Universidad de las Ciencias Informáticas

Facultad 1

Título: Arquitectura del Sistema Informatizado de Cooperación Internacional.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autores

Ariel Enrique Novo Rijo.

Laritza González Marrero.

Tutor

Ing. Julio Cesar Isaza Vázquez.

Ciudad de la Habana, Junio de 2008

"Año 50 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de Junio del año 2008.

Ariel Enrique Novo Rijo

_____
Firma del Autor

Laritza González Marrero

_____
Firma del Autor

Ing. Julio Cesar Isaza Vázquez

_____
Firma del Tutor

DATOS DE CONTACTO

Tutor
Ing. Julio Cesar Isaza Vázquez
E-mail: jisaza@uci.cu
Teléfono: 8372760.
Universidad de las Ciencias Informáticas.

AGRADECIMIENTOS

A la Revolución. A la UCI.

A nuestro tutor Julio, al Chony, a Yanier, gracias por todo.

De Laritza:

A mis padres y abuelos, por su dedicación y esfuerzo.

A mi familia y vecinos, por estar cuando yo no pude junto a los míos.

A mis hermanos, por sus oraciones.

A mis profesores, por contribuir a mi formación.

A Ariel, por su esfuerzo.

A mi esposo, Dargel, sin él no hubiera sido posible.

A mi Dios, por ser todo para mí.

De Ariel:

Ufff... hay machos.

Vaya el primero a mis padres: gracias por mi vida.

A mi familia, que es la mejor de todas: los quiero.

A mis amigos "legados" y los nuevos: somos o no somos?

A ti del tamaño del mundo: Yasny te amo.

DEDICATORIA

De Laritza:

A mis padres.

De Ariel:

A mis padres.... A quien más?

RESUMEN

La Dirección de Cooperación Internacional (DCI) de la Universidad de las Ciencias Informáticas (UCI), formada por tres grupos de trabajo: Trámites, Cooperación y Relaciones Públicas, se encarga de gestionar todos los procesos de dicho centro referentes a las misiones y vínculos profesionales en el extranjero.

Como parte de la estrategia de informatización de la UCI se ha decidido informatizar los procesos que realiza la DCI con fines de optimizar su desempeño y alcanzar alto rendimiento en su funcionamiento, por lo que se crea el Proyecto SICI para desarrollar el Sistema Informatizado de Cooperación Internacional, con el que se pretende alcanzar esta meta.

En el presente trabajo se propone una arquitectura de software que sirva de guía para desarrollar este sistema, basada en una serie de parámetros bien definidos para lograr una buena flexibilidad, escalabilidad, bajo los principios de un bajo acoplamiento y utilizando unas buenas técnicas de diseño y programación.

En el trabajo se pueden encontrar una serie de vistas arquitectónicas que describen la arquitectura del Sistema, conjuntamente con un grupo de aspectos trascendentales para esta; así como se podrá contar con una evaluación de la misma y con diferentes métodos evaluativos definidos.

ÍNDICE DE CONTENIDO

INTRODUCCIÓN 1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 5
1.1 Introducción 5
1.2 Fundamentación del tema 5
1.2.1 Arquitectura de Software 5
1.2.1.1 El rol de Arquitecto de Software 6
1.2.1.2 Por qué documentar la Arquitectura 7
1.2.2 Estilos Arquitectónicos 7
1.2.2.1 Estilos arquitectónicos más comunes 7
1.2.3 Patrones 8
1.2.3.1 Características de los patrones 10
1.3 Estado del arte 10
1.4 Metodologías de desarrollo, Herramientas y Lenguajes 13
1.4.1 Metodologías de desarrollo de software 13
1.4.1.1 RUP 13
1.4.1.2 XP 15
1.4.2 Lenguaje de modelado UML 16
1.4.3 Herramienta CASE: Visual Paradigm 17
1.4.4 Sistemas de gestión de contenidos 17
1.4.4.1 Clasificación de los CMS 18
1.4.4.2 CMS de código abierto para sitios web 18
1.4.5 PHP 21
1.4.6 PostgreSQL 22
1.5 Propuesta de solución final 23
1.6 Conclusiones 24
CAPÍTULO 2: DESCRPCIÓN DE LA ARQUITECTURA 25
2.1 Introducción 25
2.2 Representación Arquitectónica 25
2.2.1 4+1 vistas arquitectónicas 27
2.3 Objetivos y restricciones arquitectónicas 28
2.3.1 Interoperabilidad con sistemas legados 28
2.3.2 Distribución física de la DCI 28
2.3.3 Política de manejo de datos 29
2.3.4 Restricciones de acuerdo a la estrategia de diseño 29
2.3.5 Requerimientos de hardware 29
2.3.6 Requerimientos de software 31
2.3.7 Estructura del equipo de desarrollo 31
2.4 Tamaño y rendimiento 32

2.5 Vista de Casos de Uso 32
2.5.1 Subsistema de Trámites 33
2.5.2 Subsistema de Relaciones Públicas 37
2.5.3 Subsistema de Cooperación Internacional 39
2.5.4 Subsistema de Economía 42
2.6 Vista lógica 44
2.7 Vista de Implementación 49
2.8 Vista de Despliegue 52
2.9 Conclusiones 53
CAPÍTULO 3: EVALUACIÓN DE LA ARQUITECTURA 55
3.1 Introducción 55
3.2 Evaluando una arquitectura de software 55
3.2.1 ¿Cuándo una arquitectura puede ser evaluada? 55
3.2.2 ¿Quiénes participan? 56
3.2.3 Planificación de las evaluaciones. 56
3.2.4 ¿Qué resultados produce la evaluación de una arquitectura? 56
3.2.5 Atributos por los cuales puede ser evaluada la arquitectura. 57
3.2.6 ¿Por qué los atributos de calidad son demasiados imprecisos para el análisis? 58
3.2.7 ¿Cuáles son las salidas de una evaluación arquitectónica? 59
3.2.8 ¿Cuáles son los beneficios de realizar una evaluación arquitectónica? 59
3.3 Técnicas de evaluación 59
3.3.1 ¿Cómo se puede realizar una evaluación arquitectónica? 60
3.4 Evaluando la arquitectura propuesta 61
3.5 Conclusiones 63
CONCLUSIONES 64
RECOMENDACIONES 65
BIBLIOGRAFÍA REFERENCIADA 66
BIBLIOGRAFÍA CONSULTADA 67
GLOSARIO 69

ÍNDICE DE FIGURAS

FIGURA 1: GRÁFICA DE RUP EN DOS DIMENSIONES. 14
FIGURA 2: METODOLOGÍA XP (PROGRAMACIÓN EXTREMA). 15
FIGURA 3: REPRESENTACIÓN DE LOS COMPONENTES FUNDAMENTALES DE DRUPAL....25
FIGURA 4: MODELO DE LAS VISTAS ARQUITECTÓNICAS. 27
FIGURA 5: ESTRUCTURA ORGANIZACIONAL DEL EQUIPO DE DESARROLLO....31
FIGURA 6: DIAGRAMA DE CASOS DE USO CRÍTICOS DEL SUBSISTEMA DE TRÁMITES....33
FIGURA 7: DIAGRAMA DE CASOS DE USO CRÍTICOS DEL SUBSISTEMA DE RELACIONES PÚBLICAS. 37
FIGURA 8: DIAGRAMA DE CASOS DE USO CRÍTICOS DEL SUBSISTEMA DE COOPERACIÓN....40
FIGURA 9: DIAGRAMA DE CASOS DE USO CRÍTICOS DEL SUBSISTEMA DE ECONOMÍA. 42
FIGURA 10: REPRESENTACIÓN DE LOS SUBSISTEMAS. 44
FIGURA 11: ESTRUCTURA LÓGICA. 45
FIGURA 12: SISTEMA DE MÓDULOS. 46
FIGURA 13: SISTEMA DE TEMAS. 47
FIGURA 14: SISTEMA DE BÚSQUEDA. 48
FIGURA 15: SISTEMA DE ACCESO A LOS NODOS. 49
FIGURA 16: IMPLEMENTACIÓN DEL SISTEMA DE MÓDULOS. 50
FIGURA 17: DIAGRAMA DE DESPLIEGUE. 52

ÍNDICE DE TABLAS

TABLA 1: DESCRIPCIÓN DE LA VISTA LÓGICA. 46
TABLA 2: DESCRIPCIÓN DEL SISTEMA DE MÓDULOS. 47
TABLA 3: DESCRIPCIÓN DEL SISTEMA DE TEMAS. 48
TABLA 4: DESCRIPCIÓN DEL SISTEMA DE BÚSQUEDA. 48
TABLA 5: DESCRIPCIÓN DEL SISTEMA DE ACCESO A LOS NODOS. 49
TABLA 6: IMPLEMENTACIÓN DEL MÓDULO TRAMITACIÓN. 50
TABLA 7: DESCRIPCIÓN DE ELEMENTOS E INTERFACES DE COMUNICACIÓN. 53

INTRODUCCIÓN

El mundo actual se presenta a la puerta matizado por un sinnúmero de situaciones y acontecimientos que deja a todos con una gama muy amplia de polémicas y puntos de vista divergentes, pero si algo esta bastante claro en estos momentos es la situación mundial de Las tecnologías de la informática y las comunicaciones, o TICs como se le suele llamar. Los ordenadores invaden todos los sectores de la sociedad, incluso ya se han convertido en una herramienta indispensable para muchos, y ya se emplean en funciones tan amplias que han superado con creces las expectativas de casi todos. Las TICs son una herramienta fundamental para alcanzar el desarrollo en cualquier esfera. Todos los países desarrollados del mundo dedican grandes sumas de dinero para llevar a cabo proyectos e inversiones en el sector informático. Es una carrera sin precedentes que evoluciona constantemente y de una forma sorprendentemente rápida, lo que hoy podemos considerar como tecnología de punta, puede dentro de un año considerarse como algo anacrónico o pasado.

Cuba, como una pieza más de la gran maquinaria mundial, se empeña por llevar adelante proyectos y poner en práctica ideas que la sitúen en un escalón decoroso y en una posición en que sea capaz de asimilar todos los beneficios que pueden proporcionar las TICs. El primer paso fue la creación de los Joven Club de Computación, por los que han pasado miles de cubanos y han recibido una preparación en diversas áreas y materias del campo informático. Todo siguiendo la perspectiva de inculcar la necesidad y la importancia del empleo de estas tecnologías para el desarrollo del país. Así mismo se crearon las carreras universitarias en Ingeniería Informática y Ciencias de la Computación para formar personal profesional en este campo. Innumerables son los pasos que ha dado la Revolución sobre el tema, tanto es así se crea una universidad donde estudian jóvenes de todo el país y se dedica única y exclusivamente a las Ciencias Informáticas.

La Universidad de las Ciencias Informáticas (UCI), surge como una idea del Comandante en Jefe Fidel Castro, el mismo expresó: “esta universidad de excelencia debe ser una escuela flexible y capaz de metamorfosearse, de cambiarse, de perfeccionarse” [1], y a la vez ser un “símbolo del futuro de nuestra enseñanza universitaria y del desarrollo venidero del país, basado en el talento colectivo de su pueblo y los cuantiosos y bien preparados recursos humanos que posee” [1]. El papel más importante que le toca desempeñar a la UCI es convertirse en aquella institución que lleve al país a alcanzar un desarrollo informático maduro en todos sus renglones. Para esto se ha propuesto una estrategia de informatización, automatización de procesos, servicios y actividades, logrando excelentes resultados y escalando cada día para llegar a materializar en un futuro muy cercano su misión principal.

La UCI crece constantemente, y al mismo tiempo va creciendo su experiencia en el desarrollo de software, los que son empleados en distintas aristas de la sociedad cubana. En la economía, en el sistema de Salud Pública y educacional; incluso se han obtenido resultados en la exportación. Todo este proceso va unido al incremento sistemático de la informatización del centro, que por sus características requiere de complejos sistemas de gestión.

La Dirección de Cooperación Internacional (DCI) de la UCI, es una de las máximas exponente del dinamismo y la responsabilidad que caracteriza la Universidad. Esta Dirección se encarga de gestionar, esencialmente, todo lo referente a las misiones que cumple la Universidad en el extranjero y todo lo que se pueda derivar de las mismas. La DCI está formada por tres grupos que se especializan en diferentes actividades para poder llevar a cabo su función principal, el grupo de Cooperación internacional, el grupo de Trámites y el de Relaciones Públicas.

Todo este trabajo implica una serie de procesos y actividades con alto nivel de complejidad. Actualmente no todos los objetivos se logran con la eficiencia y organización que se desea. No existe una herramienta automatizada que ayude a optimizar el desempeño de todos los grupos que conforman la Dirección. Solamente cuenta con un sistema de gestión muy pobre que resuelve algunas dificultades el grupo de Trámites. Dicho sistema está basado en tecnología propietaria y no se considera una solución para resolver los problemas de la Dirección, incluso ni del propio grupo de Trámites que lo emplea. De manera general toda la información que se maneja está en formato duro, y aunque se trabaja con buena organización se hace muy engorrosa la búsqueda de algún dato específico puesto que puede suponer varias consultas a diferentes archivos y no se logra en el tiempo deseado. Sin contar que la DCI debe interactuar con otros agentes externos y los canales de comunicación no son los más óptimos. Se debe adicionar además los errores humanos que pueden ser introducidos, por ejemplo a la hora de registrar una determinada información.

Con esta ilustración de la **situación problémica** en la DCI, llega el siguiente **problema científico:** ¿cómo construir una arquitectura para el Sistema Informatizado de Cooperación Internacional (SICI) que sea capaz de automatizar los procesos de la DCI de la UCI, con suficiente flexibilidad y escalabilidad para asimilar las características de esta Dirección?

Constituye el **objeto de estudio:** Los estilos arquitectónicos en aplicaciones Web apropiados para sistemas informatizados de cooperación internacional. El **campo de acción** será: La arquitectura del SICI de la UCI.

El **objetivo general** de este trabajo es**:** diseñar una arquitectura de software, que permita el desarrollo

de un sistema informatizado para la DCI basada en un bajo acoplamiento que a su vez permita una alta integración y cohesión entre sus componentes; teniéndose así los siguientes **objetivos específicos**:

- Estudiar las tendencias arquitectónicas actuales.
- Elaborar y construir la propuesta arquitectónica para desarrollar el sistema.
- Crear una propuesta arquitectónica que facilite desarrollar un sistema que cumpla con los requerimientos de los usuarios.

Para alcanzar estos objetivos se han trazado las siguientes **tareas**:

- Consulta de la bibliografía especializada en el desarrollo de arquitecturas en sistemas de gestión.
- Análisis y definición de las herramientas y tecnologías para el desarrollo de sistema.
- Estudio de los patrones de diseño más adecuados.
- Definición de las políticas para el manejo de transacciones, concurrencia, rendimiento, acceso a sistemas legados.

La **idea a defender** que se persigue es: la arquitectura del SICI desarrollada sobre un sistema de gestión de contenidos servirá para desarrollar un sistema que garantizará una alta flexibilidad y escalabilidad.

Para la realización de este trabajo se emplearon los siguientes **métodos de investigación científica:**

Métodos Teóricos:

- Analítico-Sintético: Este método permite extraer lo esencial de la bibliografía consultada de Arquitectura.
- Análisis Histórico Lógico: Permite deducir lógicamente cómo ha evolucionado la arquitectura.

Métodos Empíricos:

- Entrevista: Se realizaron entrevistas en diferentes instituciones del país que trabajan con sistemas como el que se pretende modelar su arquitectura, además al personal de la DCI para recopilar información y establecer acuerdos.

El presente trabajo consta de tres capítulos, el primero dedicado al estudio de los conceptos fundamentales de la arquitectura de software y los principales estilos arquitectónicos, así como ha estudiar diferentes herramientas y tecnologías que se pueden utilizar para la construcción de una arquitectura de software. El segundo contiene la descripción de la arquitectura propuesta para el SICI, basado entre otros aspectos en una serie de vistas que proporcionan una visión global de dicha arquitectura. En el tercer capítulo se recogen una serie de parámetros que sirven de guía para evaluar una arquitectura de software y además se evalúa la arquitectura propuesta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Este capítulo tiene como objetivo exponer los conceptos básicos sobre la arquitectura, tratar temas claves del campo como los patrones y estilos arquitectónicos. Se realizará un análisis de las tendencias actuales de la arquitectura de software, así como un análisis de las tecnologías, herramientas y lenguajes que van a ser usadas en el proceso de desarrollo del SICI.

1.2 Fundamentación del tema

1.2.1 Arquitectura de Software

La arquitectura de software ha sido tratada por muchos autores y por su puesto todos tienen diferentes visiones del tema y exponen diferentes ideas y criterios al respecto.

Una de las definiciones más reconocidas del concepto de arquitectura de software es la de Clements:

“La AS es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión del detalle inherente a la mayor parte de las abstracciones.” [2]

La definición oficial de Arquitectura del Software es la establecida por la IEEE en su documento Std 1471-2000:

“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”.

En un sentido amplio se puede estar de acuerdo en que la Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de:

- Definir los módulos principales.
- Definir las responsabilidades que tendrá cada uno de estos módulos.
- Definir la interacción que existirá entre dichos módulos.
- Control y flujo de datos.

- Secuenciación de la información.
- Protocolos de interacción y comunicación.
- Ubicación del software en el hardware.

La Arquitectura del Software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño.

1.2.1.1 El rol de Arquitecto de Software

El Arquitecto Software tiene una responsabilidad técnica global sobre el proyecto. Por ello, para el correcto desempeño de su rol debe poseer un amplio conjunto de conocimientos tecnológicos que le permitan desarrollar un sólido liderazgo técnico, imprescindible a la hora de tomar decisiones efectivas y a tiempo, y convencer de su idoneidad a los responsables de gestión y desarrollo. Su trabajo fundamental consiste en (1) desarrollar un conjunto de vistas tempranas del sistema en su conjunto que incluyan sus “componentes” principales, el comportamiento de estos componentes según lo percibe el resto del sistema y la forma en que interactúan y se coordinan para cumplir con la misión del sistema y (2) entender y comunicar cómo trasladar estas vistas a la implementación.

Una de las facetas en que el Arquitecto Software debe poseer profundos conocimientos y amplia experiencia es la relacionada con los mecanismos de comunicación e integración entre componentes software existentes, con el propósito de elegir la opción pertinente y las herramientas adecuadas en cada caso.

Estas son las funciones que debe desempeñar el Arquitecto en un proyecto de desarrollo de software:

- Identifica las tecnologías que serán usadas en el proyecto.
- Recomienda una metodología de desarrollo.
- Proporciona la estructura general y diseño de la aplicación.
- Se asegura que el proyecto está adecuadamente definido y el diseño debidamente documentado.
- Establece las guías para la codificación, para el manejo de excepciones.
- Identifica las tareas de implementación.
- Proporciona la guía para desarrollar la lógica de negocio.

- Orienta a los desarrolladores en las tareas difíciles.
- Establece los lineamientos de codificación y los hace cumplir.
- Ayuda al administrador del proyecto a estimar los costos.
- Ayuda a ubicar al personal según sus habilidades en las posiciones adecuadas dentro del proyecto.
- Chequea que el diseño gráfico propuesto sea factible.
- Se asegura que los requerimientos de negocio determinados por el analista de negocio sean suficientes.
- Proporciona consejos técnicos y guía al administrador del proyecto.
- Se asegura de que los patrones de diseño sean usados, mantenidos y extendidos.

1.2.1.2 Por qué documentar la Arquitectura

La documentación de la Arquitectura debe ser lo suficientemente abstracta como para ser entendida por nuevos miembros, teniendo propósitos de educación. Suficientemente detallada como para servir de plan para la construcción, sirve de vehículo primario para la comunicación entre *stakeholders*. Debe tener suficientemente información para servir de base al análisis y desarrollo del sistema.

1.2.2 Estilos Arquitectónicos

Un estilo arquitectónico es un concepto descriptivo que define una forma de articulación u organización arquitectónica. Los estilos arquitectónicos son familias o grupos de sistemas de software que siguen el mismo patrón estructural.

Es una especialización de tipos de elementos y tipos de relaciones, junto con un conjunto de restricciones de cómo pueden ser usadas. Un estilo es una especialización de un tipo de vista y refleja patrones recurrentes de interacción independientes de cualquier sistema.

1.2.2.1 Estilos arquitectónicos más comunes

Dentro de los principales estilos arquitectónicos, se pueden identificar un conjunto de subgrupos que recogen sistemas muy específicos:

Estilos de flujos de datos: Sucesión de transformaciones de los datos de entrada.

- Sistemas de filtros y tuberías (*pipe & filter*).
- Sistemas de procesamiento por lotes (*batch*).

Estilos basados en llamada y retorno: Son sistemas que reflejan la estructura del lenguaje de programación.

- Sistemas de programación principal y subrutina.
- Sistemas orientados a objetos.
- Sistemas organizados en capas.

Estilos de componentes independientes: Grupo de sistemas cuyos componentes se comunican mediante el paso de mensajes.

- Sistemas cliente/servidor.
- Sistemas de eventos.
- Sistemas orientados a servicios.

Estilos centrados en los datos: Estos sistemas se caracterizan por el acceso compartido a un banco de datos central.

- Repositorios (datos pasivos).
- Pizarras (datos activos).

Estilos de máquinas virtuales: Los sistemas bajo estos estilos simulan una funcionalidad no nativa del entorno.

- Intérpretes.
- Sistemas basados en reglas.

Estilos heterogéneos: Son sistemas que combinan características de varios estilos.

1.2.3 Patrones

Todo equipo de desarrollo de software con un nivel medio de experiencia, aplica diferentes técnicas muy particulares para solucionar determinados problemas. Estos equipos de desarrollo han tenido que

enfrentar diferentes situaciones problemáticas similares anteriormente y dándole solución a estas, han adquirido ciertos métodos o estructuras que ha su vez se convierten en la solución de un problema determinado.

De ahí que la reutilización efectiva de las experiencias de solución a problemas específicos de profesionales experimentados se ha convertido en patrones.

“Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma”. [3]

Esta definición se refiere a patrones de ciudades y edificios, pero son perfectamente aplicables en el del desarrollo de software.

Los Patrones inicialmente fueron aplicados en la fase de diseño de los sistemas de información, sin embargo existen otros ámbitos de la ingeniería del software donde se puede aplicar el concepto genérico de patrón.

Hay patrones que describen soluciones para todo, desde el análisis hasta el diseño y desde la arquitectura hasta la implementación. Por ejemplo mostramos algunos:

- Patrones organizativos: Describen la estructura y prácticas de las organizaciones humanas, especialmente las productoras de software.
- Patrones de análisis: Describen un conjunto de prácticas destinadas a elaborar modelos de los conceptos principales de la aplicación que se va a construir. La intención principal de estos patrones es ayudar a las personas que realizan el trabajo de modelado, pues no siempre tienen experiencia al respecto y, en la mayoría de los casos, construyen sus modelos sin referencia alguna.
- Patrones de arquitectura: Expresan un paradigma fundamental para estructurar u organizar un sistema software. Proporcionan un conjunto de subsistemas o módulos predefinidos, con reglas y guías para organizar las relaciones entre ellos.
- Patrones de diseño: Proporciona un esquema para refinar los subsistemas o componentes de un sistema software y las relaciones entre ellos. Describe estructuras recurrentes de comunicar componentes que resuelven un problema de diseño en un contexto particular. Son patrones de un

nivel de abstracción menor que los patrones de arquitectura. Están por lo tanto más próximos a lo que sería el código fuente final. Su uso no se refleja en la estructura global del sistema.

- Patrones de programación (*Idioms patterns*): Son patrones de bajo nivel, específicos a un lenguaje de programación determinado. Describen cómo implementar aspectos particulares de los componentes de un patrón de diseño usando las características y potencialidades de un lenguaje de programación concreto.

1.2.3.1 Características de los patrones

El concepto de patrón, aunque tiene un carácter general, se piensa que debe contar con las siguientes propiedades para que sea eficientemente reutilizado.

- Ser una solución de sentido común que forma parte del conocimiento de un diseñador experto.
- Debe facilitar la comunicación entre diseñadores, promulgan un vocabulario común.
- Debe facilitar el aprendizaje de diseñadores no expertos.
- La solución que describe ha debido ser probada más de una vez en distintos casos.
- Debe proveer una plantilla conceptual que muestra el núcleo de una solución a un problema de diseño específico.
- Debe describir participantes y relaciones entre ellos: describen módulos, estructuras del sistema y mecanismos complejos.

1.3 Estado del arte

Cuba esta conformado por una enorme cantidad de instituciones, ministerios, centros de estudio, organizaciones gubernamentales, por citar una parte de la gran maquinaria que conforma todo el sistema cubano. En casi todos estos centros se gestionan de alguna manera procesos referentes a las relaciones internacionales, pero en sentido general, el país no presenta un alto de nivel de informatización y es por esto que casi ninguna de estas estructuras antes mencionadas cuenta con sistemas informáticos para llevar a cabo estos procesos.

Cabe entonces la lógica de apuntar hacia horizontes internacionales en busca de sistemas que se encarguen de automatizar procesos de relaciones internacionales, pero es ahí donde viene la simple y

negativa noticia que cada país maneja estos asuntos de maneras muy diferentes y particulares en dependencia de sus características y regulaciones, que ciertamente son muy disímiles.

En el escaso grupo de sistemas en Cuba que se dedican a la gestión de procesos de relaciones internacionales se pueden encontrar:

- El Sistema de Trámites del Ministerio de las Informáticas y las Comunicaciones (MIC): es un sistema web del cual no hay prácticamente ninguna documentación que pueda proporcionar niveles de detalles sobre su arquitectura pero a grandes rasgos se puede mencionar que es un aplicación muy sencilla desarrollada sobre Microsoft.Net, específicamente utilizando ASP.Net, no esta prevista para grandes volúmenes de datos, emplea Microsoft Access como sistema gestor de base de datos, que desde este momento ya se puede descartar emplear alguna de estas tecnologías para desarrollar el SICI puesto que son software propietario y no son multiplataforma.
- El Sistema de Atención a Misiones del Ministerio de la Industria Sideromecánica (SIME): solamente con mencionar que consiste en una aplicación de escritorio ya no se hace necesario realizar un análisis de este sistema puesto que la arquitectura del SICI tiene que basarse primariamente en el modelo cliente/servidor donde el cliente solamente necesita de un navegador web para acceder al sistema - que estará corriendo sobre un servidor web - a través de la red, mientras que en una aplicación de escritorio cada usuario debe tener instalada la aplicación en su estación de trabajo lo que representa un consumo de recursos no deseado por ningún usuario, además que realizar un mantenimiento de este tipo de software implica la reinstalación del sistema cada vez que se le haga una actualización o salga una nueva versión del producto. En el caso específico de este sistema no hay una separación lógica de ningún componente o capa de la aplicación como podría ser la presentación, la lógica del negocio o el acceso a datos en un sistema tradicional 3 capas.
- El Sistema de Tramitación del Ministerio de Educación Superior (MES): Este sistema opera sobre Web, e incluye una potente herramienta para realizar cualquier tipo de reporte. Tiene el inconveniente que por cada cliente que hace una petición al sistema, se ejecuta una aplicación diferente en el servidor, además su base de datos está diseñada en Microsoft Access, como se mencionaba anteriormente no es una tecnología aceptada en la UCI para el desarrollo.

Visto que no hay muchas posibilidades de arquitecturas candidatas – prácticamente ninguna – en sistemas que realizan funciones relacionadas a las que debe realizar el SICI, sería prudente chequear cuáles son las tendencias arquitectónicas en los sistemas de gestión de la Universidad, ya que

independientemente a que el SICI se enfoca directamente hacia los procesos de relaciones internacionales, este no deja de ser en fin uno de estos – sistema de gestión -.

- Akademos: Su función principal es la gestión académica del Centro. De manera general su arquitectura se basa en el estilo arquitectónico 3 capas, hace una perfecta implementación de este patrón arquitectónico, las capas se encuentran bien definidas, logrando un bajo acoplamiento entre sus componentes. Tiene su propio *framework* implementado lo que permite una escalabilidad adecuada al sistema. El inconveniente principal es que se basa totalmente en la plataforma .Net y su sistema gestor de base de datos es SQLServer, ambas tecnologías propietarias descartadas para el desarrollo en la UCI. Vale aclarar que actualmente ya está la nueva propuesta de arquitectura para Akademos basada en software libre, utilizando el *framework* Symnfony, el cual se basa en el patrón modelo-vista-controlador (MVC), y como gestor de base de datos PostgreSQL.
- Kainos: Este proyecto surge con el objetivo principal de informatizar los procesos de la Federación Estudiantil Universitaria (FEU) de la UCI, extendiéndose después a la FEU nacional, la Unión de Jóvenes Comunistas de la Universidad y del país, por lo que han desarrollado varios sistemas, todos basados en arquitecturas diferentes. El uso de PostgreSQL como sistema gestor de base de datos ha sido común, mientras que se han empleado *frameworks* diferentes como J2EE, Symnfony y el CMS Drupal. En todos los casos la arquitectura se basa en tecnologías libres. Se observan patrones como MVC, en el caso de la plataforma J2EE se utilizan los *frameworks* Hibernate, Spring y JSF para lograr una separación de los componentes fundamentales del sistema, por su parte Drupal se basa en una arquitectura modular totalmente desacoplada en módulos.
- Sistema de Control de Acceso: Como su nombre lo indica, este sistema se encarga, de manera general, de gestionar el control del acceso a la Universidad. Está desarrollado sobre la plataforma .Net siguiendo el estilo arquitectónico tres capas. A pesar de utilizar tecnología propietaria – utiliza además SQLServer como gestor de base de datos – posee otras cualidades arquitectónicas como la seguridad, la flexibilidad y la interoperabilidad que destacan por su implementación. El uso de .Net como plataforma de desarrollo se debe a las facilidades que brinda este *framework* para interactuar con diversos periféricos que son necesarios dentro de los procesos de este sistema.

1.4 Metodologías de desarrollo, Herramientas y Lenguajes

1.4.1 Metodologías de desarrollo de software

En el transcurso de las dos últimas décadas se consideró que el éxito en el desarrollo del software se alcanzaba a partir de las notaciones de modelado y posteriormente las herramientas, sin embargo las expectativas no fueron satisfechas. Esto se debe en gran medida a que un elemento de gran importancia en el desarrollo del software había sido postergado, la metodología de desarrollo. De nada sirven buenas notaciones de modelado y herramientas si no se garantizan directivas para su aplicación.

Una metodología de desarrollo de software es un proceso en el que se define “quién” esta haciendo “qué”, “cómo” y “cuándo”.

La primera pregunta que debemos plantearnos a la hora de elegir una metodología de desarrollo es: ¿Una metodología ágil o una metodología robusta? No cabe duda de que la gran mayoría de los proyectos pueden beneficiarse en gran medida del uso de una metodología ágil, pero indudablemente existen otros proyectos y entornos de trabajo en los que es necesario que el proyecto se desarrolle con mayor control o guiados por un plan.

Actualmente las dos metodologías que más se destacan en el campo de la Ingeniería de Software son: RUP (*Rational Unified Proccess*), metodología de desarrollo software robusta y XP (*eXtreme Programming*), metodología ágil.

1.4.1.1 RUP

La metodología RUP, divide en 4 fases el desarrollo del software: inicio, elaboración, construcción y transición, cada una de ellas con los objetivos de: determinar la visión del proyecto, definir la mejor arquitectura, obtener la capacidad operacional inicial y llegar a obtener el *release* del proyecto respectivamente. Estas fases son desarrolladas a través de iteraciones y ciclos de desarrollo, en una misma fase puede que exista más de una iteración, en cambio un ciclo se extiende desde la fase de inicio hasta la de transición, al final del mismo se obtiene una versión del producto. Durante una iteración se desarrollan los nueve flujos de trabajo, en unos el esfuerzo requerido es mayor que en otros, en dependencia de la fase en que se encuentre el desarrollo del proyecto. Este esquema brinda una descripción de las fases y flujos de trabajo que conforman la metodología RUP.

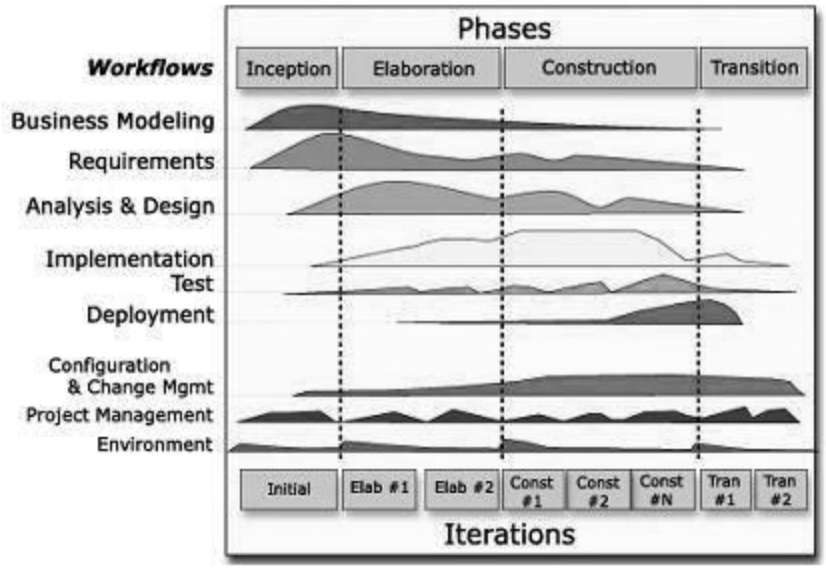

Figura 1: *Gráfica de RUP en dos dimensiones.*

Flujos de trabajo:

- Modelamiento del Negocio: Identifica y detalla los procesos de negocios, así como quién inicia o participa, y cuáles son las actividades necesarias a ser automatizadas.
- Requerimientos: Establece qué es lo que el sistema debe hacer, incluyendo funcionalidades y restricciones.
- Análisis y Diseño: Describe cómo será realizado el sistema, indica cómo se debe programar.
- Implementación: Define la organización de los componentes y los nodos, así como la estructura de capas de la aplicación.
- Pruebas: Busca los defectos del software durante el ciclo de vida.
- Despliegue: Produce el *release* del producto.
- Gestión y Configuración de cambios: Describe cómo controlar los elementos producidos por todos los miembros del equipo de desarrollo en cuanto a: utilización/actualización concurrente de elementos, controlador de versiones, etc.
- Administración del proyecto: Realiza actividades con las que se busca que el producto satisfaga al cliente.
- Ambiente: Administrando el ambiente de desarrollo.

RUP define como sus principales elementos: trabajadores (quién), actividades (cómo), artefactos (qué), flujo de actividades (cuándo).

1.4.1.2 XP

La Programación Extrema o XP (en inglés *eXtreme Programing*) es una metodología de desarrollo muy utilizada para proyectos de corta duración, cuyos equipos son pequeños y su fecha de entrega está retardada.

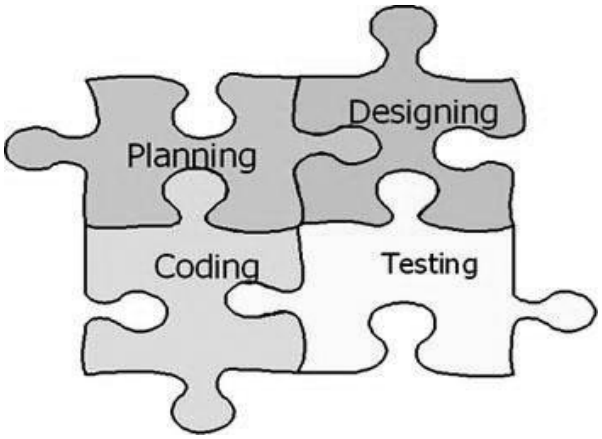

Figura 2: *Metodología XP (Programación Extrema).*

Las actividades de XP son: escuchar, diseñar, codificar y probar; las mismas se realizan indefinidamente.

Prácticas Básicas de XP

- El juego de la Planificación (*Planning Game*): Los objetivos de la siguiente versión están definidos por la prioridad de los módulos, fechas de entrega, estimaciones de funciones y sus consecuencias.
- Versiones Pequeñas (*Short Releases*): Se realizan versiones pequeñas con el objetivo de añadir periódicamente (en cada iteración) aquellas funcionalidades que el cliente necesite.
- Metáfora del Sistema (*Metaphor*): Este término es el que reemplaza al que conocemos como arquitectura, debe estar en lenguaje común, entendible para todos (Cliente y Desarrolladores), puede cambiar permanentemente. Básicamente consiste en que cada proyecto es guiado por una historia simple de cómo funciona el sistema en general.

- Diseño Simple (*Simple Designs*): El diseño se plasma en tarjetas CRC (Clase-Responsabilidad-Colaboración), no se implementan características innecesarias.
- Pruebas Continuas (*Testing*): Los desarrolladores escriben pruebas unitarias y los clientes especifican pruebas funcionales. Las mismas son escritas antes que el código.
- Refactorización (*Refactoring*): Esta práctica consiste en cambiar el código con el propósito de que este sea más legible, pero sin cambiar el funcionamiento.
- Programación por parejas (*Pair Programming*): El código es escrito por dos personas trabajando en la misma computadora.
- Posesión Colectiva del Código (*Collective Code Ownership*): Cualquier programador puede cambiar cualquier parte del sistema en cualquier momento, siempre se utilizan estándares y se excluyen los comentarios, es decir nadie es dueño de un módulo. Para realizar integraciones con todo el código permanentemente, los test siempre deben funcionar al 100%.
- Integración continua (*Continuous Integration*): Antes y después de la integración se deben realizar los casos de pruebas, además el cliente debe estar presente para hacerle pruebas funcionales.
- Semana laboral de 40 horas (*40-Hour Week*): Sólo se trabaja 40 horas por semana por cada trabajador.
- Cliente en el Sitio (*On Site Customer*): El cliente (o un representante del mismo que conozca el negocio) es parte del equipo de trabajo, debe estar a tiempo completo con el equipo de desarrollo. Lo ideal sería que fuera analista.
- Estándares de Codificación (*Coding Standard*): Todo el código debe estar escrito de acuerdo a un estándar de codificación.

La metodología XP tiene cuatro fases: Planeamiento, Producción, Mantenimiento y Muerte.

1.4.2 Lenguaje de modelado 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. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas”. [4]

“Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional), pero no especifica en sí mismo qué metodología o proceso usar”. [5]

1.4.3 Herramienta CASE: Visual Paradigm

Visual Paradigm es una herramienta CASE (Ingeniería de Software Asistida por Ordenador, en inglés *Computer Aided Software Engineering*) muy potente que utiliza UML como lenguaje de modelado. Soporta el ciclo de vida completo de desarrollo de un software, desde la fase de análisis hasta el despliegue del mismo. Permite realizar ingeniería directa o inversa sobre el software, es capaz a partir de un modelo relacional en diferentes SGBD, desplegar todas las clases asociadas al las tablas. Soporta múltiples usuarios trabajado sobre el mismo proyecto. Se caracteriza por su robustez, usabilidad y portabilidad.

1.4.4 Sistemas de gestión de contenidos

Los sistemas de gestión de contenidos (*Content Management Systems* o CMS) son un software que se utiliza principalmente para facilitar la gestión de webs, ya sea en Internet o en una intranet, y por eso también son conocidos como gestores de contenido web (*Web Content Management* o WCM). Hay que tener en cuenta, sin embargo, que la aplicación de los CMS no se limita sólo a las webs. [6]

Los CMS son una interfaz que administran una o varias bases de datos donde se encuentra el contenido del sitio web. Permiten modificar el diseño del sitio en cualquier momento sin preocuparse del contenido, o viceversa, puesto que manejan ambos de manera independiente.

Lo más trascendental de los CMS es que son aplicaciones prefabricadas, esto tiene la ventaja de que con conocimientos técnicos muy básicos o casi ninguno, se puede lograr un resultado de mucha calidad. Por otra parte permiten a los desarrolladores más avanzados interactuar con el mismo profundamente, ya que los CMS son una plataforma muy escalable y mediante el uso de *plug-ins* que pueden ser desarrollados por cualquier desarrollador, se puede extender ampliamente su funcionalidad.

1.4.4.1 Clasificación de los CMS

Existe una gama muy amplia de Sistemas de gestión de contenido. Todos no permiten gestionar el mismo tipo de sitio, por lo que se pueden clasificar teniendo en cuenta este criterio. A continuación se muestran algunas clasificaciones:

- Genéricos: Están diseñados para dar solución a problemas específicos, constituyen una plataforma flexible para desarrollar e implementar diversas aplicaciones. Pueden servir para construir soluciones de gestión de contenidos, para soluciones de comercio electrónico, blogs, portales, etc.
- Foros: Son espacios de debate *online*, estos sitios son utilizados por los usuarios para la discusión de diversos temas.
- Portales: Sitio web con contenido y funcionalidad diversa que sirve como fuente de información o como soporte a una comunidad.
- e-Learning: Son entornos virtuales de aprendizaje, o sea aulas virtuales, permiten exponer materiales para diferentes cursos y además son muy útiles para evaluaciones online.
- Wikis: Son sitios donde cualquier usuario puede aportar cualquier tipo información o enriquecer una ya existente. Usualmente esta información se expone en forma de artículo.

Es importante mencionar que como todo tipo de software que hay en el mercado, los CMS también se agrupan en dos grupos, (1) los CMS de código abierto y (2) los propietarios.

1.4.4.2 CMS de código abierto para sitios web

- Drupal: Es un software que permite de una manera sencilla a una persona o una comunidad de usuarios publicar, gestionar y organizar una gran variedad de contenido en un sitio web. Decenas de miles de personas y organizaciones han utilizado Drupal para establecer diferentes tipos de sitios web. Drupal es un software de código abierto bajo la licencia GPL, y es mantenido y desarrollado por una comunidad de miles de usuarios y desarrolladores.

  Requerimientos:

  - Lenguaje de programación: PHP.
  - Sistema gestor de base de datos: MySQL ó PostgreSQL.
  - Servidor web: Apache, IIS.

- Licencia: GPL.

- e107: CMS codificado en PHP y utiliza el popular sistema gestor de base de datos MySQL para almacenar el contenido. Es totalmente libre y personalizable, y en constante desarrollo. Posee una gran selección de *themes* y módulos, además es muy flexible.

  Requerimientos:

  - Lenguaje de programación: PHP.
  - Sistema gestor de base de datos: MySQL.
  - Servidor web: Apache, IIS.
  - Licencia: GPL.

- eZ Publish: Es uno de los líderes de los CMS y plataforma de desarrollo en el mundo. Incluye funcionalidades avanzadas para publicaciones web, intranets, sitios de comercio electrónico y mas. Es usado por miles de empresas en el mundo. Esta respaldado por la firma eZ systems, que le brinda soporte y desarrollo.

  Requerimientos:

  - Lenguaje de programación: PHP.
  - Sistema gestor de base de datos: MySQL ó PostgreSQL.
  - Servidor web: Apache u otro con soporte para PHP.
  - Licencia: GPL.

- Jomala: Muy usado mundialmente para soportar desde las más simples páginas personales hasta las más complejas aplicaciones web corporativas. Ofrece una plataforma de desarrollo potente para que los desarrolladores puedan desarrollar sus propios *add-ons* que puedan extender las prestaciones del CMS.

  Requerimientos:

  - Lenguaje de programación: PHP.
  - Sistema gestor de base de datos: MySQL.
  - Servidor web: Apache.
  - Licencia: GPL.

- Plone: Basado en Zope y desarrollado en Phyton, es un CMS muy potente. Cuenta con una comunidad de desarrollo muy grande en todo el mundo. Puede utilizarse como servidor intranet o extranet, un Sistema de Publicación de documentos y una herramienta de trabajo en grupo para colaborar entre entidades distantes.

  Requerimientos:

  - Lenguaje de programación: Phyton.
  - Sistema gestor de base de datos: Zope.
  - Servidor de aplicación: Zope.
  - Servidor web: Apache, IIS, Zope.
  - Licencia: GPL.

- Postnuke: su código es orientado a objetos y muy modular. Contiene un grupo de herramientas que permiten desarrollar sitios muy complejos con mucha facilidad. Cuenta con una comunidad de usuarios muy grande por lo que es muy fácil conseguir ayuda online.

  Requerimientos:

  - Lenguaje de programación: PHP.
  - Sistema gestor de base de datos: MySQL.
  - Servidor web: Apache, IIS.
  - Licencia: GPL.

- XOOPS: Es una herramienta ideal para desarrollar pequeños o grandes sitios web dinámicos de comunidades, portales de intranets, portales corporativos, web blogs. XOOPS es el acrónimo de Sistema de Portal extensible Orientado a Objetos (en inglés: *eXtensible Object Oriented Portal System*). La comunidad de XOOPS tiene decenas de sitios web oficiales de ayuda online a usuarios.

  Requerimientos:

  - Lenguaje de programación: PHP.
  - Sistema gestor de base de datos: MySQL.
  - Servidor web: Apache, IIS.

- Licencia: GPL.

1.4.5 PHP

PHP es un lenguaje de secuencia de comandos de servidor diseñado específicamente para la Web. Dentro de una página Web puede incrustar código
PHP que se ejecutará cada vez que se visite una página. El código PHP es interpretado en el servidor Web y genera código HTML y otro contenido que el visitante puede ver. [7]

Algunas de sus características:

- Alto rendimiento: PHP es muy eficiente, mediante el uso de un único servidor, puede servir millones de accesos al día. [7]
- Interfaces para una gran cantidad de sistemas de base de datos diferentes: Dispone de una conexión propia a todos los sistemas de base de datos. Además de MySQL, puede conectarse directamente a las bases de datos de PostgreSQL, mSQL, Oracle, dbm, filepro, Hyperwave, Informix, InterBase y Sybase, entre otras. El uso de ODBC (del inglés *Open Database Connectivity Standard*, Estándar de conectividad abierta de base de datos) permite establecer una conexión a cualquier base de datos que suministre un controlador ODBC. [7]
- Bibliotecas incorporadas para muchas tareas Web habituales: Como se ha diseñado para su uso en la Web, PHP incorpora una gran cantidad de funciones integradas para realizar útiles tareas relacionadas con la Web. Puede generar imágenes GIF al instante, establecer conexiones a otros servicios de red, enviar correos electrónicos, trabajar con *cookies* y generar documentos PDF, todo con unas pocas líneas de código. [7]
- Bajo coste: PHP es gratuito. [7]
- Facilidad de aprendizaje y uso: La sintaxis de PHP se basa en otros lenguajes de programación, principalmente en C y Perl. [7]
- Portabilidad: PHP esta disponible para una gran cantidad de sistemas operativos diferentes. Se puede escribir código PHP en todos los sistemas operativos gratuitos del tipo Unix, como Linux y FreeBSD, versiones comerciales de Unix, como Solaris e IRIX o en las diferentes versiones de Microsoft Windows. Su código funcionará sin necesidad de aplicar ninguna modificación a los diferentes sistemas que ejecute PHP. [7]

- Acceso al código abierto: Dispone de acceso al código fuente de PHP. A diferencia de los productos comerciales y de código cerrado, si desea modificar algo o agregar un elemento al programa, puede hacerlo con total libertad. No necesitará esperar a que el fabricante publique parches, ni tendrá que preocuparse porque el fabricante cierre sus puertas o decida abandonar el producto. [7]

1.4.6 PostgreSQL

PostgreSQL es un Sistema Administrador de Bases de Datos Relacionales –RDBMS - orientada a objetos que tiene las características tradicionales de los sistemas de bases de datos comerciales con mejoras que se encuentran en la próxima generación de sistemas DBMS. PostgreSQL es libre y el código fuente completo está disponible. [8]

Se distribuye bajo la licencia BSD perteneciente al grupo de licencias de software libre. PostgreSQL es concebido como una arquitectura cliente / servidor, lo cual requiere procesos separados para cada cliente y servidor, y diversos procesos de ayudante.

Características esenciales:

PostgreSQL permite una alta concurrencia, elimina la necesidad de bloqueos explícitos, ya sea por tablas o por filas, su estrategia se basa un sistema denominado MVCC o Acceso Concurrente Multiversión (en inglés *Multi Version Concurrent Access*) este permite que mientras un proceso está escribiendo en una tabla, otros accedan a la misma tabla, siempre se obtiene una versión consistente de lo último a lo que se le hizo *commit*.

Además cuenta con características comunes a otros RDBMS, como el uso de llaves Foráneas, disparadores, vistas, integridad transaccional, herencia de tablas, tipos de datos y operaciones geométricas.

En cuanto al tipo de datos, PostgreSQL posee un abundante grupo de tipos de datos nativos, brinda soporte para números de precisión arbitraria, texto de largo ilimitado, figuras geométricas (y funciones asociadas), direcciones IP, direcciones MAC, arreglos. Además los usuarios pueden crear sus propios tipos de datos, que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL.

1.5 Propuesta de solución final

Para construir la propuesta de solución del presente trabajo, atendiendo al rol que corresponde el mismo (arquitectura), fue necesario utilizar como guía los lineamientos arquitectónicos de la Universidad, que tienen como objetivos organizar y centrar el Proyecto de Arquitectura y Soporte de la Informatización de la UCI. Luego guiándose bajo esta línea de trabajo, se hace un estudio de diferentes conceptos, tecnologías y herramientas que pueden dar solución a los objetivos de este trabajo.

Con vistas a definir los lineamientos arquitectónicos para construir el Sistema Informatizado de Cooperación Internacional se define la propuesta de esta manera:

En primer lugar se decide el empleo RUP como metodología de desarrollo de software; es una metodología que dará buen soporte al desarrollo por sus cualidades antes mocionadas.

Teniendo en cuenta que RUP hace uso de UML como lenguaje de modelado para sus diagramas y plantillas, se utilizará la herramienta CASE Visual Paradigm (versión 6.0 Enterprise Edition), esta además de cumplir perfectamente con todos los estándares UML es una herramienta gratis que la Universidad ha pagado por su licencia.

La plataforma de desarrollo será el CMS Drupal (versión 5.2), por su potencia y flexibilidad, y a la vez sencillez lo que permitirá construir un sistema de alta calidad y excelentes prestaciones que den respuesta a los requerimientos de los clientes, además por su basta documentación y amplio conocimiento de uso por desarrolladores de la Universidad y del equipo del proyecto.

Esta versión de Drupal utiliza PHP 5 como lenguaje de programación, - lenguaje muy potente con amplias funciones - con vistas a incorporar nuevas funciones a Drupal o modificar las ya existentes, se utilizará el Entorno de Desarrollo Integrado (IDE, siglas en inglés de *Integrated Development Environment*) Zend Studio 5.0, disponible para desarrolladores profesionales que agrupa todos los componentes de desarrollo necesarios para ciclo de desarrollo de aplicaciones PHP.

Como sistema gestor de base de datos se optará por PostgreSQL, se integra perfectamente con PHP y posee un rendimiento muy alto, actualmente es la opción de miles de sitios web de prestigiosas compañías profesionales que atienden millones de usuarios y recogen una basta información en su sistema.

Se utilizarán además otros software de apoyo al diseño gráfico del sistema, estos son Adobe Photoshop CS2, Macromedia Dreamweaver 8.0, excelentes para el tratamiento de imágenes y

desarrollo de plantillas CSS respectivamente. Ambos muy profesionales y de uso extendido; realmente hacen muy sencillo el tratamiento de imágenes y estilos para la web.

1.6 Conclusiones

En el presente capítulo a partir de un análisis de diferentes estilos y tendencias arquitectónicas, así como mediante el estudio de diferentes metodologías de desarrollo de software, herramientas y lenguajes, se definieron los fundamentos de la arquitectura del SICI mediante la propuesta de solución final.

CAPÍTULO 2: DESCRPCIÓN DE LA ARQUITECTURA

2.1 Introducción

En este capítulo se describe la arquitectura del SICI principalmente mediante las vistas arquitectónicas de Casos de Uso, Lógica, Implementación y Despliegue. Además se analizan y definen aspectos de importancia para la arquitectura del Sistema. De manera general proporciona un acercamiento a la estructura del mismo, su organización y distribución.

2.2 Representación Arquitectónica

El Sistema se desarrollará completamente sobre Drupal, este CMS esta basado principalmente en una arquitectura desacoplada en módulos, cada uno de los cuales tiene definido su propio grupo de funciones; además la plataforma está conformada por otros componentes que son imprescindibles para Drupal. Este estilo proporcionará una alta flexibilidad y escalabilidad al sistema; la plataforma de Drupal permite extender e implementar nuevas funcionalidades no previstas.

El siguiente esquema proporciona una ilustración de la estructura del CMS:

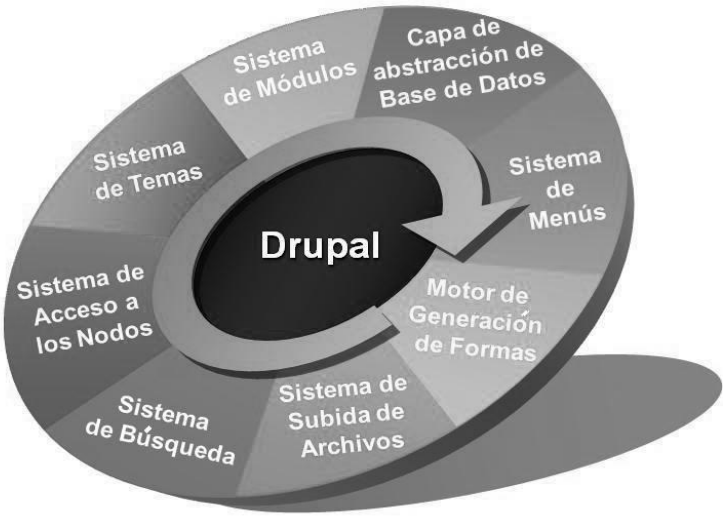

Figura 3: *Representación de los componentes fundamentales de Drupal.*

Otros aspectos que son características importantes de Drupal y que por consiguiente son determinantes en la arquitectura del Sistema se comentan a continuación:

Nodos y Tipos de contenido (*Nodes* and *Content types*, en inglés): Todo el contenido de un sitio web basado en Drupal es almacenado y tratado como un "nodo ". Un nodo es cualquier página, formulario, texto de un fórum, encuesta, cualquier tipo de entrada. Tratar todo el contenido como nodos permite la flexibilidad de crear nuevos tipos de contenido, además de hacer muy fácil el cambio sobre cualquier contenido.

En un sitio web se pueden tener diferentes tipos de nodos y utilizarlos en diferentes propósitos, Drupal utiliza para esto los "tipos de contenido", por ejemplo una historia determinada puede ser un nodo, del tipo de contenido historia, es algo similar a objetos y clases en la programación orientada a objetos. En el Sistema que se pretende desarrollar un tipo de contenido puede ser misionero, cada vez que se cree un nuevo misionero pasará a ser un nuevo nodo.

Módulos: Como ya se ha mencionado Drupal está desacoplado en Módulos, estos son archivos de código, contienen la implementación de funciones que son necesarias para diferentes objetivos. Existen dos grupos de módulos: (1) Los que conforman el núcleo del CMS (*Core Modules*, en inglés), estos se instalan automáticamente cuando se instala Drupal y poseen las funciones necesarias para que funcione correctamente el CMS y (2) los módulos contribuidos (*Contrib Modules*, en inglés), que son *plug-ins* para extender, construir o modificar las funcionalidades del núcleo de Drupal, estos pueden ser desarrollados por cualquier programador, siempre y cuando se sigan las políticas de desarrollo que rigen el CMS.

Ganchos (Hooks): Cada módulo no es más que un fichero que contiene código PHP con una serie de funciones *hooks* (funciones gancho) que son llamadas por Drupal. Los *hooks* son funciones de nombre foo_bar (), donde "foo" es el nombre del módulo (cuyo nombre de archivo es, pues, foo.module) y "bar" es el nombre del *hook*. Cada *hook* tiene un conjunto definido de parámetros y un resultado determinado.

El Sistema se puede ubicar dentro de los sistemas de estilo de componentes independientes. Por una parte tenemos en este grupo los sistemas cliente/servidor, cuyas características concuerdan perfectamente con las de Drupal: El Servidor de la Aplicación proporciona una serie de servicios o prestaciones, estos son accedidos a través de solicitudes de los clientes y toda esta comunicación se realiza a través de la red. El otro subgrupo, los sistemas de eventos, coinciden satisfactoriamente con el comportamiento de Drupal: desde la perspectiva del propio Sistema, el sistema de módulos y mediante las APIs comunes del *core*, actúan como un gestor de lo eventos, que en este caso serían los *hooks* invocados, los que son expuestos por cada módulo, cada uno de los cuales –módulos- recibe a su ves la información que sea de su interés al ser lanzado un *hook*.

2.2.1 4+1 vistas arquitectónicas

En este trabajo se descompone la representación arquitectónica en una serie de vistas: de Casos de uso, Lógica, Implementación y Despliegue, que brindan una visión global del sistema, lo que permite una mejor comprensión de la descripción. Para elaborar estas vistas se ha tomado como convención el uso de los estereotipos de UML, como lenguaje de modelado, con el que se generarán diferentes artefactos propuestos por la metodología de desarrollo RUP.

Figura 4: *Modelo de las vistas arquitectónicas.*

- Vista de Casos de Uso: Esta vista nos brinda información de escenarios y casos de usos arquitectónicamente significativos con funcionalidades imprescindibles para el sistema. Los casos de uso arquitectónicamente significativos son aquellos que sirven para validar la arquitectura propuesta y describen las funcionalidades imprescindibles para el sistema.
- Vista Lógica: En esta vista se describen los paquetes más abstractos que conforman el sistema y las relaciones de dependencia o de uso que existen entre ellos.
- Vista de Procesos: Esta vista describe los aspectos de concurrencia y sincronización del diseño (No se realiza en el presente trabajo).
- Vista de Implementación: La vista de implementación proporciona una descripción de las principales capas y subsistemas de componentes de la aplicación.
- Vista de Despliegue: Mediante esta vista se describe el mapeo del software en el hardware y se reflejan los aspectos de distribución.

2.3 Objetivos y restricciones arquitectónicas

2.3.1 Interoperabilidad con sistemas legados

Por las características de la Universidad, el sistema tendrá que interactuar con diferentes sistemas legados que brindan funcionalidades que se pueden considerar de alta prioridad, es por esto que se propone el uso del estándar XML para el intercambio de datos con estos sistemas, haciendo uso de las tecnologías de los servicios web a través del protocolo SOAP. Además constituye un objetivo exponer diferentes servicios web a partir de la información que se manejará en el Sistema que puedan ser útiles para otros procesos ajenos al mismo.

Es imprescindible además, hacer uso del directorio activo; para controlar el acceso y autenticación con el Sistema y los niveles de permisos se usará integración con el protocolo LDAP. Esto no quiere decir que la autenticación y la seguridad en cuanto a los niveles de privilegios en el Sistema dependen del directorio activo, en caso de no existir, la plataforma de Drupal tiene sus propios mecanismos de autenticación, módulos que brindan todas las funcionalidades necesarias para proporcionar una seguridad adecuada al Sistema, como por ejemplo el módulo *User*. Para gestionar estos procesos mediante LDAP, se usará el módulo *LDAP_Integration*, él mismo cuenta con todas las funciones necesarias para el trabajo con este protocolo y se integra perfectamente con el dominio de la Universidad.

2.3.2 Distribución física de la DCI

La Dirección de Cooperación Internacional está formada por tres grupos: Cooperación Internacional, Trámites y Relaciones Publicas, todos no están ubicados en la misma oficina pero si dentro de una única edificación donde radica la DCI. Cada uno de estos grupos tiene una serie de especialistas, todos con sus puestos de trabajos con ordenadores, además en la oficina de la directora de esta Dirección también hay un ordenador. Por tal motivo es la decisión de desarrollar una aplicación distribuida, garantizando así el acceso a través de la web a la misma desde cualquier estación de trabajo de la red. También con el Sistema tienen que interactuar otros agentes externos a la DCI que juegan roles imprescindibles y todos deben acceder a la Aplicación mediante la web puesto que se encuentran en diferentes puntos del centro.

2.3.3 Política de manejo de datos

Resulta vital, para mantener la integridad del sistema y la información que fluye por el mismo mantener canales de comunicación segura, por lo que se empleará el protocolo SSL, este proporciona autenticación y privacidad de la información entre extremos sobre Internet (o intranet) mediante el uso de criptografía, permite a las aplicaciones cliente-servidor comunicarse de una forma diseñada para prevenir escuchas, la falsificación de la identidad del remitente y mantener la integridad del mensaje. Para cumplir con esta restricción se hará uso del módulo mod_ssl del servidor web Apache que dará soporte a la aplicación.

2.3.4 Restricciones de acuerdo a la estrategia de diseño

El proceso de desarrollo se debe regir por las políticas de desarrollo de Drupal. Para extender, construir o modificar las funcionalidades del núcleo del CMS se deben emplear los estándares de codificación establecidos por la organización que creó Drupal. [9]

Drupal no está programado orientado a objetos por varias razones, sin embargo si está diseñado con características de un sistema orientado a objetos, utiliza patrones de diseño que son propios de estos sistemas. En la arquitectura de Drupal se observan comportamientos de patrones de diseño de diferentes categorías como son *Singleton*, *Decorator*, *Bridge*, *Observer*, *Command*, *Chain of Responsibility*; y un patrón determinante es el "*Reflection Pattern*", que describe el comportamiento de todo sitio basado en Drupal. Gracias a este patrón el sistema se convierte en una aplicación muy adaptable a diferentes entornos puesto que permite prácticamente la modificación de todo su comportamiento a través de los módulos instalables, sin la necesidad de modificar del núcleo.

2.3.5 Requerimientos de hardware

Estaciones de trabajo (PC Cliente)

- Periféricos: Mouse y Teclado.
- Tarjeta de Red.
- 128 MB de RAM.
- Procesador Pentium 4 (ó similar).
- 40 GB de espacio en disco.

PC impresora

- Periféricos: Mouse y Teclado.
- Tarjeta de Red.
- 128 MB de RAM.
- Procesador Pentium 4 (ó similar).
- 40 GB de espacio en disco.
- Puerto USB.

Impresora

- Conexión USB.
- Controladores multiplataforma.

Servidor Web

- Tarjeta de Red.
- 1 GB de RAM.
- 80 GB de espacio en disco.
- Procesador Pentium 4 (ó similar).

Servidor de Base de Datos

- Tarjeta de Red.
- 1 GB de RAM.
- 80 GB de espacio en disco.
- Procesador Pentium 4 (ó similar).

2.3.6 Requerimientos de software

Estaciones de trabajo (PC Cliente)

- Sistema operativo: Multiplataforma.
- Navegador web: Internet Explorer, Mozilla, NetScape (Drupal prácticamente funciona con todos los navegadores web del mercado).

Servidor Web

- Servidor web Apache.
- PHP 5.

Servidor de Base de Datos

- Sistema Gestor de Base de Datos PostgreSQL 8.1.

2.3.7 Estructura del equipo de desarrollo

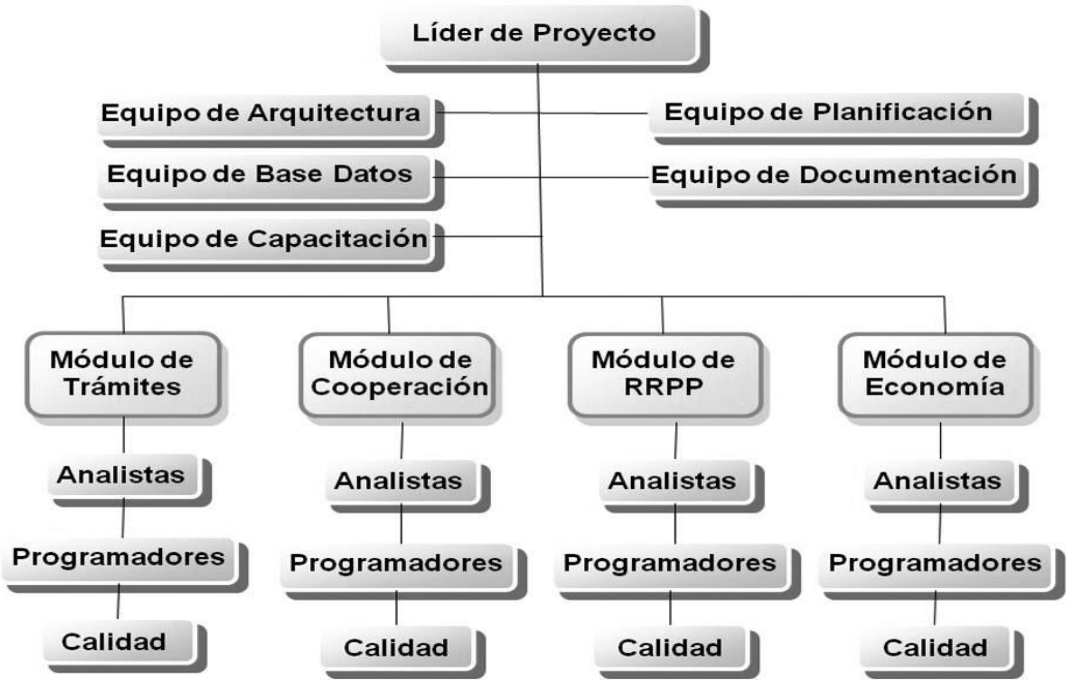

Figura 5: *Estructura organizacional del equipo de desarrollo.*

El equipo de desarrollo cuenta con 8 ordenadores Pentium 4, con sus periféricos, 512 MB de RAM, 120 GB de capacidad de disco duro y tarjeta de red, los cuales tienen instalados todo el software necesario para que cada miembro del equipo independientemente de su rol pueda trabajar en cualquier ordenador.

2.4 Tamaño y rendimiento

Una pregunta muy frecuente es qué tanta escalabilidad se puede obtener en un sistema basado en Drupal y qué rendimiento se espera de este. Estos factores dependen de varias variables, en primer lugar se debe mirar cuántos y cuáles módulos están instalados; saber las características del tráfico con el sistema, si es mayormente de usuarios autentificados o anónimos; es muy determinante el hardware con que se cuente y además influye también toda la configuración del software empleado. La respuesta no consiste en una afirmación exacta, pero para cada problema que pueda surgir siempre habrá una solución en dependencia de la categoría de la situación.

Para el Sistema se contará con uno de los mecanismos implementados por Drupal para lograr una mayor eficiencia: el mecanismo de caché, el cual elimina considerablemente las consultas a la base de datos incrementando así el rendimiento y reduciendo la carga del servidor.

Por otra parte existen varios módulos contribuidos que pueden ser utilizados para aumentar el rendimiento, o se pueden emplear estrategias ya probadas por otros usuarios de Drupal, como la desactivación temporal del motor de búsqueda, el cual esta considerado como uno de los más críticos en cuanto al nivel de interacción con la base de datos por la cantidad de recursos que emplea. Otro aspecto fundamental que se tendrá en cuenta para el Sistema es la utilización del módulo *Throttle,* mecanismo que controla la congestión, detectando automáticamente el incremento del tráfico hacia el Sistema, este mecanismo es utilizado por otros módulos para optimizar sus funcionamientos deshabilitando temporalmente funcionalidades intensas para el CPU.

2.5 Vista de Casos de Uso

Dado que Drupal está caracterizado por una arquitectura modular y por tanto el término módulo se utiliza para nombrar los archivos de código fuente que implementan la mayoría de las funcionalidades de la plataforma, se ha optado por llamar cada módulo del Sistema que se pretende desarrollar subsistema, por lo que a partir de este momento cuando aparezca el termino subsistema, se refiere a los módulos del Sistema y el término módulo, a los módulos del CMS.

2.5.1 Subsistema de Trámites

El Subsistema de trámites está previsto para el Grupo de Trámites de la DCI, en el que se manejan los asuntos migratorios y de extranjería en la UCI. Se encarga de la gestión de la documentación necesaria para que una persona pueda viajar al exterior o un extranjero pueda entrar al país por motivos de trabajo relacionados con la Universidad. Elabora los reportes estadísticos y económicos de las personas que viajan y las misiones, además de los partes sobre el estado de estas últimas y del proceso de tramitación de los misioneros. Controla y archiva los documentos y la información obtenida de dichos procesos.

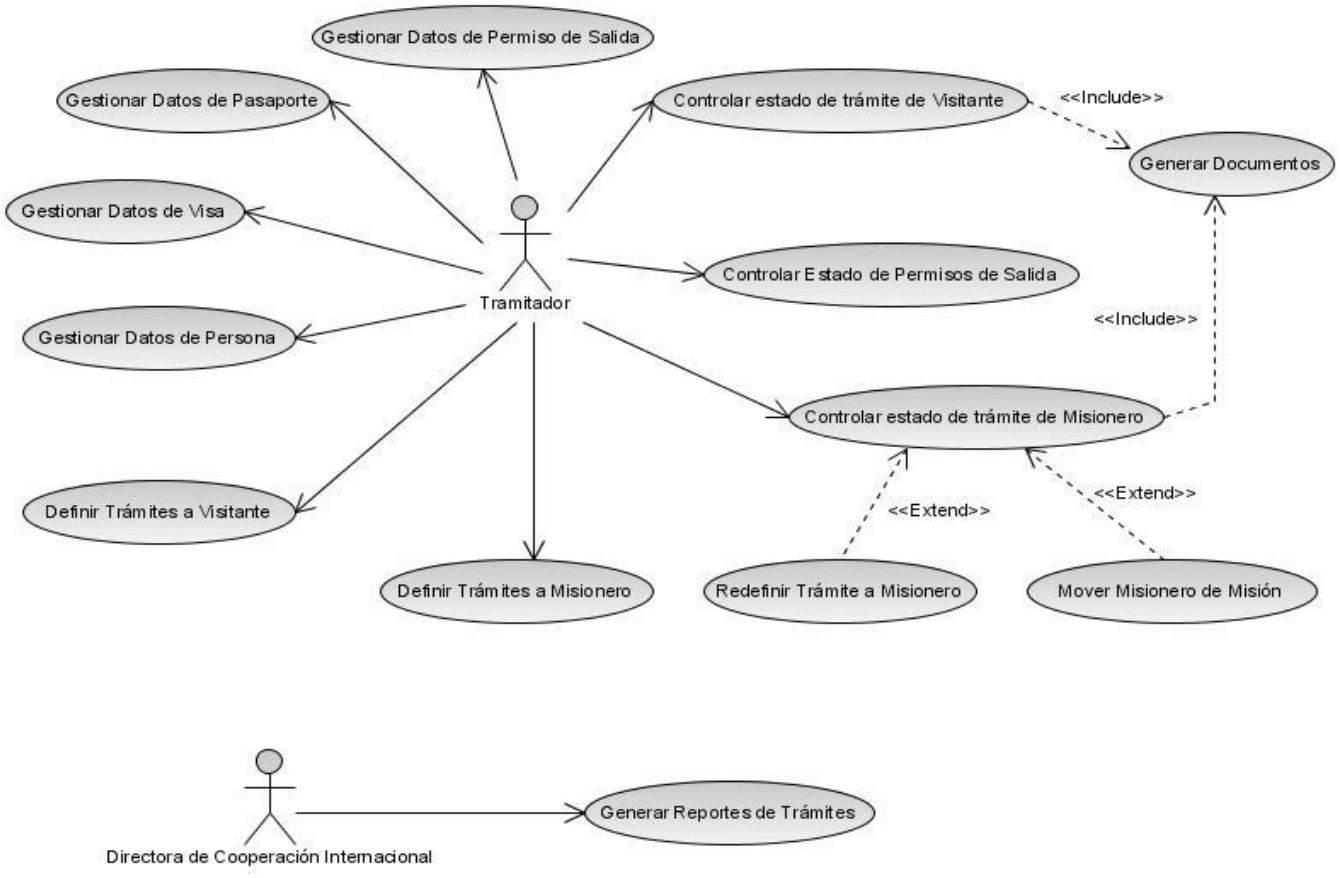

Figura 6: *Diagrama de casos de uso críticos del Subsistema de Trámites.*

Nombre del Caso de Uso: Gestionar Datos de Permiso de Salida.

Descripción:

El caso de uso inicia cuando el Tramitador necesita realizar una operación sobre un Permiso de Salida determinado, el sistema ejecuta la acción indicada terminando así el caso de uso.

Precondiciones:

El usuario está autenticado con el rol de tramitador.

Nombre del Caso de Uso: Gestionar Datos de Pasaporte.

Descripción:

El caso de uso inicia cuando el Tramitador necesita realizar una operación sobre un determinado Pasaporte, el sistema ejecuta la acción indicada terminando así el caso de uso.

Precondiciones:

El usuario está autenticado con el rol de tramitador.

Nombre del Caso de Uso: Gestionar Datos de Visa.

Descripción:

El caso de uso inicia cuando el Tramitador necesita realizar una operación sobre una determinada Visa, el sistema ejecuta la acción indicada terminando así el caso de uso.

Precondiciones:

El usuario está autenticado con el rol de tramitador.

Nombre del Caso de Uso: Gestionar Datos de Persona.

Descripción:

El caso de uso inicia cuando el Tramitador necesita realizar una operación sobre los datos de una persona, el sistema ejecuta la acción indicada terminando así el caso de uso.

Precondiciones:

El usuario está autenticado con el rol de tramitador.

Nombre del Caso de Uso: Definir Trámites a Misionero.

Descripción:

El caso de uso inicia cuando el Tramitador necesita definir los trámites a realizar a un misionero, el sistema registra los trámites definidos terminando así el caso de uso.

Precondiciones:

El usuario está autenticado con el rol de tramitador.

Nombre del Caso de Uso: Controlar estado de trámite de Misionero.

Descripción:

El caso de uso inicia cuando el Tramitador necesita registrar los datos de los documentos oficiales del viaje del misionero o modificar los trámites definidos con anterioridad. El sistema registra los datos de los documentos oficiales y actualiza el estado de los procesos de trámites y del misionero terminando así el caso de uso.

Precondiciones:

El usuario está autenticado con el rol de tramitador.

Nombre del Caso de Uso: Mover Misionero de Misión.

Descripción:

El caso de uso inicia cuando el Tramitador necesita mover al misionero de la misión en que esta propuesto a otra misión, el sistema actualiza la misión en la cual va a participar el misionero terminando así el caso de uso.

Precondiciones:

El usuario está autenticado con el rol de tramitador.

Nombre del Caso de Uso: Redefinir Trámite de Misionero.

Descripción:

El caso de uso inicia cuando el Tramitador necesita modificar los trámites confeccionados a un misionero, el sistema actualiza el nuevo trámite terminando así el caso de uso.

Precondiciones:

El usuario está autenticado con el rol de tramitador.

Nombre del Caso de Uso: Controlar Estado de Permisos de Salida.

Descripción:

El caso de uso inicia cuando el Tramitador necesita tener un control sobre el estado de los permisos de salida que estén en uso, el sistema muestra todos los permisos de salida diferenciando los que están vigentes y los que no, terminado así el caso de uso.

Precondiciones:

El usuario está autenticado con el rol de tramitador.

Nombre del Caso de Uso: Definir Trámites a Visitante.

Descripción:

El caso de uso inicia cuando el Tramitador necesita definir los trámites a realizar a un Visitante. Así como seleccionar qué documentos oficiales se les va a confeccionar a esa persona.

Precondiciones:

El usuario está autenticado con el rol de tramitador.

Nombre del Caso de Uso: Controlar Estado de Trámite de Visitante.

Descripción:

El caso de uso inicia cuando el Tramitador necesita controlar el estado de trámite de cada Visitante. El Tramitador introduce los datos necesarios para confeccionar los documentos requeridos para el visitante, el sistema registra los datos y actualiza el estado del visitante, terminado así el caso el caso de uso.

Precondiciones:

El usuario está autenticado con el rol de tramitador.

Nombre del Caso de Uso: Generar Documentos.

Descripción:

El caso de uso inicia cuando el Tramitador necesita generar los diferentes documentos a imprimir. El Tramitador selecciona el documento a generar y el sistema carga los datos que debe mostrar el documento y luego lo genera para que el Tramitador lo imprima, terminado así el caso de uso.

Precondiciones:

El usuario está autenticado con el rol de tramitador.

Nombre del Caso de Uso: Generar Reportes de Trámites.

Descripción:

El caso de uso inicia cuando la Directora de Cooperación Internacional realiza una búsqueda por diferentes criterios, el sistema le muestra la información que ella necesita terminando así el caso de uso.

Precondiciones:

El usuario está autenticado con el rol de tramitador.

2.5.2 Subsistema de Relaciones Públicas

El Grupo de Relaciones Públicas se encarga de manejar lo relacionado con la imagen institucional así como la promoción y divulgación de las actividades de interés de la Universidad. Este grupo, en conjunto con el de Cooperación Internacional, dirige metodológica y funcionalmente el sistema de colaboración internacional de la UCI, gestiona y conforma la información de apoyo a los intercambios inter-institucionales, además de garantizar las acciones necesarias para el éxito de las misiones de la UCI en el exterior y la satisfacción de los visitantes en Cuba. El Subsistema de Relaciones Públicas tiene como objetivo cumplir los requerimientos de este Grupo.

Figura 7: *Diagrama de casos de uso críticos del Subsistema de Relaciones Públicas.*

Nombre del Caso de Uso: Gestionar Visitas.

Descripción:

El caso de uso inicia cuando el Controlador accede a la opción Adicionar si quiere insertar una visita y si quiere modificar, eliminar o buscar el sistema le muestra un listado de las visitas entradas en el sistema.

Precondiciones:

El Controlador se haya autentificado para poder tener acceso a la interfaz gestionar Visitas.

Nombre del Caso de Uso: Gestionar Datos del Visitante.

Descripción:

El caso de uso inicia cuando el Controlador accede a la opción Adicionar si quiere insertar un visitante y si quiere modificar, eliminar o buscar el sistema le muestra un listado de los visitantes entrados en el sistema.

Precondiciones:

El Controlador se haya autentificado para poder tener acceso a la interfaz Gestionar Datos del Visitante y haya insertado una visita.

Nombre del Caso de Uso: Gestionar Datos del Acompañante.

Descripción:

El Controlador accede a la opción Adicionar si quiere insertar un acompañante y si quiere modificar, eliminar o buscar el sistema le muestra un listado de los Acompañantes entrados en el sistema.

Precondiciones:

El caso de uso inicia cuando el Controlador se haya autentificado para poder tener acceso a la interfaz Gestionar Datos del Acompañante y haya insertado un visitante.

Nombre del Caso de Uso: Gestionar Informe de Visita.

Descripción:

El caso de uso inicia cuando el Documentador se encarga de adicionar un informe, así como poder realizar alguna modificación a los informes, poder eliminarlos y buscarlos. Esto se realiza mediante la selección de una visita corta en una lista que muestra la interfaz.

Precondiciones:

Que se haya insertado un informe para poder eliminar y modificar.

Nombre del Caso de Uso: Gestionar Programa de Visita.

Descripción:

El caso de uso inicia cuando el Controlador se encarga de adicionar un programa de visita, así como poder realizar alguna modificación a los programa, poder eliminarlos y buscarlos. Esto se realiza mediante la selección de una visita en una lista que muestra la interfaz.

Precondiciones:

Que se haya insertado una visita corta para poder realizar un Programa de visita.

Nombre del Caso de Uso: Gestionar Atenciones.

Descripción:

El caso de uso inicia cuando el Controlador se encarga de adicionar un programa de atenciones, así como poder realizar alguna modificación a los programas, poder eliminarlos y buscarlos. Esto se realiza mediante la selección de una visita larga en una lista que muestra la interfaz.

Precondiciones:

Que se haya insertado una visita larga para poder realizar un Programa de atenciones.

Nombre del Caso de Uso: Reportes.

Descripción:

El Controlador se encarga de consultar la información deseada a través de diferentes criterios, el sistema le muestra la información deseada, terminando así el caso de uso.

Precondiciones:

El Controlador debe encontrarse autenticado de forma satisfactoria. Exista la información deseada registrada en la base de datos.

2.5.3 Subsistema de Cooperación Internacional

El Subsistema de Cooperación es el encargado de desarrollar las funcionalidades para satisfacer los requerimientos del Grupo de Cooperación Internacional, que tiene entre sus funciones gestionar la estrategia de colaboración internacional de la Universidad, coordinar la participación de la UCI en organizaciones internacionales de excelencia, diseñar y fomentar los vínculos inter-institucionales y contribuir a la identificación y gestión de programas de cooperación internacional.

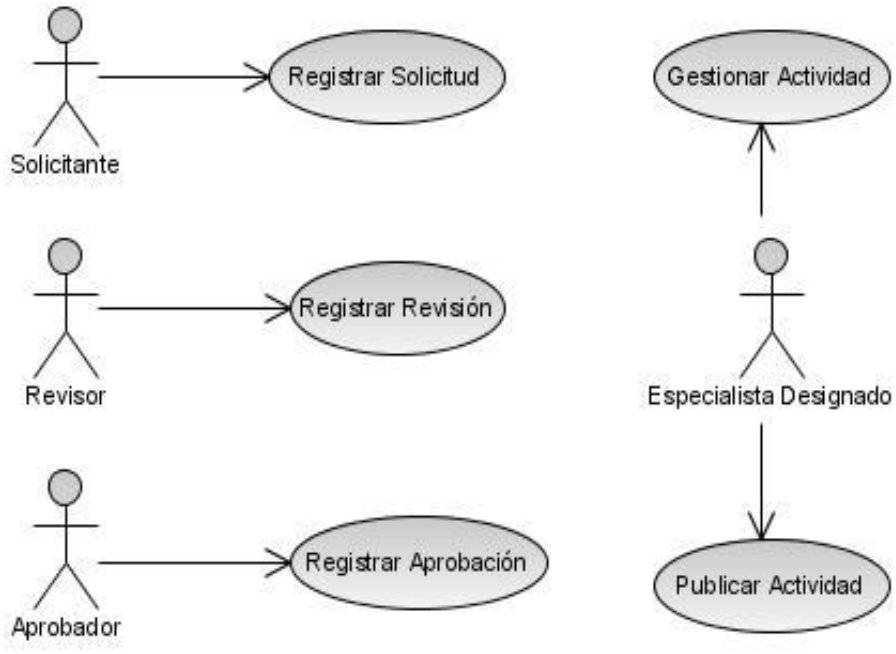

Figura 8: *Diagrama de casos de uso críticos del Subsistema de Cooperación.*

Nombre: Registrar solicitud.

Descripción:

El caso de uso inicia cuando el Solicitante decide hacer o modificar una solicitud de misión, ver el estado en que se encuentran las solicitudes que ha hecho o cancelar una solicitud. Este introduce en el sistema los datos necesarios para realizar la operación deseada y a su vez el sistema ejecuta dicha acción finalizando el caso de uso.

Precondiciones:

Los viajeros que aparezcan en la solicitud deben estar aprobados por las organizaciones políticas pertinentes. El usuario correspondiente al Solicitante debe encontrarse en el sistema SICI autenticado de forma satisfactoria.

Nombre: Registrar revisión.

Descripción:

El caso de uso inicia cuando el Revisor decide revisar una solicitud de misión o ver todas las solicitudes que están en curso. Este introduce en el sistema los datos necesarios para llevar a cabo la operación deseada y el sistema la ejecuta finalizando el caso de uso.

Precondiciones:

La solicitud debe estar creada. El usuario correspondiente al Revisor debe encontrarse en el sistema SICI autenticado de forma satisfactoria.

Nombre: Registrar Aprobación.

Descripción:

El caso de uso se inicia cuando el Aprobador decide ver las solicitudes o convocatorias, según sea el caso, que están esperando el análisis y tomar una decisión final sobre cada una de ellas. Selecciona la opción aprobar, rechazar o aprobar con cambios y el sistema registra el cambio de estado finalizando el caso de uso.

Precondiciones:

La solicitud debe estar revisada. El usuario correspondiente al Aprobador debe encontrarse en el sistema SICI autenticado de forma satisfactoria.

Nombre: Gestionar Actividad.

Descripción:

El caso de uso inicia cuando el Especialista Designado decide registrar en el sistema información de una actividad, es decir, una posible misión. Solicita adicionar, modificar o eliminar una actividad, entra los datos necesarios para la operación y el sistema la ejecuta, almacenando un registro de la actividad en todos los casos y finalizando el caso de uso.

Precondiciones:

El usuario correspondiente al Especialista Designado debe encontrarse en el sistema SICI autenticado de forma satisfactoria.

Nombre: Publicar Actividad.

Descripción:

El caso de uso inicia cuando el Especialista Designado decide publicar en el sistema una actividad. Selecciona adicionar o dar de baja a una publicación. Introduce los datos necesarios en el sistema y este ejecuta operación solicitada finalizando el caso de uso.

Precondiciones:

La actividad tiene que estar en estado Aprobada. El usuario correspondiente al Especialista Designado debe encontrarse en el sistema SICI autenticado de forma satisfactoria.

2.5.4 Subsistema de Economía

El Subsistema de Economía está concebido para que los Grupos de Cooperación y Trámites puedan llevar a cabo sus funciones que necesitan de gestiones económicas continuamente. A pesar de que en la DCI no existe un grupo de economía específicamente se hace necesario desarrollar un subsistema que se encargue de la gestión de cheques, presupuestos para misiones, obsequios.

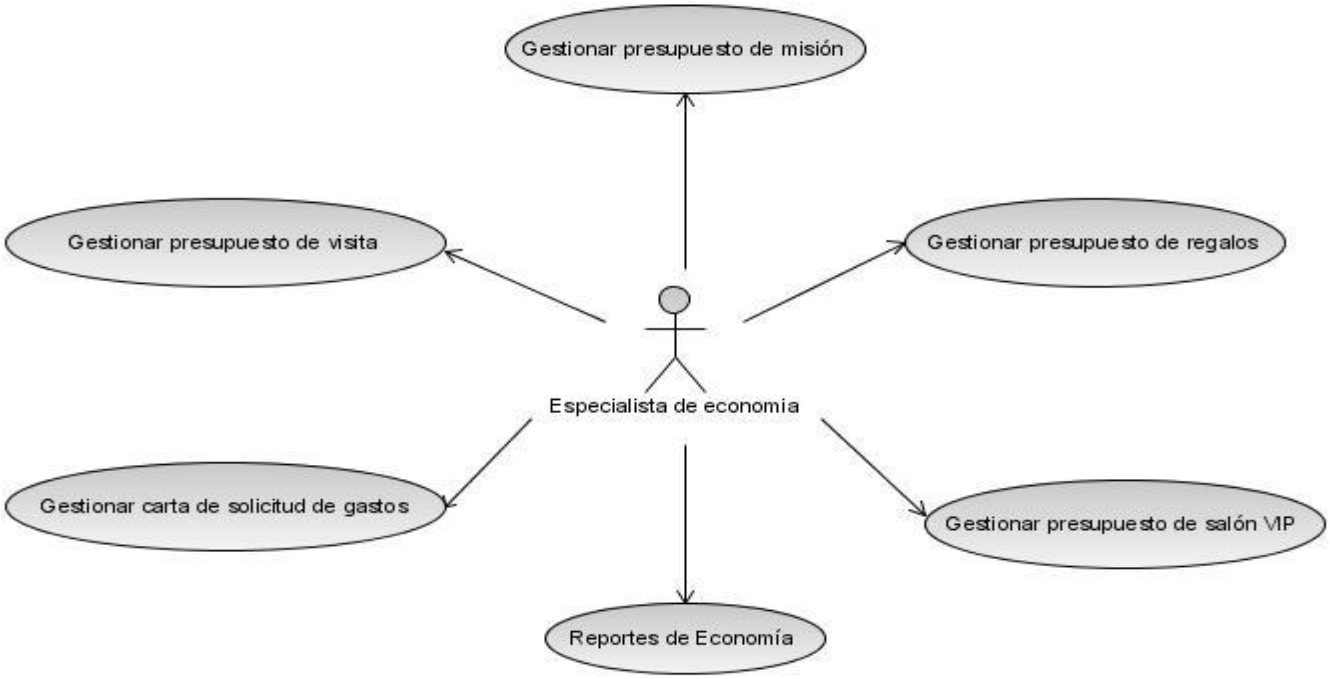

Figura 9: *Diagrama de casos de uso críticos del Subsistema de Economía.*

Nombre del Caso de Uso: Gestionar Presupuesto de Visitas.

Descripción:

El Especialista de Economía se encarga de adicionar los datos en el formulario para la obtención del presupuesto de la visita.

Precondiciones:

El usuario correspondiente al Especialista de Economía designado debe encontrarse autenticado de forma satisfactoria. Que el sistema tenga alguna solicitud de presupuesto de visita archivado.

Nombre del Caso de Uso: Gestionar Presupuesto de Misión.

Descripción:

El Especialista de Economía se encarga de adicionar los datos en el formulario para la obtención del presupuesto de misión.

Precondiciones:

El usuario correspondiente al Especialista de Economía designado debe encontrarse autenticado de forma satisfactoria. Que el sistema tenga alguna solicitud de presupuesto de visita archivado.

Nombre del Caso de Uso: Gestionar Carta de Solicitud de Gastos.

Descripción:

El Especialista de Economía se encarga de adicionar los posibles gatos de las atenciones que se les dará a la visita, además de poder eliminar o modificar en caso necesario.

Precondiciones:

El usuario correspondiente al Especialista de Economía designado debe encontrarse autenticado de forma satisfactoria. El especialista de Economía haya adicionado una carta de solicitud anteriormente para luego modificar o eliminar.

Nombre del Caso de Uso: Gestionar Presupuesto de Salón VIP.

Descripción:

El Especialista de Economía se encarga de adicionar una solicitud de salón VIP, así como realizar alguna modificación a la solicitud o poder eliminarla.

Precondiciones:

El usuario correspondiente al Especialista de Economía designado debe encontrarse autenticado de forma satisfactoria. El especialista de Economía haya adicionado una solicitud de salón VIP anteriormente para luego modificar o eliminar.

Nombre del Caso de Uso: Gestionar Presupuesto de Regalo.

Descripción:

El Especialista de Economía se encarga de gestionar los posibles gastos para la obtención del regalo que ofrecerá la UCI, logrando así adicionarlo, además de poder eliminar o modificar en caso necesario.

Precondiciones:

El usuario correspondiente al Especialista de Economía designado debe encontrarse autenticado de forma satisfactoria. El especialista de Economía haya adicionado un presupuesto anteriormente para luego modificar o eliminar.

Nombre del Caso de Uso: Reportes.

Descripción:

El Especialista de Economía se encarga de consultar la información deseada a través de diferentes criterios, el sistema le muestra la información deseada, terminando así el caso de uso.

Precondiciones:

El usuario correspondiente al Especialista de Economía designado debe encontrarse autenticado de forma satisfactoria. Exista la información deseada registrada en la base de datos.

2.6 Vista lógica

Mediante la siguiente figura se pretende mostrar los subsistemas que conforman el Sistema que será desarrollado, son cuatro fundamentales, tres coinciden con los grupos que conforman la DCI y un cuarto subsistema, el de Economía que es necesario desarrollar para garantizar las gestiones económicas que llevan a cabo los grupos de Trámites y Relaciones Públicas. Cada uno de estos cuatro subsistemas tiene definidos sus propios tipos de contenidos, pero se apoyan en módulos comunes para realizar sus procesos, en dependencia del tipo de proceso que sea.

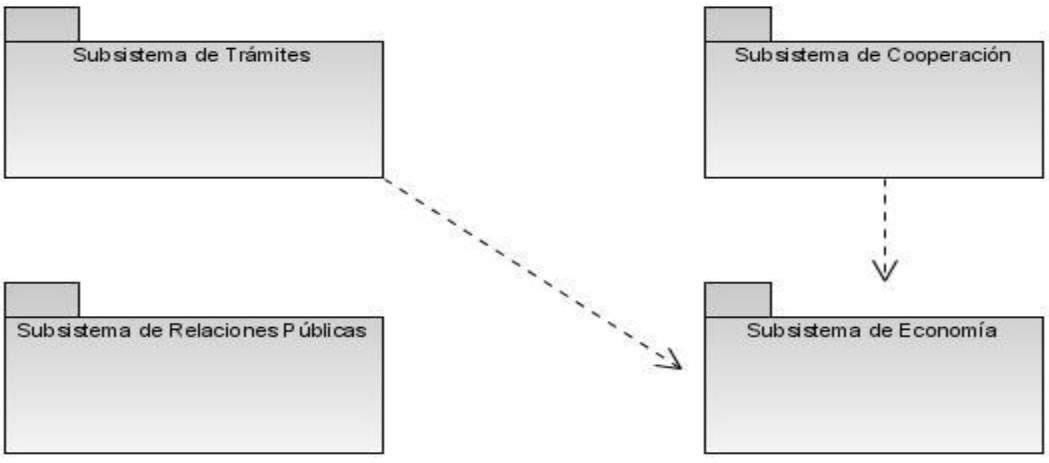

Figura 10: *Representación de los subsistemas.*

La estructura lógica del Sistema:

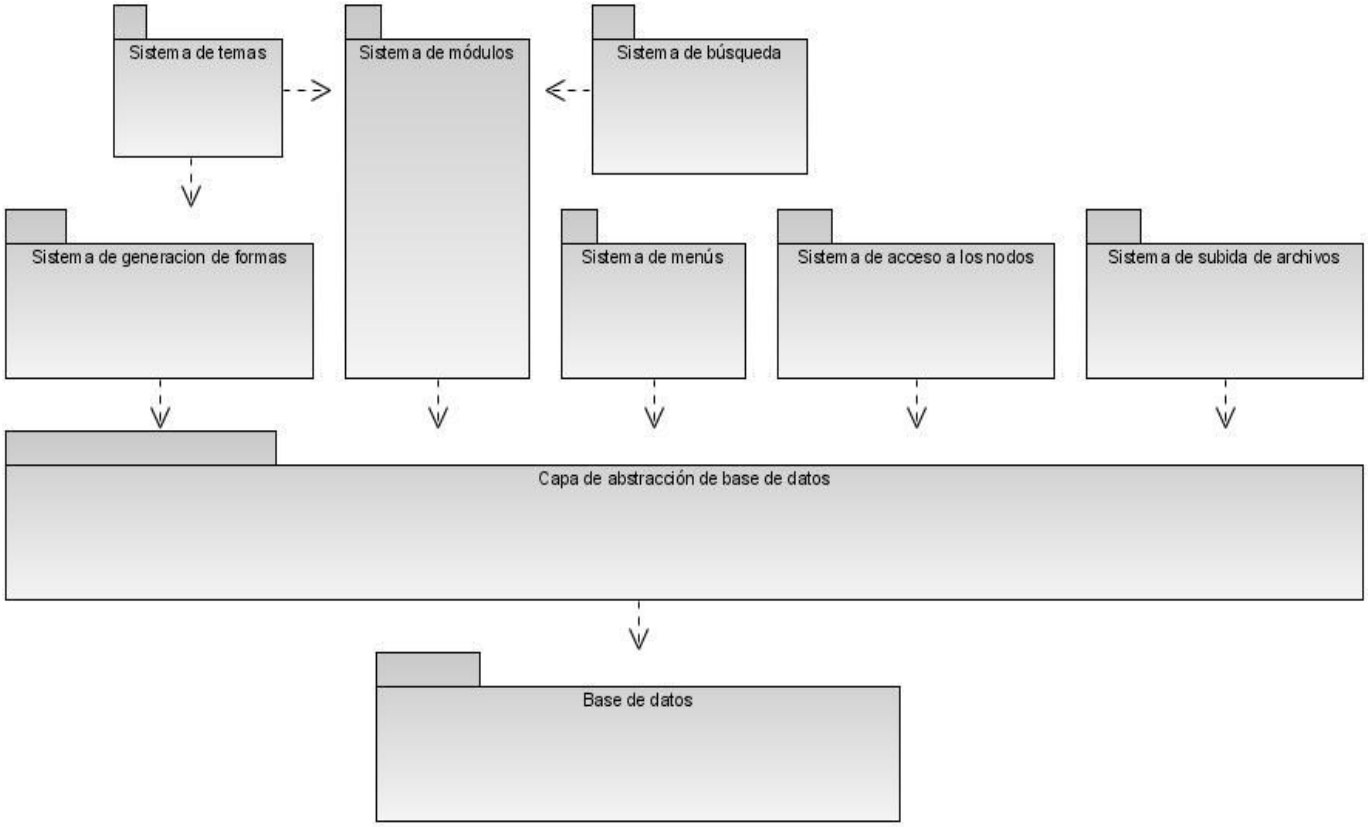

Figura 11: *Estructura lógica.*

En la tabla a continuación se describen todos los paquetes de este diagrama, los mismos no aparecen físicamente de esta manera en el SICI, la imagen representa una aproximación lógica del funcionamiento de un sistema basado en Drupal de acuerdo a las funcionalidades del mismo, las que se pueden agrupar en mecanismos o sistemas como ha sido representado en el diagrama.

Tabla 1: *Descripción de la vista lógica.*

Descripción de los paquetes		
Nombre	Estereotipo	Descripción
Sistema de módulos	Paquete	Conjunto de módulos que conforman el núcleo de Drupal y módulos contribuidos. Aquí se encuentran implementadas casi todas las funcionalidades del Sistema.
Sistema de temas	Paquete	Conjunto de módulos y funciones utilizadas para generar código html; puede ser personalizado en cada tema.
Sistema de búsqueda	Paquete	Contiene las funciones necesarias para constituir una interfaz de búsqueda que gestiona un mecanismo de búsqueda global.
Sistema de generación de formularios	Paquete	Grupo de funciones que permiten la visualización y procesamiento de formularios HTML.
Sistema de menús	Paquete	Contiene las funciones necesarias para definir los menús de navegación y rutear las páginas solicitadas a código basado en URLs.
Sistema de acceso a los nodos	Paquete	Contiene las funciones necesarias para definir quién puede hacer qué con cuáles nodos.
Sistema de subida de archivos	Paquete	Contiene las funciones comunes necesarias para el manejo de archivos.
Capa de abstracción de Base de Datos	Paquete	Grupo de funciones que se utilizan para controlar el manejo de los datos, soporta múltiples servidores de Base de Datos.
Base de datos	Paquete	Conjunto de tablas relacionadas que contienen toda la información.

En el paquete sistema de módulos se tiene:

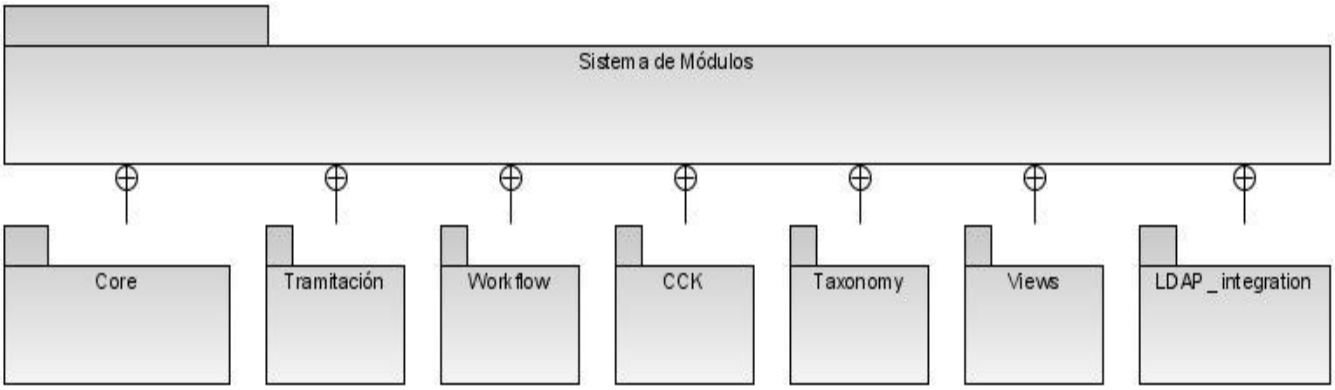

Figura 12: *Sistema de módulos.*

La tabla siguiente describe el sistema de módulos del SICI.

Tabla 2: *Descripción del sistema de módulos.*

Estructura del sistema de módulos		
Nombre	Estereotipo	Descripción
Core	Paquete	Representa todos los módulos que se instalan por defecto con cada distribución de Drupal. Algunos son requeridos para que funcione correctamente.
Tramitación	Paquete	Módulo que contiene funcionalidades indispensables para el Subsistema de Trámites. Fue desarrollado por el equipo de programación del proyecto.
Workflow	Paquete	Módulo que contiene funcionalidades para gestionar los flujos de trabajo de los tipos de contenido.
CCK	Paquete	Módulo que permite crear y personalizar campos sobre los tipos de contenido.
Taxonomy	Paquete	Módulo utilizado para la organización y clasificación de los tipos de contenido.
Views	Paquete	Módulo que provee métodos muy flexibles para controlar la presentación de los tipos de contenidos.
LDAP_Integration	Paquete	Módulo utilizado para integrar el Sistema con el dominio de la Universidad a través del protocolo LDAP

El paquete sistema de temas:

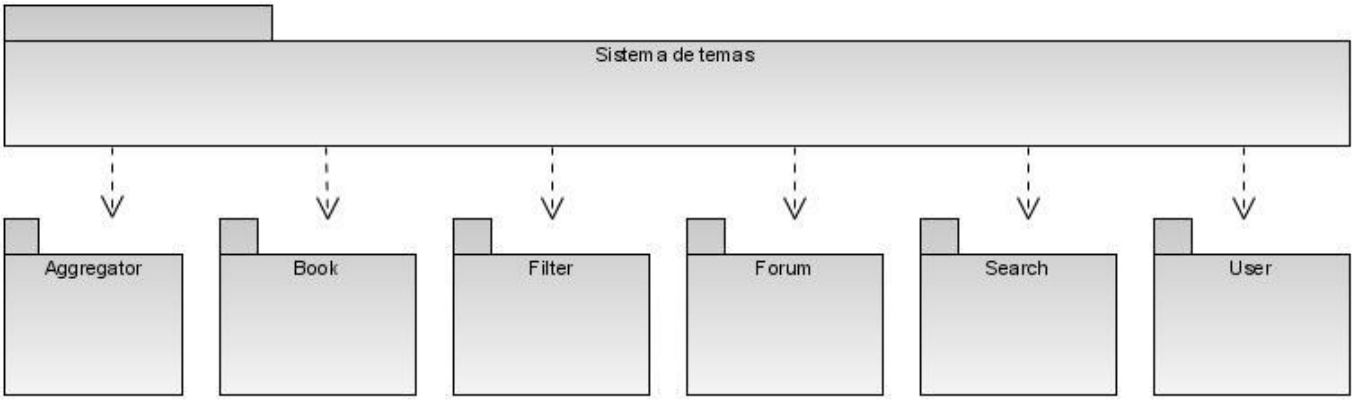

Figura 13: *Sistema de temas.*

A continuación una tabla con los detalles del sistema de temas del SICI.

Tabla 3: *Descripción del sistema de temas.*

Módulos que intervienen en el sistema de temas		
Módulo	Estereotipo	Función dentro del sistema de temas.
Aggregator	Paquete	Provee funciones que permiten darle formato a diferentes objetos que fueron enriquecidos, para posteriormente mostrarlos.
Book	Paquete	Provee funciones para definir cómo puede ser personalizado el código html exportado por los *books* y prepara los URLs para la navegación dentro de estos.
Filter	Paquete	Provee funciones para darle formato a una serie de criterios de filtros.
Forum	Paquete	Provee funciones para darle formato a una serie de aspectos dentro de un fórum.
Search	Paquete	Provee funciones para darle formato a páginas que se generan como resultado de una consulta o una simple entrada para hacer una consulta.
User	Paquete	Provee funciones para hacer listas de usuarios o personalizar una página de usuario.

En el paquete sistema de búsqueda:

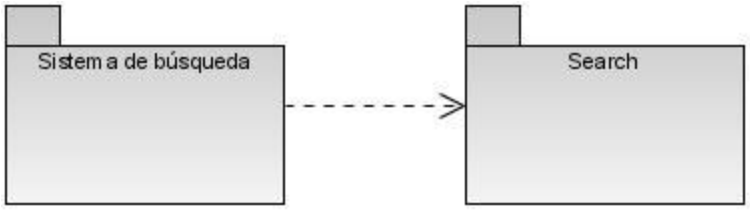

Figura 14: *Sistema de búsqueda.*

Tabla 4: *Descripción del sistema de búsqueda.*

Módulo que interviene en el sistema de búsqueda		
Módulo	Estereotipo	Función dentro del sistema de búsqueda.
Search	Paquete	Prácticamente contiene todas las funciones que permiten el desempeño del sistema de búsqueda, está concebido exclusivamente para este fin.

En el paquete sistema de acceso a los nodos:

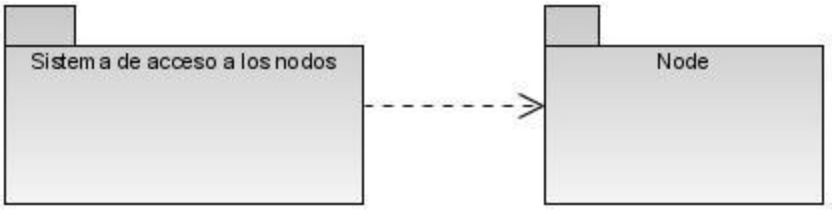

Figura 15: *Sistema de acceso a los nodos.*

Tabla 5: *Descripción del sistema de acceso a los nodos.*

Módulo que interviene en el sistema de acceso a los nodos		
Módulo	Estereotipo	Función dentro del sistema de búsqueda
Node	Paquete	Prácticamente contiene todas las funciones que permiten el desempeño del sistema de acceso a los nodos, como por ejemplo comparación de los Id de usuarios con arreglos de permisos de acceso, generación de consultas SQL para estos fines, entre otras.

2.7 Vista de Implementación

En la Vista anterior se describieron los paquetes lógicos más abstractos que conforman el Sistema, algunos de estos están compuestos solamente por funciones implementadas por la plataforma de Drupal, por lo que no se les hizo hincapié. Mediante esta Vista se pretende ilustrar las funciones que se consideran de mayor importancia para algunos sistemas y las desarrolladas por los grupos de programación del Proyecto; no es objetivo de este trabajo documentar las funcionalidades de Drupal, que además están muy bien documentadas en diferentes sitios web oficiales acerca del desarrollo de este CMS.

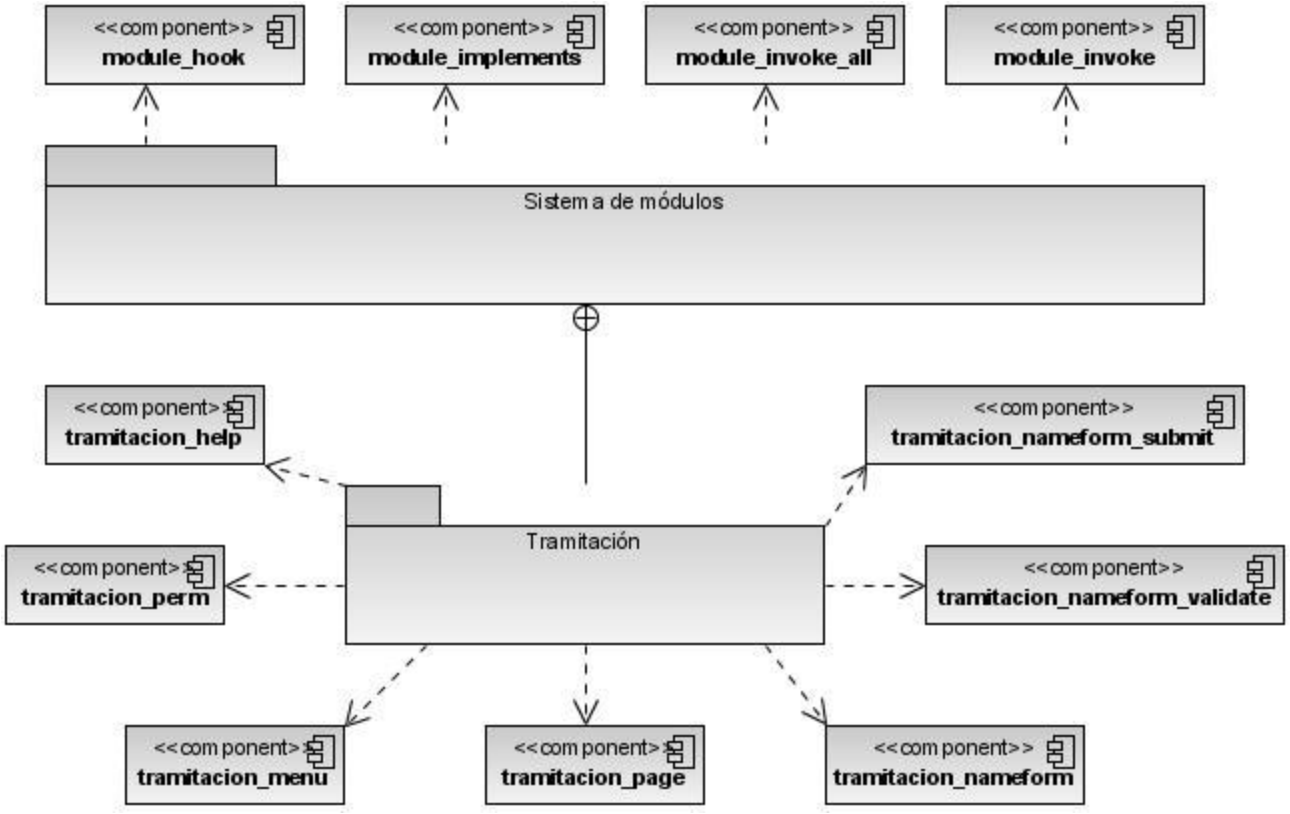

Figura 16: *Implementación del sistema de módulos.*

Tabla 6: *Implementación del módulo Tramitación.*

Descripción de los *Hooks* o funciones		
Función	Estereotipo	Descripción
Module_hook	Componente	Función empleada por el sistema de módulos para determinar si un módulo implementa un *hook* determinado.
Module_implements	Componente	Función empleada por el sistema de módulos para determinar que módulos están ejecutando un *hook* determinado.
Module_invoke	Componente	Función empleada por el sistema de módulos para invocar un *hook* en un módulo determinado.
Module_invoke_all	Componente	Función empleada por el sistema de módulos para invocar un *hook* en todos los módulos que lo tienen implementado.
Tramitacion_help	Componente	A través de este *hook* el módulo proporciona información de interés sobre él mismo que puede ser usada por otros módulos y servir como ayuda a los usuarios.

Tramitacion_perm	Componente	Este *hook* provee los permisos definidos en el módulo, así estos pueden ser seleccionados en la pagina de permisos de usuarios para restringir el acceso a las acciones que realiza el módulo.
Tramitacion_menu	Componente	Mediante este *hook* se definen los elementos del menú de este módulo y las llamadas al servidor que se harán a través de estos.
Tramitacion_page	Componente	Este *hook* es invocado cuando se el servidor recibe una petición proveniente de los elementos del menú de este módulo. Retorna todo el html generado a partir de la estructura de datos del formulario pedido.
Tramitacion_nameform	Componente	A través de este *hook* se obtiene el formulario que se muestra en cada estado del flujo de trabajo del Subsistema de Trámites para crear o modificar los nodos necesarios.
Tramitacion_nameform_validate	Componente	Este *hook* es invocado inmediatamente después que se le hace *submit* a cualquier formulario del módulo, se encarga de realizar ciertas validaciones de condiciones necesarias para poder llevar a cabo esta operación.
Tramitacion_nameform_submit	Componente	Mediante este *hook* se realizan todas las consultas a la base de datos necesarias para gestionar la información enviada a través del formulario.

2.8 Vista de Despliegue

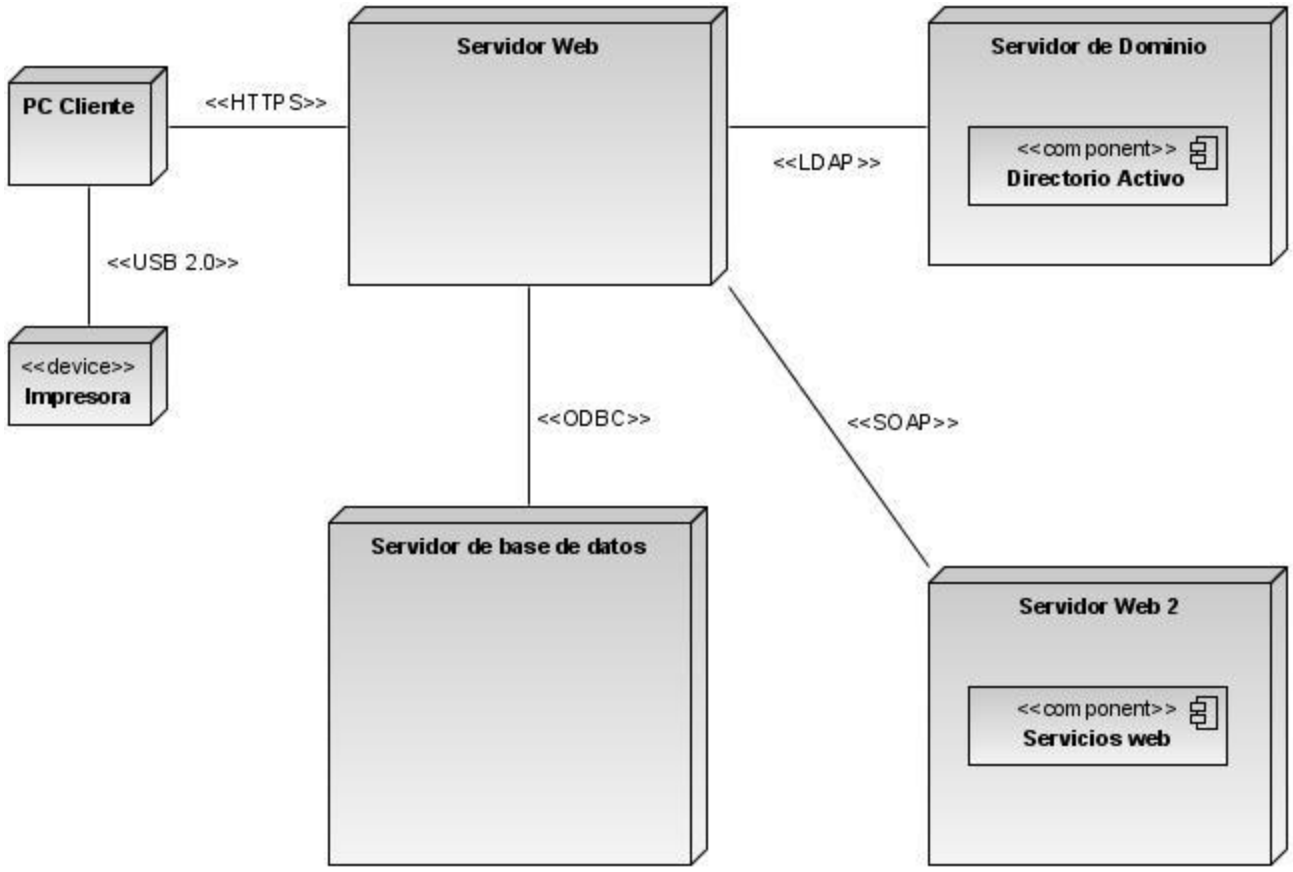

Figura 17: *Diagrama de despliegue.*

En la tabla que se muestra a continuación aparecen detallados todos los elementos de este diagrama, con el objetivo de proporcionar una explicación de las funciones y características de los mismos:

Tabla 7: *Descripción de elementos e interfaces de comunicación.*

Descripción de elementos e interfaces de comunicación		
Nombre	Tipo	Descripción
Impresora	Dispositivo	Estos dispositivos son utilizados para satisfacer las necesidades de impresión de reportes y documentos generados por el Sistema.
PC Cliente	Procesador	Son los nodos clientes del Sistema, a través de estos se accede al mismo mediante los navegadores web. Se encuentran distribuidos por toda la Universidad incluyendo la DCI, cliente principal del Sistema.
Servidor de Aplicaciones	Procesador	Es el entorno de ejecución del Sistema cuenta con toda la tecnología necesaria, tanto de software como de hardware para la puesta en marcha y funcionamiento del Sistema.
Servidor de Base de Datos	Procesador	Contiene la base de datos del Sistema, donde se encuentra almacenada toda la información que se necesita y se almacena toda la nueva introducida a través de dicho Sistema.
Servidor de Dominio	Procesador	Contiene el Directorio Activo de la Universidad (componente representado), donde se encuentra registrado los datos de usuarios y contraseñas, permisos, certificados, etc.
Servidor de Aplicaciones 2	Procesador	Entorno de ejecución de otros sistemas con los que se hace necesario interactuar mediante servicios web (componente representado) que contienen funcionalidades imprescindibles para el Sistema.
USB 2.0	Puerto de conexión	Se utiliza para conectar diferentes clases de dispositivos con un ordenador, en este caso se utilizará para conectar impresoras a los nodos clientes, se caracteriza por la alta velocidad en la trasferencia de los datos.
HTTPS	Protocolo	Protocolo seguro de transferencia de hipertexto, utiliza el cifrado de paquetes basado en las SSL, garantizando así un canal seguro de comunicación.
ODBC	Protocolo	Es un estándar de acceso a base de datos, que garantiza la interoperabilidad con disimiles sistemas gestores de base de datos. Se integra perfectamente con PostgreSQL.
SOAP	Protocolo	Protocolo basado en el estándar XML para la transferencia de datos entre sistemas, garantizando así la independencia de plataforma.
LDAP	Protocolo	Es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

2.9 Conclusiones

En este capítulo se elaboró la propuesta de solución del sistema mediante la línea base de la arquitectura. Se definieron los patrones de diseño utilizados por el Sistema así como también se define el estilo arquitectónico que lo caracteriza. Se definieron los requisitos no funcionales del Sistema, es decir, las herramientas y tecnologías necesarias para desarrollar la Aplicación, además se expusieron

las diferentes vistas planteadas por RUP, imprescindibles para la comprensión y el desarrollo del mismo.

CAPÍTULO 3: EVALUACIÓN DE LA ARQUITECTURA

3.1 Introducción

Evaluar la arquitectura es indispensable si se desea obtener un producto de calidad. Es tan importante como difícil pero es una práctica sumamente valiosa, pues los defectos de diseño que se corrigen a tiempo, eliminan cadenas de errores que se pudieran generar, es decir, sería aún más costoso en términos de tiempo y dinero dejar pasar por alto un error que pudo corregirse con anterioridad.

En el presente capítulo se realiza un análisis sobre cuándo, cómo y por qué se evalúa la arquitectura, así como quiénes participan y qué se evalúa. Posteriormente es necesario hacer una valoración de la arquitectura propuesta en este trabajo.

3.2 Evaluando una arquitectura de software

La calidad de un sistema depende en gran medida de su arquitectura. La evaluación de la misma permite obtener una visión de las decisiones más riesgosas asociadas al software en cuestión y además, aquellas que son buenas y correctas. Mientras más temprano se mitigue un riesgo o se corrija un error, mejor es, tanto para el equipo de desarrollo como para el cliente.

“Una mala arquitectura puede llevar a un proyecto al fracaso. Todos los requerimientos de calidad pueden quedar insatisfechos.

La arquitectura también determina la estructura del proyecto: configuración, agenda y presupuesto, alcance, entre otros aspectos. Es mejor cambiar la arquitectura antes que otros artefactos, que están basados en ella, se estabilicen.

Realizar una evaluación de la arquitectura es la manera más económica de evitar desastres.” [10].

3.2.1 ¿Cuándo una arquitectura puede ser evaluada?

Habitualmente, la evaluación de la arquitectura ocurre después que ésta ha sido especificada, pero antes que empiece la implementación, esto se hace con el objetivo de validar el diseño propuesto. No obstante, uno de los aspectos más interesantes de la evaluación de arquitecturas es que se puede efectuar en cualquier etapa de la vida de una arquitectura. Básicamente existen dos etapas: temprana y tardía. En la primera no tiene por qué estar especificada completamente la arquitectura, en la

mayoría de los casos es utilizada para examinar las decisiones arquitectónicas ya tomadas y decidir entre las opciones que están pendientes. Por otro lado la evaluación tardía es realizada tanto cuando la arquitectura está terminada como cuando la implementación está completa.

"En general, una evaluación debe realizarse cuando hay suficiente de la arquitectura como para justificarlo. Una buena regla sería: *realizar una evaluación cuando el equipo de desarrollo empieza a tomar decisiones que dependen de la arquitectura y el costo de deshacerlas sobrepasa al costo de realizar una evaluación*." [10].

3.2.2 ¿Quiénes participan?

Durante la evaluación de la arquitectura participan los *stakeholders*: personal interesado en el desarrollo de la arquitectura y del software, pueden ser desarrolladores como el arquitecto, los diseñadores, el líder del proyecto, los implementadores o los verificadores, de ellos, tres primeros son *dicision markers* (tomadores de decisiones); por otro lado se tiene al equipo de evaluación, que no son más que personas que guiarán el proceso de evaluación y además realizarán el análisis.

3.2.3 Planificación de las evaluaciones.

Las evaluaciones de la arquitectura pueden planearse o no. Una evaluación planeada es aquella que fue concebida dentro del ciclo de desarrollo, en cambio una no planeada ocurre cuando se han descubierto defectos en la arquitectura, por lo general en etapas tardías. Existen grandes riesgos de tener retrasos en la fecha de entrega como consecuencia de realizar una evaluación no planificada, es por ello que se recomienda planificar una o dos evaluaciones.

3.2.4 ¿Qué resultados produce la evaluación de una arquitectura?

La evaluación de una arquitectura no produce resultados cuantitativos. El objetivo de mitigar los riesgos, es aprender cómo un atributo de calidad es afectado por una decisión de diseño arquitectónico, para que de esta manera se pueda estudiar con cuidado dicha decisión. La evaluación ayuda a encontrar debilidades, no dirá "si" o "no", "correcto" o "incorrecto", dirá donde se encuentran los riesgos. En caso de que existan contradicciones, el líder del proyecto deberá tomar la decisión.

"En términos concretos, la evaluación de la arquitectura produce un informe, la forma y contenido del mismo varía según el método utilizado. En particular, produce repuestas a dos tipos de preguntas:

¿Es esta arquitectura adecuada para el sistema para la cual fue diseñada?

¿Cuál de dos o más arquitecturas propuestas es la más adecuada para el sistema?“ [10].

3.2.5 Atributos por los cuales puede ser evaluada la arquitectura.

Evaluar la arquitectura es difícil ya que en muchas ocasiones los atributos de calidad no son comprendidos correctamente, la arquitectura ha sido descrita con un nivel alto de abstracción o falta experiencia al equipo de evaluación. Al evaluar la arquitectura no se alcancanzará el cien por ciento de la clidad requerida, pero muchos de los atributos de calidad se encuentran directamente en el mundo de la arquitectura. Los siguientes atributos ayudan a evaluar la arquitectura:

- *Performance*: Es el tiempo requerido para responder a un estímulo (evento), o número de eventos por unidad de tiempo.
- *Reliability*: Es la habilidad del sistema de continuar operando sobre el tiempo. Es usualmente medida en tiempo promedio entre fallas.
- *Availability*: Es la porción de tiempo en que el sistema esta levantado y corriendo. Se mide como el tiempo transcurrido entre fallas, así como, cuan rápido el sistema esta apto para reanudar y quedar operativo ante una falla.
- *Security*: Es la medida de la habilidad del sistema de resistirse al uso no autorizado y negar los servicios, mientras los provee a usuarios legítimos.
- *Modifiability*: Es la habilidad de realizar cambios al sistema en forma rápida y a bajo costo.
- *Portability*: Es la habilidad del sistema de correr sobre diferentes ambientes. Estos ambientes pueden ser de hardware, de software o una combinación de ambos. *Portability* es un caso particular de *modifiability*.
- *Functionality*: Es la habilidad del sistema de hacer el trabajo para el cual fue construido.
- *Variability*: Es la capacidad de la arquitectura de ser expandida o modificada para producir nuevas arquitecturas. *Variability* es importante cuando la arquitectura se va a utilizar como piedra fundamental de toda una familia de productos relacionados, como puede ser una línea de producto.
- *Subsetability*: Es la habilidad de soportar la producción de un subconjunto del sistema. *Subsetability* permite el desarrollo incremental. Es un tipo especial de *variability*.
- *Conceptual integrity*: Es la visión que unifica el diseño del sistema en todos los niveles. La arquitectura debe hacer cosas similares en forma similar. Debe mostrar consistencia en los

mecanismos, decisiones y patrones aplicados.

3.2.6 ¿Por qué los atributos de calidad son demasiados imprecisos para el análisis?

Los atributos de calidad son la base para la evaluación de una arquitectura, pero por si solos no son suficientes para juzgar la adecuabilidad de la arquitectura. Generalmente, los requerimientos son escritos de la siguiente manera [10]:

"El sistema debe ser robusto."
"El sistema debe ser modificable."
"El sistema debe ser seguro."
"El sistema debe tener una performance aceptable."

Cada una de las frases anteriores está sujeta a diferentes interpretaciones y malos entendidos. Lo que uno puede considerar robusto, otro puede considerarlo apenas aceptable.

El punto es que los atributos de calidad no son cantidades absolutas, existen en un contexto con metas específicas. En particular:

Un sistema es modificable (o no) con respecto a un tipo de cambio específico.
Un sistema es seguro (o no) con respecto a un tipo de amenaza específica.
Un sistema es confiable (o no) con respecto a la ocurrencia de un tipo de falta específica.
Un sistema es performante (o no) con respecto a un criterio específico de performance.
Una arquitectura es construible (o no) con respecto a restricciones de tiempo y presupuesto específicas.

No parece razonable, considerar que un sistema pueda alguna vez, por ejemplo, ser completamente confiable bajo toda circunstancia (pensar en problemas de energía, meteorológicos, entre otros). Dado esto, es importante que el arquitecto entienda perfectamente bajo que circunstancias un sistema debe ser confiable para ser considerado aceptable. Por lo tanto, el primer trabajo que debe realizar una evaluación de arquitectura es obtener las metas específicas de calidad ante las cuales la arquitectura será juzgada.

Si algunas de estas metas no son específicas o son ambiguas, se debe pedir a los *stakeholders* que ayuden al equipo de evaluación a reescribirlas. El mecanismo a utilizar para representar estas metas es el de escenario. Un escenario es una pequeña descripción de la interacción de un *stakeholder* con el sistema. Los escenarios son parecidos a los casos de uso.

3.2.7 ¿Cuáles son las salidas de una evaluación arquitectónica?

Sólo se puede proceder a realizar una evaluación arquitectónica si se conoce el criterio de adecuabilidad. Es imprescindible obtener los atributos de calidad requeridos contra los cuales será calificada la arquitectura. Además se adquiere conocimiento acerca de cuáles son los riesgos o decisiones potencialmente peligrosas y cuáles son los no riesgos o decisiones correctas.

"Documentar riesgos y no riesgos consiste en:

Una decisión arquitectónica (o una decisión que no ha sido tomada).

Una respuesta específica al atributo de calidad que esta siendo tratado, junto con las consecuencias del nivel predecible de la respuesta.

- Una base lógica por el efecto positivo o negativo que la decisión tuvo en alcanzar el requerimiento del atributo de calidad." [10].

3.2.8 ¿Cuáles son los beneficios de realizar una evaluación arquitectónica?

Cuando se evalúa la arquitectura es una excelente oportunidad para reunir a las *stakeholders* y brindarles una explicación clara y minuciosa de la arquitectura, de aquí surgen nuevas ideas, se descubren oportunidades de rehúso, se mejora la calidad de la documentación y si alguna no esta terminada todavía, se le asigna un responsable.

"El mayor beneficio que brinda la evaluación de una arquitectura, es que descubre los problemas que si se hubiesen dejado sin descubrir, habría sido mucho, más costoso corregirlos luego. En breve, una evaluación produce una mejor arquitectura." [10].

3.3 Técnicas de evaluación

Existen varias técnicas de evaluación de la arquitectura de un software, básicamente se dividen en dos grupos: cualitativas y cuantitativas, ellas se utilizan a consecuencia del estado actual de la arquitectura. Cuando la arquitectura del sistema ha sido implantada se ponen en práctica técnicas cuantitativas, más cuando está en proceso de construcción se aplican técnicas cualitativas para su evaluación.

Dentro de las técnicas cualitativas se hallan cuestionarios, listas de verificación o escenarios. Por su parte las métricas, simulaciones, prototipos, experimentos y modelos matemáticos se clasifican como técnicas cuantitativas.

3.3.1 ¿Cómo se puede realizar una evaluación arquitectónica?

Una de las características fundamentales de la tecnología RUP es que el sistema es centrado en la arquitectura, consecuentemente, la calidad del sistema depende en gran medida de ella. Es posible estimar la calidad de la arquitectura a través de los requisitos no funcionales. Para ello existen métodos como el ATAM (*Architecture Trade-Off Analysis Method*), SAAM (*Software Architecture Analysis Method*) y ARID (*Active Reviews Intermediate Designs*) de los cuales se ofrecen los pasos que se realizan en cada uno de ellos, a continuación.

- Método SAAM
  - ✓ PASO 1: Desarrollo de escenarios.
  - ✓ PASO 2: Descripción de la arquitectura.
  - ✓ PASO 3: Clasificación de escenarios.
  - ✓ PASO 4: Evaluación de escenarios.
  - ✓ PASO 5: Interacción de escenarios.
  - ✓ PASO 6: Evaluación general.
- Método ATAM

  Presentación
  - ✓ PASO 1: Presentar el ATAM.
  - ✓ PASO 2: Presentar pautas del Negocio.
  - ✓ PASO 3: Presentar la arquitectura.

  Investigación y análisis.
  - ✓ PASO 4: Identificar las propuestas arquitectónicas.
  - ✓ PASO 5: Generar el árbol de utilidad de los atributos de calidad.
  - ✓ PASO 6: Analizar las propuestas arquitectónicas.

  Pruebas.
  - ✓ PASO 7: Lluvia de ideas y priorización de escenarios.
  - ✓ PASO 8: Analizar las propuestas arquitectónicas.

Informes.

- ✓ PASO 9: Presentar los resultados.

- Método ARID

  PRE-reunión.

  - ✓ Identificar los revisores.
  - ✓ Preparar la presentación de diseño.
  - ✓ Preparar los escenarios.
  - ✓ Preparar los materiales.

  Evaluación.

  - ✓ Presentación del ARID
  - ✓ Presentación del diseño.
  - ✓ Lluvia de ideas y priorización de escenarios.
  - ✓ Se realiza la revisión.
  - ✓ Conclusiones.

Cualquiera de estos métodos con sus concernientes técnicas se utiliza para realizar la evaluación de una arquitectura de software. Luego de estudiar los métodos más relevantes (aunque solo se mencionan los respectivos pasos de cada uno de ellos) y teniendo en cuenta el propósito, el contexto, las debilidades y fortalezas, y además valorando que la arquitectura se encuentra en sus primeras etapas, se decidió aplicar el método ARID.

3.4 Evaluando la arquitectura propuesta

Dado el estado actual de la arquitectura, se aplica la técnica cualitativa de evaluación por escenarios, específicamente el método ARID. “El método ARID es conveniente para realizar la evaluación de diseños parciales en las etapas tempranas de desarrollo.” [10].
A continuación se definen los escenarios fundamentales de la arquitectura del sistema:

- **Rendimiento**

Para evitar la congestión del sistema se aplican estrategias como la suspensión temporal de servicios recurrentes o iterativos que estén activos hasta tanto no se reciban peticiones del cliente. Por otra parte se utiliza el módulo *Throttle*, de Drupal (encargado de detectar el incremento en el tráfico hacia el sistema y de desactivar funciones criticas del CPU de forma temporal sobre todo en módulos en desuso).

- **Seguridad**

Sin dudas la seguridad es uno de los elementos más primordiales de cualquier arquitectura, y por supuesto no es pasada por alto. La solución propuesta hace uso de canales de comunicación segura, a través del protocolo SSL; evitando así que intrusos falsifiquen su identidad y puedan interceptar o modificar cualquier información. El sistema permite la inserción de varios usuarios con iguales o diferentes privilegios dentro de él y se sirve del módulo *User* de Drupal, para validar así los privilegios de acceso y demás funciones del usuario activo dentro del sistema (se puede acceder como invitado si se desea, este es el nivel de privilegios más bajo y es como se inicia el sistema, hasta que el cliente decida autenticarse). Por otra parte, se utiliza el módulo *LDAP_integration*, que permite validar usuarios de dominios existentes previamente especificados, de esta forma también tendrían autoridad dentro del sistema, sin necesidad de predefinir un usuario para cada miembro del dominio que se desea incluir.

- **Portabilidad**

El sistema utiliza el lenguaje estándar de marcado XML y el lenguaje de programación PHP, emplea PostgreSQL como gestor de base de datos relacionales, además de Apache como servidor web y el CMS Drupal. Todo lo antes mencionado está liberado por licencias libres, lo cual implica que fusionan un sistema perfectamente funcional en diferentes plataformas, fundamentalmente Windows, Unix, Linux y otras.

- **Disponibilidad**

Drupal presenta la posibilidad de mantener disponible el sistema en la red, pues permite crear y configurar un *virtualhost* dentro de los ficheros de configuración del mismo. Dentro del fichero “settings.php” se declara el nombre y dirección con la cual se brinda el acceso. De esta forma Drupal se sirve del protocolo HTTP para que otras estaciones se conecten al sistema, siempre y cuando utilicen el nombre predefinido en el servidor.

- **Independencia entre la base de datos y la aplicación**

Este es un factor fundamental dentro de la eficiencia de la arquitectura. Esta característica facilita el mantenimiento del sistema y garantiza daños menores en caso de fallas o errores del mismo. Además, si fuera necesario sustituir el sistema de base de datos o el CMS los cambios respectivos serán únicamente de comunicación entre ambos.

Drupal logra esta independencia a través de la capa de abstracción de base de datos, la misma está implementada y mantenida para MySQL y PostgreSQL, aunque permite incorporar fácilmente soporte para otras bases de datos.

3.5 Conclusiones

Dada la situación y características que presenta el software cubano no es un secreto la necesidad de migrar cada sistema informático a software libre. Se tiene la posibilidad de iniciar de cero un nuevo sistema, por lo cual no se debe diseñar una arquitectura dependiente de licencias propietarias, pues sería idear un sistema que debe ser sustituido en cualquier momento. Con esta premisa, la prioridad fue lograr una arquitectura que cumpliera con todas las normas de calidad y requisitos necesarios, basado en tecnologías y licencias libres; consecuentemente la propuesta de solución expuesta es idónea.

Es una arquitectura robusta, asimila cambios con facilidad (altamente escalable), capaz de responder a un sin número de peticiones de usuarios concurrentes y además consta de un eficiente control de acceso. Sus niveles de seguridad y disponibilidad son aceptables.

CONCLUSIONES

Se realizó un estudio del estado del arte sobre arquitectura para sistemas de gestión vinculados a procesos de relaciones internacionales y a la gestión de maneras diversas, encontrando así, cual de las características presentes en ellos eran la más apropiadas para aplicar a la descrita en este trabajo.

La arquitectura diseñada es reusable porque permite la creación de nuevos sistemas de gestión con características semejantes, con tiempos de desarrollado relativamente cortos y sin cambios a la arquitectura.

Al ser diseñada y desarrollada sobre Drupal, la arquitectura permite la agregación, eliminación e implementación de nuevas funcionalidades sin que la arquitectura tenga que cambiar su estructura, poniéndose así de manifiesto su flexibilidad.

En el trabajo desarrollado se describieron características arquitectónicas esenciales y se generaron las vistas necesarias que permiten la comprensión y desarrollo del Sistema.

Todos los resultados obtenidos y analizados son positivos a pesar que las métricas de arquitectura no son medibles numéricamente, evaluando así a la arquitectura como buena.

RECOMENDACIONES

- Se recomienda al equipo de desarrollo del SICI este documento como una guía para el desarrollo puesto que proporciona una visión abstracta de alto nivel de todo el Sistema.
- Se recomienda al equipo de arquitectura del SICI continuar con un refinamiento de la descripción arquitectónica a lo largo del ciclo de desarrollo, para alcanzar asi un Documento de Arquitectura más enriquecido.
- Se recomienda al equipo de arquitectura del SICI refinar la evaluación de la arquitectura utilizando varios métodos evaluativos para lograr una identificación mas profunda de posibles errores.
- Se recomienda el uso del presente trabajo de manera general por parte de proyectos que pretendan desarrollar sobre el CMS Drupal puesto que contiene varias descripciones que pueden ser útiles para la comprensión de su arquitectura.

BIBLIOGRAFÍA REFERENCIADA

[1] VALLE, A. E. D. *Inauguró un nuevo curso Universidad de las Ciencias Informáticas*, 2006. [Disponible en: http://www.somosjovenes.cu/index/semana33/uci.htm

[2] PAUL CLEMENTS, F. B., LEN BASS, DAVID GARLAN, JAMES IVERS, REED LITTLE, ROBERT NORD, JUDITH STAFFORD. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2002. 512 p.

[3] CHRISTOPHER ALEXANDER, S. I., MURRAY SILVERSTEIN, MAX JACOBSON, INGRID FIKSDAHL-KING Y SHLOMO ANGEL. *A Pattern Language: Towns/Building/Construction*. Oxford Press, 1977.

[4] CRAIG LARMAN, B. M. V. *UML y Patrones: una introducción a análisis y diseño orientado a objetos y al proceso unificado.* . Prentice-Hall, 2006. 590 p.

[5] *Lenguaje Unificado de Modelado*. Disponible en: http://es.wikipedia.org/wiki/Uml

[6] XAVIER CUERDA GARCIA, J. M. A. *Introducción a los Sistemas de Gestión de Contenidos (CMS) de código abierto*, 2004. [Disponible en: http://mosaic.uoc.edu/articulos/cms1204.html

[7] LUKE WELLING, L. T. *Desarrollo Web con PHP y MySQL*. Grupo Anaya Comercial, 2005. 976 p.

[8] *Sitio oficial de postgreSQL*. Disponible en: http://www.postgresql.org/docs/faqs.FAQ.html

[9] *Coding standards*. Disponible en: http://drupal.org/coding-standards

[10] PAUL CLEMENTS, R. K., MARK KLEIN. *Evaluating Software Architectures: Methods and Case Studies*. Addison-Wesley, 2001. 323 p.

BIBLIOGRAFÍA CONSULTADA

1. ANACLETO, V. A. *El arquitecto de software como un ser visionario* 2006. [Disponible en: http://www.epidataconsulting.com/tikiwiki/tiki-read_article.php?articleId=40http://drupal.org
2. *Desarrollo sobre el CMS Drupal*. Disponible en: http://api.drupal.org/api/5
3. BARCO, A. *SOA y los Servicios Web (II)*, 2006. [Disponible en: http://arquitecturaorientadaaservicios.blogspot.com/
4. *Sitio oficial de postgreSQL*. Disponible en: http://www.postgresql.org/docs/faqs.FAQ.html
5. XAVIER CUERDA GARCIA, J. M. A. *Introducción a los Sistemas de Gestión de Contenidos (CMS) de código abierto*, 2004. [Disponible en: http://mosaic.uoc.edu/articulos/cms1204.html
6. *Lenguaje Unificado de Modelado*. Disponible en: http://es.wikipedia.org/wiki/Uml
7. VALLE, A. E. D. *Inauguró un nuevo curso Universidad de las Ciencias Informáticas*, 2006. [Disponible en: http://www.somosjovenes.cu/index/semana33/uci.htm
8. *Coding standards*. Disponible en: http://drupal.org/coding-standards
9. PAUL CLEMENTS, F. B., LEN BASS, DAVID GARLAN, JAMES IVERS, REED LITTLE, ROBERT NORD, JUDITH STAFFORD. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2002. 512 p
10. PAUL CLEMENTS, R. K., MARK KLEIN. *Evaluating Software Architectures: Methods and Case Studies*. Addison-Wesley, 2001. 323 p.
11. CRAIG LARMAN, B. M. V. *UML y Patrones: una introducción a análisis y diseño orientado a objetos y al proceso unificado.* Prentice-Hall, 2006. 590 p.
12. LEN BASS, P. C., RICK KAZMAN. *Software Architecture in Practice*. Addison-Wesley, 2003. 528 p.
13. SHAW, D. G. Y. M. *An introduction to Software Architecture*, 1994. [Disponible en: http://www.cs.cmu.edu/afs/cs/project/vit/ftp/pdf/intro_softarch.pdf
14. *Introducción a los Sistemas de Gestión de Contenidos (CMS)*. 2005. [Disponible en: http://www.uoc.edu/mosaic/articulos/cms1204.html.
15. CHRISTOPHER ALEXANDER, S. I., MURRAY SILVERSTEIN, MAX JACOBSON, INGRID FIKSDAHL-KING Y SHLOMO ANGEL. *A Pattern Language: Towns/Building/Construction*. Oxford Press, 1977.

16. CLEMENTS, P. *A Survey of Architecture Description Languages*, lEEE Computer Society, 1996. [Disponible en: http://portal.acm.org/citation.cfm?id=858261
17. LUKE WELLING, L. T. *Desarrollo Web con PHP y MySQL*. Grupo Anaya Comercial, 2005. 976 p.
18. JOHN K. VANDYK, M. W. *Pro Drupal Development* Apress, 2007. 428 p.
19. IVAR JACOBSON, G. B. Y. J. R. *The Unified Software Development Process.* Addison-Wesley, 1999. 463 p.
20. GRADY BOOCH, J. R. E. I. J. *El Lenguaje Unificado de Modelado.* Madrid, Addison-Wesley, 1999. 432 p.

GLOSARIO

BSD: Es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Pertenece al grupo de licencias de software Libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

COMMIT: (acción de cometer) Se refiere a la idea de hacer que un conjunto de cambios "tentativos, o no permanentes" se conviertan en permanentes. Un uso popular es al final de una transacción de base de datos.

COOKIES: Fragmento de información que se almacena en el disco duro del visitante de una página web a través de su navegador, a petición del servidor de la página. Esta información puede ser luego recuperada por el servidor en posteriores visitas. Principalmente se usa para llevar el control de usuarios, ofrecer opciones de diseño (colores, fondos, etc.) o de contenidos al visitante, y/o para conseguir información sobre los hábitos de navegación del usuario.

Framework: Es un marco de trabajo definido en la cual otro software puede ser construido, puede incluir soporte de programas, bibliotecas, entre otros software para ayudar a desarrollar o unir componentes de un proyecto. También se le suele llamar plataforma.

GPL: Es una licencia creada por la Free Software Foundation, está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

HARDWARE: Es la parte física de un computador y más ampliamente de cualquier dispositivo electrónico.

HTTP (HyperText Transfer Protocol): protocolo de transferencia de hipertexto es el protocolo usado en cada transacción de la Web (WWW). El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido.

IEEE: El instituto de ingenieros eléctricos y electrónicos (*The Institute of Electrical and Electronics Engineers**)*** es una asociación técnico-profesional mundial dedicada a promover la creatividad, el desarrollo y la integración, compartir y aplicar los avances en las tecnologías de la información, electrónica y ciencias en general para beneficio de la humanidad.

INTRANET: es una red de computadoras dentro de una red de área local (LAN) privada, empresarial o educativa que proporciona herramientas de Internet. Tiene como función principal proveer lógica de negocios para aplicaciones de captura, informes y consultas con el fin de facilitar la producción de dichos grupos de trabajo; es también un importante medio de difusión de información interna a nivel de grupo de trabajo.

ONLINE: En general, se dice que algo está en línea, on-line u online si está conectado a una red o sistema mayor.

PLUG-INS: Es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica, generalmente tienen la función de reproducir determinados formatos de gráficos, reproducir datos multimedia, codificar/decodificar emails, filtrar imágenes de programas gráficos.

RELEASE: Término empleado para indicar una versión de un software, ya se Alfa, Beta, releases estables, etc.

SOFTWARE: Se refiere al equipamiento lógico o soporte lógico de un computador digital, comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica. Incluyen, entre otras, aplicaciones informáticas y software de sistema.

STAKEHOLDERS: Son aquellos que serán afectados por el proyecto y que pueden ejercer cierta influencia sobre él, pero que no están directamente involucrados con la ejecución del trabajo.

THEMES: en Drupal se emplea para llamar al conjunto de estilos CCS, imágenes, scripts que conforman una interfaz de usuario.