Universidad de las Ciencias Informáticas
Facultad 3

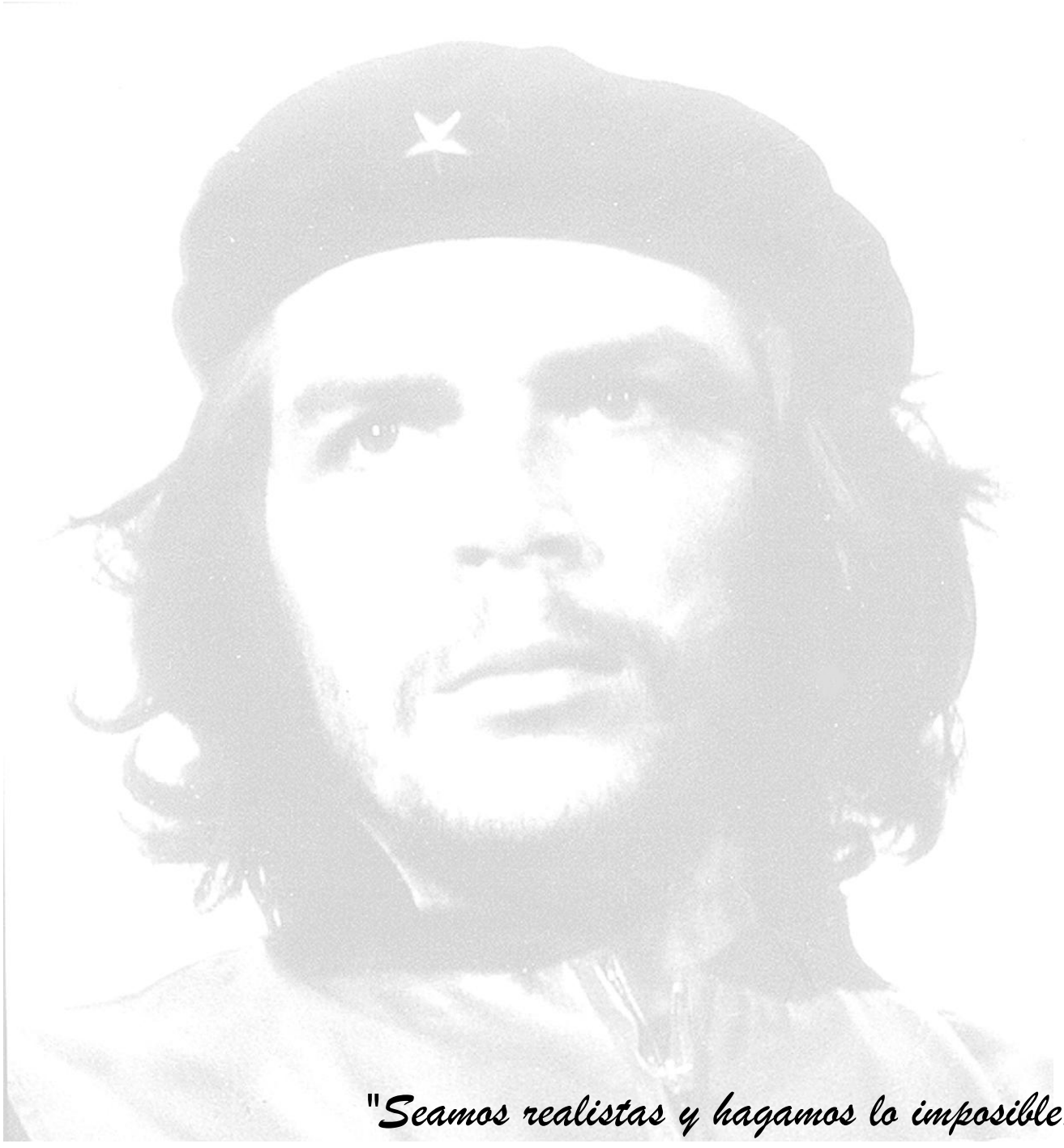
Título: Propuesta de arquitectura para el Sistema de Gestión de Información de los Recursos de la Facultad 3

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

Autor: Larisa González Álvarez
Tutor: Pascual Verdecia Vicet
Asesor: Pedro Yobanis Piñero Pérez

Ciudad de la Habana

Junio, 2008



"Seamos realistas y hagamos lo imposible."

DECLARACIÓN DE AUTORÍA

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

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

Larisa González Álvarez

_____

Firma del Autor

Dr.C Pascual Verdecia Vicet

_____

Firma del tutor

DATOS DE CONTACTO

Pascual Verdecia Vicet
Ingeniero de Minas, Ingeniero Civil (1994), Máster en Voladura con Explosivos, Doctor en Ciencias Técnicas (2002), 21 años profesor de Física, Categoría Asistente.
pverdecia@uci.cu

Pedro Yobanis Piñero Pérez
Licenciado en Ciencias de la Computación (2000), Doctor en Ciencias Técnicas (2005), Categoría Auxiliar.
ppp@uci.cu

AGRADECIMIENTOS

A mi mamá, mi papá y mi hermana por ser el pilar fundamental de todas las obras de mi vida.

A mi Roberkys, por su apoyo en los momentos difíciles, sus enseñanzas imperecederas y su amor incondicional.

A mi familia que siempre está ahí para apoyarme.

A mis amigos, por estar sobre todo y para todo.

A mis compañeros de estudios en la UCI, que me sacan las castañas tantas veces del fuego y me dan el apoyo que necesito.

A Pascual y Pedro por sus experiencias inigualables de aporte a este trabajo.

A todas las personas que de una forma u otra, contribuyeron a mi formación profesional así como al desarrollo de este trabajo.

DEDICATORIA

A Fidel y la Revolución por dejarme ser parte de la UCI.

A mi familia, por su apoyo estos cinco años.

Al nene, por estar.

A mis amigos.

RESUMEN

La Facultad 3 de la Universidad de las Ciencias Informáticas se ha propuesto automatizar el Proceso de Gestión de Información de los Recursos que ella maneja, de ahí que surja la necesidad de determinar los elementos bases que determinen el sistema informático a crearse, de manera que se redunde también en un software con calidad.

El presente trabajo cuenta con un estudio del estado del arte enmarcado en elementos básicos relativos a la arquitectura de software pudiendo, a partir de ellos, definir una propuesta arquitectónica para el Sistema de Gestión de Información de los Recursos de la Facultad 3 que contribuya a garantizar su mantenibilidad, eficiencia, seguridad, portabilidad y estabilidad. También se enfoca el estudio a partir de las responsabilidades a cargo de un arquitecto, y las características usuales en los Sistemas de Información para probar la utilidad y actualidad de la propuesta.

Asimismo la proposición se documenta a partir de la generación del artefacto fundamental definido para el rol de arquitecto en la metodología RUP, teniendo en cuenta los puntos establecidos para el documento según la plantilla que se destina para tal efecto. En vistas a la validación de la propuesta se empleó el método de evaluación de arquitecturas QUASAR, que permite determinar el cumplimento de la arquitectura con los atributos de calidad establecidos y los requerimientos asociados.

PALABRAS CLAVE

- ✓ Arquitectura de software.
- ✓ Factor de calidad.
- ✓ Método QUASAR.
- ✓ Requerimiento no funcional.

TABLA DE CONTENIDOS

INTRODUCCIÓN

La aparición de mayores y mejores tecnologías es uno de los atributos más llamativos del presente siglo. Muchas son las ramas de la ciencia que se han visto favorecidas con este auge del conocimiento y el desarrollo científico-técnico. La informática se puede incluir entre estas materias vanguardistas.

Es tanto así el desarrollo en esta rama que es inevitable en cualquier sector de la vida socio-económica de un país no pensar en la aparición de un sistema informático capaz de convertir un problema empresarial en una solución eficazmente factible. Para que ello sea posible se han invertido numerosos esfuerzos humanos y han surgido innumerables paradigmas, metodologías, herramientas y prácticas relativas a la computación y la informática.

Cuba no se ha mantenido al margen de esta situación, por el contrario, ha favorecido la creación de estructuras que permitan una “informatización” del país en vistas al desarrollo productivo y social que esto pueda reportar. En este marco de ideas revolucionarias se inserta la construcción de la Universidad de las Ciencias Informáticas (UCI) que vincula la formación, producción e investigación en su seno.

La UCI representa en estos momentos la estructura educacional de mayor magnitud en Cuba. Ello implica que maneje un elevado número de recursos tanto materiales como humanos. Si se unen estas características estructurales a las metas internas en cuanto al plan de enseñanza y producción, así como al propósito de ser “la ciudad digital de Cuba”, surge la necesidad de ejecución de numerosos proyectos informáticos que tributen a dicho fin.

La Facultad 3 de la UCI, uniéndose a este marco de trabajo, consecuente a la política de uso del software libre, se ha propuesto perfeccionar el proceso de gestión de información de los recursos de la facultad mediante la automatización del mismo. Esto reportaría muchos beneficios para el desarrollo y organización de la facultad, permitiendo un mejor manejo de los volúmenes de información asociados a este proceso. Además se podrá tener un acceso más adecuado a la información, con los privilegios que corresponda en cada caso. También se

podrán publicar todos los datos necesarios, buscarlos y manejarlos, todo de forma automática, ahorrando trabajo manual y tiempo.

Hoy día en el mundo la temática de la gestión ha tomado una nueva fuerza, adoptándose, más que como una estrategia, como una forma eficaz para garantizar el éxito de cualquier empresa. Se han desarrollado numerosos sistemas de gestión de los recursos humanos (RRHH). Ejemplo de ellos es *Nexo Digital Huma Nex,* una aplicación de gestión de recursos humanos con una amplia y variada experiencia, desarrollada para gestionar con eficacia los procesos de selección de recursos humanos. (Anónimo, 2008) También se desarrollan sistemas integrados de gestión, los que han llegado a alcanzar una fuerza mayor en las empresas. En Cuba también se han puesto en práctica una serie de sistemas de gestión, orientados en su mayoría a la gestión de inventarios como ASSETS, desarrollado en 1997.

Sin embargo, la gestión de la información sigue siendo necesaria en todos estos sistemas y en cualquier lugar, institución o grupo de instituciones, puesto que la información “determina el orden y el caos” (Aja Quiroga, 2002). Toda institución debe ser vista como una gran fuente de información. Es por ello que los sistemas de gestión de información no son genéricos, pudieran solo ser en alguna medida reutilizables en empresas del mismo sector y nunca en su totalidad. Teniendo en cuenta estos aspectos, la facultad decide enfrentar este proyecto desde la producción interna, enfrentándose a nuevos retos relativos a la solución óptima de la misma.

Cuando se traducen los elementos principales del negocio en términos de un sistema y se modelan los elementos correspondientes a este flujo, ¿cómo tener éxito en garantizar una visión global del proyecto entendible para los desarrolladores y que a la vez garantice la calidad esperada?

Determinar los elementos básicos o críticos de un sistema y su descomposición o agrupación, de forma que se redunde en calidad y fortaleza, es una buena práctica en cuanto a creación de software.

La arquitectura podría ser la respuesta, ya que “aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño” (Casanovas, 2004).

Todo lo anterior determinó el planteamiento del siguiente problema científico:

Problema

No existe una representación definida de la arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 que garantice mantenibilidad, eficiencia, seguridad, portabilidad y estabilidad de un producto de software capaz de realizar un manejo informatizado de los datos referidos a los recursos humanos y materiales.

Hipótesis

Si se define la arquitectura de software para el Sistema de Gestión de Información de los Recursos de la Facultad 3, que contenga los elementos estructurales del sistema, su comportamiento y la colaboración entre dichos elementos, se logrará garantizar mantenibilidad, eficiencia, seguridad, portabilidad y estabilidad del producto de software resultante.

Objetivo

Generar el artefacto correspondiente al rol de arquitecto, que permita llevar las decisiones arquitectónicamente significativas a tomar en el Sistema de Gestión de Información de los Recursos de la Facultad 3, para garantizar la calidad deseada del producto.

Objeto de estudio: Proceso de desarrollo de software.

Campo de acción: Descripción y diseño de la arquitectura.

El trabajo se desarrolló a partir de las siguientes tareas de investigación.

Tareas

1. Realizar un estudio del estado del arte en materia de arquitectura de software para resumir estado actual.

2. Procesar y evaluar la información obtenida de la investigación del tema y adoptar una posición.
3. Caracterizar la metodología de desarrollo utilizada, justificar su empleo y analizar su enfoque hacia la arquitectura de software.
4. Llevar a cabo un estudio de las herramientas a emplear en el desarrollo del Sistema y argumentar el uso de la seleccionada.
5. Describir componentes de arquitectura y determinar los que serán empleados en el desarrollo.
6. Definir propuesta de arquitectura.
7. Validar la propuesta arquitectónica.

Fueron empleados los siguientes métodos científicos.

Métodos

Métodos Teóricos

1. **Histórico-Lógico**: Para conocer la esencia de la arquitectura de software y la metodología de desarrollo, sus trayectorias históricas y tendencias actuales
2. **Analítico-Sintético**: Para desglosar la información obtenida del tema, obtener las características principales sintetizadas y arribar a conclusiones.
3. **Sistémico**: Para determinar los elementos arquitectónicos que componen el sistema y sus relaciones.

Métodos Empíricos

1. **Observación**: Para la determinación de los elementos definitorios y restrictivos del sistema, así como su evolución en el tiempo.
2. **Experimentación**: Para probar la validez de la propuesta de solución durante el ciclo de vida del proyecto.

El **resultado** esperado con el desarrollo de este trabajo es:

Una propuesta de arquitectura que permita modelar eficientemente las bases del Sistema.

Estructura

Este trabajo está estructurado en tres capítulos:

Capítulo 1: Fundamentación teórica

Definición del marco teórico de la investigación. Realización de un estudio del estado del arte relativo a la Arquitectura de Software, al rol de arquitecto y a los sistemas de gestión. También se tratan los patrones y estilos arquitectónicos, los lenguajes de descripción arquitectónica, así como la actualidad en herramientas para la representación de diseños arquitectónicos y de tecnologías para la construcción de sistemas de software.

Capítulo 2: Propuesta de arquitectura

Se estructura una propuesta de solución al problema planteado en el diseño teórico de la investigación, guiada por los elementos propuestos en el Documento descripción de arquitectura definido en la metodología RUP.

Capítulo 3: Valoración de la propuesta

Se establecen las demandas, los argumentos y las evidencias basadas en la propuesta de arquitectura descrita en el Capítulo 2, guiados por lo establecido en el método de evaluación QUASAR, en aras de validar el cumplimiento de los factores de calidad pre-establecidos y los requerimientos asociados.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

La arquitectura de software es hoy en día una práctica importante en la creación de software con calidad. Ello deviene sin duda de la fortaleza que brindan su correcta definición y aplicación a cualquier sistema informático que se plantee desarrollar. En este capítulo se aborda el marco de desempeño que rodea a la misma, sus tendencias actuales y caracterización más detallada. A la vez se pretende analizar su enfoque desde las metodologías y las herramientas existentes, determinándose aquellas a utilizar.

1.1 Metodologías de desarrollo y arquitectura

El auge acelerado del desarrollo de software ha conllevado a la necesidad irrefutable de mecanismos de organización para dicho proceso de creación. Es en este marco que surgen las metodologías para el desarrollo de software con sus disímiles características y formas de trabajo. Ellas se encargan de guiar la manera de documentar y controlar toda la información que se genera en proyectos grandes y/o que necesiten de equipos de trabajo. Muchas son las metodologías que hoy existen pero generalmente se agrupan en dos grandes grupos: las ágiles (MAs) y las robustas. Cada una presenta características muy específicas que ayudan a definir cuál usar en cada momento, siempre teniendo presente las particularidades de cada proyecto.

Según la norma 1074 del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, del inglés) toda metodología de desarrollo de software debe incluir la forma en que se va a realizar la captura de requisitos, el diseño, la implementación y prueba. Todo ello debe hacerse con el propósito de contar con un marco de trabajo bien definido que ayude a obtener productos que garanticen los requerimientos de calidad, que cumplan con las expectativas del cliente, se desarrollen en el tiempo estimado y con costos presupuestados (Cataldi, 2000).

Las metodologías ágiles son relativamente jóvenes y fueron diseñadas y pensadas para resolver problemas a una velocidad máxima como su propio nombre lo indica. La designación “ágil” en términos de desarrollo de software aparece en el 2001. Esta nueva concepción de las

metodologías centra el éxito de su puesta en práctica en el desarrollo en equipo y en la introducción del cliente en el equipo de desarrollo.

Entre las metodologías ágiles más utilizadas tenemos la Programación Extrema (XP, del inglés), ágil y especialmente diseñada para la resolución de problemas en el menor tiempo posible y con una alta probabilidad de cambio de requisitos, de ahí que exista una ausencia de énfasis en la arquitectura durante las primeras iteraciones y por tanto un déficit de métodos de diseño arquitectónico. Este elemento es considerado como uno de los aspectos negativos de XP.

En su mayoría las MAs defienden la idea de hacer sistemas que debieron estar" para ayer" y es en este punto donde la arquitectura comienza a perder valor. Otro caso de este tipo de metodologías es Método de Desarrollo de Sistemas Dinámicos (DSDM, del inglés) surgida en 1994 y que tiene definida 5 fases en el ciclo de vida, pero ninguna de ellas atribuye fuerza a la arquitectura de software, pues envuelve a grandes rasgos el diseño y de ahí pasa a la implementación de un producto ejecutable que será limado en próximas iteraciones. Existe realimentación en todas sus fases y se centra en el equipo de trabajo y el usuario como elementos base (Anónimo, 2006).

Sin embargo, existen las metodologías tradicionales o establecidas como Proceso Unificado de Rational (RUP, del inglés) que poseen una concepción distinta en cuanto al quehacer de los creadores de software, pues establecen una serie de fases e hitos que permiten una mayor documentación, generación de artefactos y definición de roles.

RUP es un proceso de desarrollo de software que junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Tiene un carácter dirigido a casos de uso, centrado en la arquitectura, iterativo e incremental (Booch, y otros, 2000).

RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura. Define en su ciclo de vida 4 fases y 9 flujos de trabajo, como se muestra en la Figura 1.

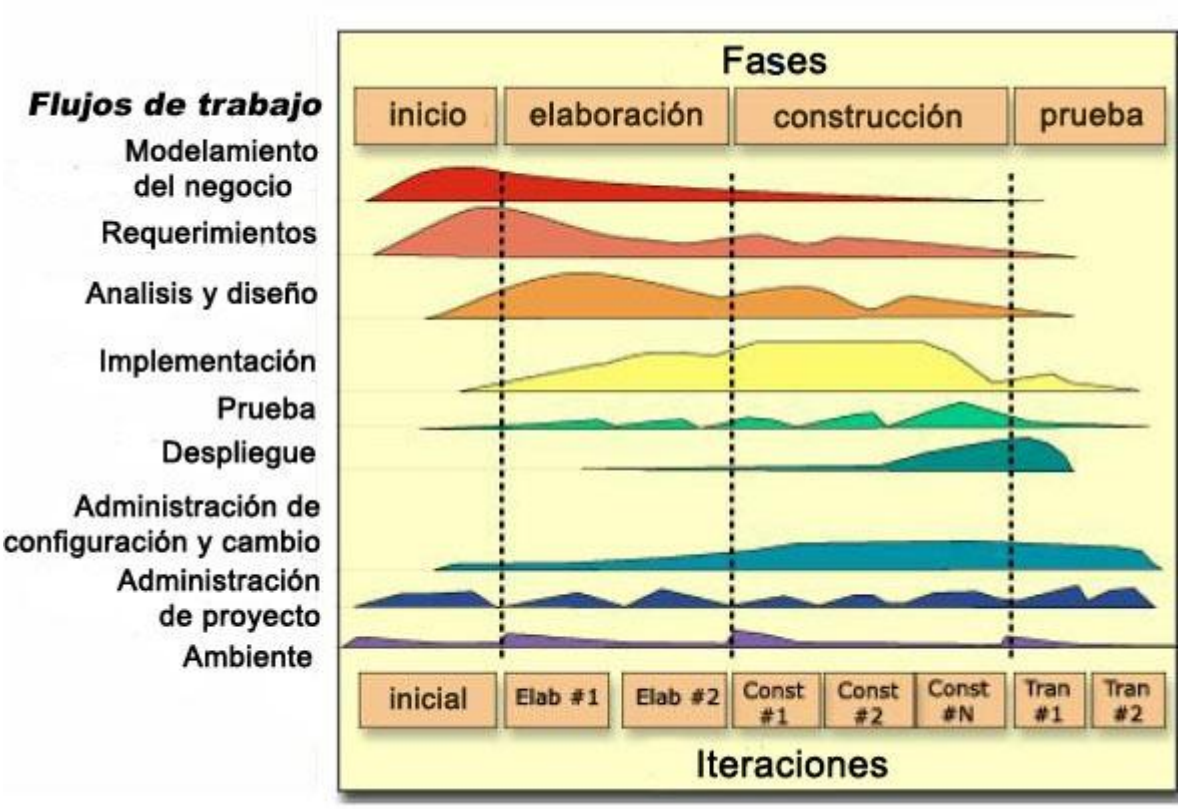


Figura 1. Fases y Flujos de trabajo según RUP (Anónimo, 2003)

RUP define una serie de roles para cada integrante del equipo de desarrollo de manera que combinan sus esfuerzos según la fase en la que se encuentren, uno de ellos es el máximo responsable de la arquitectura: el arquitecto del software.

El arquitecto del software debe ser polifacético, poseer madurez, visión y una vasta experiencia que le permita tomar decisiones rápidamente y hacer el juicio adecuado, crítico en ausencia de la información completa (Anónimo, 2003).

Seguidamente se listan las responsabilidades que establece RUP para el arquitecto de software (Anónimo, 2003):

- ✓ Priorizar los casos de uso
- ✓ Estructurar Modelo de Implementación
- ✓ Realizar análisis arquitectónico
- ✓ Construir Prueba de Concepto de la Arquitectura

- ✓ Incorporar elementos de diseño existentes
- ✓ Describir distribución
- ✓ Evaluar Viabilidad de Prueba de Concepto de la Arquitectura
- ✓ Identificar mecanismos de diseño
- ✓ Identificar elementos de diseño
- ✓ Describir la arquitectura en tiempo de ejecución

Los artefactos fundamentales a desarrollar según RUP son:

- ✓ Modelo de Análisis
- ✓ Modelo de Diseño
- ✓ Modelo de Despliegue
- ✓ Modelo de Implementación
- ✓ Arquitectura de referencia
- ✓ Documento de la Arquitectura

Valorando que el proyecto:

- ✓ Consta de 4 integrantes.
- ✓ Debe ser suficiente y adecuadamente documentado para que los futuros informáticos que deban perfeccionarlo o agrandarlo tengan una base sólida por donde guiarse.
- ✓ Consta de muchos intereses involucrados (dígase usuarios de la aplicación y clientes directos).
- ✓ No tiene inmediatez en tiempo de entrega

Y teniendo en cuenta que RUP:

- ✓ Define roles fundamentales que tienen a su cargo la generación de artefactos y documentación correctos por cada fase y flujo que tributan a un buen producto final.
- ✓ Es una metodología establecida y una de las más usadas mundialmente.
- ✓ Su aplicación ha arrojado buenos resultados en proyectos nacionales e internacionales.
- ✓ Está diseñada de forma que exista un entendimiento continuo y gradual de todos los implicados en el proyecto.

- ✓ Centra su ciclo de vida en la arquitectura.

Se determinó adoptar esta metodología para guiar el desarrollo del proyecto.

1.2 La Arquitectura de software.

La arquitectura ha estado presente en todas las etapas de la historia del hombre, incluso hoy día se levantan a cada momento maravillas arquitectónicas. Es por ello que no debería sorprender el uso de este término cuando se habla de creación de software.

Los primeros vestigios de la arquitectura de software se remontan a la década de los 60. Su concepción inicial nada tenía que ver con la madurez que ha llegado a alcanzar. En los últimos años la visión que los desarrolladores tienen de los sistemas de software cambió. Todos los sistemas, pequeños o grandes, poseen una estructura y un comportamiento que los hace diferenciables según su arquitectura. Esta nueva forma de ver el desarrollo de sistemas permite el estudio de programas ya creados e incluso la creación de algunos nuevos teniendo en cuenta las particularidades del problema y el tipo de arquitectura.

No es menos cierto que existen diversas definiciones de arquitectura de software y que no ha existido aún una común para todos los desarrolladores o arquitectos. Sin embargo, se pueden citar algunas significativas como la de Paul Clements (Clements, 1996):

"*La arquitectura de software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se le 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 o diferimiento del detalle inherente a la mayor parte de las abstracciones*."

En esta definición se hace referencia a la arquitectura como una forma de enunciar "componentes" y comportamientos pero no se detallan cuáles son dichos componentes. Se puede inferir que son componentes de diseño básico del esqueleto que compone al sistema.
La IEEE Std 1471-2000 plantea que:

"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución."

El elemento común que puede encontrarse en estas definiciones y tantas otras es que la arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

RUP plantea que se necesita una arquitectura para (Booch, y otros, 2000):

- Comprender el sistema.
- Organizar el desarrollo.
- Fomentar la reutilización.
- Hacer evolucionar el sistema.

1.2.1 Estilos arquitectónicos

Los estilos arquitectónicos en Ingeniería de software mantienen una analogía con sus homólogos en la construcción de edificios: definen y agrupan rasgos similares de construcciones previas y poseen un nombre que los identifica. "Un estilo describe entonces una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas" (Billy Reinoso, y otros, 2004).

Ellos agrupan una serie de elementos (componentes, conectores, configuraciones y restricciones) que permiten conformar una descripción con alto nivel de abstracción. Dicha

descripción puede hacerse en lenguaje natural, pero lo más recomendable es hacerlo mediante los Lenguajes de Descripción Arquitectónica (ADLs).

El tema de los estilos, desde el inicio mismo de su manejo, ha sido muy diversamente tratado, en ocasiones suele hablarse de ellos como: “tipos de arquitecturas”, “arquitecturas comunes”, “marcos de referencia arquitectónicos prototípicos”. Incluso en términos de clasificaciones de estilos, los autores no han logrado llegar a agrupaciones comunes; diversas son las formas de agruparlos según los diferentes criterios que guíen la selección y la constitución de los mismos.

La notación que se pretende adoptar en este trabajo es la dada por Carlos Billy Reinoso (Billy Reinoso, y otros, 2004), que resulta la agrupación más actual y que a su vez está respaldada por su aplicación en la Estrategia de Arquitectura de Microsoft. Se detallan solo algunos de los estilos representativos y de significación en la posterior propuesta de solución.

Estilos de Flujo de datos

- ✓Tuberías y Filtros

Estilos centrados en datos

- ✓Arquitecturas de Pizarra o repositorio

Estilos de Llamada y Retorno

- ✓Modelo – Vista – Controlador (MVC)

Este modelo suele ser visto desde dos puntos de vista: como una práctica recurrente (estilo) o muchas veces como patrón de implementación.

El patrón conocido como Modelo-Vista-Controlador (MVC) separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases (Burbeck, 1992).
Dicha separación se comporta como sigue:

Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado y a instrucciones de cambiar el estado.

Vista: Maneja la visualización de la información.

Controlador: Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

Esta forma de separar el trabajo en diferentes niveles facilita testear la aplicación independientemente de la representación visual. Además, si son necesarias modificaciones en el transcurso del trabajo pueden hacerse solo en la capa necesaria sin interferir en las demás.

Generalmente los cambios gráficos son de menor esfuerzo y más frecuentes que un cambio en requerimientos o funcionalidades, de aquí que la independencia entre estas tareas sea cómoda.

- ✓ Arquitecturas en Capas

En la actualidad existe una tendencia a modelar sistemas en términos de capas jerárquicamente distribuidas, de modo que los componentes o servicios de la aplicación se agrupen favoreciendo la alta cohesión (capa interna) y el bajo acoplamiento (capas externas).

Garlan y Shaw definen este estilo como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior (Garlan, y otros, 1994).

Entre las deficiencias más notables de este estilo se encuentra el hecho de que se hace difícil eliminar una capa sin afectar a las restantes. Puede ocurrir también que escoger los elementos a contener en cada capa sea un paso difícil, pero para ello existen, de este mismo estilo, modelaciones con diferentes cantidades de capas que definen a grandes rasgos cuál contiene qué o a quién.

La cantidad mínima de capas es dos, y en orden creciente pueden llegar a haber n niveles. En la actualidad es muy utilizado el estilo en tres capas, aunque, en dependencia de la complejidad

del sistema o de las características del ambiente de despliegue, puede aparecer una cuarta capa desdoblada de la capa de datos.

En un estilo en capas, los conectores se definen mediante los protocolos que determinan las formas de la interacción.

- ✓ Arquitecturas Orientadas a Objetos

Esta arquitectura ha sido incluida, según las diversas bibliografías que abordan el tema, en más de una agrupación de estilos, sin embargo dentro de las Llamada y Retorno ha sido más utilizada en los últimos años.

Como su nombre lo indica la definición de este estilo se acoge a las prácticas de programación OO: los componentes del estilo son los objetos y estos responden a encapsulamiento, herencia y polimorfismo. Así mismo los objetos constituyen elementos reutilizables en el ambiente de desarrollo y son responsables de preservar la integridad de su propia representación.

La principal deficiencia del estilo radica en que la modificación de un objeto implica la modificación de cada elemento (objeto, método) que lo invoca.

- ✓ Arquitecturas basadas en Componentes

Estilos de Código Móvil

- ✓ Arquitectura de Máquinas Virtuales

Estilos heterogéneos

- ✓ Sistemas de control de procesos
- ✓ Arquitecturas basadas en atributos

Estilos Peer – To – Peer

- ✓ Arquitectura basadas en Eventos

- ✓ Arquitecturas Orientadas a Servicios (SOA)
- ✓ Arquitecturas Basadas en Recursos

1.2.2 Patrones

Como dijera Christopher Alexander : "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 ni siquiera dos veces de la misma forma".

1.2.2.1 Patrones de arquitectura

Los patrones coronan una práctica de diseño que se origina antes que la arquitectura de software se distinguiera como discurso en perpetuo estado de formación y proclamara su independencia de la ingeniería en general y el modelado en particular (Billy Reinoso, y otros, 2004).

Los términos patrones y estilos arquitectónicos suelen ser utilizados indistintamente en un marco de arquitectura, sin embargo hay algunos autores que acentúan que los patrones de arquitectura se van a un alto nivel de diseño, más cercano a la implementación, menos teórico.

Según Canal, los patrones de arquitectura constituyen “esquemas de organización de los sistemas de software que determinan cuál va a ser la estructura de los mismos mediante el establecimiento de su división en subsistemas, indicando las responsabilidades de cada uno de estos subsistemas y las reglas y criterios que rigen las relaciones entre ellos” (Canal Velasco, 2000).

Existen dos corrientes básicas en términos de patrones de arquitectura:

Patrones Orientados a Arquitectura de Software (POSA): Estos patrones agrupan una vista general de distintos niveles de abstracción en concepción de patrones, entre ellos un grupo

orientado a un alto nivel de arquitectura. Los patrones aquí definidos son también tratados indistintamente como estilos.

Ejemplos:

- ✓ Layers
- ✓ Pipes and Filters
- ✓ Blackboard
- ✓ Broker
- ✓ Model-View-Controller
- ✓ Presentation-Abstraction-Control
- ✓ Microkernel
- ✓ Reflection

Patrones de Arquitectura Empresarial (PEEA): Se centran en unificar las partes de una aplicación empresarial mediante formas comunes. Se basan en experiencias recopiladas sobre formas de modelar partes de un sistema, fundamentalmente mediante capas y la forma en que estas internamente se organizan.

Ejemplos de patrones PEAA:

Capa de Dominio

- ✓ Transaction Script
- ✓ Domain Model
- ✓ Table Module

Servidor Web

- ✓ Page Controller
- ✓ Template View
- ✓ Two Step View

1.2.2.2 Patrones de diseño

Dentro del marco de la arquitectura de software se insertan lógicamente los elementos relacionados al diseño del sistema en cuestión. Cuando se habla de diseño generalmente se asocia a clases involucradas en la solución.

Si se analizan las fases de desarrollo de RUP se percibe al diseño como la parte dominante del flujo de Análisis y Diseño. A su vez, si se detallan los artefactos que se generan en el diseño de un sistema, se hace visible que están, en una gran parte, a cargo del arquitecto.
Análogamente a los patrones de arquitectura existen los patrones de diseño, que tienen a su cargo la definición de 4 elementos fundamentales, pero aplicados al modelo de clases del sistema. Dichos elementos son (Gamma, y otros, 1994):

- ✓ Nombre: describe el problema de diseño, su solución, y consecuencias en una o dos palabras.
- ✓ Problema: describe cuándo aplicar el patrón. Se explica el problema y su contexto. Puede describir estructuras de clases u objetos que son sintomáticas de un diseño inflexible. Se incluye una lista de condiciones.
- ✓ Solución: describe los elementos que forman el diseño, sus relaciones, responsabilidades y colaboraciones. No se describe un diseño particular. Un patrón es una plantilla.
- ✓ Consecuencias: resultados de aplicar el patrón.

Los patrones de diseño según su propósito se agrupan en (Luo, y otros, 2008):

- ✓ Patrones de Creación
- ✓ Patrones Estructurales
- ✓ Patrones de Comportamiento

Según su ámbito se agrupan en:

- ✓ De clase: Basados en la herencia de clases
- ✓ De objeto: Basados en la utilización dinámica de objetos

Hasta el momento la información más completa existente sobre los patrones de diseño data de 1994, el libro Design Patterns, escrito por la Banda de los Cuatro (GoF, del inglés). Según esta misma fuente (Gamma, y otros, 1994) los patrones definidos son:

Creacionales

- ✓ Abstract Factory
- ✓ Builder
- ✓ Factory Method
- ✓ Prototype
- ✓ Singleton

Estructurales

- ✓ Adapter
- ✓ Bridge
- ✓ Composite
- ✓ Decorator
- ✓ Facade
- ✓ Flyweight
- ✓ Proxy

Composición

- ✓ Chain of Responsability
- ✓ Command
- ✓ Interpreter
- ✓ Iterator
- ✓ Mediator
- ✓ Memento
- ✓ Observer
- ✓ State
- ✓ Strategy
- ✓ Template Method
- ✓ Visitor

1.2.3 Lenguajes de Descripción Arquitectónica

Hoy en día las herramientas de modelación de sistemas son variadas y suplen muchas de las necesidades de descripción y detalle de los elementos del software. Cuando se habla de

arquitectura, sin embargo, por sus propias características en cuanto a nivel de abstracción y elementos que la componen, se percibe que los más populares lenguajes y soportes no abarcan dichos terrenos.

Para suplir estas carencias se utilizan, entre otras cosas, los Lenguajes de Descripción Arquitectónica (ADLs, del inglés). Con ellos se logra modelar y razonar un problema en términos de arquitectura de software, definiendo componentes y conectores, e identificando mejor la forma en que estos se combinan (Billy Reinoso, y otros, 2004).

A decir de Carlos Billy Reinoso y Nicolás Kicillof: “Contando con un ADL, un arquitecto puede razonar sobre las propiedades del sistema con precisión, pero a un nivel de abstracción convenientemente genérico.” (Billy Reinoso, y otros, 2004).

Un ADL, como lenguaje al fin, cuenta con elementos primarios o básicos en su diseño, pero estos están acordes a la modelación de las configuraciones y conexiones que se establecen cuando de arquitectura se trata.

Garlan y Shaw, citan como elementos primarios de un ADL (Shaw, y otros, 1994):

- ✓ Componentes: Elementos a nivel de módulo.
- ✓ Operadores: Mecanismos de interconexión.
- ✓ Patrones: Composición en la que los elementos son conectados de una manera en particular.
- ✓ Clausura: Condiciones en las que una composición puede servir como un subsistema.
- ✓ Especificación: No solo de funcionalidad, sino también de rendimiento, tolerancia a fallos, etc.

Asimismo estos autores refieren que un ADL auténtico debe proveer composición, abstracción, reusabilidad, configuración, heterogeneidad y análisis.

Stéphane Faulkner y Manuel Kolp enuncian como aspectos esenciales de un ADL (Faulkner, y otros, 2003):

- ✓ Componentes: Unidades computacionales o almacenes de datos.
- ✓ Interfaces: Colección de puntos de interacción entre el mismo y el mundo exterior.
- ✓ Conectores: Usados para modelar interacciones entre componentes y reglas que regulan dichas interacciones.
- ✓ Configuraciones (o topologías): Grafos conectados de componentes y conectores que describen la estructura arquitectónica.
- ✓ Restricciones: las propiedades de un sistema o una de sus partes, cuya violación dará el sistema como inaceptable para uno o más stakeholders.
- ✓ Composiciones jerárquicas: Representan arquitecturas como componentes simples en otras.

Por su parte Vestal define un ADL como: “Lenguaje descriptivo de modelado que se focaliza en la estructura de alto nivel de la aplicación antes que en los detalles de implementación de sus módulos concretos.” (Vestal, 1993).

Aún cuando en la actualidad no se maneja una definición consensuada de ADL, se conoce que estos deben asegurar una modelación encarnada en componentes, conectores y sus configuraciones, incluyendo soporte automático para ello y teniendo en cuenta los estilos y patrones que pueden ser aplicables a cualquier solución mediante arquitectura de software.

Existen hoy en día diversos lenguajes de descripción arquitectónica que deambulan en la industria del software, algunos son considerados ADLs, otros son tomados como lenguajes que de alguna manera permiten la modelación de elementos arquitectónicos.

Algunos de ellos son:

1. Acme
   - ✓ Se destaca su capacidad de describir con facilidad sistemas “relativamente simples”.
   - ✓ Capaz de soportar el mapeo de especificaciones arquitectónicas entre diferentes ADLs.

- ✓ Soporta la definición de cuatro tipos de arquitectura: la estructura, las propiedades de interés, los tipos y estilos.

2. Aesop
   - ✓ Herramienta para construir ambientes de diseño de software basada en principios de arquitectura.
   - ✓ Ambiente de desarrollo basado en el estilo de tubería y filtros propio de UNIX.
   - ✓ Requiere manejo de toda una jerarquía de lenguajes específicos (FAM Command Language (FCL)).
   - ✓ Sólo soporta nativamente desarrollos realizados en C++.
   - ✓ No está disponible en plataforma Windows.
3. Jacal
4. CHAM
5. Darwin

1.2.4 Lenguaje Unificado 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.
Aún cuando se reconoce que UML no cuenta con un nivel de abstracción ideal para describir una arquitectura, es hoy el lenguaje más utilizado para hacerlo.

Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema (Schmuller, 2000).

El modelo más aceptado a la hora de establecer las vistas necesarias para describir una arquitectura de software es el modelo 4+1. Aunque no existe de forma explícita una vista arquitectónica, estas cinco vistas pretenden describir en su conjunto la arquitectura del sistema (Kruchten, 1995).

Este modelo define 4 vistas principales (Garzás, 2006):

- ✓ Vista Lógica: modelo de objetos, clases, entidad – relación, etc.

- ✓ Vista de Proceso: modelo de concurrencia y sincronización.
- ✓ Vista de Desarrollo: organización estática del software en su entorno de desarrollo.
- ✓ Vista Física: modelo de correspondencia software – hardware.

Y una vista más, la "+1", que se muestra y traza en cada una de las anteriores y que está formada por las necesidades funcionales que cubre el sistema, y que en ocasiones es identificada como vista de "casos de uso". A partir de ello se deduce que, según este modelo, la arquitectura es en realidad evolucionada desde escenarios.

RUP define estas vistas de la siguiente manera:

- ✓ Vista de Casos de Uso: Modela los casos de uso arquitectónicamente significativos.
- ✓ Vista Lógica: Muestra elementos de diseño arquitectónicamente significativos.
- ✓ Vista de Proceso: Muestra la estructura de procesos del sistema.
- ✓ Vista de Despliegue: Cuenta con una descripción de los nodos físicos.
- ✓ Vista de Implementación: Agrupa una colección de componentes y subsistemas de implementación.

En favor de UML se tiene que:

- ✓ Simplifica la comunicación entre desarrolladores de software.
- ✓ Sus principios principales son fáciles de entender y de aprender.
- ✓ Permite y viabiliza la comunicación entre trabajadores del proyecto y usuarios.
- ✓ Estandariza elementos de diseño de sistemas.
- ✓ Constituye el estándar más utilizado mundialmente.

1.2.5 Calidad en Arquitectura de Software

La calidad es una meta que todo proceso debe alcanzar, y en cuanto a desarrollo de software, es lógicamente también deseable.

La ISO/IEC, citada por Camacho, y otros (2004), define a la calidad de software como la totalidad de rasgos y atributos de un producto de software que le apoyan en su capacidad de satisfacer sus necesidades explícitas o implícitas.

Paralelo a todo proceso de desarrollo de software, se lleva a cabo lo que se conoce como Proceso de Aseguramiento de la Calidad (SQA, del inglés), el cual permite elaborar actividades sistemáticas que se necesitan para lograr la calidad en el producto.

La arquitectura está naturalmente ligada al aseguramiento de la calidad, puesto que la base inicial para toda arquitectura son los requisitos no funcionales, también conocidos como "atributos de calidad".

Según Bass et al. (1998), citado en (Camacho, y otros, 2004) los atributos de calidad se pueden clasificar en dos grupos (Anexos 1 y 2):

- ✓ Observables vía ejecución (disponibilidad, confidencialidad, funcionalidad, desempeño, confiabilidad, seguridad externa, seguridad interna)

- ✓ No observables vía ejecución (configurabilidad, integrabilidad, integridad, interoperabilidad, modificabilidad, mantenibilidad, portabilidad, reusabilidad, escalabilidad, capacidad de prueba)

Los atributos de calidad también se pueden clasificar en (Gómez, 2007):

- ✓ Operacionales (rendimiento, confiabilidad, disponibilidad y tolerancia a fallos)
- ✓ De desarrollo (facilidad de modificación, facilidad de reutilización y flexibilidad)

En su mayoría, los atributos de calidad se pueden organizar y descomponer de maneras diferentes, en lo que se conoce como modelos de calidad. Es interesante destacar que la organización y descomposición de los atributos de calidad han permitido el establecimiento de

modelos específicos para efectos de la evaluación de la calidad arquitectónica (Camacho, y otros, 2004).

Dada la relación tan estrecha establecida entre la arquitectura de software y los atributos de calidad que posea un sistema, se hace necesario entonces determinar el grado en que dichos atributos son satisfechos, además de prevenir posibles fallas o riesgos potenciales que puedan afectar al producto de software, evaluando la arquitectura que se proponga. Esta evaluación puede efectuarse lo mismo durante las fases tempranas de definición de la arquitectura o una vez que ya ha sido implementada.

La evaluación de arquitecturas se puede llevar a cabo mediante técnicas y métodos. Las técnicas generalmente se dividen en (Gómez, 2007) (Anexo 3):

- ✓ Cualitativas (escenarios, cuestionarios y listas de verificación)
- ✓ Cuantitativas (métricas, simulaciones, prototipos, experimentos o modelos matemáticos)

Existen diversos métodos de evaluación de arquitecturas que se enfocan en probar diversos atributos de calidad, algunos de ellos son (Camacho, y otros, 2004):

Método de Análisis de Arquitecturas de Software (SAAM, del inglés)

Centrado en la enumeración de un conjunto de escenarios que representan los cambios probables a los que estará sometido el sistema en el futuro. Como entrada principal, es necesaria alguna forma de descripción de la arquitectura a ser evaluada.

Método de Análisis de Acuerdos de Arquitectura (ATAM, del inglés)

Está llamado a ser un método de identificación de riesgos, un medio para detectar áreas de riesgos potenciales dentro de la arquitectura de un software complejo. Puede llevarse a cabo en las etapas tempranas del ciclo de vida de desarrollo. Está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM.

Estos métodos en su mayoría refieren su evaluación sobre determinadas condiciones y enfocados a evaluar la arquitectura del sistema a partir de determinados atributos. Sin embargo existe un método que se centra en la evaluación de la arquitectura del sistema antepuesto a los requerimientos arquitectónicamente significativos: Un Método para la Valoración de Calidad de Arquitecturas de un Sistema Software (QUASAR, del inglés).

Método QUASAR

Cuando se sigue este método, uno o varios equipos de validación evalúan la calidad de la arquitectura de un sistema a través de *casos de calidad*, que son desarrollados y presentados al equipo de validación por los equipos de arquitectos del sistema.

QUASAR es utilizado para probar la calidad de la arquitectura de un sistema y de sus subsistemas con vistas a (Firesmith, 2006):

- ✓ **Determinar la calidad de arquitectura del sistema**: El objetivo primario de la evaluación de QUASAR es determinar la calidad de la arquitectura del sistema por lo que se refiere al grado en que la arquitectura permite al sistema reunir sus metas de calidad asociadas y requisitos.
- ✓ **Determinar la complacencia del contrato**: Garantizar que los contratos demanden de calidad en los requerimientos como vía para que el proceso de desarrollo contemple un proceso de valoración de la calidad arquitectónica.
- ✓ **Asegurar la especificación de los requerimientos arquitectónicamente significativos.**
- ✓ **Determinar la complacencia de los requisitos**: Determinar correspondencia de la arquitectura del sistema y los subsistemas, con los requerimientos arquitectónicamente significativos, tanto solicitados por el cliente como derivados.
- ✓ **Determinar completitud y madurez de la arquitectura**: Realizar una evaluación de la integridad y madurez de la arquitectura del sistema en términos de la completitud y madurez de las arquitecturas de sus subsistemas asociados, para que los resultados puedan ser comunicados a los stakeholders.

- ✓ **Determinar consistencia e integridad de la arquitectura**: Puede usarse para determinar consistencia de arquitectura de un subsistema perteneciente a la arquitectura del sistema, o también puede ser usado como forma de identificación de falta de consistencia, haciendo cumplir la integridad arquitectónica del sistema a través de la jerarquía de agregación de arquitectura de los subsistemas.
- ✓ **Ayudar a los arquitectos a desarrollar, documentar y mejorar sus arquitecturas**: Mediante la identificación de errores y fallas arquitectónicas durante de validación.
- ✓ **Identificar defectos arquitectónicos en fases tempranas.**
- ✓ **Manejar riesgos arquitectónicos**: Lograr que los riesgos arquitectónicos sean llevados a un nivel aceptable o manejable, disminuyendo la probabilidad de aparición de defectos y aumentando la posibilidad de que estos sean rectificados en etapas tempranas del ciclo de desarrollo.
- ✓ **Proveer a los stakeholders de visibilidad de la arquitectura**.

1.3 Herramientas para la representación de diseños arquitectónicos

La modelación visual ha devenido en una útil herramienta para desglosar el diseño de aplicaciones software. Mediante el modelado visual se logra detallar los elementos que componen un sistema, sus conexiones y restricciones.

1.3.1 Herramientas Case para Diseño Orientado a Objetos

El modelado orientado a objetos ha alcanzado su punto más alto desde la aparición del Lenguaje Unificado de Modelado (UML), debido a que este constituye un estándar para la representación gráfica de la información de un sistema.

Como soporte a este lenguaje de modelado algunas herramientas han hecho su aparición en la red de redes. Las compañías que las tienen a su cargo han desarrollado más de una versión que van enfocadas, en algunos casos a mejoras respecto a versiones anteriores, y en otros casos a darles un giro a su utilidad hacia un campo más específico. Entre las más usadas actualmente se encuentran:

Rational Rose Enterprise Edition

- ✓ Herramienta propietaria.
- ✓ Perteneciente a la familia Rational Rose.
- ✓ Madura, bien establecida en el mercado.
- ✓ Incluye una Modelación Añadida de la Web que proporciona visualización, modelación y herramientas para el desarrollo de aplicaciones Web.
- ✓ Ofrece habilidades de análisis de calidad de código y generación del código, con capacidades de sincronización, así como manejo más granular y uso de modelos.

Enterprise Architect

- ✓ Interfaz de usuario intuitiva.
- ✓ Documentación flexible y comprensible.
- ✓ Ingeniería de Código Directa e Inversa.
- ✓ Plug-ins para vincular EA a Visual Studio.NET o Eclipse
- ✓ Modelado de Base de Datos.
- ✓ Soporte de esquema XML.
- ✓ Soporte en Administración de Requisitos.
- ✓ Soporta Línea Base, seguridad de usuario, prueba, mantenimiento, administración de proyecto, información de estado del sistema, control de versiones, entre otras.

Visual Paradigm

- ✓ Ofrece entorno de creación de diagramas para UML 2.0.
- ✓ Disponibilidad en múltiples plataformas.
- ✓ Disponibilidad de integrarse en los principales IDEs.
- ✓ Soporta una gama de lenguajes en la Generación de Código e Ingeniería Inversa en Java, C++, CORBA IDL, PHP, Esquema de XML, Ada y Python.
- ✓ La Generación de Código soporta C #, VB .NET, Lenguaje de Definición de Objeto (ODL), Flash Action Script, Delphi, Perl, Objetivo-C, y Ruby.

Por las anteriores características mencionadas, adicionadas a las ventajas que presenta en el diseño de sistemas y a la agilidad que proporciona en la modelación, se determinó emplear Visual Paradigm en la posterior propuesta de solución.

1.4 Sistemas de Gestión de Contenidos

La creación y administración de los sitios Web puede llegar a ser un trabajo extenso y hasta tedioso, por ello en los últimos años se han desarrollado herramientas para agilizar este proceso. Es con este propósito que se crean, y se utilizan mucho hoy en día, los Sistemas de Gestión de Contenidos (CMS, del inglés) que devienen en una potente vía para viabilizar el trabajo con el Web que antes recaía mayormente en los administradores.

Un CMS es 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 (Cuerda Garcia, y otros, 2004). También puede verse como “un sistema que separa los archivos relacionados con el contenido de un sitio web (texto, imágenes, etc.) del marco operativo que enlaza las páginas y las muestra de acuerdo al diseño” (Cadena, 2006).

Los CMS brindan un medio donde la actualización, mantenimiento y ampliación de la Web se hacen tareas sencillas y simples, todo ello con la colaboración de múltiples usuarios.

En términos de creación, gestión, publicación y presentación del contenido, presentan características que los distinguen. En cuanto a creación, la utilización de editores de texto en los que el usuario ve el resultado final mientras escribe, la edición de los documentos en XML y la importación de documentos existentes, son comunes. A su vez la documentación se almacena en una base de datos central donde también se guardan el resto de datos de la Web. Además, es posible asignar un grupo a cada área, con responsables, editores, autores y usuarios con diferentes permisos y hacer un seguimiento del estado de cada paso del ciclo de trabajo.

Un CMS posee soporte de normas internacionales de accesibilidad, compatibilidad con los diferentes navegadores disponibles en todas las plataformas y capacidad de internacionalización.

De manera general, los mayores atributos de usabilidad de los sistemas de gestión de contenidos están dados por (Cuerda Garcia, y otros, 2004):

- ✓ Inclusión de nuevas funcionalidades en la Web.
- ✓ Mantenimiento de gran cantidad de páginas
- ✓ Reutilización de objetos o componentes
- ✓ Páginas interactivas
- ✓ Cambios del aspecto de la Web
- ✓ Consistencia de la Web
- ✓ Control de acceso

1.4.1 CMS de código libre

Existen Sistemas gestores de contenidos cuyo código fuente es libre o abierto, lo que permite que pueda ser modificado una vez adquirido y que dichas modificaciones puedan ser patentadas.

Entre los CMS de código abierto que se utilizan actualmente contamos con: Joomla CMS, ASP Nuke, Plone y Drupal.

1.5 Sistemas de gestión

Los sistemas de gestión son tan antiguos como el surgimiento de la tecnología que alcanza a desarrollarlos. En la actualidad la mayor expresión de los sistemas de gestión se ha visto en la creación de diversos sistemas integrados de gestión para las empresas (ERP) que automatizan todos los procesos empresariales con gran eficiencia. Más notable aún es el auge de sistemas

de este tipo con Open Source, permitiendo mejoras graduales ante cambios frecuentes en el negocio de las empresas.

En este campo tenemos dignos ejemplares como:

Openbravo (2002)

- ✓ Destinada a empresas de pequeño y mediano tamaño.
- ✓ Arquitectura cliente/servidor Web.
- ✓ Escrita en Java.
- ✓ Se ejecuta sobre Apache y Tomcat.
- ✓ Soporte para bases de datos PostgreSQL y Oracle.

Estos sistemas poseen diversas ventajas como la reducción en los costos de mantenimiento, y la facilidad de actualización de versiones, que se derivan directamente de las ventajas Open Source.

En Cuba también se han implantado software de gestión pero mayormente orientados a ramas específicas dentro de la gestión empresarial, como los inventarios, las nóminas salariales y los recursos humanos. Como ejemplo de sistemas integrados de gestión aplicados en el país tenemos a:

Assets

- ✓ Proporciona opciones de seguridad que le permiten limitar el acceso a los diferentes procesos del sistema de acuerdo con el perfil de cada usuario.
- ✓ Aplicación cliente-servidor programada en Visual Basic 6.0 y Microsoft SQL Server 2000, utilizando adicionalmente Crystal Reports 7.0 para la generación de reportes de salidas.
- ✓ Plataforma SQL.
- ✓ Procesos implementados con inicio y final de transacciones.

Este sistema ha sido adoptado por diversas instituciones cubanas con éxito en sus funciones, entre ellas está la UCI.

En la UCI también se está desarrollando un ERP con idea de crear cada uno de sus módulos genéricos en función de las empresas cubanas y el resto de los módulos específicos aplicados a la empresa en cuestión.

Sin embargo, los requerimientos del sistema que se pretende realizar para la Facultad 3 no incluyen ningún proceso empresarial, como compras, ventas, captación de personal, sino más bien el manejo de la información relativa a esos recursos que ya hayan pasado por esos procesos a un nivel central y finalmente hayan sido asignados a la Facultad. Lo anterior expuesto justifica la no reutilización o empleo de los sistemas integrados de gestión existentes sino la generación de un sistema de gestión de información enmarcado en las necesidades justas del cliente.

1.5.1 Sistemas de Gestión de Información

La información es un ente manejable en toda empresa y por ello, un punto clave en el éxito de cualquier proceso empresarial es la gestión de la información asociada al mismo.

La Gestión de Información es el proceso mediante el cual se obtienen, despliegan o utilizan recursos básicos (económicos, físicos, humanos, materiales) para manejar información dentro y para la sociedad a la que sirve. Tiene como principales funciones (Contreras Díaz, y otros, 2007):

- ✓ Determinar necesidades internas de información, relativas a las funciones, actividades y procesos administrativos de la organización y a su satisfacción.
- ✓ Optimizar el flujo organizacional de la información y el nivel de la comunicación.
- ✓ Manejar eficientemente los recursos organizacionales de información, mejorar las inversiones sucesivas en los mismos y optimizar su aprovechamiento.
- ✓ Entrenar a los miembros de la organización en el manejo o la utilización de los recursos informacionales.
- ✓ Contribuir a modernizar u optimizar las actividades organizativas y los procesos administrativos relacionados con los mismos.

- ✓ Garantizar la calidad de los productos de la organización y asegurar su diseminación efectiva.
- ✓ Determinar las necesidades de información externa de la organización y satisfacerlas.

Cuando las necesidades de gestionar la información encuentran respaldo en las tecnologías de la información entonces se crean herramientas poderosas como lo son los sistemas (software) de gestión de la información.

Un sistema de información es un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio, y realiza cuatro actividades básicas (Peralta, 1997) (Anexo 4):

Entrada

Proceso mediante el cual el Sistema de Información toma los datos que requiere para procesar la información.

Almacenamiento

El sistema puede recordar la información guardada en la sección o proceso anterior.

Procesamiento

Es la capacidad del Sistema de Información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida.

Salida de información

La salida es la capacidad de un Sistema de Información para sacar la información procesada o bien datos de entrada al exterior.

Los Sistemas de Gestión de Información constituyen hoy, no sólo soportes de los negocios, sino, además, un instrumento de ventajas competitivas sostenibles al permitir gestionar los activos tangibles e intangibles y convertirse en una herramienta integral de gerencia (Contreras Díaz, y otros, 2007).

Existen sistemas de este tipo en el mundo como:

Sistema de Información de Ciencia y Tecnología Argentino (SICyTAR)

- ✓ Montado sobre la Plataforma Lattes, que utiliza el motor de búsqueda sobre base de datos ORACLE, que combinada con el Sistema Visualizador XML/XSL y con el Directorio de archivos XML permite búsquedas y visualización de currículos.

Los producidos por Sidart:

- ✓ Implanta, personaliza y desarrolla aplicaciones sobre Alfresco para dotar de inteligencia los repositorios documentales de su organización.
- ✓ Alfresco es la alternativa de Código Abierto para la gestión de contenido empresarial (ECM).
- ✓ A través de una adecuada implementación, sus datos serán fácilmente accesibles (con un sistema de búsquedas semejante a Google).

1.6 Conclusiones

- ✓ Existen numerosas metodologías de desarrollo de software, que se agrupan básicamente en dos grupos, sin embargo, RUP, por sus disímiles ventajas, su estructura bien definida y su adaptabilidad al proyecto, es la seleccionada para guiar su proceso de desarrollo.
- ✓ La arquitectura es el esqueleto de un sistema software, enfocada en términos de componentes y sus relaciones, y que engloba a su vez una alta reutilización, mediante el uso de patrones y estilos. Es un paso inviolable en la construcción de un software de calidad.
- ✓ Los patrones y estilos arquitectónicos son términos comúnmente inseparables. Su utilización en arquitectura es muy útil, por cuanto al nivel de abstracción que brindan al software. Sus clasificaciones y agrupaciones son diversas, pero su utilización es recomendada acorde al marco problémico.
- ✓ Para la representación arquitectónica de sistemas los lenguajes de descripción arquitectónicos han devenido en una potente vía de agrupamiento de los elementos

distintivos de una arquitectura, además de permitir la aplicación de patrones y estilos. Sin embargo, estos no han alcanzado el lugar que les corresponde por la presencia de UML, su frecuente utilización para los desarrollos de software y la representación de las 4 + 1 vistas.

- ✓ La calidad de la arquitectura es también medible y esos resultados son los que determinan la medida de cuan cerca de lo necesario y/o de lo esperado está el software que se pretende construir. Para desarrollar una evaluación de arquitectura existen diversos métodos, que se sustentan en valores de atributos de calidad.
- ✓ Existen diversas herramientas para la representación de diseños arquitectónicos, y entre ellas el Visual Paradigm, por su facilidad de manejo, su agilidad para la creación de componentes y diseños, y su soporte sobre Linux, ha sido elegido para la solución del proyecto en cuestión.
- ✓ Los sistemas de gestión de información constituyen una herramienta muy útil en todas las empresas por el ahorro manual y de tiempo que reportan, son creados en dependencia de las características de la institución que los requiera.

CAPÍTULO 2: PROPUESTA DE ARQUITECTURA

Introducción

En este capítulo se propone una arquitectura candidata para el Sistema de Gestión de Información de los Recursos de la Facultad 3. Para ello se estructura la propuesta mediante el artefacto principal que propone RUP:

- ✓ Documento descripción de la arquitectura

Los tópicos que este documento contiene, engloban en sí los aspectos fundamentales que se necesitan definir y delimitar para constituir la arquitectura.

2.1 Documento Descripción de la Arquitectura

El Documento de Arquitectura de Software proporciona una apreciación global arquitectónica comprensiva del sistema, usando varias vistas arquitectónicas para describir diferentes aspectos del sistema. Está destinado a capturar y llevar las decisiones arquitectónicamente significativas que han sido tomadas en el sistema (Anónimo, 2003).

Propósito

Este documento persigue:

- ✓ Establecer las bases arquetípicas del sistema.
- ✓ Fomentar la reutilización de tecnologías.
- ✓ Esclarecer entendimiento del sistema.

Se estructura fundamentalmente mediante la representación arquitectónica del sistema, los elementos restrictivos de su diseño y el soporte al desarrollo. Mediante este documento la comunicación entre desarrolladores, demás miembros del equipo de desarrollo y clientes debe hacerse más clara y detallada.

Alcance

El Documento Descripción de la Arquitectura abarca aspectos de todo el Sistema de Gestión de Información de los Recursos de la Facultad 3 e influye en la determinación de consideraciones arquitectónicas, de diseño e implementación en el sistema.

2.1.1 Representación arquitectónica

La representación arquitectónica establece las bases organizacionales del sistema, teniendo en cuenta los niveles de abstracción y la aplicación de arquitecturas típicas. En este punto se definen los componentes, conexiones y restricciones, así como la forma en que estos se integran para dar forma al sistema.

2.1.1.1 Estructuración mediante estilos

El Sistema de Gestión de Información de los Recursos de la Facultad 3 está destinado a ser una herramienta de control del volumen de información asociado a los recursos humanos y materiales con los que cuenta la facultad. Esta información se pretende manejar mediante un sitio Web, permitiendo:

- ✓ Organización y presentación de los datos de los recursos
- ✓ Búsqueda de recursos según diversos filtros.
- ✓ Generación de reportes relativos al estado de los recursos.
- ✓ Publicación de menús informativos útiles para la facultad.

Lo anteriormente detallado implica manejo y almacenamiento de gran volumen de información, así como la utilización de numerosas vistas de presentación de los datos.
Teniendo en cuenta también que este proyecto tiene perspectivas de crecimiento y modificación en un futuro, con la aparición de nuevos módulos, la arquitectura del sistema se estructuró mediante el estilo Capas (Layers), como se muestra en la Figura 2.

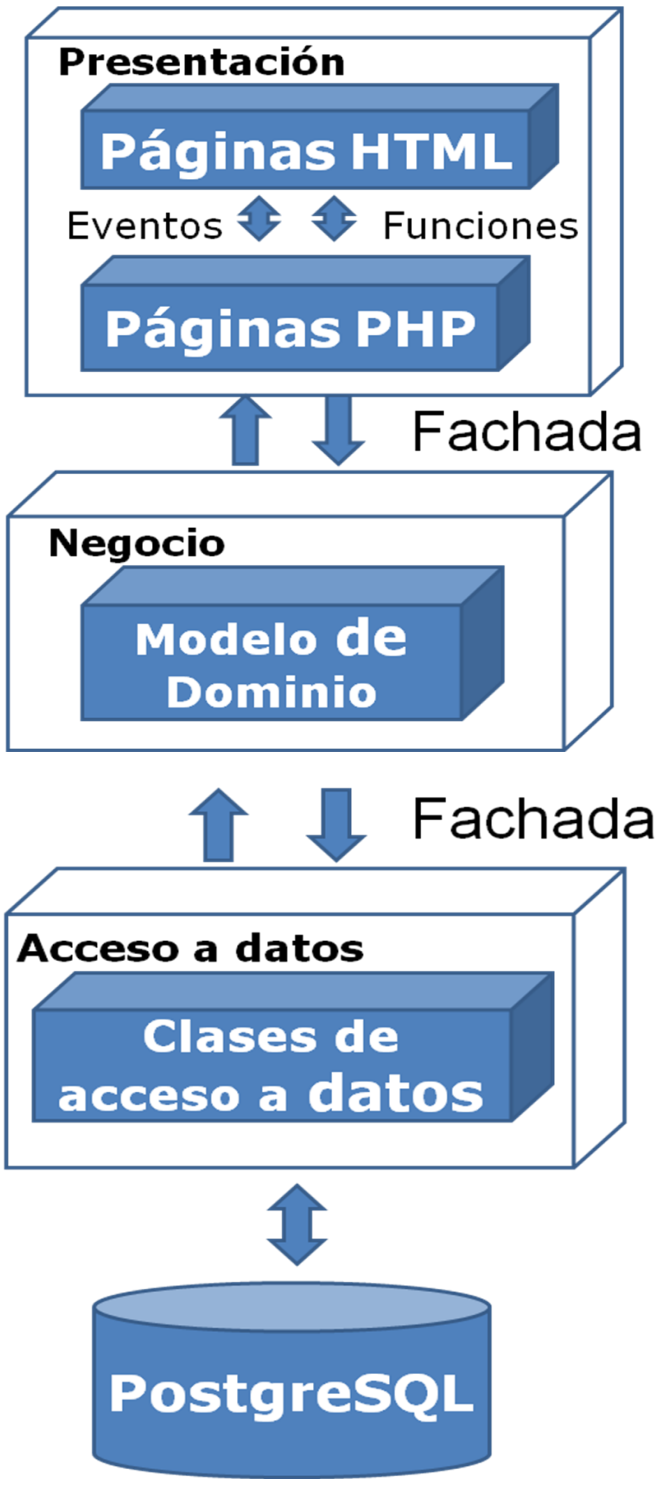


Figura 2 Sistema estructurado en tres capas

Los elementos que conforman este estilo son sus capas:

Presentación

Esta capa agrupa los elementos relativos a la presentación de la información, la cual se divide en dos subcapas fundamentales:

- ✓ Interfaces de usuario

  Esta capa contiene las páginas HTML que son el punto de contacto de los usuarios con la aplicación, permitiendo entrada y salida de datos del sistema. Estas páginas tienen aplicado formato de hojas de estilo (CSS, del inglés), y se estructuran en cuadros de texto para la presentación de la información. Además, poseen caja de autenticación de usuario y entradas de datos según se requieran. Contiene el respaldo para la presentación de los datos a los usuarios, dígase formatos de textos, fondo, imágenes y primeras validaciones de los datos. La vista de esta subcapa se muestra en la Figura 3.

Figura 3. Interfaz de usuario

- ✓ Eventos

Esta capa contiene las páginas PHP que son las encargadas de manejar y gestionar todos los eventos asociados a las interfaces que soportan. Estas páginas garantizan la lógica de presentación de los datos. Además constituyen el puente donde se realiza la captura de los datos de entrada que serán pasados a la fachada de negocio para su posterior tratamiento en el sistema (Ver Ejemplo 1).

```
<?php

require_once '../FachadaNegocio/class.FachadaNegocio.php';

$nombre=$_GET['nombre'];
$solapin=$_GET['solapin'];

RecursosHelper::getRecursosHelper()->InsertarProfesor($nombre, $solapin);
```

Ejemplo 1. Página PHP que contiene la captura de datos de entrada en presentación para pasarlos a la fachada RecursosHelper

Conexión entre Interfaces de usuarios y Eventos

Las Interfaces de usuario se relacionan con los Eventos para:

1. Realización de operaciones predefinidas por el CMS (validación de usuarios, manejo de excepciones).

2. Realización de eventos relativos a las nuevas funcionalidades propias del negocio (inserción de datos y búsquedas).

3. La propiedad Action del formulario HTML contiene la referencia a las páginas PHP que contienen la captura de datos. (Ver Ejemplo 2).

```
<form method="post" action="BuscarPersona.php"><input type="text"
      name="usuario"> <br>
<br>
<input type="submit" name="submit" value="Enviar">
```

Ejemplo 2. Llamada de "BuscarPersona.php" desde la interfaz correspondiente

Negocio

Esta capa encierra toda la lógica del negocio: está compuesta por la declaración de las clases representativas del diseño (entidades y controladora), teniendo en cuenta las reglas del negocio. Ver Figura 4.

Figura 4. Clase Profesor perteneciente al diseño de clases del sistema

El modelo de dominio contenido en esta capa contiene las clases persistentes del diseño. Estas clases en su mayoría carecerán de funcionalidades puesto que las implementaciones de esta lógica serán manejadas a través de consultas a la base de datos, exceptuando aquellas funciones que por su complejidad lógica deban ser manejadas mediante objetos.

Acceso a datos

Esta capa contiene la información requerida para la conexión a la base de datos y la transmisión de los datos necesarios en un sentido u otro (entrada o salida de la BD) según la transacción.

- ✓ Clases de acceso a datos

  Las Clases de Acceso a Datos contienen las funcionalidades de conexión a la base de datos y demás funciones que permiten la descomposición de los objetos de tipo fachada en campos de las tablas de la base de datos conjuntamente con el llamado de ejecución de las consultas. Ver Ejemplo 3.

```php
<?php
class PostgreSQL {
	private $host;
	private $port;
	private $user;
	private $pass;
	private $db;
	private $cnx;
	private $result;
	private $querycount;

	private  function __construct($host = '10.8.11.113', $port = '5432', $user = 'postgres', $pass = 'postgres', $db =
		$this->host = $host;
		$this->port = $port;
		$this->user = $user;
		$this->pass = $pass;
		$this->db = $db;
	}

	static private  $single = NULL;
	static public   function getInstancia()
	{
		if (self::$single == NULL) {
			self::$single = new PostgreSQL();
		}
		return self::$single;
	}


	/*Coneccion a la base de datos*/
	public   function conexion() {
		try {
			$this->cnx = @pg_connect ( "host=$this->host port=$this->port dbname=$this->db user=$this->user passwo
			if (! $this->cnx) {
				throw new Exception ( "No es posible conectarse con el servidor" );
			}
		} catch ( Exception $e ) {
			echo $e->getMessage ();
```

Ejemplo 3. Clase contenedora de los atributos de conexión a la base de datos, incluye la aplicación del patrón Singleton para creación de sesiones de usuario.

Conectores entre capas generales

Presentación – Negocio

La capa de presentación mediante la invocación de eventos permite el envío de los datos de entrada visual a la fachada que media entre presentación y negocio, interfaz que se encargaría de la creación de objetos de tipo de negocio. Esta fachada sería el puente

principal entre ambas capas, de manera que se logre un menor acoplamiento entre ambas capas. Estos objetos permiten la invocación de las funcionalidades de la manejadora principal de la capa de negocio y a partir de ahí las responsabilidades quedan del lado de esta capa. Ver Ejemplo 4.

```
<?php

require_once ('../Negocio/class.RHMFAC3.php');

class RecursosHelper {

	static private  $single = NULL;
	static public   function getRecursosHelper()
	{
		if (self::$single == NULL) {
			self::$single = new RecursosHelper();
		}
		return self::$single;
	}

	private function __construct() {

	}

	public  function InsertarProfesor($nombre, $solapin)
	{
		$profesor = new Profesor($nombre,$solapin);
		Recursos::InsertarProfesor($profesor);
	}
```

Ejemplo 4. Clase RecursosHelper, fachada entre capa de presentación y negocio

Cuando existen funcionalidades que devuelvan información que necesite ser presentada al usuario entonces las funciones de negocio serán las que devuelvan esa información a la fachada y esta a la Presentación. Una vez obtenida la información, Presentación se encargará de gestionar la presentación de los datos.

Negocio – Acceso a datos

Esta conexión se realiza mediante la fachada encargada de mantener aisladas las capas, de manera que se conserve el estado más puro del estilo. El primer paso de la conexión está contenido en la implementación de los métodos de la clase controladora principal, según sea

el caso, desde donde se llama a la fachada y esta recibe los datos que espera según la funcionalidad. Ver Ejemplo 5.

```
public  function InsertarProfesor(Profesor $P)
{
      PostgresHelper::getPostgresHelper()->InsertarProfesor($P);
}
```

Ejemplo 5. Llamada al método de Inserción de un Profesor de la clase PostgresHelper, fachada entre Negocio y Acceso a Datos.

En caso de haber información de retorno esta conexión seguirá activa hasta que dicha información esté de regreso en el negocio.

Acceso a datos – Base de datos

En la capa de acceso a datos están contenidos los atributos de la conexión, y la funcionalidad de generar la cadena de la conexión que será pasada a la clase que posee el método de conexión y que llevará esta a efecto. De esta manera queda establecida la conexión para la ejecución de cualquier consulta a la base de datos. Ver Ejemplo 6.

```
public function insertarProfesor($p)
{
     $sql = "select * from InsertarProfesor($p->GetNombre(),$p->GetSolapin())";
     $this->query ( $sql );
}
```

Ejemplo 6. Paso de argumentos a la consulta y llamado a ejecución de la misma

Restricciones

Teniendo en cuenta el estilo propuesto, la restricción que impone su concepción es (Reinoso, 2004):

- ✓ Cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior.

2.1.1.2 Aplicación de patrones arquitectónicos

Capa de negocio

Existen algunos patrones que permiten agrupar de una forma lógica y eficiente la lógica de dominio, entre los que se encuentra el aplicado en el Sistema de Gestión de Información de los Recursos de la Facultad 3, definido por Fowler (2002):

Nombre: Modelo de Dominio (Domain Model)

Problema: Se tienen reglas de negocio complicadas y/o cambiantes que involucran una serie de cálculos, validaciones, derivaciones. El tener sencillas transacciones para modelar las acciones no permite implementar eficientemente la complejidad del sistema.

Solución: Insertar una capa entera de objetos que modelan el área de negocio en que se está trabajando. Esta capa contendrá objetos que imitan los datos en el negocio y objetos que capturan las reglas que utiliza el negocio. Principalmente se combinan los datos y procesos para agrupar los procesos cerca de los datos con los que estos trabajan.

Elementos: Modelo de Dominio: conjunto de clases que modelan la lógica del negocio, incluyendo atributos y funcionalidades (en la medida en que se requieran).

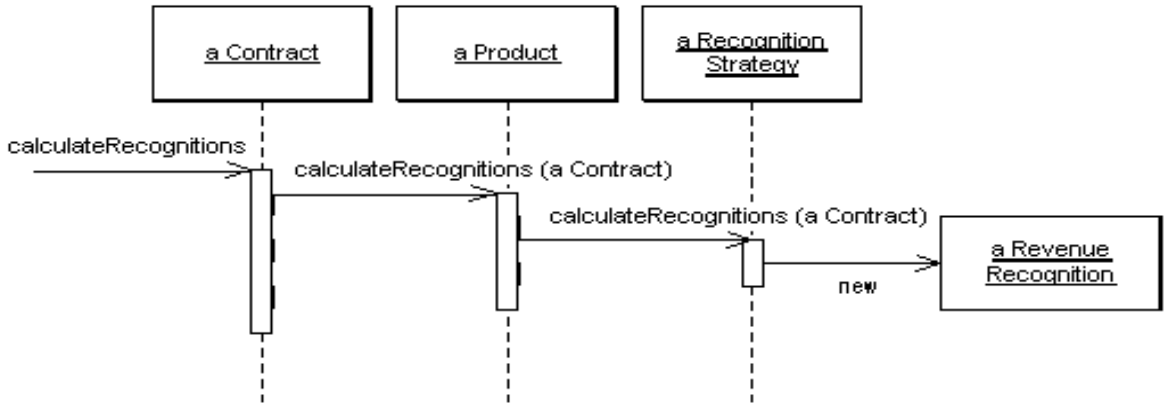


Figura 5. Diagrama de secuencia representativo del funcionamiento del patrón Domain Model (Fowler, 2002)

2.1.1.3 Frameworks de desarrollo

CodeIgniter es un framework para el desarrollo de aplicaciones Web con el uso de PHP. Su meta es preparar al desarrollador para enfrentar proyectos lo más rápido que pueda, proveyéndolo de un rico conjunto de bibliotecas para la realización de tareas comúnmente necesitadas, así como de una interfaz simple y una estructura lógica para acceder a estas bibliotecas. CodeIgniter permite enfocarse creadoramente en un proyecto, minimizando la cantidad de código necesitado para una tarea dada.

Este framework implementa una arquitectura de tipo Modelo- Vista- Controlador (MVC). La mayoría del trabajo se hace en el Controlador, cargando en las bibliotecas, recibiendo los datos del Modelo, y mostrando en la Vista. Todo está en un sentido llano y realmente puede verse cómo funcionan las cosas, en eso radica su simplicidad.

El manejo del Modelo de CodeIgniter es recto y le permite imitar una sentencia SQL con comandos sencillos, pero a su vez permite la creación de modelo de objetos, cargarlos y construir métodos que se encarguen de determinadas tareas. Esto pudiera hacerse en el modelo o bien recargarlo sobre la base de datos, nunca en el controlador para no corromper las bases del MVC.

Este framework cuenta entre sus facilidades la generación de la capa de acceso a datos, elemento que garantizará en el Sistema de Gestión de Información de los Recursos de la Facultad 3.

2.1.2 Metas y Restricciones arquitectónicas

Las metas y restricciones arquitectónicas impuestas para el sistema vienen dadas en alguna medida por los requerimientos no funcionales del sistema.

Requerimientos No Funcionales

Propiedades o cualidades que el producto debe tener.

Requerimientos de hardware

PC Cliente

- ✓ Tener mouse y teclado.
- ✓ Tarjeta de red.
- ✓ Procesador Intel Pentium III de 450 MHz(o equivalente) y versiones posteriores.
- ✓ 128 MB de RAM.

PC Servidor Web

- ✓ Tener mouse y teclado.
- ✓ Tarjeta de red.
- ✓ Procesador Intel Pentium IV de 3.0 GHz (o equivalente) y versiones posteriores.
- ✓ 512 MB de RAM.
- ✓ 256 MB de almacenamiento.

PC Servidor de base de datos

- ✓ Tener mouse y teclado.
- ✓ Tarjeta de red.
- ✓ Procesador Intel Pentium IV de 3.0 GHz (o equivalente) y versiones posteriores.
- ✓ 5 GB de almacenamiento.
- ✓ 256 MB de RAM.

Requerimientos de software

PC Cliente

- ✓ Internet Explorer 4.0 o posterior, Mozila firefox.

PC Servidor

- ✓ Sistema Operativo Microsoft Windows 95 o superior, Linux.
- ✓ Internet Explorer 4.0 o posterior, Mozila firefox, PHP 5.0 o superior, Apache 2.0 o superior.

Restricciones de diseño e implementación

- ✓ Las herramientas y tecnologías para el desarrollo de la aplicación deberán soportarse sobre sistemas operativos Linux y Windows.
- ✓ El diseño de la aplicación será orientado a objetos.

- ✓ El sistema deberá soportar la alta cohesión y el bajo acoplamiento.
- ✓ Se garantizará la posibilidad de inclusión de nuevas funcionalidades y/o módulos de diseño para futuras iteraciones.

Requerimientos de apariencia o interfaz externa

- ✓ La aplicación será Web con el menor dominio requerido en cada caso.
- ✓ La interfaz será amigable y no sobre cargada de información por página.

Requerimientos de Seguridad

- ✓ Los niveles de acceso a la información se manejarán mediante roles, unos para la aplicación (usuarios de la Web) y otro para la base de datos (solo administrador).
- ✓ La autenticación de usuarios en la aplicación se hará mediante roles uci y roles de administración, con validación de sus niveles de acceso.
- ✓ La asignación de usuarios y sus opciones sobre el sistema se garantizarán desde el módulo de Administración del sistema.
- ✓ El CMS garantizará el registro de las operaciones en el sistema (quién, cuándo, dónde, cómo).
- ✓ La disponibilidad de la información para cada usuario quedará garantizada una vez se autentique en el sistema.
- ✓ Las máquinas servidoras contarán con backups para garantizar la seguridad de las salvas de información y la mayor disponibilidad de la Web en cada momento.

Requerimientos de Usabilidad

- ✓ La aplicación Web será amigable y de fácil acceso para todo usuario del dominio uci que lo visite.
- ✓ Las habilidades requeridas para el manejo de la aplicación consistirán solo en conocimientos ofimáticos básicos (impresión, navegabilidad Web, inserción de datos).
- ✓ La modificación y/o inserción de datos relativos a los recursos requerirán además de requerimientos ofimáticos, un dominio básico del negocio.
- ✓ Existirá documentación para el uso de la aplicación orientada a clientes y usuarios finales.

Requerimientos de Eficiencia

- ✓ El sistema deberá tener un tiempo de respuesta ante peticiones de cómo máximo 2.0 segundos.
- ✓ El almacenamiento de los datos deberá ser mediante una base de datos relacional y normalizada.

Requerimientos de Estabilidad

- ✓ El servidor Web deberá mantenerse estable hasta 200 conexiones simultáneas.
- ✓ El sistema debe mantener aislado al usuario de las fallas internas y mantenerse estable ante estas.

Requerimientos de Soporte

- ✓ Se dispondrá de documentación técnica del sistema, acorde a lo establecido por las pautas de la metodología empleada.
- ✓ Deberá designarse un personal con conocimientos medios de informática y con dominio de las tecnologías utilizadas.

Estructura del equipo de desarrollo

El equipo de desarrollo está conformado por 4 miembros activos que responden a 5 roles fundamentales propuestos por la metodología RUP: Líder de proyecto, Analista, Arquitecto, Diseñador de base de datos e Implementador. El resto de los roles se asumirán por los mismos integrantes según se necesite. La jerarquía de roles se presenta en la Figura 6.

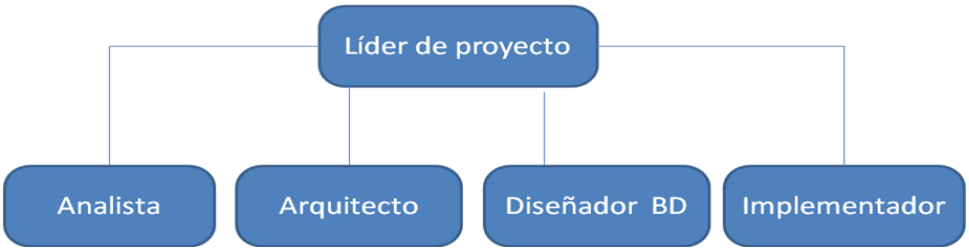


Figura 6. Estructura del equipo de proyecto

2.1.3 Tamaño y Rendimiento

La cantidad de registros que deberá estar asociado a cada tabla del Sistema de Gestión de Información de los Recursos de la Facultad 3, incluyendo un estimado de crecimiento de los datos de alrededor de un 15%, se establece en la Tabla 1:

Tabla	**Registros**
Estudiante	1200
Alumno Ayudante	300
Profesor	250
Personal Paradocente	20
Tesis	200
Evaluación de Tesis	1200
Evaluación Persona	2390
Departamento	6
Disciplina	13
Asignaturas	55
Perfil	20
Curso Optativo	200
Medio	1000
Local	10
EvaluaciónCO	36000
PersonaB	1470
MedioB	1000
ReporteMedio	4000
Persona	1470

Tabla1. Cantidad estimada de registros por tabla de la base de datos

Dado el volumen de datos relacionado anteriormente, la estimación del tamaño en memoria que estos ocuparán se determinó de la siguiente forma:

Por cada tabla se determinó el espacio que ella ocupa según sus atributos y cantidad de registros asociados, así como el espacio que se necesita para manejar sus índices asociados. Ejemplo:

Tabla Departamento

Longitud por registro	**54**
Espacio disponible por bloque(Byte)	**7180**
Registros disponibles por bloque	**132**
Total de bloques por tabla	**1**
Tamaño total por tabla(Kbyte)	**8**

Tabla 2. Valores de registros, bloques y tamaño total de Tabla Departamento

Longitud por registro	**63**
Espacio disponible por bloque(Byte)	**7250**
Registros disponibles por bloque	**115**
Total de bloques por tabla	**1**
Tamaño total por tabla(Kbyte)	**8**

Tabla 3. Valores por manejo de índices de Tabla Departamento

Para detallar más en la forma de calcular estos valores consultar Anexo 7.

Análogamente se realizan estos cálculos por cada tabla del modelo de datos, obteniendo los valores como se muestra en la Tabla 4.

Tabla	Tamaño por tabla(Kbyte)	Tamaño por índices(Kbyte)	Tamaño total(Kbyte)
Estudiante	40	32	72
Alumno Ayudante	24	8	32
Profesor	96	8	104

Personal Paradocente	8	8	16
Tesis	32	8	40
Evaluación de Tesis	64	32	96
Evaluación Persona	800	56	856
Departamento	8	8	16
Disciplina	8	8	16
Asignatura	8	8	16
Perfil	8	8	16
Curso Optativo	24	16	40
Medio	128	40	168
Local	8	8	16
ReporteMedio	1336	96	1432
MedioB	120	24	144
PersonaB	472	40	512
EvaluaciónCO	1128	800	1928
Persona	384	40	424
Total Final	**4696**	**1248**	**5944**

Tabla 4. Tamaños totales en memoria por cada tabla de la base de datos

Para la introducción de estas y otras informaciones adicionales al sistema se destinarán 10 roles, lo cual determina que para la escritura en el servidor los niveles de concurrencia serán bajos. Sin embargo por el volumen de información que se almacenará será necesario un gestor potente.

PostgreSQL en sus últimas versiones presentó como limitación de almacenamiento: solo poseer hasta 1600 columnas por tabla y cada columna no exceder 1GB de datos, lo cual garantiza que el volumen de información asociado a la lógica de negocio será posible de manejar eficientemente mediante dicho gestor.

En cuanto a los niveles de concurrencia para la lectura se hace necesario que el servidor de los datos sea rápido en términos de tiempo de respuesta y que permita un bloque de 200 solicitudes simultáneas.

En este sentido PostgreSQL posee un reconocido sistema de manejo de la concurrencia conocido como Multi Version Concurrency Control (MVCC) que permite que varios usuarios puedan leer y escribir a la misma vez sobre una misma tabla sin bloquearse ni denegarse servicios. Dado que PostgreSQL, en tanto a cantidad de conexiones es ilimitada, mediante el MVCC se crea una equivalencia a varias salvas de versiones de los datos de una misma tabla, de manera que diversos usuarios que soliciten información de esta puedan ver la última versión estable hasta tanto una nueva no haya sido correctamente salvada.

2.1.4 Vistas Arquitectónicas

La arquitectura abarca tantos elementos significativos del software que se hace difícil modelarla mediante un único escenario, por ello una forma comúnmente utilizada son vistas arquitectónicas, que permiten conjugar cada una de las partes esenciales del sistema mediante diagramas y especificaciones.

Vista de Casos de Uso

La Vista de Casos de Uso permite describir y mostrar los casos de uso priorizados desde el punto de vista de la construcción del sistema.

El Sistema de Gestión de Información de los Recursos de la Facultad 3 tiene como principal objetivo manejar la información de la que dispone la Facultad, proceso que se garantiza mediante búsquedas de los datos por campos específicos o combinación de ellos. Para lograr esto se necesita la realización de varios casos de uso con un peso priorizado desde el punto de vista arquitectónico. Parte de estos casos de uso lo representan los que manejan la gestión de usuarios. Esta representación se modela en la Figura 7.

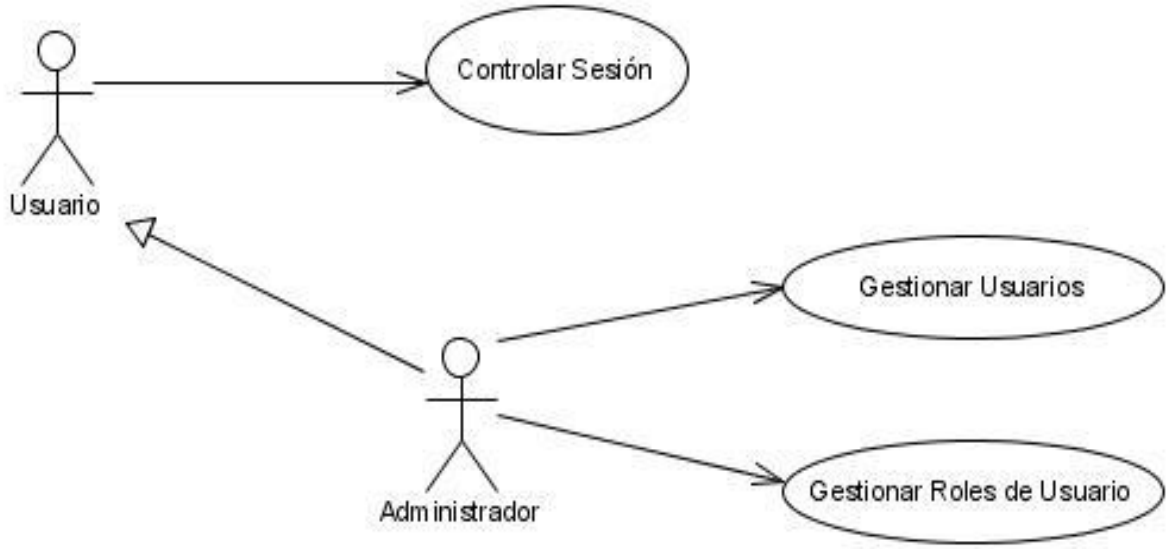


Figura 7. Vista de casos de uso para Gestión de Usuarios

Caso de uso Autenticar Usuario

Permite que todo personal que intente acceder al sitio tenga permiso correcto para hacerlo.

Caso de uso Gestionar Roles de Usuario

Permite que el administrador gestione los roles posibles a ocupar en el sistema por los usuarios, quedando establecidos de esta manera los niveles de acceso en cada caso.

Caso de Uso Gestionar Usuarios

Permite que el administrador del sistema gestione (cree, elimine o modifique) los usuarios del sistema.

A partir de que se realice la gestión de usuarios, se hace posible operar cualquier acción posterior en el sistema, con los permisos pertinentes de manera que se garantice el manejo seguro de la información.

Se hace necesario también garantizar la gestión de los datos en el sistema, dígase inserción, modificación y eliminación, puesto que a partir de que la información exista en el sistema se puede operar de la manera deseada con ella. Para la garantía de estas prioridades se tienen los casos de uso que se muestran en la Figura 8.

Caso de uso Gestionar Cursos Optativos

Permite efectuar la inserción, modificación o eliminación de los cursos optativos que ofrece la Facultad 3, así como asignar o causar baja de estudiante a un curso optativo.

Caso de uso Gestionar Información de Medios

Permite efectuar la inserción, modificación o eliminación de los datos referentes a los recursos materiales de la Facultad 3.

Caso de uso Gestionar Información de Personal

Permite efectuar la inserción, modificación o eliminación de los datos referentes al personal docente y para docente de la Facultad 3.

Caso de uso Gestionar Tesis

Permite que se pueda efectuar la inserción, modificación o eliminación de tesis en el sistema, así como asignársele autor y tutor a las mismas.

Caso de uso Gestionar Perfil

Permite que se pueda efectuar la inserción, modificación o eliminación de perfiles en el sistema, así como asignársele estudiantes.

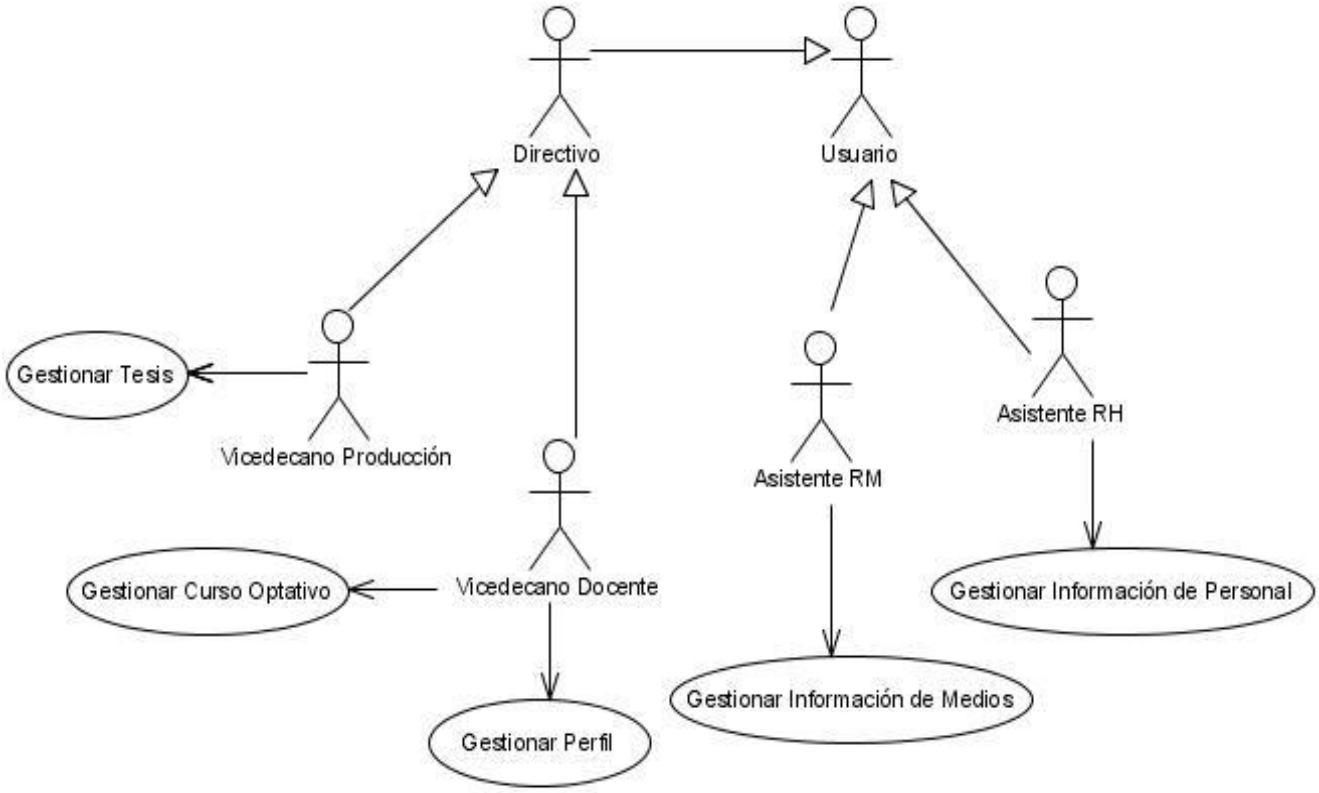


Figura 8. Vista de casos de uso para Gestión de Información

Los anteriores casos de uso del sistema son la base para que los demás procesos puedan llevarse a cabo, de ahí la prioridad que se les atribuye arquitectónicamente.

Vista Lógica

La Vista Lógica muestra la estructura del diseño del sistema, expresada en sus subsistemas, paquetes y clases. Esta vista incluye la descripción de las principales clases de diseño que intervienen en la conformación de la aplicación.

El Sistema de Gestión de Información de los Recursos de la Facultad 3 se estructuró en 4 subsistemas de diseño: dos que incluyen los módulos centrales derivados de la lógica del negocio y dos que representan los módulos a utilizar de Drupal y del framework CodeIgniter. Esta división se realizó teniendo en cuenta la (Abreu Bartomeo, y otros, 2007) posible encapsulación de elementos comunes en lógica, de forma que pueda en algún momento

reutilizarse o modificarse el sistema o alguno de sus subsistemas. La representación de dichos subsistemas se muestra en la Figura 9.

Subsistema Recursos Humanos

Contiene los paquetes de la lógica referente a los Recursos Humanos (Estudiantes y Trabajadores). Es preciso señalar que el paquete Trabajadores contiene la lógica de personal docente y profesores. Estos paquetes a su vez contienen la división por paquetes acorde a la estructuración en capas propuesta para el sistema.

Subsistema Recursos Materiales

Encapsula un único paquete contenedor de la lógica de los Medios materiales. Este paquete a su vez contiene la división por paquetes acorde a la estructuración en capas propuesta para el sistema.

Subsistema Drupal

Contiene los principales módulos utilizados del CMS en cuestión. Entre ellos se encuentran los que garantizan la autenticación de usuarios (User) y la seguridad (Workflow, Watchdog) y otros de uso básico (Node).

Subsistema CodeIgniter

Agrupa los paquetes más significativos que posee el framework. Se remarca el único a utilizar en esta iteración (Database), para la generación de la capa de acceso a datos.

También se incluyen en la Figura 9 dos paquetes que contienen las clases relativas a la ubicación física de los medios y a los reportes que de ellos se requieren.

Paquete Ubicaciones

Encapsula las clases relativas a las ubicaciones que tienen los recursos materiales y humanos una vez que son asignados a la facultad, dígase Locales, Departamentos, Especialidades y Asignaturas respectivamente. El interés que se tiene de estas entidades es solo por la relación

que guardan con los dos módulos centrales de recursos pero se agrupan en paquetes para su posterior reutilización.

Paquete Reportes

Agrupa las clases que representan los datos necesarios a almacenar una vez que un recurso es dado de baja o movido de ubicación. Estos reportes por sus características específicas son modelados como clases y por su similitud son ubicados en este paquete.

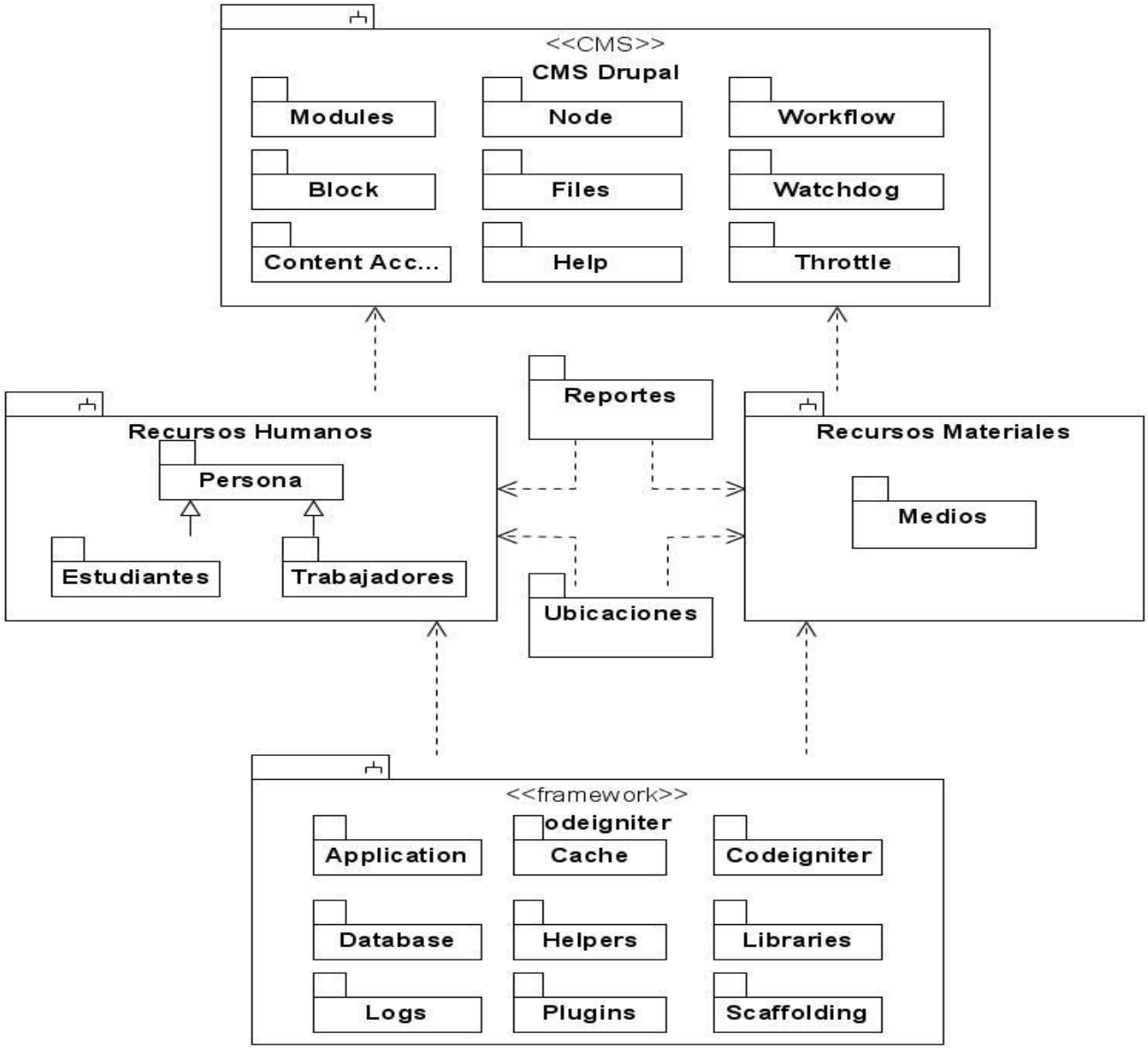


Figura 9. División del sistema en subsistemas con sus respectivos paquetes incluidos

Los paquetes contenedores de la lógica propia del negocio seguirán la estructura en capas según muestra la Tabla 5.

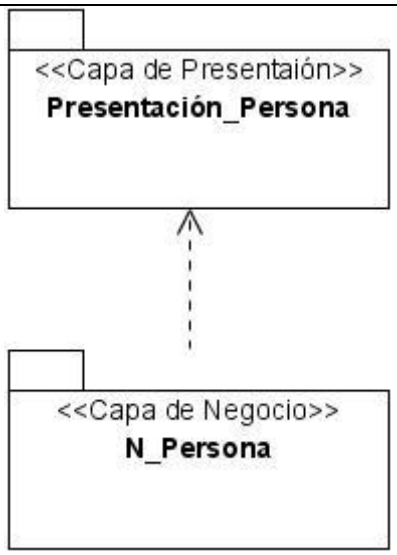
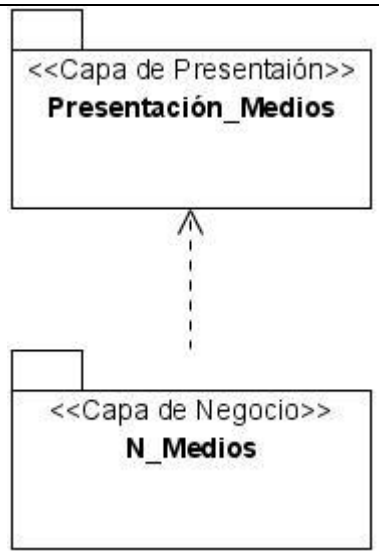
Estudiantes	Trabajadores	Persona
		
Medios	**Ubicaciones**	**Reportes**
		

Tabla 5. Estructura de paquetes para cada paquete básico de los subsistemas centrales

En el caso de los paquetes de presentación de la información y de negocio guardarán en todos los casos el mismo tipo de contenido, pero referente al campo específico sobre el que se trabaje (persona, trabajadores, medios).

Presentación_Trabajadores

Contiene los ficheros php que manejan la transición de la información relativa a los Trabajadores (profesores y personal para docente) entre la presentación de los datos y el negocio.

N_Trabajadores

Contiene las clases persistentes del diseño que agrupan los campos que se manejan de profesores y personal para docente.

La especificación se realiza análogamente para el resto de los paquetes, solo con enfoque a su campo específico.

El diagrama de clases del diseño presenta la utilización de estereotipos definidos por UML para la Web (Ver Anexo 6). Las clases más significativas del diseño se relacionan de la forma que muestra la Figura 10.

En este diseño se definen las relaciones y multiplicidades correspondientes para las clases más significativas que se involucran en la propuesta de solución. Es necesario remarcar que el peso en el diseño de cada una de estas clases es igual debido a la necesidad de manejar la información relativa a todos esos medios y los mismos no guardan relaciones de dependencia que acentúen su existencia ante otros, exceptuando la clase **Recurso**, que destaca su peso por ser la contenedora principal de la lógica del negocio, encargada a su vez de manejar las conexiones y transacciones con la fachada que media entre las capas de Negocio y Acceso a datos.

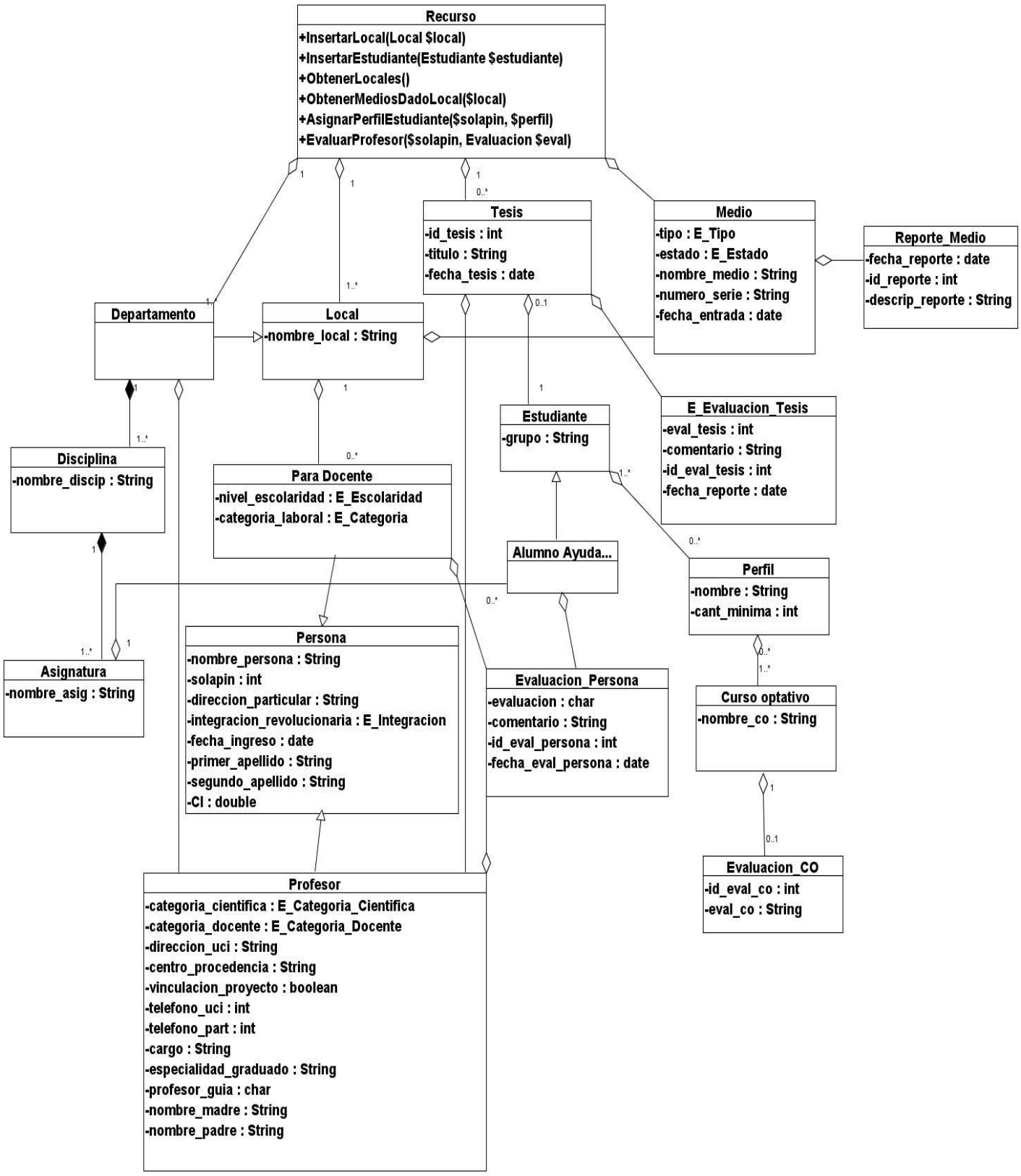


Figura 10. Diagrama de clases principales del diseño

Vista de Despliegue

La vista de despliegue permite establecer la distribución física del sistema encarnada en sus nodos, los componentes que los integran y las formas de conexión que se establecen entre ellos.

El diagrama de despliegue contiene la información de esta vista a grandes rasgos, como se muestra en la Figura 11.

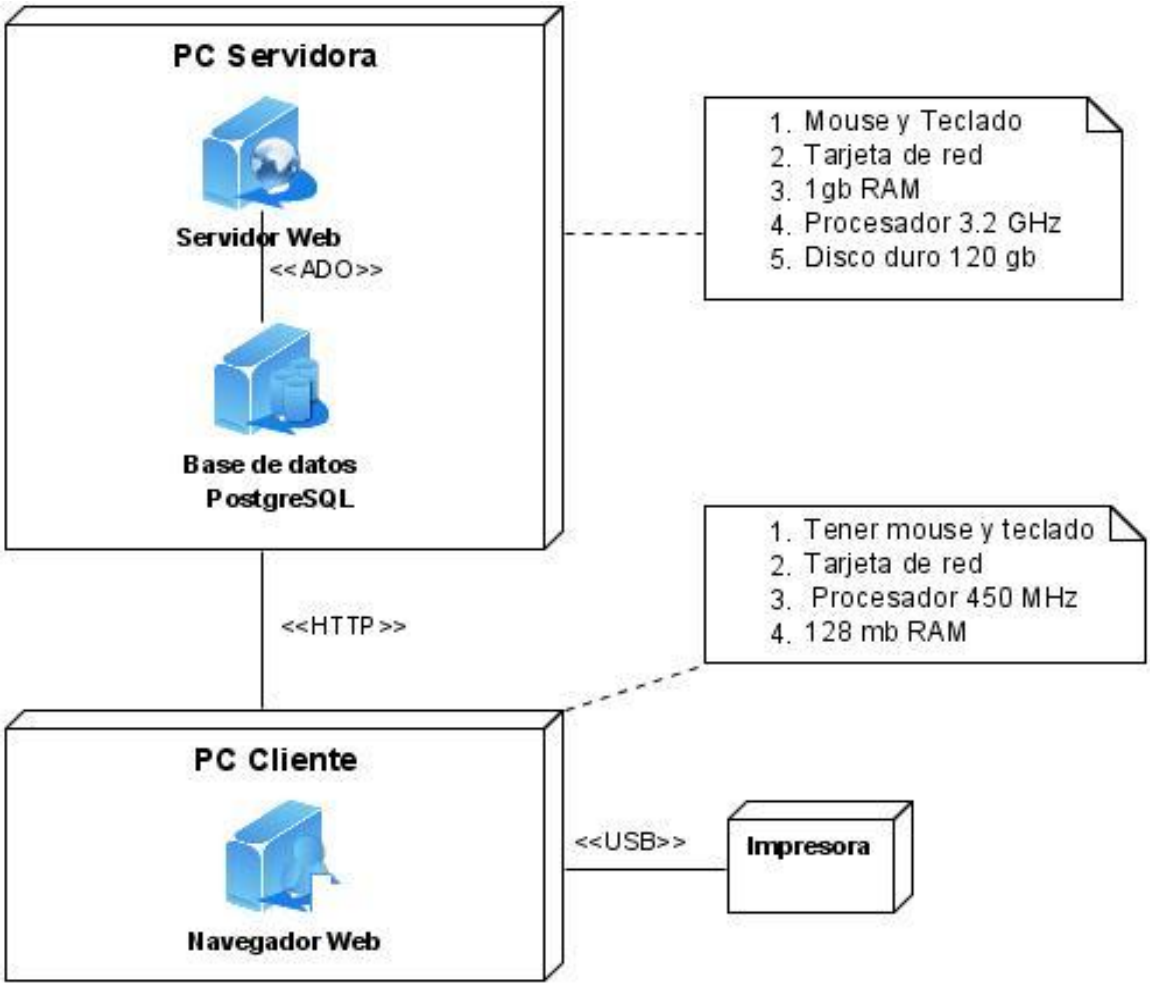


Figura 11. Modelo de despliegue

PC Servidora

Es la computadora servidora que contendrá la base de datos y donde se mantendrá corriendo el servidor Web. Esta máquina es la que ha dispuesto la facultad 3 para el despliegue y posee características de hardware que le permiten soportar ambos servidores (Ver Requisitos no Funcionales en 2.1.2 Metas y Restricciones arquitectónicas). El nodo será accesible para las máquinas clientes que requerirán del sitio, y este, para responder a los pedidos, accederá a la base de datos.

PC Cliente

Este nodo representa a las máquinas desde donde podrá conectarse cualquier usuario del sitio con los permisos correspondientes a su rol. Requiere de la conexión de dispositivos externos para la impresión de reportes. Para detallar en los requisitos básicos para la disponibilidad del sitio desde este nodo ver sección 2.1.2 Metas y Restricciones arquitectónicas.

Impresora

Dispositivo externo que permitirá la impresión de los reportes desde las máquinas clientes. La forma de conexión de estos dispositivos con la máquina cliente estará acorde a la disponibilidad que posea la facultad.

Vista de Implementación

La Vista de implementación muestra y describe el conjunto de subsistemas, paquetes y componentes de implementación en que se desdobla la aplicación. Esta vista puede realizarse con la ayuda del Modelo de Implementación, artefacto principal del flujo de implementación definido por RUP.

El Sistema de Gestión de Información de los Recursos de la Facultad 3 cuenta con una distribución física en términos de implementación representada en la Figura 12.

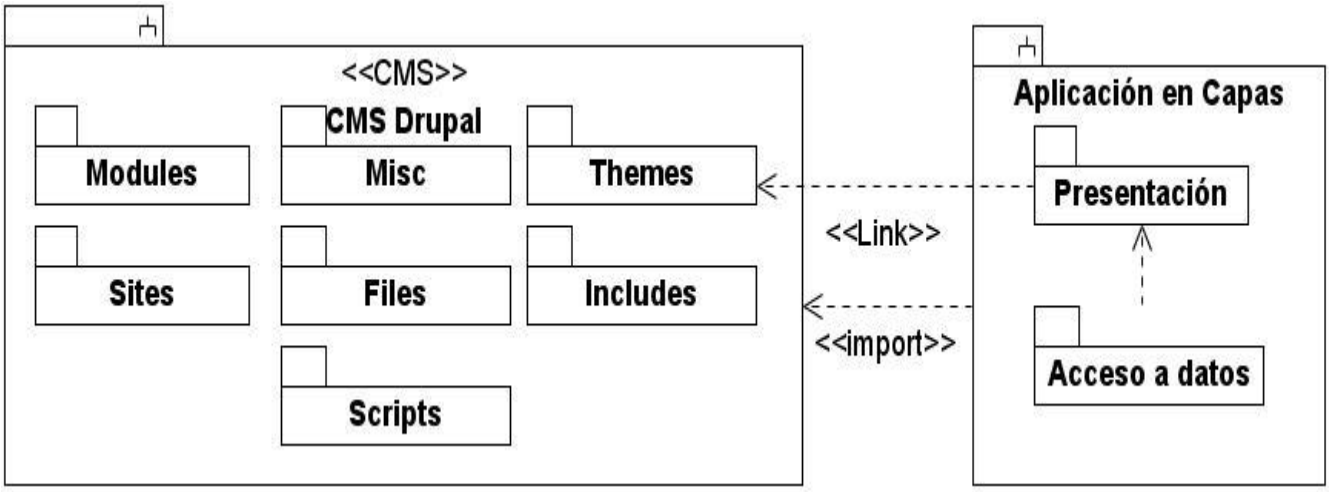


Figura 12. Vista general de subsistemas de implementación

En la figura se muestran los dos subsistemas de implementación y los componentes que los integran, así mismo la forma en que estos interactúan. La relación entre subsistemas viene dada por la relación import y entre los paquetes que interactúan se genera una relación de dependencia de tipo <<link>>. Ello se traduce en que el paquete Presentación necesita de un paquete del CMS para poder ejecutarse, puesto que en el mismo se maneja el motor de plantillas.

El CMS en este caso se modela como un subsistema de implementación, ya que muchas de las funcionalidades que garantizan el funcionamiento del sitio se encuentran en sus paquetes. Es de destacar que este subsistema también garantiza acceso a datos, pero a la base de datos propia del CMS. En el otro subsistema solo se modela el acceso a datos a la base de datos que contiene los datos persistentes de la lógica propia del negocio.

Los paquetes que intervienen en la propuesta de solución se descomponen a su vez en componentes y paquetes de la siguiente manera:

Presentación

El paquete contenido en el paquete de Presentación tributa a la estructura en capas que se definió para la capa de presentación en la estructuración mediante estilos (ver sección 2.1.1 Representación arquitectónica).

Las páginas HTML son las que se guardan en la base de datos del CMS y que se levantan una vez que se hace llamado a ellas, páginas que mediante vinculación hacen referencia a las contenidas en el paquete PHP pages.

PHP pages

Contiene las páginas PHP que validan la ejecución de acciones para las páginas HTML correspondientes, como sigue en la Figura 13.

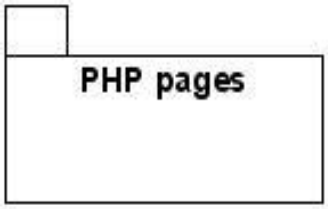


Figura 13. Paquete PHP pages contenido en paquete Presentación

Acceso a datos

El paquete de acceso a datos contiene los ficheros que garantizan la conexión a la base de datos contenedora de la información de los elementos propios del negocio, ficheros generados por el framework CodeIgniter. La relación entre estos componentes se muestra en la Figura 14.

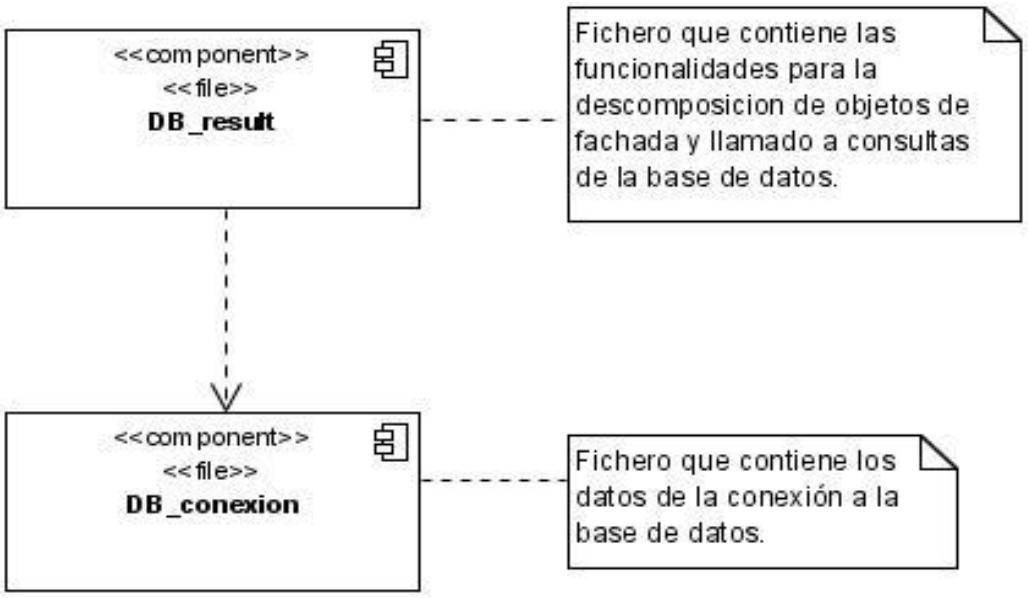


Figura 14. Diagrama de componentes para el paquete Acceso a datos

Vista de Datos

Esta vista permite mostrar los elementos persistentes arquitectónicamente significativos en el modelo de datos, teniendo en cuenta la tecnología utilizada.

En el Sistema de Gestión de Información de los Recursos de la Facultad 3 se determinó un total de 16 clases persistentes que modelan los datos que se requieren manejar en el sistema. Estas entidades son las que se agrupan en la base de datos generada por el equipo de desarrollo para gestionar la información propia del negocio, porque la información relativa a la gestión de usuarios la maneja la base de datos del CMS. El modelo físico de los datos se muestra en la Figura 15.

La lógica del negocio será resuelta mayormente mediante consultas o procedimientos almacenados, debido a que la mayor parte de las funcionalidades requeridas son búsquedas y el resto son procedimientos básicos de almacenamiento, modificación y eliminación de datos.
Para las funciones más complejas entonces se extraerán los datos y se realizará el análisis en la capa de negocio. Asimismo, por la cantidad de relaciones que se establecen entre tablas, serán necesarios disparadores (triggers) en operaciones como la de eliminación de una persona, puesto que ello implicaría eliminar la misma de todas aquellas tablas donde aparezca (dígase Departamento, Disciplina, Asignatura, etc.).

También se utilizarán las transacciones para garantizar la integridad de la información, de manera que si existen fallos en la seguridad, no quede nada a medias, como por ejemplo Adicionar un Profesor, que implicaría almacenar datos en la tabla Persona y en la tabla Profesor, de manera que no puede ocurrir que exista una persona sin su especificación como Profesor, almacenada.

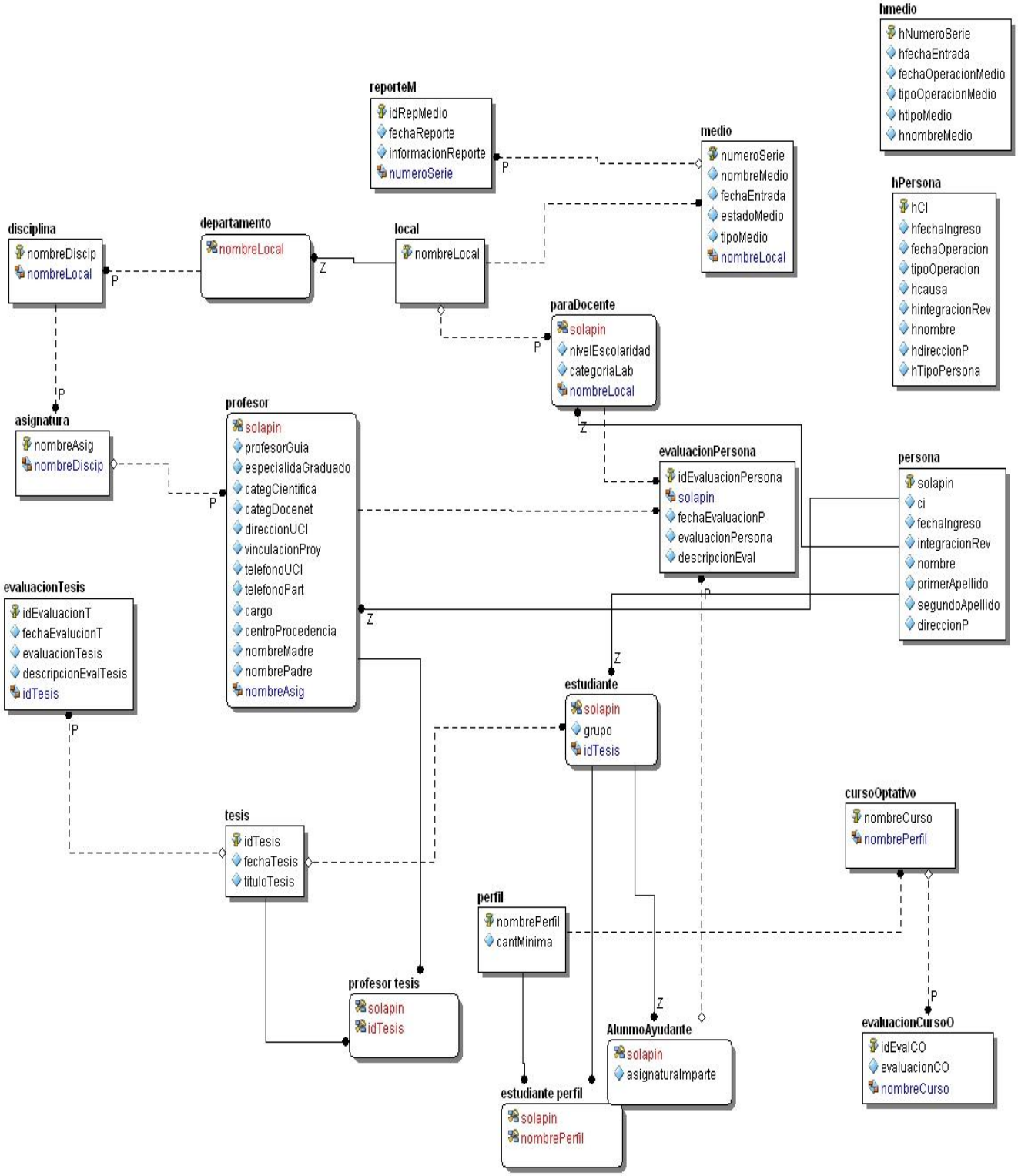


Figura 15. Modelo físico de los datos para el Sistema.

2.1.5 Soporte al desarrollo

Para la realización del Sistema de Gestión de Información de los Recursos de la Facultad 3 se analizaron los disímiles lenguajes y tecnologías capaces de tributar al mejor desarrollo del mismo. Finalmente se determinó el empleo de los que a continuación se listan. Algunos de los elementos que guiaron su selección también son nombrados.

Lenguaje de desarrollo

PHP (Hypertext Preprocessor) es un lenguaje interpretado de alto nivel y del lado del servidor. Es considerado un pre-procesador de páginas HTML, cuya utilidad fundamental está orientada a la construcción de páginas Web. Incluye la programación orientada a objetos.
PHP surge y se mantiene como un lenguaje de scripts y ello le atribuye cualidades valiosas (Dondo, 2007):

- ✓ **Velocidad**: No solo la velocidad de ejecución, la cual es importante, sino además no crear demoras en la máquina. Por esta razón no debe requerir demasiados recursos de sistema. PHP se integra muy bien junto a otro software.
- ✓ **Estabilidad:** PHP utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.
- ✓ **Seguridad**: El sistema debe poseer protecciones contra ataques. PHP provee diferentes niveles de seguridad, estos pueden ser configurados desde el archivo .ini.
- ✓ **Simplicidad:** Se les debe permitir a los programadores generar código productivamente en el menor tiempo posible. Usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente.

Hoy día grandes empresas usan PHP como herramienta Web, entre ellas Cisco, NTT DoCoMo, CMG, Vodafone, Motorola, Siemens, Ericsson, CBS, Unilever, Philips, BMC, NTT, Air Canadá, JAL, Lufthansa, OnVista, Lycos Europe y Deutsche Bank (Abreu Bartomeo, y otros, 2007).

PHP no solo ha ganado numerosos usuarios debido a sus ventajas devenidas de ser un lenguaje basado en scripts, sino por muchas otras características añadidas que lo hacen

confiable, seguro y accesible. Uno de los elementos que han propiciado el avance de PHP ha sido el ser un lenguaje “libre”, de modo que muchas personas han podido y pueden aportar funciones y modificaciones que lo hagan cada vez más cotizado y confiable.

Algunos de los atributos ventajosos propios de PHP son:

- ✓ **Multiplataforma**: como en todos los sistemas se utiliza el mismo código base, los scripts pueden ser ejecutados de manera independiente al sistema operativo.
- ✓ **Derivado del C**: asume muchas de sus sintaxis y su forma de trabajo, lo cual lo hace fácil de emprender y entender.
- ✓ **Permite conexión a diversos gestores de bases de datos**: MySQL, PostgreSQL, Oracle, entre otros.
- ✓ **Modular y extensible**: Variados módulos bases y pueden acoplarse otros de ser necesario.
- ✓ **Ejecución rápida**: completamente escrito en C, se ejecuta utilizando poca memoria.

IDE de desarrollo

Zend Studio para Eclipse es la última versión hasta el 2008 del ya conocido entorno de programación integrado Zend. Esta nueva edición de Zend Studio está orientada a desarrolladores profesionales de PHP, y se presenta combinado con las capacidades de expansión del ecosistema del proyecto Eclipse.

Esta renovada versión de Zend agrupa las cualidades de sus versiones anteriores, como son (Anónimo, 2007):

- ✓ Integra Java fácilmente en su código utilizando las características del completado de Código y especifica Jars adicionales o carpetas de Clase que pueden utilizarse para el completado de códigos.
- ✓ Integración del uso y completado de código personalizado de Zend Framework y vista de la lista de las funciones del framework desde la Visualización de Funciones PHP.
- ✓ Visualización de los eventos de Zend Platform en una ventana de lista de eventos personalizada y dedicada. Clic en cada evento para ver el detalle completo del evento en la ventana del navegador.

- ✓ Aumentar la productividad con: Soporte PHP 5 completo, Analizador de Código, carpeta de Código, completado de Código, coloreado de Sintaxis, Administrador de Proyecto, Editor de Código, Depurador de gráficos y asistentes.
- ✓ Documentación del código de forma más sencilla, aplicaciones, y proyectos con PHPDocumentor, la herramienta de documentación estándar para PHP.
- ✓ Simplificar el despliegue con la integración FTP y SFTP de forma tal que permita a los programadores en forma segura subir y descargar archivos de proyectos de modo transparente hacia y desde servidores remotos.
- ✓ Conectarse directamente con la bases de datos profesionales más utilizadas tales como IBM DB2/Cloudscape/ Derby/, MySQL, Oracle, Microsoft SQL Server, PostgreSQL y SQlite.
- ✓ Escribir y realizar consultas a servidores conectados usando el editor de consultas SQL de Zend con SQL92 y soporte de coloreado de Sintaxis.
- ✓ Visualizar las estructuras de la base de datos y administrar el contenido con el explorador SQL de Zend.
- ✓ Características de depuración avanzadas, incluyendo: condiciones límites, visualización de errores, vistas avanzadas, variables y buffer de salida.
- ✓ Depurar en forma local y remota en un medio conocido utilizando el depurador PHP más poderoso.
- ✓ Asegurar la protección máxima de ubicaciones de proyectos o en Internet con depuradores remotos seguros.
- ✓ Depurar y analizar su código directamente desde el nuevo Navegador IE con “un clic en el depurador de navegador”.

Para más detalles ver Anexo 5.

Zend Studio para Eclipse también brinda nuevas facilidades que lo hacen ser superior a las versiones anteriores (Dalcero, 2008):

- ✓ Disponer de un entorno mucho más flexible y profesional para controlar todo el ciclo de vida de un desarrollo.
- ✓ Capacidades de refactorización del código fuente: permite adecuar el comportamiento externo de una función o clase sin cambiar el funcionamiento interno.

- ✓ Disponer de un buen debugger local con la conexión a los servidores de desarrollo.
- ✓ Política de trabajo en equipo.
- ✓ Sistema de control de versiones.

Sistema Gestor de Contenidos (CMS)

Los Sistemas gestores de contenidos han encontrado un lugar en la creación de software para la Web, de calidad y con calidad. Ellos proporcionan el soporte al proceso de desarrollo, con facilidades sugerentes y ahorro en tiempo de trabajo.

Drupal se distribuye bajo la licencia GNU GPL, y por lo tanto es software libre. Permite hacer un eficiente manejo de los recursos en la Web. Es un sistema multiusuario, multiplataforma, multilenguaje, extensible y modular.

Se compone de una infraestructura base y un conjunto de módulos que ofrecen un amplio conjunto de funciones, incluyendo chat, foros, búsquedas, entre otros. Es posible añadir módulos de terceros para modificar el comportamiento de Drupal u ofrecer nuevas funciones (Abreu Bartomeo, y otros, 2007).

Entre sus características principales podemos contar (Reyero, 2005):

- ✓ **Ayuda on-line**: Un robusto sistema de ayuda online y páginas de ayuda para los módulos del 'núcleo', tanto para usuarios como para administradores.
- ✓ **Búsqueda**: Todo el contenido en Drupal es totalmente indexado en tiempo real y se puede consultar en cualquier momento.
- ✓ **Personalización**: Un robusto entorno de personalización está implementado en el núcleo de Drupal. Tanto el contenido como la presentación pueden ser individualizados de acuerdo las preferencias definidas por el usuario.
- ✓ **URLs amigables**: Drupal usa el modo de reescritura de Apache para crear URLs que son manejables por los usuarios y los motores de búsqueda.
- ✓ **Autenticación de usuarios:** Los usuarios se pueden registrar e iniciar sesión de forma local o utilizando un sistema de autenticación externo como Jabber, Blogger, LiveJournal

u otro sitio Drupal. Para su uso en una intranet, Drupal se puede integrar con un servidor LDAP.

- ✓ **Permisos basados en roles**: Los administradores de Drupal no tienen que establecer permisos para cada usuario. En lugar de eso, pueden asignar permisos a un 'rol' y agrupar los usuarios por roles.
- ✓ **Control de versiones**: El sistema de control de versiones de Drupal permite seguir y auditar totalmente las sucesivas actualizaciones del contenido: qué se ha cambiado, la hora y la fecha, quién lo ha cambiado, y más. También permite mantener comentarios sobre los sucesivos cambios o deshacer los cambios recuperando una versión anterior.
- ✓ **Enlaces permanentes**: Todo el contenido creado en Drupal tiene un enlace permanente asociado a él para que pueda ser enlazado externamente sin temor de que el enlace falle en el futuro.
- ✓ **Objetos de Contenido (Nodos)**: El contenido creado en Drupal es, funcionalmente, un objeto (Nodo). Esto permite un tratamiento uniforme de la información, como una misma cola de moderación para envíos de diferentes tipos, promocionar cualquiera de estos objetos a la página principal o permitir comentarios sobre cada objeto.
- ✓ **Agregador de noticias**: Drupal incluye un potente Agregador de Noticas para leer y publicar enlaces a noticias de otros sitios web. Incorpora un sistema de caché en la base de datos, con temporización configurable.
- ✓ **Independencia de la base de datos**: Aunque la mayor parte de las instalaciones de Drupal utilizan MySQL, existen otras opciones. Drupal incorpora una 'capa de abstracción de base de datos' que actualmente está implementada y mantenida para MySQL y PostgreSQL, aunque permite incorporar fácilmente soporte para otras bases de datos.
- ✓ **Administración vía Web**: La administración y configuración del sistema se puede realizar enteramente con un navegador y no precisa de ningún software adicional.
- ✓ **Análisis, Seguimiento y Estadísticas**: Drupal puede mostrar en las páginas web de administración informes sobre enlaces entrantes, popularidad del contenido, o de cómo los usuarios navegan por el sitio.
- ✓ **Registros e Informes**: Toda la actividad y los sucesos del sistema son capturados en un 'registro de eventos', que puede ser visualizado por un administrador.

- ✓ **Libro Colaborativo**: Esta característica es única de Drupal y permite crear un proyecto o "libro" a ser escrito y que otros usuarios contribuyan contenido. El contenido se organiza en páginas cómodamente navegables.
- ✓ **Control de congestión**: Drupal incorpora un mecanismo de control de congestión que permite habilitar y deshabilitar determinados módulos o bloques dependiendo de la carga del servidor. Este mecanismo es totalmente configurable y ajustable.
- ✓ **Sistema de Caché**: El mecanismo de caché elimina consultas a la base de datos incrementando el rendimiento y reduciendo la carga del servidor.

La arquitectura de Drupal se sustenta en su distribución por nodos, módulos y taxonomías.
Drupal introduce el concepto de nodo como sinónimo de tipos de contenido, cualquier recurso que se ingrese al sistema pasa a ser un nodo, que puede ser variable e incluir artículos, historias, posts, encuestas, imágenes, libros colaborativos, reseñas, recetas, etcétera. Este nuevo concepto nos permite estandarizar la información asignándoles las mismas características a distintos tipos de objetos y la posibilidad de tener toda la información centralizada y a la vez catalogada (Martin, y otros, 2005).

Según (Martin, y otros, 2005)., las taxonomías en Drupal consisten en una organización del contenido en categorías que se arman a través del módulo taxonomías, el cual permite generar vocabularios controlados con términos que pueden ordenarse jerárquicamente y asociarse a un tipo de nodo en particular si hiciera falta. Se puede configurar que un nodo pueda ser clasificado bajo uno o múltiples términos de un vocabulario.

Sistema Gestor de Base de Datos

Un Sistema de Información tiene como eslabón fundamental un almacén de datos, de forma que la información pueda ser correctamente accedida y preservada.

El Sistema de Gestión de Información de los Recursos de la Facultad 3 consecuente al uso del software libre, y teniendo en cuenta que necesita un potente gestor que permita estabilidad en el manejo del gran volumen de datos asociados, utilizará **PostgreSQL**.

PostgreSQL es un genuino gestor de base de datos objeto relacional libre, por lo cual cuenta con un grupo de desarrolladores y organizaciones comerciales como respaldo a su crecimiento. Cuenta con una serie de cualidades que lo hacen muy confiable y utilizado, y que en este caso también lo hacen ideal para su empleo en el Sistema de Gestión de Información de los Recursos de la Facultad 3 (Anónimo, 2003):

- ✓ Instalación ilimitada: No hay costo asociado a la licencia del software.
- ✓ Estabilidad y confiabilidad legendarias: Nunca ha presentado caídas en varios años de operación de alta actividad
- ✓ Alta concurrencia: Mediante un sistema denominado MVCC PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- ✓ Amplia variedad de tipos nativos: Soporta variedad de tipos estándares y además permite creación de nuevos tipos de datos, los que pueden ser por completo indexables.

Es un magnífico gestor de bases de datos, es multiplataforma. Se encuentra bajo la licencia BSD. Permite una fácil gestión de los usuarios y de las bases de datos que contenga el sistema. Sirve de soporte a los lenguajes más populares como PHP, C, C++, Java, Python, Ruby, etc., y al protocolo de comunicación encriptado por SSL. El número de base de datos que puede contener es ilimitado (Abreu & Colomé, 2007).

Servidor Web

El Servidor **Apache** es un servidor Web de tipo libre que se difunde en el mundo hoy día, cuya máxima es garantizar seguridad, eficiencia y extensibilidad mediante servicios HTTP. Su arquitectura modular permite que sea compatible con diversas plataformas (Unix y Windows) y lenguajes.

Una de las principales razones de emplear módulos en Apache, es que no toda instalación requiere de las mismas funcionalidades. Por lo tanto, si fueran incluidas todas las funcionalidades posibles en una versión única de Apache, esto lo haría sumamente pesado en cuanto a requerimientos de Memoria RAM y espacio en Disco Duro, por esto se opta por modularizar e incluir solo lo necesario (Anónimo, 2005).

Apache 2.0 presenta algunas mejoras con respecto de sus versiones anteriores (Anónimo, 2006):

- ✓ **Hebrado en Unix**

Esto mejora la escalabilidad para muchas aunque no para todas las configuraciones.

- ✓ **Nuevo sistema de configuración y compilación**

El sistema de configuración y compilación ha sido escrito de nuevo desde cero para basarlo en autoconf y libtool.

- ✓ **Soporte Multiprotocolo**

La nueva versión tiene la infraestructura necesaria para servir distintos protocolos.

- ✓ **Soporte mejorado para las plataformas que no son tipo Unix**

La versión 2.0 de Apache es más rápida y más estable en sistemas que no son tipo Unix. Estas plataformas tienen ahora implementada su propia API nativa, evitando las capas de emulación POSIX que provocan problemas y un bajo rendimiento.

- ✓ **Nueva interfaz de programación (API) de Apache**

Apache 2.0 hace automáticamente mucho de lo que es necesario, y la ordenación de módulos se hace ahora por hooks, lo que ofrece una mayor flexibilidad.

- ✓ **Filtros**

Los módulos de Apache pueden ahora escribirse para que se comporten como filtros que actúan sobre el flujo de contenidos tal y como salen del servidor o tal y como son recibidos por el servidor.

- ✓ **Mensajes de error en diferentes idiomas**

Los mensajes de error que se envían a los navegadores están ahora disponibles en diferentes idiomas, usando documentos SSI.

- ✓ **Configuración simplificada**

Muchas directivas que podían inducir a confusión han sido simplificadas.

- ✓ **Soporte de Unicode Nativo para Windows NT**

Apache 2.0 en Windows NT usa ahora utf-8 para la codificación de los nombres de fichero.

- ✓ **Actualización de la librería de expresiones regulares**

Apache 2.0 incluye la Librería de expresiones regulares compatibles con Perl (PCRE). Ahora, cuando se evalúan las expresiones tipo, se usa siempre la potente sintaxis de Perl 5.

Además, Apache 2.0 incluye módulos como:

- ✓ mod_ssl
- ✓ mod_dav
- ✓ mod_deflate
- ✓ mod_auth_ldap
- ✓ mod_auth_digest
- ✓ mod_charset_lite
- ✓ mod_file_cache
- ✓ mod_headers
- ✓ mod_proxy
- ✓ mod_negotiation
- ✓ mod_autoindex
- ✓ mod_include
- ✓ mod_auth_dbm

De manera general Apache cuenta entre sus virtudes el ser:

- ✓ Multiplataforma
- ✓ Seguro
- ✓ Modular
- ✓ Libre

2.2 Conclusiones

Una vez establecidas las bases arquitectónicas para el Sistema de Gestión de Información de los Recursos de la Facultad 3, se puede arribar a las siguientes conclusiones:

- ✓ El estilo tres capas se ajusta a las condiciones y restricciones impuestas para el Sistema de Gestión de Información de los Recursos de la Facultad 3, atribuyéndole una lógica organizacional que conserva la alta cohesión y el bajo acoplamiento.
- ✓ El patrón Domain Model le permite al Sistema establecer una capa lógica orientada a objetos, de manera que se beneficie de sus ventajas (dígase herencia, polimorfismo, encapsulación).

- ✓ La utilización de la base de datos como algo más que un almacén de datos presupone un empleo eficiente del gestor.
- ✓ La orientación de tecnologías y herramientas a software libre y además multiplataforma, conlleva a un abaratamiento de los costos presupuestados para el software, además de una portabilidad en condiciones reales y actuales de despliegue.

CAPÍTULO 3: VALORACIÓN DE LA PROPUESTA

Introducción

La calidad es un tema de primer orden cuando se modela arquitectura de aplicaciones, principalmente porque las decisiones arquitectónicas en un sistema influyen directamente en la calidad del software. En este sentido existen actualmente disímiles métodos para evaluar el nivel de cumplimiento de una propuesta arquitectónica relativa a los atributos de calidad asociados. En este capitulo se realiza la evaluación de la propuesta elaborada en el Capítulo 2 mediante el Método para la Evaluación de la Arquitectura de un Sistema de Software (QUASAR, del inglés).

3.1 QUASAR y la calidad arquitectónica

El desarrollo de formas sistemáticas para relacionar atributos de calidad de un sistema a su arquitectura provee una base para la toma de decisiones objetivas sobre acuerdos de diseño y permite a los ingenieros realizar predicciones razonablemente exactas sobre los atributos del sistema que son libres de prejuicios y asunciones no triviales (Camacho, y otros, 2004).

Cuando se emplea el método QUASAR, varios equipos de evaluación evalúan la calidad de la arquitectura del software por medio de casos de calidad que son presentados a estos equipos por los arquitectos del sistema.

Un caso de calidad es orientado a un atributo o factor de calidad, o incluso a un subfactor en el que un atributo se descomponga, pero también debe perseguir cumplir los requerimientos establecidos para el sistema o subsistema en cuestión (Firesmith, 2006).

Un caso de calidad de arquitectura se constituye de un grupo relacionado de (Firesmith, 2006):

Demandas

Afirmaciones del arquitecto de que el sistema o subsistema adecuadamente logra sus objetivos de calidad y cumple con sus requisitos asociados.

Argumentos

Razones claras y convincentes de los arquitectos que justifican la creencia en las demandas asociadas (es decir, razones por las cuales los evaluadores deben creer que el sistema o subsistema cumple adecuadamente sus objetivos y requisitos).Típicamente estas razones son las decisiones arquitectónicas de los arquitectos, dígase determinación apropiada de componentes arquitectónicos, patrones, etc.

Evidencias

Documentación confiable y suficiente que soporta los argumentos arquitectónicos (diagramas, modelos, documentos y ejecutables de prototipos de arquitectura).

En el Sistema de Gestión de Información de los Recursos de la Facultad 3 con la propuesta de arquitectura se pretende tributar básicamente 5 factores de calidad: mantenibilidad, eficiencia, seguridad, portabilidad y estabilidad. En vistas a plasmar el cumplimiento de estos atributos a partir de la propuesta se formularon los siguientes casos de calidad de arquitectura.

3.2 Casos de Calidad

Caso de Calidad de Mantenibilidad

Demandas

1. Demanda Factor Mantenibilidad
   - ✓ **La aplicación será fácil de mantener**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente el cumplimiento del siguiente objetivo de calidad: “El Sistema de Gestión de Información de los Recursos de la Facultad 3 podrá ser soportado una vez desplegado”.
   - ✓ **El Sistema será modificable**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente el cumplimiento del siguiente objetivo de calidad: “El Sistema de Gestión de Información de los Recursos de la Facultad 3 será fácilmente modificable en futuras ediciones”.
2. Demanda de requerimientos asociados al factor Mantenibilidad

- ✓ **El Sistema deberá estar correctamente documentado**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la inclusión del siguiente requerimiento de mantenibilidad: “Se dispondrá de documentación técnica del sistema, acorde a lo establecido por las pautas de la metodología empleada*”.*
- ✓ **El Sistema será fácilmente modificable**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la inclusión del siguiente requerimiento de mantenibilidad: “El sistema deberá soportar la alta cohesión y el bajo acoplamiento”.
- ✓ **El crecimiento del Sistema será posible**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la inclusión del siguiente requerimiento de mantenibilidad: “Se garantizará la posibilidad de inclusión de nuevas funcionalidades y/o módulos de diseño para futuras ediciones”.

Argumentos

En la garantía del cumplimiento del factor mantenibilidad y de los requisitos asociados en el Sistema de Gestión de Información de los Recursos de la Facultad 3 intervienen principalmente las siguientes determinaciones arquitectónicas:

- ✓ Estructurar el sistema en capas, cumpliendo las restricciones que impone su aplicación, garantiza una independencia entre las capas, de forma tal que la modificación en alguna de ellas afecte lo menos posible en las restantes. Esto redunda en beneficio para el mantenimiento (en caso de crecimiento o modificación) del sistema.
- ✓ La orientación de todo el diseño e implementación de la aplicación con herramientas y tecnologías libres tiene en su contra la no tenencia de una Empresa Institucionalizada que respalde el mantenimiento de la misma, sin embargo esto se compensa con la tenencia de numerosos seguidores que aportan modificaciones, funciones y nuevos atributos a cada una de las tecnologías libres empleadas en la solución. A esto último también se adiciona la posibilidad de modificar el código en beneficio específico de la facultad, contando con un informático que le de soporte a la aplicación.

Evidencias

- ✓ Estructuración en capas propuesta para el sistema, ver en sección **2.1.1 Representación arquitectónica** de este documento.
- ✓ Utilización de tecnologías y herramientas orientadas al software libre para el desarrollo de la aplicación, ver en sección **2.1.5 Soporte al desarrollo** de este documento.

Caso de Calidad de Eficiencia

El modelo de McCall según Camacho, y otros (2004), establece una relación entre atributos de calidad y sus criterios asociados. Refiriéndose a eficiencia quedaría como en la Figura 16.

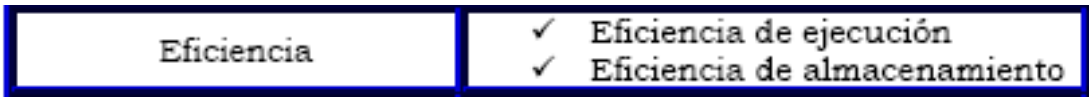


Figura 16: Eficiencia según Modelo de McCall (Camacho et al., 2004).

Demandas

1. Demanda Factor Eficiencia

- ✓ **La aplicación será eficiente en sus respuestas**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente el cumplimiento del siguiente objetivo de calidad: “El Sistema de Gestión de Información de los Recursos de la Facultad 3 deberá brindar respuestas rápidas y correctas en el menor tiempo posible”.

2. Demanda de requerimientos asociados al factor Eficiencia

- ✓ **El hardware deberá responder a las necesidades**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la inclusión del siguiente requerimiento de eficiencia: “La máquina servidora deberá poseer un mínimo de 512 MB de memoria RAM”.
- ✓ **La aplicación será rápida en sus respuestas**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la inclusión del siguiente requerimiento de eficiencia: “El sistema deberá tener un tiempo de respuesta ante peticiones de cómo máximo 2.0 segundos”.

- ✓ **Almacenamiento eficiente**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la inclusión del siguiente requerimiento de eficiencia: “El almacenamiento de los datos deberá ser mediante una base de datos relacional y normalizada”.

Argumentos

Se puede afirmar que el cumplimiento del factor eficiencia y de sus requisitos asociados en el Sistema de Gestión de Información de los Recursos de la Facultad 3 quedaría garantizado fundamentalmente mediante:

- ✓ El manejo de la caché que realiza el CMS, garantiza una velocidad mayor en cuanto a respuestas de pedidos al servidor.
- ✓ El empleo del patrón Domain Model en la capa de Negocio es eficiente al erradicar la repetición de código y la no orientación a objetos que involucran el uso del patrón Transaction Script, que pudiera en este punto resolver también la lógica de dominio.
- ✓ La utilización de una capa de Acceso a datos cuya única función es la de conectar la capa de Negocio con el almacén de los datos, garantizando un uso eficiente de la base de datos, explotando sus facilidades mas allá de simple almacén de la información.
- ✓ La máquina servidora cuenta con una RAM de 1 GB lo cual excede las condiciones mínimas para una eficiencia en tiempos de respuesta, garantizando un mejor desempeño en este factor.

Evidencias

En las primeras fases de obtención de un producto funcional se realizaron pruebas en tiempos de respuesta. Mediante el empleo de la extensión Firebug para el explorador Mozilla Firefox, que cuenta entre sus facilidades la de devolver los tiempos de carga de páginas Web (teniendo en cuenta HTML, CMS, imágenes, etc.), se determinaron los tiempos de respuesta promedio para algunas operaciones definidas para el Sistema de Gestión de Información de los Recursos de la Facultad 3.

Para las operaciones probadas los tiempos encontrados arrojaron resultados como se muestra en las Figuras 17 y 18.

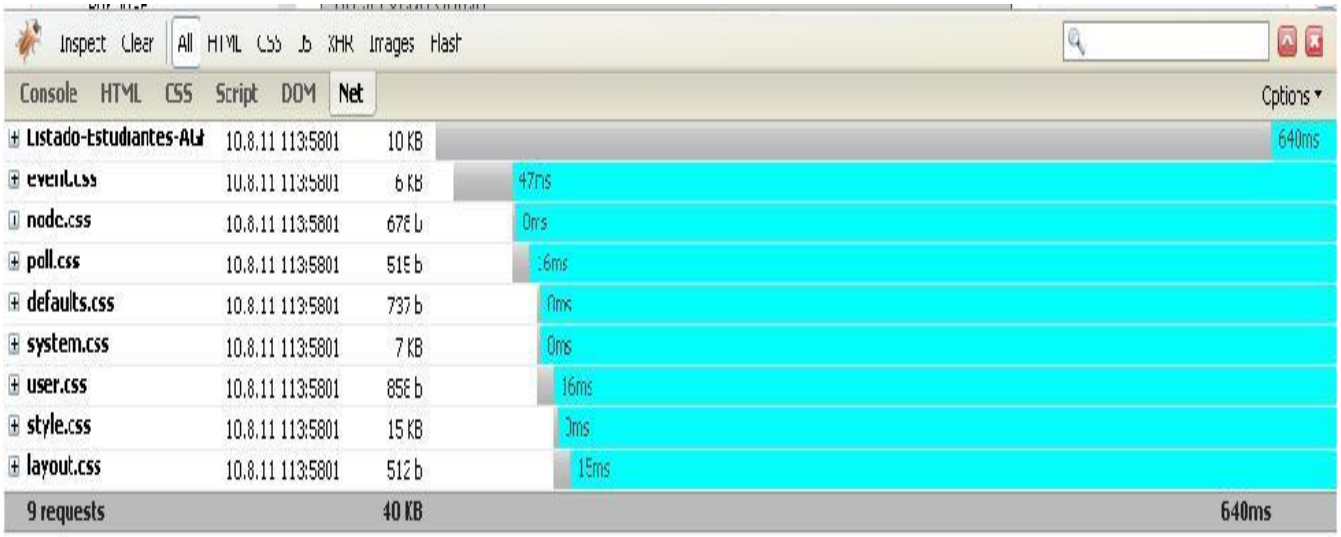


Figura 17. Tiempos de respuesta para mostrar Reporte de estudiantes conociendo año, grupo y facultad

Resultado: 9 pedidos 40 Kb 640 ms

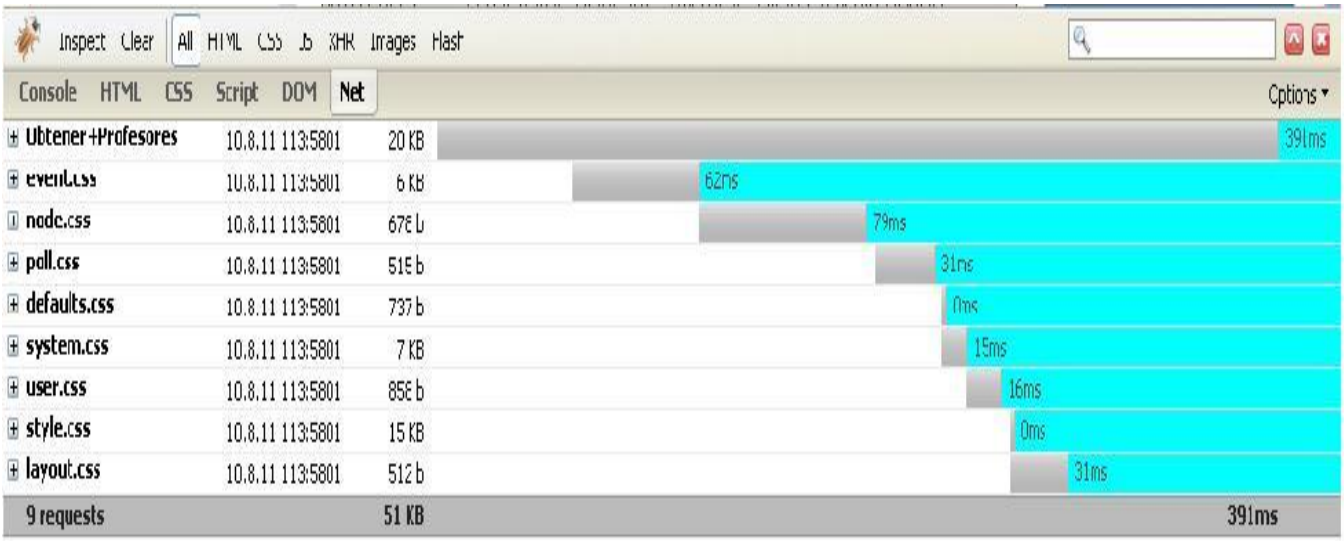


Figura 18. Tiempos de respuesta para mostrar Listado de profesores de la facultad

Resultado: 9 pedidos 51 Kb 640 ms

Las pruebas arrojan cifras por debajo de los 2 segundos máximos que se establecen para lograr una eficiencia en este aspecto. Es importante tener en consideración que gracias al manejo de la caché, si alguna de estas páginas fuera cargada nuevamente mientras el sistema esté activo, los tiempos de respuesta serían menores.

Teniendo en cuenta los resultados del cronómetro de la extensión Fasterfox para el Mozilla Firefox, en pedidos que implican consultas a la base de datos, los tiempos de respuesta obtenidos son:

Listado de estudiantes conociendo año, grupo y facultad
Tiempo total: 1.781 s. Ver Anexo 8.
Listado de profesores de la facultad
Tiempo total: 1.485 s. Ver Anexo 9

Caso de Calidad de Seguridad

La seguridad es un tema de primer orden en cualquier negocio, empresa o sistema. Se le puede definir como el conjunto de elementos que garantizan el bienestar de una entidad. Generalmente la seguridad en términos de sistemas informáticos se aborda desde tres puntos básicos: confidencialidad, integridad y disponibilidad.

Demandas

1. Demanda Factor Seguridad
   - ✓ **El Sistema será seguro para su utilización**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente el cumplimiento del siguiente objetivo de calidad: “El Sistema de Gestión de Información de los Recursos de la Facultad 3 será seguro para su utilización”.

2. Demanda subfactor Integridad
   - ✓ **La información conservará su integridad**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente el cumplimiento

del siguiente objetivo de calidad: “El Sistema de Gestión de Información de los Recursos de la Facultad 3 garantizará la integridad de la información que maneja”.

3. Demanda subfactor Disponibilidad

- ✓ **La información conservará su disponibilidad**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente el cumplimiento del siguiente objetivo de calidad: “El Sistema de Gestión de Información de los Recursos de la Facultad 3 garantizará la disponibilidad de la información que maneja”.

4. Demanda subfactor Confidencialidad

- ✓ **La información conservará su confidencialidad**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente el cumplimiento del siguiente objetivo de calidad: “El Sistema de Gestión de Información de los Recursos de la Facultad 3 garantizará la confidencialidad de la información que maneja”.

5. Demanda de requerimientos asociados al factor Seguridad

- ✓ **El acceso a los datos deberá ser controlado**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la inclusión del siguiente requerimiento de seguridad: “Los niveles de acceso a la información se manejarán mediante roles, unos para la aplicación (usuarios de la Web) y otro para la base de datos (sólo administrador)*”.*
- ✓ **El acceso a la aplicación Web deberá ser controlado**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la inclusión del siguiente requerimiento de seguridad: “La autenticación de usuarios en la aplicación se hará mediante roles UCI y roles de administración, con validación de sus niveles de acceso*”.*

6. Demanda de requerimientos asociados al subfactor Integridad

- ✓ **Las trazas de la información deberán almacenarse.** La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la

inclusión del siguiente requerimiento de seguridad: “Se garantizará el registro de las operaciones en el sistema (quién, cuándo, dónde, cómo)*”.*

7. Demanda de requerimientos asociados al subfactor Disponibilidad

- ✓ **La disponibilidad del sitio será en todo momento*.*** La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la inclusión del siguiente requerimiento de seguridad: “Las máquinas servidoras contarán con backups para garantizar la seguridad de las salvas de información y la mayor disponibilidad de la Web en cada momento”.

8. Demanda de requerimientos asociados al subfactor Confidencialidad

- ✓ **La información necesaria por cada rol estará disponible*.*** La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la inclusión del siguiente requerimiento de seguridad: “La disponibilidad de la información para cada usuario quedará garantizada una vez se autentique en el sistema”.

Argumentos

En aras de garantizar el cumplimiento del factor seguridad y de los requerimientos asociados del Sistema de Gestión de Información de los Recursos de la Facultad 3 se adoptaron los siguientes lineamientos:

- ✓ El módulo de Administración de Drupal permitirá manejar la gestión de usuarios y roles, de manera que se garanticen los correctos accesos a la información en la aplicación.
- ✓ La utilización del módulo Workflow de Drupal permitirá establecer flujos de trabajo, de manera que puedan establecerse permisos para actualización y eliminación de la información según roles de acceso, garantizando en alguna medida la integridad de la información.
- ✓ El empleo del módulo Statistics de Drupal permitirá mantener una traza de acciones sobre la aplicación Web (quién, cómo, cuándo, qué), de manera que pueda determinarse la integridad referencial de la información.

- ✓ La utilización de los backups y las transacciones en la base de datos que garanticen la recuperación e integridad de la información ante fallas internas o externas.

Caso de Calidad de Portabilidad

Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos (Camacho, y otros, 2004).

Demandas

1. Demanda Factor Portabilidad

- ✓ **El Sistema será portable**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente el cumplimiento del siguiente objetivo de calidad: “El Sistema de Gestión de Información de los Recursos de la Facultad 3 será posible de migrar a otros ambientes”.

2. Demanda de requerimientos asociados al factor Portabilidad

- ✓ **Tecnologías multiplataforma**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la inclusión del siguiente requerimiento de portabilidad: “Las herramientas y tecnologías para el desarrollo de la aplicación deberán soportarse sobre sistemas operativos Linux y Windows”.

Argumentos

En el de Sistema de Gestión de Información de los Recursos de la Facultad 3 el cumplimiento del factor portabilidad y de los requerimientos asociados se ve garantizado mediante:

- ✓ La utilización del CMS Drupal y del gestor de base de datos PostgreSQL que cuentan entre sus características el ser multiplataforma.
- ✓ El empleo de PHP como lenguaje para la creación de la Web puesto que al ser un lenguaje de scripts puede ser ejecutado independientemente del sistema operativo.
- ✓ La utilización del Servidor Web Apache que se soporta sobre varias plataformas (Unix y Windows).

Evidencias

- ✓ Utilización de tecnologías y herramientas multiplataforma para el desarrollo y puesta en marcha de la aplicación, ver en sección **2.1.5 Soporte al desarrollo** de este documento.
- ✓ Ver Vista de Despliegue en la sección **2.1.4 Vistas arquitectónicas** de este documento.

Caso de Calidad de Estabilidad

La estabilidad de un sistema software es vista como la posibilidad de mantener la disponibilidad de su servicio ante fallos.

Demandas

1. Demanda Factor Estabilidad

- ✓ **El Sistema será estable**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente el cumplimiento del siguiente objetivo de calidad: “El Sistema de Gestión de Información de los Recursos de la Facultad 3 será posible de mantenerse estable ante cambios o errores”.

2. Demanda de requerimientos asociados al factor Estabilidad

- ✓ **Estabilidad Servidor Web**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la inclusión del siguiente requerimiento de estabilidad: “El servidor Web deberá mantenerse estable hasta 200 conexiones simultáneas”.
- ✓ **Estabilidad ante fallos**. La arquitectura del Sistema de Gestión de Información de los Recursos de la Facultad 3 soporta adecuadamente la inclusión del siguiente requerimiento de estabilidad: “El sistema debe mantener aislado al usuario de las fallas internas y mantenerse estable ante estas”.

Argumentos

El Sistema de Gestión de Información de los Recursos de la Facultad 3 ve garantizado el cumplimiento del factor estabilidad y de sus requerimientos asociados mediante:

- ✓ La utilización de PostgreSQL como gestor de base de datos que posee un Control de Concurrencia Multi Versión (MVCC), lo cual posibilita atender varias peticiones de lectura

y escritura sobre una misma tabla sin que se desestabilice el sistema ni se bloqueen o denieguen las acciones de ningún usuario.

- ✓ El tratamiento de errores o excepciones del sistema, de manera que no exista posibilidad de que el cliente vea afectado su servicio debido a fallas internas.

Evidencias

Por realizar pruebas de carga y rendimiento una vez se despliegue el software en el hardware final.

3.4 Conclusiones

- ✓ La velocidad del Sistema de Gestión de Información de los Recursos de la Facultad 3 en tiempos de respuesta queda establecida para tiempos menores a 2 segundos.
- ✓ El Sistema de Gestión de Información de los Recursos de la Facultad 3 se soporta en ambientes Linux y Windows.
- ✓ El Sistema de Gestión de Información de los Recursos de la Facultad 3 es mantenible para las condiciones de estructuración arquitectónica establecidas en la propuesta.

CONCLUSIONES

- ✓ La determinación de la arquitectura provee al sistema de una representación de su estructura en términos de componentes, conexiones y restricciones, de manera que puede determinarse la calidad del producto y los posibles riesgos a enfrentar en su proceso de creación.
- ✓ Se le dio cumplimiento al objetivo del trabajo mediante la generación del Documento Descripción de Arquitectura, artefacto fundamental establecido por RUP.
- ✓ El estilo tres capas se ajusta a las condiciones y restricciones impuestas para el Sistema, atribuyéndole una lógica organizacional que conserva la alta cohesión y el bajo acoplamiento.
- ✓ La propuesta realizada, validada mediante el método Quasar, es factible tributando al cumplimiento de los factores de calidad planteados.

RECOMENDACIONES

- ✓ Valorar la introducción de la capa de acceso a datos como módulo adicional del CMS para próximas iteraciones, con vista a hacer un uso más eficiente de las virtudes de Drupal.
- ✓ Profundizar en el tema de seguridad a partir de la propuesta, teniendo en cuenta aspectos como la encriptación de los datos, con el fin de garantizar la integridad de la información.

BIBLIOGRAFÍA

Abreu Bartomeo, Yanedi y Colomé Cedeño, Dunia Maria. 2007. *Portal de los Institutos Politécnicos de informática. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* [pdf] Ciudad de La Habana : s.n., 2007.

Aja Quiroga, Lourdes. 2002. La Gestión de información, gestión del conocimiento y gestión de la calidad en las organizaciones. [En línea] 2002. [Citado el: 6 de marzo de 2008.] http://bvs.sld.cu/revistas/aci/vol10_5_02/aci04502.htm. Acimed Vol 10 05 2002.

Anónimo. 2006. Apache 2.0. *Visión general de las nuevas funcionalidades de Apache 2.0. .* [En línea] 2006. [Citado el: 13 de mayo de 2008.] http://www.apache2.es/2.0.58/new_features_2_0.html.

—. 2004. Assets. [En línea] 2004. [Citado el: 13 de febrero de 2008.] http://www.assets.co.cu/.

—. 2003. *Ayuda Proceso Unificado de Rational.* [html] 2003.

—. IBM. [En línea] [Citado el: 13 de febrero de 2008.] http://www-306.ibm.com/software/awdtools/developer/rose/enterprise/index.html.

—. 2005. Osmosis Latina. *Guia de Apache 2.* [En línea] 2005. [Citado el: 12 de 5 de 2008.] http://www.osmosislatina.com/apache2/modulos.htm.

—. 2003. TiendaLinux.com. *Ventajas de PostgreSQL.* [En línea] 2003. [Citado el: 12 de abril de 2008.] http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html.

—. 2006. Navegapolis.net. *¿Qué es DSDM?* En línea] 2006. [Citado el: 20 de marzo de 2008.] http://www.navegapolis.net/content/view/361/59/.

—. 2007. *Zend Studio 5 I LAS SOLUCIONES MÁS COMPLETAS PARA EL DESARROLLO DE PHP .* [pdf] s.l. : The PHP Company, 2007.

—. 2008. NexoDigital. [En línea] 2008. [Citado el: 13 de febrero de 2008.] http://www.nexo-digital.com

Billy Reinoso, Carlos y Kicillof, Nicolás. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* [doc] Buenos Aires : Universidad de Buenos Aires, 2004.

—. 2004. *Lenguajes de Descripción de Arquitectura (ADL).* [doc] Buenos Aires : Universidad de Buenos Aires, 2004.

Booch, Grady, Rumbaugh, James y Jacobson, Ivar. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Education. S.A, 2000.

Burbeck, Steve. 1992. The UIUC Smalltalk Archive. *Application programming in Smalltalk-80: How to use Model-View-Controller (MVC).* [En línea] 1992. [Citado el: 16 de marzo de 2008.] http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html.

Cadena, Sylvia. 2006. EsLaRed. *Comparación de CMS .* [En línea] 28 de marzo de 2006. [Citado el: 30 de marzo de 2008.] http://eslared.org.ve/walc2004/apc-aa/archivos-aa/1e60354f4717edb9fb793dbc5219499d/walc04_martesam_sc.ppt.

Calegari, Daniel, Perovich, Daniel y Vignaga, Andrés. 2006. Facultad de Ingeniería. Universidad de la República de Uruguay. *Impacto de la Evolución de la Base de Datos en el Diseño de un Sistema de Información. XXXII Conferencia Latinoamericana de Informática (CLEI).* [En línea] 2006. [Citado el: 2008 de mayo de 18.] http://www.fing.edu.uy/inco/grupos/coal/investigacion/publicaciones/cvp06.pdf.

Camacho, Erika, Cardeso, Fabio y Nuñez, Gabriel. 2004. *Arquitecturas de Software. Guía de estudio.* [pdf] 2004.

Canal Velasco, Carlos. 2000. *Un Lenguaje para la Especificación y Validación de Arquitecturas de Software.* [pdf] Málaga : s.n., 2000.

Casanovas, Josep. 2004. desarrolloweb.com. *Usabilidad y arquitectura del software.* [En línea] 2004. [Citado el: 25 de octubre de 2007.] http://www.desarrolloweb.com/articulos/1622.php .

Cataldi, Zulma. 2000. *Metodología de diseño, desarrollo y evaluación de software educativo.*[pdf]. Tesis de Magíster en Informática. Facultad de Informática. UNLP.

Cedeño Prado, Yordan. 2004. Gestiopolis.com. *La gestión de los recursos humanos y las tecnologías de la información.* [En línea] 2004. [Citado el: 8 de noviembre de 2007.] abril 2004. [Cited: noviembre 8, 2007.] http://www.gestiopolis.com/canales/derrhh/articulos/57/gesrechum.htm.

Clements, Paul C. 1996. *A Survey of Architecture Description Languages.* Pittsburg : Software Engineering Institute Carnegie Mellon University : s.n., 1996.

Contreras Díaz, Yimian de Lyz y Rivero Amador, Soleydi. 2007. *Diseño del Sistema de Gestión de Información del Centro de Estudios de Medio Ambiente y Recursos Naturales (CEMARNA) de la Universidad de Pinar del Río.* [pdf] Pinar del Río : 2do Simposio Internacional sobre Tecnologías de la Información en las Organizaciones Informacionales., 2007.

Cuerda Garcia, Xavier y Minguillón Alfonso, Julià. 2004. Mosaic. *Introducción a los Sistemas de Gestión de Contenidos (CMS) de código abierto.* [En línea] 2004. [Citado el: 30 de marzo de 2008.] http://mosaic.uoc.edu/articulos/cms1204.html.

Dalcero, Diego. 2008. Crossing the chasm . *Zend Studio para Eclipse y Zend Platform 3.6 disponibles.* [En línea] 2008. [Citado el: 30 de marzo de 2008.] http://ddalcero.blogspot.com/2008/01/zend-studio-para-eclipse-y-zend.html.

Dondo, Agustín. 2007. PHP en castellano. *¿Por qué elegir PHP?* [En línea] 2007. [Citado el: 6 de marzo de 2008.] http://www.programacion.net/php/articulo/porquephp/.

Evaluando la arquitectura de software.Parte 1 . **Gómez, Omar. 2007.** 01, México : Revista Software Guru Conocimiento en practica, 2007, Vol. 03 .

Faulkner, Stéphane y Kolp, Manuel. 2003. *Towards an agent architectural description language for information systems.* [pdf] 2003.

Firesmith, Donald. 2006. *QUASAR: A Method for the QUality Assessment of Software-Intensive System Architectures.* [pdf] Pittsburgh : Carnegie Mellon Software Engineering Institute, 2006.

Fowler, Martin. 2002. *Patterns of Enterprise Application Architecture.* [chm] s.l. : The Addison-Wesley Signature Series., 2002.

Gamma, Erich, y otros. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software.* [html] s.l. : Addison-Wesley Profesional Computing Series., 1994.

Garlan, David y Shaw, Mary. 1994. *An introduction to software architecture.* [pdf] s.l. : School of Computer Science. Carnegie Mellon University., 1994.

Garzás, Javier. 2006. *LA ARQUITECTURA SOFTWARE. EL MODELO 4+1.* [En línea] 2006. [Citado el: 12 de febrero de 2008.] http://jgarzas.googlepages.com/4mas1.

Kruchten, Philippe. 1995. *The "4+1" View Model of Software Architecture.* [pdf] s.l. : IEEE Software, 1995.

Kuroki, Christian. 2005. *PostgreeSQL 8. Migración a PostgreSQL desde otras bases de datos.* [pdf] 2005.

Luo, Yuhua y Turmeda, Anselm. 2008. *Ampliación de Programación Orientada a Objectos.* [pdf] 2008.

Martin, Roberkys, Enriquez, Derick y Viltres, Hubert. 2005. *Trabajo Investigativo sobre Sistemas de Gestión de Contenido. CMS Drupal.* [pdf] Ciudad de La Habana : Universidad de las Ciencias Informáticas., 2005.

Maturana, Sergio. 1999. *¿Cuánto ayudan los sistemas ERP en la planificación y programación de las actividades de una cadena de abastecimiento?* [pdf] Chile : Universidad Católica de Chile, 1999.

Palenque Terry, Efraín y Cepero Morales, Mirtha. 2006. *Los Sistemas de Información de Gestión y la eficiencia empresarial.* [pdf] Ciudad de La Habana : Facultad de Economía, Universidad de La Habana, 2006.

Peralta, Manuel. 1997. Monografías. com. *Sistema de Información.* [En línea] 1997. [Citado el: 13 de febrero de 2008.] http://www.monografias.com/trabajos7/sisinf/sisinf.shtml.

Reyero, Jose A. 2005. Drupal Hispano. *Características de Drupal.* [En línea] 2005. [Citado el: 25 de marzo de 2008.] http://www.drupal.org.es/caracteristicas.

Schmuller, Joseph. 2000. *Aprendiendo UML en 24 horas.* [pdf] Mexico : PEARSON EDUCACION, 2000.

Shaw, Mary y Garlan, David. 1994. *Characteristics of Higher-level Languages for Software Architecture.* [pdf] Pittsburg : School of Computer Science Carnegie Mellon University, 1994.

Vestal, Steve. 1993. *A cursory overview and comparison of four Architecture Description Languages.* [pdf] s.l. : Honeywell Technology Center, 1993.

Vizcaíno, Aurora, García, Felix Óscar y Caballero, Ismael. *Una Herramienta CASE para ADOO: Visual Paradigm.* [pdf] s.l. : Universidad de Castilla La Mancha.

ANEXOS

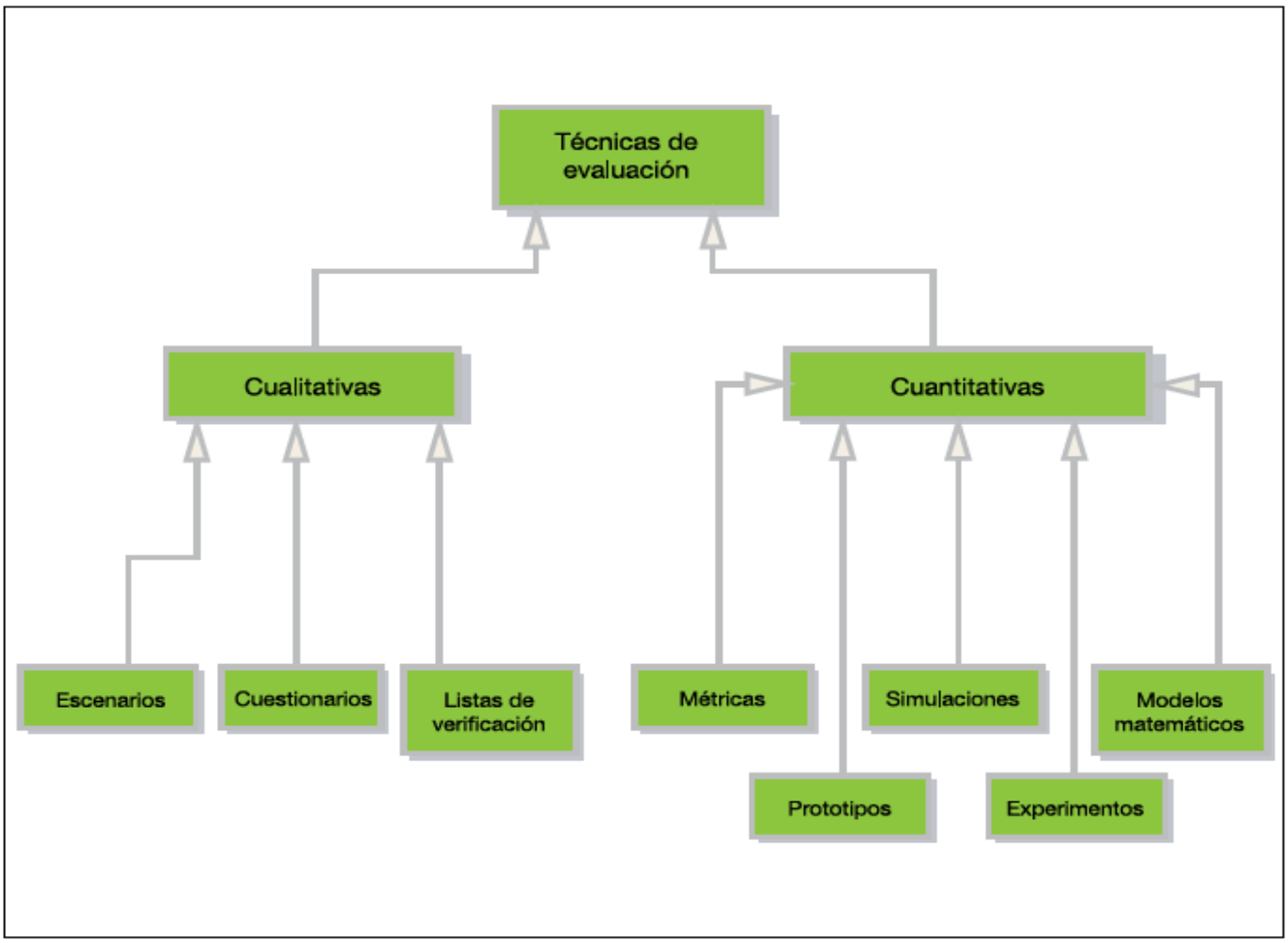
Anexo 1. Especificación de atributos de calidad observables vía ejecución (Camacho et al., 2004).

Atributo de Calidad	Descripción
Disponibilidad *(Availability)*	Es la medida de disponibilidad del sistema para el uso (Barbacci et al., 1995).
Confidencialidad *(Confidentiality)*	Es la ausencia de acceso no autorizado a la información (Barbacci et al., 1995).
Funcionalidad *(Functionality)*	Habilidad del sistema para realizar el trabajo para el cual fue concebido (Kazman et al., 2001).
Desempeño *(Performance)*	Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. (IEEE 610.12). Según Smith (1993), el desempeño de un sistema se refiere a aspectos temporales del comportamiento del mismo. Se refiere a capacidad de respuesta, ya sea el tiempo requerido para responder a aspectos específicos o el número de eventos procesados en un intervalo de tiempo. Según Bass et al. (1998), se refiere además a la cantidad de comunicación e interacción existente entre los componentes del sistema.
Confiabilidad *(Reliability)*	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo (Barbacci et al., 1995).
Seguridad externa *(Safety)*	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información (Barbacci et al., 1995).
Seguridad interna *(Security)*	Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos (Kazman et al., 2001).

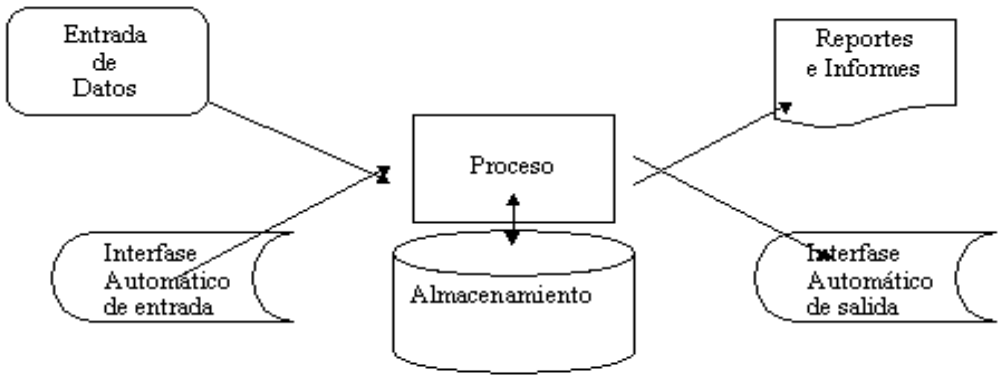
Anexo 2. Especificación de atributos de calidad No observables vía ejecución (Camacho et al., 2004).

Atributo de Calidad	Descripción
Configurabilidad *(Configurability)*	Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema (Bosch et al., 1999).
Integrabilidad *(Integrability)*	Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente al ser integrados. (Bass et al. 1998)
Integridad *(Integrity)*	Es la ausencia de alteraciones inapropiadas de la información (Barbacci et al., 1995).
Interoperabilidad *(Interoperability)*	Es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema. Es un tipo especial de *integrabilidad* (Bass et al. 1998)
Modificabilidad *(Modifiability)*	Es la habilidad de realizar cambios futuros al sistema. (Bosch et al. 1999).
Mantenibilidad *(Maintainability)*	Es la capacidad de someter a un sistema a reparaciones y evolución (Barbacci et al., 1995). Capacidad de modificar el sistema de manera rápida y a bajo costo (Bosch et al. 1999).
Portabilidad *(Portability)*	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos (Kazman et al., 2001).
Reusabilidad *(Reusability)*	Es la capacidad de diseñar un sistema de forma tal que su estructura o parte de sus componentes puedan ser reutilizados en futuras aplicaciones (Bass et al. 1998).
Escalabilidad *(Scalability)*	Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental (Pressman, 2002).
Capacidad de Prueba *(Testability)*	Es la medida de la facilidad con la que el software, al ser sometido a una serie de pruebas, puede demostrar sus fallas. Es la probabilidad de que, asumiendo que tiene al menos una falla, el software fallará en su próxima ejecución de prueba (Bass et al. 1998).

Anexo 3. Clasificación de técnicas de evaluación (Gómez, 2007).



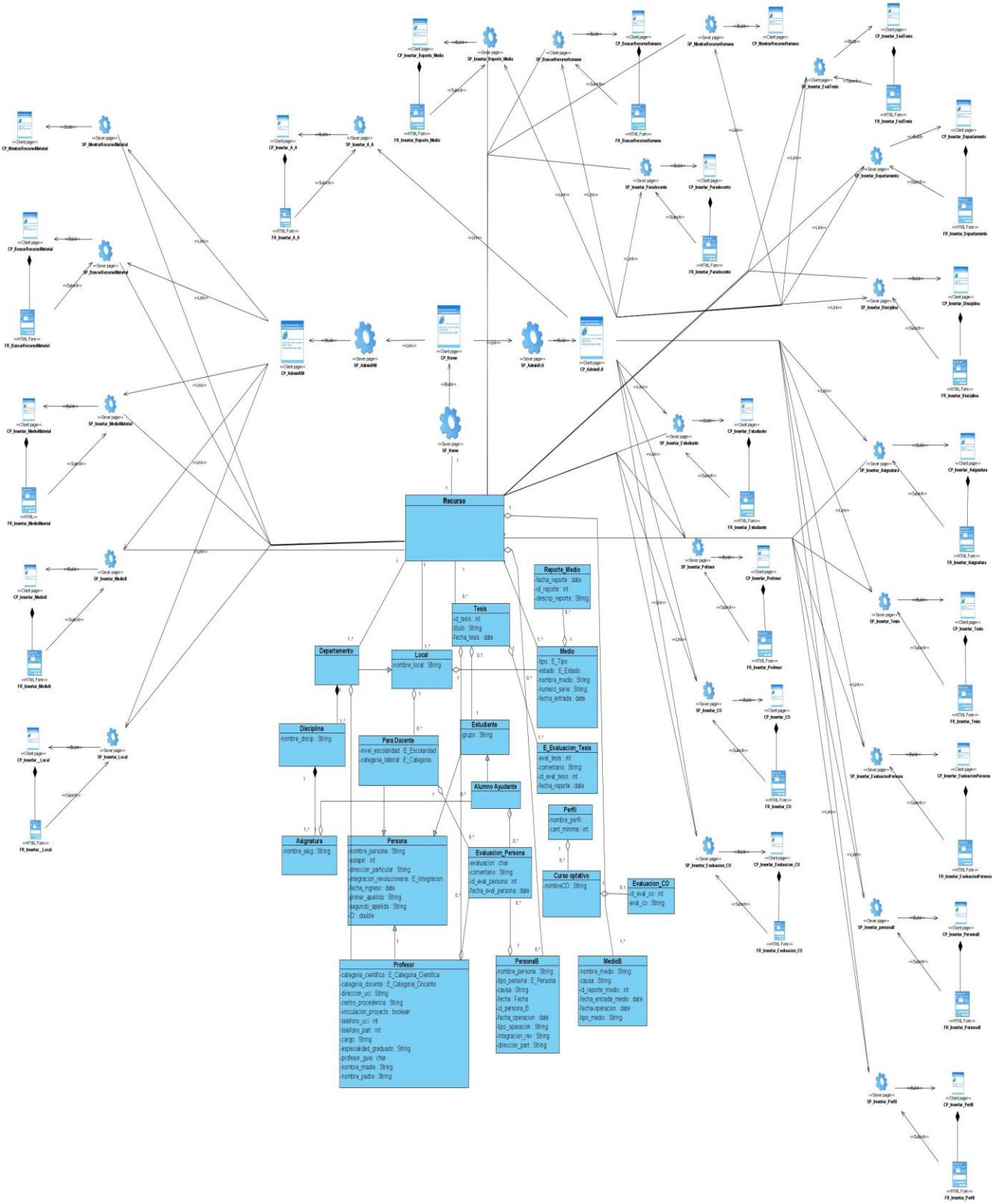
Anexo 4. Actividades básicas de un Sistema de Información (Peralta, 1997).



Anexo 5. Cualidades del IDE Zend Development en su versión profesional (The PHP Company, 2007).

Características de Alto Nivel	Prof.
IDE wProfesional/Editor, Depurador y Ayuda	•
Windows, Linux, versiones Mac	•
Soporte Multilingüe	•
Editor profesional	•
Coloreado de Sintaxis para PHP, HTML, y JavaScript	•
Coloreado de Sintaxis para XML y CSS	•
Plantillas de Código	•
Carpetas de Código	•
Soporte PHP 4 y PHP 5	•
Completado de código avanzado	•
Análisis de Código PHP (+PHPDoc)	•
Depurador profesional Interno	•
Código Snippets	•
Integración de Framework Zend	•
Completado de Código Java	•
Depurador remoto y Depurar URL	•
Analista remoto	•
Conectividad e integración de la base de datos para: IBM DB2/Cloudscape/Derby, MySQL, Oracle Microsoft SQL Server, SQLite y PostgreSQL	•
Establecer herramientas SQL: Editor de consultas SQL, Explorador SQL y más...	•
PHPDocumentor	•
Integración CVS para el equipo de desarrollo	•
Integración de subversión	•
Soporte FTP	•
Soporte de servicios web (SOAP)	•
Errores scripts e informes sobre la actividad	•
Normas de alerta	•
Integración XML	•
QA / soluciones de prueba	•
Plataforma en etapas	•
Consola de administración centralizada	•

Anexo 6. Diagrama general de clases del diseño para el Sistema de Gestión de Recursos de la Facultad 3.



Anexo 7. Cálculos para estimación de tamaño de datos en memoria de la tabla Departamento.

Departamento

Aspectos	Valor	Unidad
Total de registros	6	

Nombre columna	Tipo de dato	Precisión (p)	Escala (s) *	Valor Negativo *	Tamaño de columna (Byte)	Tamaño de cabecera (Byte)
nombreLocal	VARCHAR2	50			50	1

	Longitud por registro	54
	Espacio disponible por bloque(Byte)	7180
	Registros disponibles por bloque	132
	Total de bloques por tabla	1
	Tamaño total por tabla(Kbyte)	8

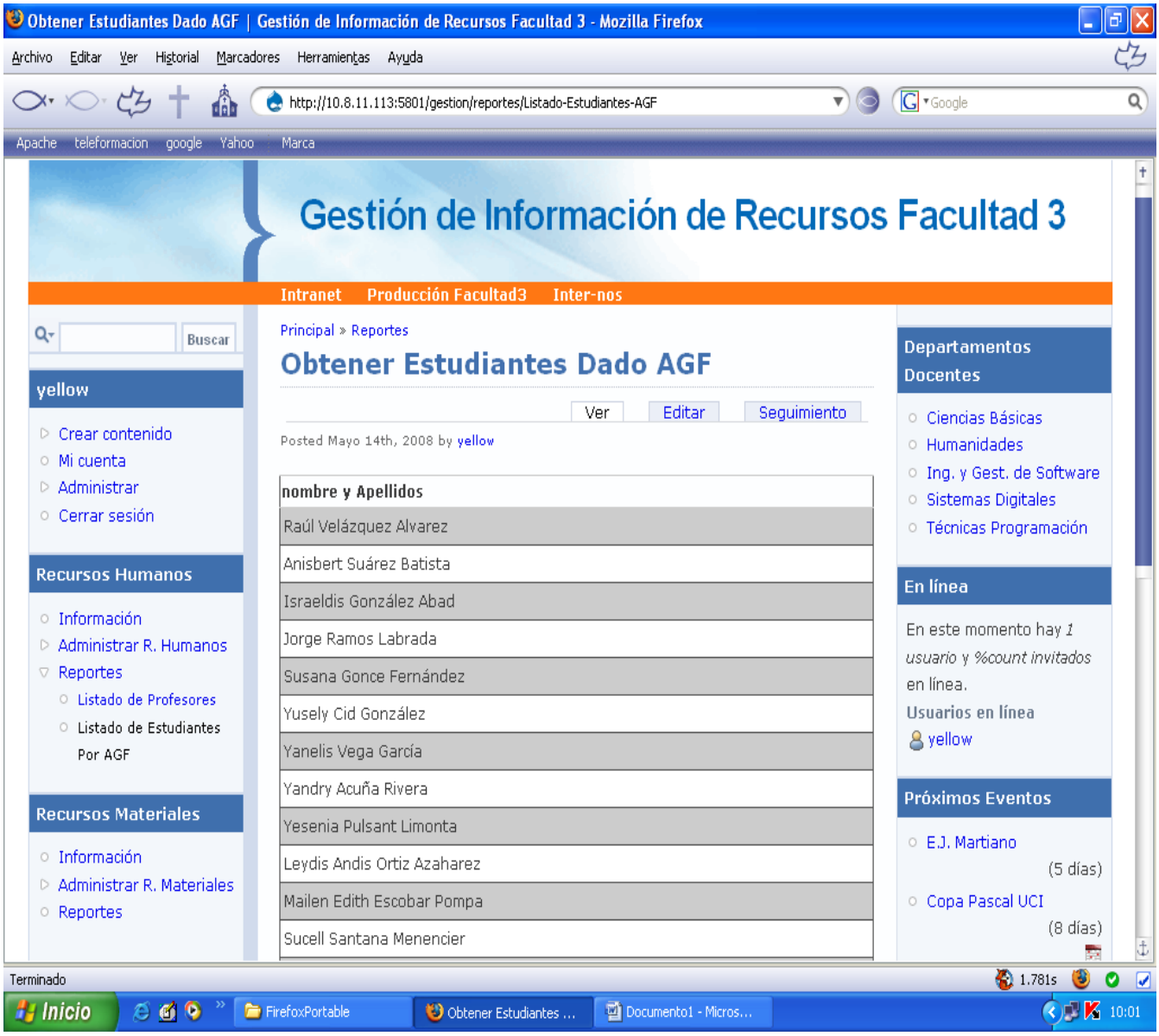
Índice nombreLocal

Nombre columna	Tipo de dato	Precisión (p)	Escala (s) *	Valor Negativo *	Tamaño de columna (Byte)	Tamaño de cabecera (Byte)
nombreLocal	VARCHAR2	50			50	1

	Longitud por registro	63
	Espacio disponible por bloque(Byte)	7250
	Registros disponibles por bloque	115

	Total de bloques por tabla	1
	Tamaño total por tabla(Kbyte)	8

Anexo 8. Tiempos de respuesta para obtener Listado de estudiantes conociendo año, grupo y facultad.



Anexo 9. Tiempos de respuesta para obtener Listado de profesores de la Facultad 3.

GLOSARIO

Artefacto: Pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar actividades. Puede ser un modelo, un elemento de un modelo, o un documento.

CASE: Ingeniería de Software Asistida por Ordenador (Computer Aided Software Engineering, del inglés).

Caso de uso: Conjunto de secuencia de acciones que un sistema ejecuta y que produce un resultado observable para un actor.

Framework: Marco de aplicación o conjunto de bibliotecas orientadas a la reutilización a muy gran escala de componentes software para el desarrollo rápido de aplicaciones.

Nodo: Elemento físico que dispone de memoria y con frecuencia capacidad de almacenamiento.

Stakeholders: Interesado en que el proyecto resulte, alguien a quien impactará el éxito o fracaso del sistema. Cualquier persona o grupo que se verá, directa o indirectamente, afectado por el sistema a ser desarrollado.