje de alerta y finaliza el caso de uso.
Poscondiciones	Se registran o modifican los datos de un módulo independiente.

Tabla 16 Descripción textual del CU Gestionar Módulos Independientes.

Caso de Uso:	Gestionar Productos Terminados
Actores:	Vicedecano de investigación.
Resumen:	El caso de uso se inicia cuando el usuario accede a las fuentes de indicadores para registrar, modificar o eliminar los datos de un determinado producto terminado. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	Para registrar o modificar, debe primeramente haber realizado una búsqueda por los criterios área y año de este indicador, donde el año debe estar en estado ACTIVO.
Referencias	**R 8**
Prioridad	**Crítico**.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario accede a la opción "Productos terminados".	2. Si el usuario accede a la opción *Registrar*, ir al escenario "Registrar productos terminados". Si el usuario accede a la opción *Modificar,* ir al escenario "Modificar productos terminados".
Escenario "Registrar productos terminados"	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario para registrar los datos de un producto terminado.
4. El usuario inserta los datos y pincha el botón "Guardar"	5. El sistema valida los datos.
	6. El sistema registra los datos y los muestra en pantalla, finalizando así el caso de uso.
Prototipo de Interfaz	

Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema muestra un mensaje de alerta y finaliza el caso de uso.
Escenario "Modificar productos terminados"	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario con los campos del producto terminado habilitados para que sean modificados.
4. El usuario modifica los datos y pincha el botón "Guardar"	5. El sistema valida los datos.
	6. El sistema registra los cambios y muestra los cambios recién registrados, finalizando así el caso de uso.
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema un mensaje de alerta y finaliza el caso de uso.
Poscondiciones	Se registran o modifican los datos de un producto terminado.

Tabla 17 Descripción textual del CU Gestionar Productos Terminados.

Caso de Uso:	Gestionar Departamento
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el usuario accede a las fuentes de indicadores para registrar, modificar o eliminar los datos de un departamento determinado. El caso de uso finaliza cuando se accede a otra opción.
Precondiciones:	Para registrar, eliminar y modificar, debe primeramente haber listado los

	departamentos.
Referencias	**R 11**
Prioridad	**Crítico**.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario accede a gestionar departamentos	2. Si el usuario accede a la opción *Registrar*, ir al escenario "Registrar departamento". Si el usuario accede a la opción *Modificar,* ir al escenario "Modificar departamento". Si el usuario accede a la opción *Eliminar,* ir al escenario "Eliminar departamento".

Escenario "Registrar departamento"	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario para registrar los datos de un departamento.
4. El usuario inserta los datos y pincha el botón "Guardar"	5. El sistema valida los datos.
	6. El sistema registra los datos y los muestra en pantalla, finalizando así el caso de uso.

Prototipo de Interfaz

Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema muestra un mensaje de alerta y finaliza el caso de uso.

Escenario “Modificar departamento”	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un formulario con los campos del departamento habilitados para que sean modificados.
4. El usuario modifica los datos y pincha el botón “Guardar”	5. El sistema valida los datos.
	6. El sistema registra los cambios y muestra los cambios recién registrados, finalizando así el caso de uso.
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	5.1. El sistema un mensaje de alerta y finaliza el caso de uso.
Escenario “Eliminar departamento”	
Acción del Actor	**Respuesta del Sistema**
	3. El sistema muestra un mensaje de alerta.
4. El usuario pincha el botón “Aceptar”	5. El sistema elimina el departamento seleccionado y finaliza el caso de uso.
Poscondiciones	Se registran, modifican o eliminan los datos de un departamento.

Tabla 18 Descripción textual del CU Gestionar Departamento.

2.7 Conclusiones

- La forma actual en que se gestionan los indicadores haciendo uso del SIndiCIT, no es la más viable, teniendo en cuenta la inconsistencia de información entre este sistema y la plataforma de gestión de CTI.

- Por tanto, una propuesta de solución es el desarrollo de una aplicación que se integre totalmente a la plataforma de gestión de CTI, de manera tal que se eliminen las inconsistencias en la información.

- A través de los diagramas de Casos de Uso del sistema y las descripciones textuales de los mismos, se obtiene una mayor información sobre lo que el sistema debe permitir y quiénes tendrán acceso a cada funcionalidad del mismo.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

El presente capítulo tiene como objetivo principal realizar el modelo de diseño del *software* Sistema de Indicadores Cienciométricos, efectuando los diagramas de clases y la realización de los casos de uso teniendo en cuenta los patrones de diseño, lo que permite que el producto sea escalable, reutilizable y flexible, reduciendo así, los esfuerzos de desarrollo y mantenimiento. Este flujo de trabajo tiene una gran importancia en el ciclo de desarrollo, pues traduce los requisitos definidos a funcionalidades que debe realizar el producto, las cuales van a formar parte de la arquitectura candidata; el modelo de diseño que se obtiene como resultado es una entrada principal para la disciplina de implementación.

3.2 Diseño del sistema

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Esta es la entrada fundamental durante las actividades de implementación. [14]

Para lograr una mejor comprensión en cuanto al diseño, se realiza a continuación un breve resumen donde se explica el funcionamiento del framework Symfony.

Estructura de Symfony:

Symfony está basado en el patrón arquitectónico Modelo-Vista-Controlador, pero este implementa todas las ventajas de dicha arquitectura, separa la vista (interfaz) y el modelo (base de datos) mediante el controlador que es el encargado de procesar las interacciones del usuario y realizar los cambios apropiados en el modelo o en la vista, pero además, Symfony hace otra división en el modelo.

Modelo: Symfony divide el modelo en una capa de acceso a datos y otra de abstracción de datos. La abstracción indica qué quiere de la base de datos, y la capa de acceso hace las consultas necesarias para obtener esa información, de esta forma si se cambia de base de datos, solamente se cambiaría la capa de acceso, y la capa de abstracción podría seguir haciendo las mismas operaciones.

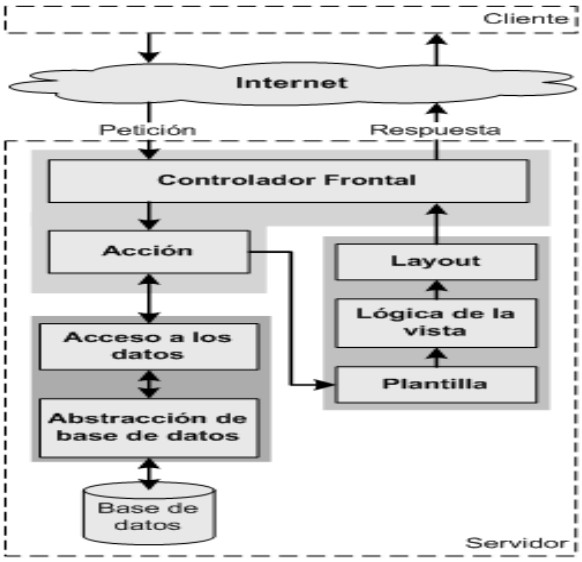

Figura 8 Funcionamiento de Symfony basado en el patrón MVC

Vista: En la presentación de la mayoría de las páginas existen varios elementos comunes como son: la cabecera, la navegación, el pie de página y la plantilla global conocido como *layout*, cambiando tan solo el interior o contenido de la página. Así, estos tres elementos quedan separados.

Controlador: El trabajo del controlador se repite para muchas acciones. Symfony crea un controlador frontal, único en la aplicación, que está encargado de realizar labores comunes, como son: el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares. Además, es el único punto de entrada a la aplicación.

Symfony utiliza un ORM para gestionar el acceso a la base de datos, por lo que puede guardar, obtener, modificar y borrar información en una base de datos sin crear sentencias SQL manualmente y sin tener que descender hasta los detalles más técnicos de cada base datos. Otra gran ventaja de los ORM es que se puede cambiar de una base de datos a otra simplemente cambiando una opción en un archivo de configuración.

[18]

3.2.1 Diagrama de clases de diseño *web*

Los diagramas de clases del diseño son utilizados con el objetivo de representar las relaciones que existen entre los distintos tipos de clases. Se realizó un diagrama de clases de diseño *web* por cada

caso de uso del sistema, descritos en el capítulo anterior. Describiéndose a continuación el diagrama de clases del diseño web del CU Gestionar Área y quedan representado el resto en el anexo 3.

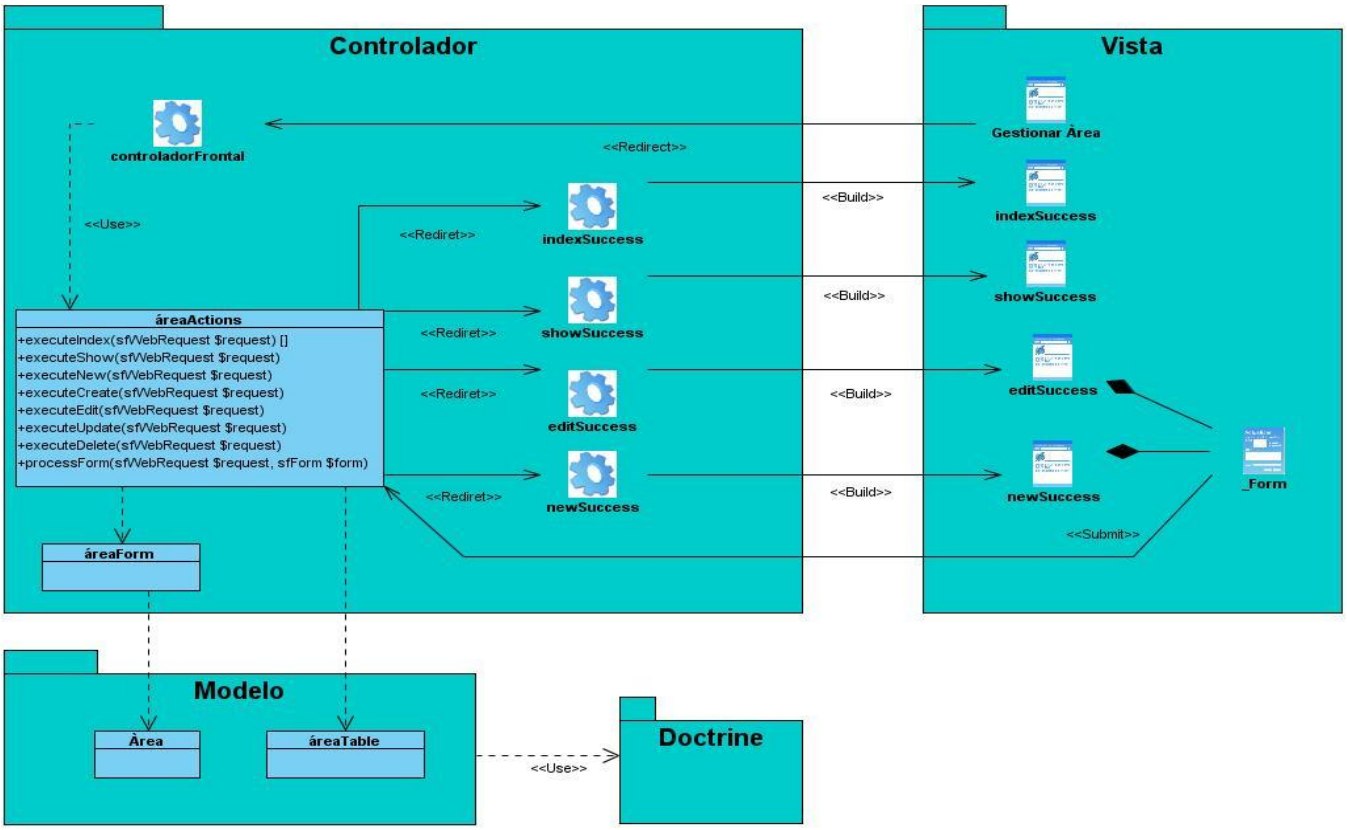

Figura 9 Diagrama de Clases del diseño *Web* Gestionar Área

3.2.2 Diagramas de interacción

Un diagrama de secuencia es un diagrama de interacción que destaca el orden temporal de los mensajes, se utiliza para modelar un flujo de control particular de un caso de uso o los aspectos dinámicos de un sistema [6]. En el anexo 4 se muestran los diagramas de secuencia para cada uno de los casos de uso del sistema.

3.3 Diseño de la base de datos

Una vez definidas las clases del sistema se pueden determinar qué clases requieren que la información que poseen se mantenga a lo largo del tiempo. Para obtener las mismas se toman las clases persistentes que están involucradas en el sistema y se realiza el modelado de la base de datos.

El objetivo del diseño de una base de datos relacional es generar un conjunto de esquemas de relaciones que permitan almacenar la información con un mínimo de redundancia, pero que a la vez faciliten la recuperación de la información. Una de las técnicas para lograrlo consiste en diseñar esquemas que tengan una forma normal adecuada. Para determinar si un esquema de relaciones tiene una de las formas normales se requiere mayor información del "mundo real" sobre lo que se intenta modelar con la base de datos. La información adicional la proporciona una serie de limitantes que se denominan dependencias de los datos.

3.3.1 Diagrama de clases persistentes

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Por lo general las clases persistentes tienen como origen las clases clasificadas como entidad porque ellas modelan la información del sistema y el comportamiento asociado de algún fenómeno o concepto. [6, 14]

El diagrama de clases persistentes describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. A continuación se muestra el diagrama de clases persistentes.

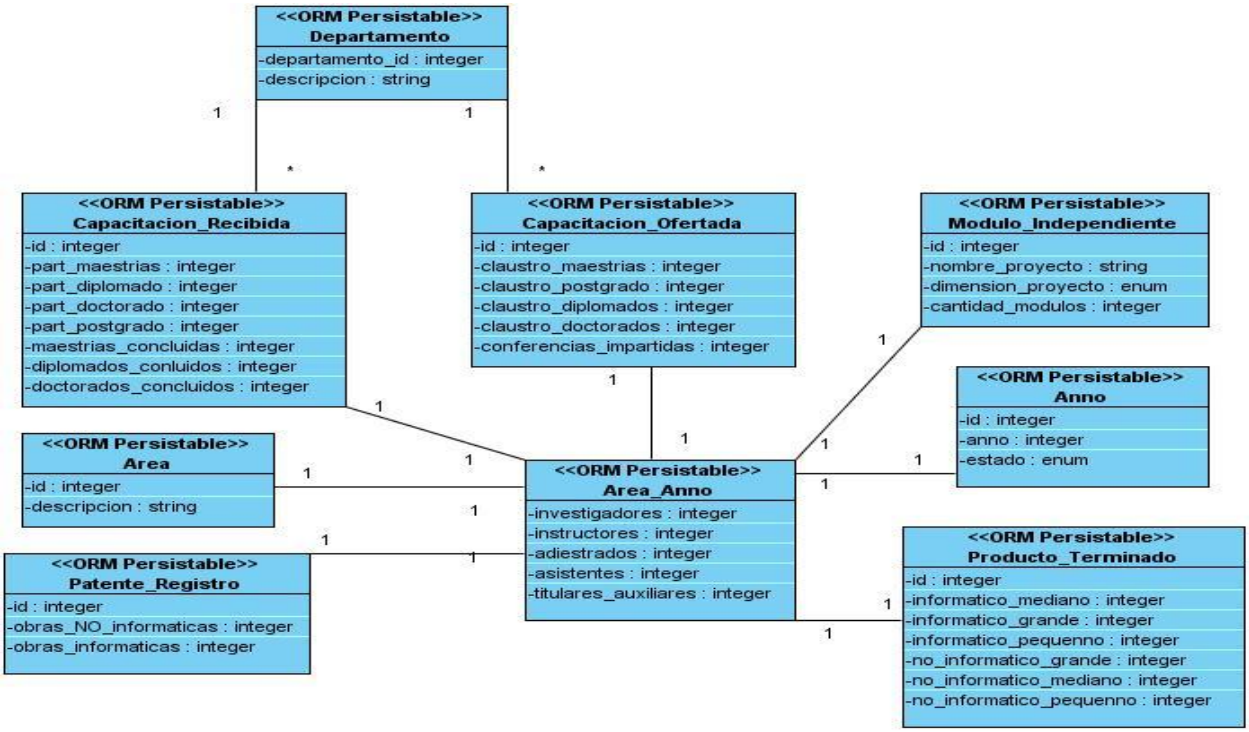

Figura 10 Diagrama de Clases Persistentes

3.3.2 Descripción de las tablas

En este punto se muestran las tablas de la base de datos con sus atributos y una breve descripción de los mismos.

Nombre: tb_dcapacitación_recibida		
Descripción: capacitación recibida de los profesores del área		
Atributo	Tipo	Descripción
capacitación_recibida_id	INT	Identificador de la capacitación recibida
part_maestrias	INT	Cantidad de participantes en maestrías.
maestrias_concluidas	INT	Cantidad de maestrías concluidas
part_diplomado	INT	Cantidad de participantes en diplomados.
diplomados_concluidos	INT	Cantidad de diplomados concluidos.
part_doctorado	INT	Cantidad de participantes en doctorado.
doctorados_concluidos	INT	Cantidad de doctorados concluidos
part_posgrado	INT	Cantidad de participantes en postgrado.

Tabla 19 Descripción de la tabla: tb_dcapacitación_recibida.

Nombre: tb_dcapacitación_ofertada		
Descripción: capacitación ofertada por profesores del área		
Atributo	Tipo	Descripción
capacitación_ofertada_id	INT	Identificador de la capacitación ofertada
claustro_maestrias	INT	Cantidad de profesores en el claustro de maestría.
claustro _diplomado	INT	Cantidad de profesores en el claustro de diplomado.
claustro_doctorados	INT	Cantidad de profesores en el claustro de doctorado curricular.
claustro_posgrado	INT	Cantidad de profesores en el claustro de postgrado.
conferencias_impartidas	INT	Cantidad de conferencias impartidas.

Tabla 20 Descripción de la tabla: tb_dcapacitación_ofertada.

Nombre: tb_danno		
Descripción: Año de análisis donde se guarda el año y si está permitido hacer cambios en los datos (activo).		
Atributo	Tipo	Descripción
anno_id	INT	Identificador del año.
estado	INT	Si el año esta activo.
anno	INT	Número del año

Tabla 21 Descripción de la tabla: tb_danno.

Nombre: tb_dmodulo_independiente		
Descripción: Módulos independientes referidos a resultados introducidos.		
Atributo	Tipo	Descripción
modulo_independiente_id	INT	Identificador de los módulos independientes referidos a resultados introducidos
nombre_proyecto	INT	Nombre del proyecto.
dimension_proyecto	INT	Dimensión del proyecto.
cantidad_modulos	INT	Cantidad de módulos del proyecto.

Tabla 22 Descripción de la tabla: tb_dmodulo_independiente.

Nombre: tb_dpatente_registro		
Descripción: Patentes y registros.		
Atributo	Tipo	Descripción

patente_registro_id	INT	Identificador de las patentes y registros.
obras_NO_informaticas	INT	Cantidad de obras no informáticas registradas.
obras_informaticas	INT	Cantidad de obras informáticas registradas.

Tabla 23 Descripción de la tabla: tb_dpatente_registro.

Nombre: tb_ dproducto_terminado		
Descripción: Productos terminados referidos a resultados introducidos.		
Atributo	Tipo	Descripción
producto_terminado_id	INT	Identificador de los productos terminados referidos a resultados introducidos
informatico_mediano	INT	Cantidad de productos terminados medianos.
informatico_grande	INT	Cantidad de productos terminados grades.
informatico_pequenno	INT	Cantidad de productos terminados pequeños.
no_informatico_grande	INT	Cantidad de productos terminados grandes no informáticos.
no_informatico_mediano	INT	Cantidad de productos terminados medianos no informáticos.
no_informatico_pequenno	INT	Cantidad de productos terminados pequeños no informáticos.

Tabla 24 Descripción de la tabla: tb_dproducto_terminado.

Nombre: tb_area		
Descripción: Áreas del centro		
Atributo	Tipo	Descripción
Area_id	INT	Identificador del área
Descripción	String	Nombre que identifica el área en el centro

Tabla 25 Descripción de la tabla: tb_darea.

Nombre: tb_darea_anno		
Descripción: Datos del personal.		
Atributo	Tipo	Descripción
Area_anno_id	INT	Identificador de los datos del personal
investigadores	INT	Cantidad de investigadores.
instructores	INT	Cantidad de instructores.
adiestrados	INT	Cantidad de adiestrados
titulares_auxiliares	INT	Cantidad de titulares y auxiliares

Tabla 26 Descripción de la tabla: tb_darea_anno.

3.5 Conclusiones

En este capítulo se describió en detalles la aplicación a través del diagrama de secuencia, diagramas de clases y el diseño de la base de datos.

- El diseño de la solución propuesta implementa una nueva versión personalizada del patrón MVC a través del marco de trabajo Symfony, garantizando la organización de las clases dentro del sistema y de la estructura de trabajo.
- El modelo de clases persistentes se traduce en tablas de la base de datos de las cuales se ha hecho una descripción.
- El modelo de Diseño constituye una entrada importante para el flujo de trabajo de Implementación.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

Este capítulo aborda aspectos como implementación y prueba de la solución propuesta. Se describe el modelo de implementación utilizado y se muestran los diagramas de componente y de despliegue. Se realiza además el modelo de prueba.

4.2. Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo con los nodos específicos en el modelo de despliegue. [14]

4.2.1. Diagrama de despliegue

El diagrama de despliegue muestra la distribución de los componentes de *software* desarrollados en el entorno donde será aplicada la solución [6]. La siguiente figura muestra como será desplegado el *software*.

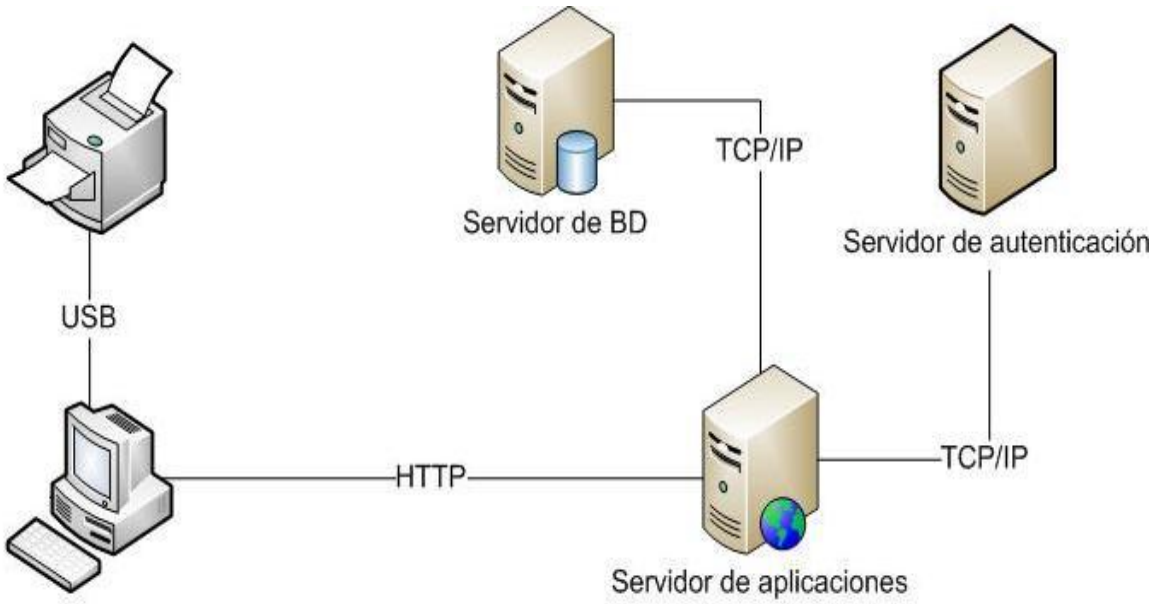

Figura 11 Diagrama de Despliegue

- **PC Cliente:** Estación de trabajo desde donde interactúa el cliente con la aplicación.
- **Servidor *Web* Apache + PHP:** Contiene la aplicación *Web*.
- **Servidor BD PostgreSQL**: Servidor donde se almacenan los datos de la aplicación.
- **Impresora:** Dispositivo que permite imprimir las informaciones generados en la aplicación.
- **Servidor LDAP:** Servidor de autenticación con el dominio uci.cu.

4.2.2 Diagrama de componentes

Un diagrama de componentes representa cómo un sistema es dividido en componentes, mostrando las organizaciones y dependencias lógicas entre componentes de *software*, sean estos componentes de código fuente, binarios o ejecutables. Pueden ser utilizados para modelar y documentar la arquitectura, la vista estática y el código fuente, versiones ejecutables y bases de datos físicas del sistema, entre otros. [6]

En los diagramas realizados, se muestran como están distribuidos los componentes según el patrón arquitectónico Modelo-Vista-Controlador que utiliza Symfony como paradigma en su organización interna. El componente *sfFrontWebController* o Controlador Frontal maneja todas las peticiones *web*, siendo el punto de entrada de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL escrita o pinchada por el usuario. El controlador frontal se encarga de despachar las peticiones, lo que implica algo más que detectar la acción que se ejecuta. De hecho, ejecuta el código común a todas las acciones. En pocas palabras el controlador frontal es el encargado de determinar qué combinación de módulo-acción se ejecutará. El paquete Vista se encarga de producir las páginas que se muestran como resultado de las acciones que se soliciten, las cuales se integran con el *layout*. En el paquete controlador quedan representadas todas las acciones, estas *actions.php* están relacionadas con todos los archivos *Success.php* de la vista que contiene el código que liga la lógica de negocio con la presentación. Al acceder a las diferentes acciones lo primero que se hace es verificar si el usuario está autenticado y tiene los permisos correspondientes a la acción que desea realizar.

El modelo es la capa que contiene las clases: *clasesTable* y las clases. Estas clases son construidas por el subsistema Doctrine de Symfony para el acceso a datos, permitiendo el acceso a la base de

datos mediante el mapeo de objetos. A continuación, se muestra el diagrama principal de componentes de la aplicación.

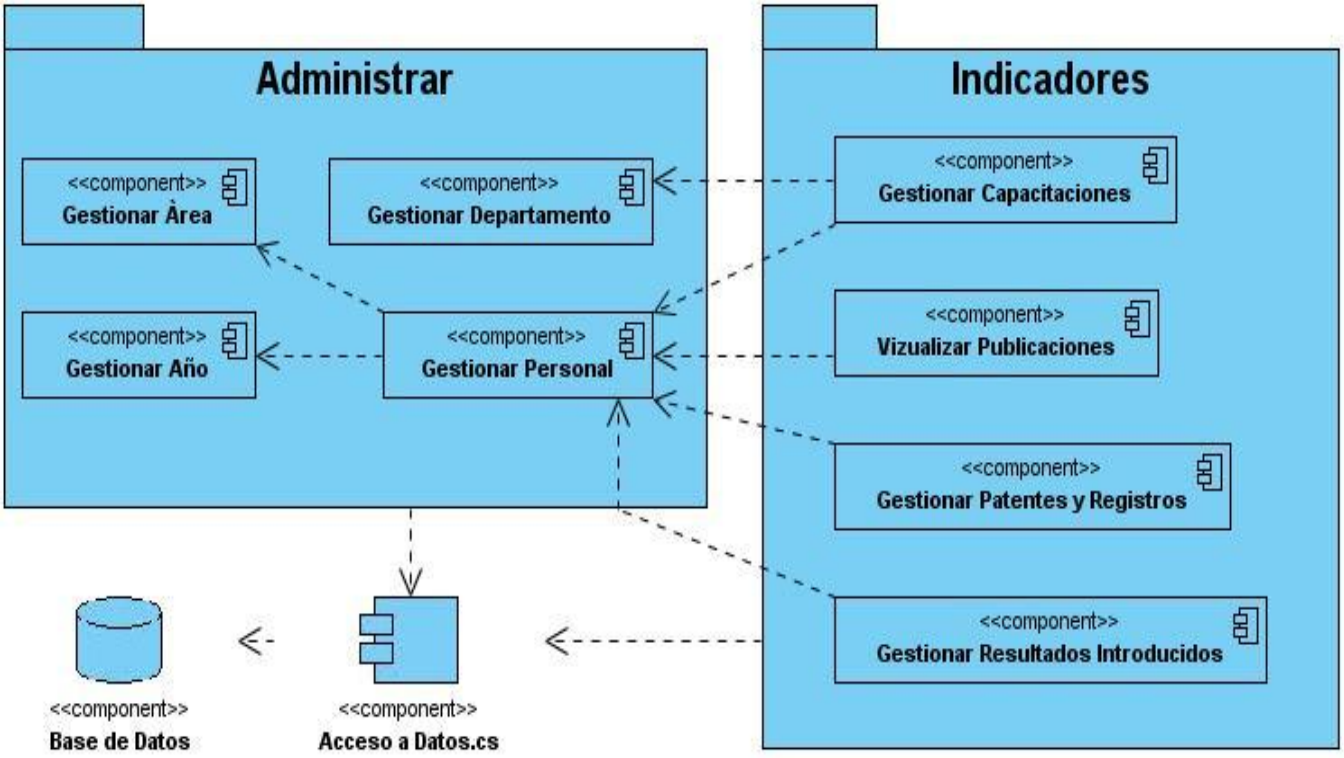

Figura 12 Diagrama de Componentes

4.3 Casos de prueba

Un Caso de Prueba especifica una forma de probar el sistema incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse, generalmente lo que se prueba puede venir dado por un requisito o por una colección de ellos cuya implementación justifica una prueba que es posible realizar, siendo muy comunes en este entorno las pruebas de Caja Negra. [14]

4.3.1 Prueba de Caja Negra

Una Prueba de Caja Negra verifica el comportamiento observable externamente del sistema, o sea, se refiere a las pruebas que se llevan a cabo sobre la interfaz del *software*. Pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce

un resultado correcto, así como que la integridad de la información externa se mantiene. Especifica como probar un caso de uso o un escenario específico de un caso de uso. Dentro de las pruebas de caja negra se encuentran las pruebas de sistema. Acontinuación se muestran el resultado de este tipo de prueba realizada a la aplicación. [14]

Se realizaron pruebas de cajas negras, basadas en el método de tres pasos para obtener un conjunto de casos de prueba del sistema a partir de los CU; los pasos son los siguientes:

Paso	Descripción	Resultado
1	Generar escenarios de uso.	Obtener los posibles caminos de ejecución de cada caso de uso. Cada camino es un escenario de uso.
2	Identificar casos de pruebas.	Conjunto de casos de prueba a partir de los escenarios anteriores.
3	Identificar los valores a probar.	Valores de prueba asociados a cada caso de prueba anterior.

Tabla 27 Pasos para obtener los casos de pruebas.

Para generar los casos de prueba primero se generan todos los posibles escenarios, o caminos de ejecución, de cada caso de uso. Luego, se identifican los casos de prueba a partir de esos escenarios y por último se identifican los valores a probar de cada caso de prueba. Lo más importante de los CU para generar los casos de prueba son los caminos de ejecución. Este camino se divide en dos: en el camino principal o secuencia normal y en el camino alterno. El camino principal son los pasos que sigue el sistema siempre y cuando no surja algún imprevisto, mientras que el camino alterno son las variaciones que van surgiendo durante el camino principal a causa de los errores. Cada uno de estos caminos constituye el escenario de casos de uso. Un caso de prueba es el conjunto de entradas de datos, las condiciones para la ejecución y los resultados esperados. Esto se hace con el propósito de identificar y comunicar las condiciones que se llevarán a cabo en la prueba. Los casos de prueba son necesarios para verificar el éxito de la aplicación y la aceptabilidad del producto. Para generar los casos de prueba se seguirán los siguientes pasos:

1. Para cada CU, generar un sistema completo de los escenarios.
2. Para cada escenario, identificar los casos de pruebas y las condiciones que permiten que se ejecute.

3. Para cada caso de prueba, identificar los valores de los datos con los cuales se harán las pruebas.

Las celdas de la tabla contienen V o I. Donde V indica válido, I indica inválido y CP que expresa Casos de pruebas.

A continuación se muestran las pruebas para los casos de usos Gestionar Área y Gestionar Patentes y Registros. Ver pruebas para cada uno de los casos de uso en el Anexo 3.

Prueba para el CU: Gestionar Área.

Escenario	Nombre	Flujo de comienzo	Alternativo
1	Registrar área.	Flujo normal	
2	Modificar área.	Flujo normal	
3	Eliminar área.	Flujo normal	

Tabla 28 Escenarios del CU.

ID	Escenario	Datos	Resultado esperado
CP1	1	V	Se registran los datos en la BD.
CP2	1	I	Mensaje de error, campo vacío.
CP3	2	V	Se actualizan los datos en la BD.
CP4	3	V	Se elimina el área seleccionada.

Tabla 29 Casos de pruebas para el CU.

ID	Escenario	Datos	Resultado esperado
CP1	1	Se introducen los datos correctamente.	Se registran los datos en la BD.
CP2	1	Se deja el campo vacío.	Mensaje de error, campo requerido.
CP3	2	Se cambian datos necesarios.	Se actualizan los datos en la BD.
CP4	4	Se selecciona el área a eliminar.	Se elimina el área seleccionada.

Tabla 30 Casos de prueba con los valores de los datos.

4.4 Conclusiones

- Se verificaron los requisitos propuestos para la implementación del sistema a través de la representación de los diagramas de despliegue y componentes.
- Queda representada la implementación del sistema brindando resultados visibles donde confirma al cliente el cumplimiento de los requisitos propuestos para el desarrollo de la aplicación.
- Se realizó el modelo de pruebas al *software* utilizando la técnica de caja negra lo que permite la detección de errores desde los primeros ciclos de desarrollo del proyecto.

CONCLUSIONES

- Se realizó un estudio detallado del estado del arte sobre los sistemas de gestión de indicadores, empleando métodos de investigación a nivel teórico y empírico. Y se arrojó como resultado la necesidad de crear un sistema de gestión de indicadores integrado a la plataforma de gestión de Ciencia, Tecnología e Innovación de la universidad.

- Se especificaron las funcionalidades y cualidades con que el sistema debe cumplir, dando paso a un mejor entendimiento de las actividades a automatizar.

- El sistema desarrollado contribuye a eliminar la inconsistencia de los datos entre el actual sistema de indicadores cienciométricos SIndiCIT y la plataforma de gestión de CTI, aumentando la confiabilidad de la información que se maneja como parte de la actividad científica de la universidad.

RECOMENDACIONES

- Profundizar en el estudio de sistemas de indicadores para ampliar las funcionalidades del *software*. Incluyendo nuevos parámetros en el análisis de los resultados.
- Realizar la ponderación de los indicadores.
- Generar reportes mediante graficas, lo que permitirá determinar hacia donde deben enfocarse los mayores esfuerzos para incrementar la producción de *software* con alto valor agregado.

BIBLIOGRAFÍA

[1] **Albornoz, Mario y Jaramillo, Hernán.** *El universo de la medición.* s.l. : La perspectiva de la ciencia y la tecnología. pág. xiii.

[2] **Albornoz, Mario y Martínez, Eduardo.** *Indicadores de ciencia y tecnología: balance y perspectivas.* págs. 11-12.

[3] **Alvarez, Rubén. 2007.** Desarrolloweb. [En línea] 2007. http://www.desarrolloweb.com/articulos/239.php.

[4] **Apache.** Apache. [En línea] Apache Foundation. http://www.apache.org..

[5] **Arebas, Jesus Barranco.** *Metodologías del análisis estructurado de sistemas.* s.l. : Madrid. 84-8468-043-6..

[6] **C, Larman.** *UML y patrones.* s.l. : UML y patrones. Tomo I.

[7] **CAVSI. 2010.** Sistemas Gestores de Bases de Datos. [En línea] 2010. http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

[8] **Cazco, Rafael. 2009.** Programación Extrema - Ingenieria de Software UDB. [En línea] 2009.

[9] **Craig, Larman. 2006.** *UML y Patrones.* La Habana : Fèlix Varela, 2006. Vol 1.

[10] CYTCES. [En línea] http://www.ricyt.org/interior/normalizacion/V_taller/rodriguez.pdf.

[11] **Eguíluz Pérez, Javier. 2010.** Introduccion a Ajax. *LibrosWeb.es.* [En línea] LibrosWeb, 2010. http://www.librosweb.es/ajax/index.html.

[12] **—. 2008.** *Introducción a JavaScript.* 2008.

[13] Guía Breve de XHTML. *Guía Breve de XHTML.* [En línea] W3C. http://www.w3c.es/Divulgacion/Guiasbreves/XHTML.

[14] **2006.** *Ingenieía del Software un enfoque práctico.* La Habana : Fèlix Varela, 2006. Vol. 1.

[15] **2008.** Lenguajes del lado servidor o cliente. *PHPNuke.* [En línea] 2008. http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

[16] **México.gov.** Sistema Integrado de Información sobre Investigación Científica y Tecnológica de México. [En línea] México.gov. http://www.siicyt.gob.mx/siicyt/.

[17] **Moodle. 2008.** CVS para Desarrolladores de Moodle . *Documentación de Moodle* . [En línea] 2008. http://aulauvs.sld.cu/doc/?file=cvs.html.

[18] **PostgreSql.** The world´s most advanced open-source database. [En línea] PostgreSql. http://www.postgresql.org/. PostgreSQl.

[19] **Potencie, Fabien. 2007.** La Guia Definitiva. [En línea] Symfony Project, 2007. Sencio Labs.

[20] **Seth, Ladd y Darren, Davidson. 2006.** *Expert Spring MVC and Web Flow.* s.l. : Appress, 2006.

[21] **Spinak, Ernesto. 2008.** Bibliometría y Cienciometría. *Bibliometría y Cienciometría.* [En línea] 2008. http://bvs.sld.cu/revistas/aci/vol9_s_01/sci07200.htm.

[22] Viual Paradigm for UM. [En línea] Visual Paradigm. http://www.visual-paradigm.com/product/vpuml/.

[23] **Z, Ortiz, y otros.** SISTEMA DE MEDICIÓN DE LA CAPACIDAD DE INNOVACIÓN TECNOLÓGICA APLICADO A NA EMPRESA MANUFACTURERA. [En línea] http://www.scielo.org.ve/pdf/uct/v11n42/art03.pdf.

GLOSARIO

A continuación, en orden alfabético, se muestra el significado de algunos términos usados en este documento que pueden dificultar la compresión del mismo:

A_____

ASP: *Active Server Pages* (ASP), también conocido como ASP clásico, es una tecnología de *Microsoft* del tipo "lado del servidor" para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a *Internet Information Services* (IIS).

B _____

Bibliometría: Es un método de análisis cuantitativo de la producción, distribución y uso de la literatura editada o semieditada.

C_____

Cliente: Cliente es un ordenador que accede a recursos y servicios brindados por otro llamado Servidor, generalmente en forma remota.

CASE: *Computer Aided Software Engineering*, siglas en ingles.

CUS: Caso de uso del sistema.

CVS: Es un **Sistema Concurrente de Versiones**, una forma de trabajo habitualmente utilizada para almacenar el código fuente de grandes proyectos de *software*. CVS almacena todas las versiones de todos lo ficheros de tal forma que nada es nunca perdido, y su utilización por varias personas es registrada.

D _____

DCD: Diagrama de clases del diseño.

DS: Diagrama de secuencia.

DOM: El *Document Object Model* (una traducción al español no literal, pero apropiada, podría ser Modelo en Objetos para la representación de Documentos o también Modelo de Objetos del Documento), abreviado DOM, es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.

H _____

Herramienta CASE: Aplicación informática destinada a aumentar la productividad en el desarrollo de *software* reduciendo el coste de las mismas en términos de tiempo y de dinero, se utiliza para la modelación del sistema.

HTTP*:* Protocolo de transferencia de hipertexto cuyas siglas vienen dadas por su nombre en inglés *HyperText Transfer Protocol,* es el protocolo usado en cada transacción de la *web*.

I _____

IDE: Entorno Integrado de Desarrollo, es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse exclusivamente para un lenguaje de programación o bien para varios.

K _____

SDK: Un kit de desarrollo de software o SDK (siglas en inglés de *software development kit*) es generalmente un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de *software*, *frameworks*, plataformas de *hardware*, computadoras, videoconsolas, sistemas operativos.

M _____

MVC: Es un patrón de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones *web*.

P _____

PHP*: Hypertext Pre-processo*, es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo *web* y puede ser incrustado dentro de código XHTML.

U _____

UML: Lenguaje visual para especificar, construir y documentar un *software*. Sus siglas vienen dadas por su nombre en inglés *Unified Modeling Language*.

X _____

XML: *Extensible Markup Language* («lenguaje de marcas ampliable»), es un metalenguaje extensible de etiquetas.

XSLT: Es un estándar de la organización W3C que presenta una forma de transformar documentos XML e incluso a formatos que no son XML. Las hojas de estilo **XSLT** - aunque el término de hojas de estilo no se aplica sobre la función directa del XSLT - realizan la transformación del documento utilizando una o varias reglas de plantilla.

1