Universidad de las Ciencias Informáticas

Facultad 10

Título: Arquitectura de Software del Sistema de Gestión de Información Intranet 2

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

Autor:

Carlos Rafael Ballestero Fernández

Tutores:

Ing. Annia Surós Vicente

Ing. Alexander Hernández Chapman

Ciudad de la Habana

Julio 2007

"La verdadera grandeza del hombre

radica en optar por la realización personal

en circunstancias en que otros optan por la locura."

Iraida Álvarez Sadín

Declaración de Autoría

Por este medio declaro que soy el único autor de este trabajo, y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

---	---	---
Firma del Autor	Firma del Tutor	Firma del Tutor

Agradecimientos

A mis padres Margarita Leonor Fernández Vega y Carlos Fidel Ballestero Labrada, por su amor, su cariño y la confianza que depositaron en mí durante todo este tiempo.

A mi familia, en especial a mis tíos Héctor, Rafael, Manuel, Mariza, Magali y a mis primos Gretel y Yudier por haber estado a mi lado.

A mis tutores Annia Surós Vicente y Alexander Hernández Champan, que sin su apoyo y dedicación no hubiese sido posible la realización del presente trabajo.

A mis compañeros, por ser mi familia y brindarme se apoyo durante todo el periodo de vida en la universidad.

A nuestro Comandante en Jefe Fidel Castro, por habernos enseñado que lo más importante en los hombres se lleva en el corazón.

Dedicatoria

Dedico el presente trabajo a mis padres,

mi familia y amigos

Resumen

Resumen

Los procesos de la gestión de la información y los conocimientos adquieren en la actualidad un papel protagónico cuando de lograr eficiencia y calidad se trata, ya sea en el campo investigativo, de formación o producción. Gracias al constante desarrollo de las Tecnologías de la Información y las Comunicaciones (TICs) estos procesos son automatizados.

En la Universidad de las Ciencias Informáticas (UCI) se manipula gran cantidad de información, las cuales se necesitan centralizarlas y que sean accesibles uniformemente para los usuarios. El presente trabajo de diploma pretende diseñar una arquitectura de software que permita optimizar todo el proceso de gestión de información y aumentar la interoperabilidad entre los distintos servicios que se generan en la universidad, además, que sirva de guía para desarrollar una intranet que le de cumplimiento a estos objetivos, con el propósito de lograr una mayor eficiencia en el entorno productivo y en las actividades docentes principalmente.

Tabla de Contenido

Tabla de Contenido

Introducción

Con la incorporación de la computación en diferentes esferas de la sociedad, y a pesar de que los estilos de programación distaban de ser una metodología coherente y principalmente eficiente surgió una demanda de software, con el propósito de resolver múltiples problemas como el procesamiento de datos, la ejecución de diversas operaciones, etc. Con este avance muchos problemas fueron resueltos, pero al mismo tiempo dieron lugar a otros, por ejemplo, la comunicación entre los miembros de trabajo, el compartimiento de conocimientos y herramientas. La falta de un medio de comunicación entre los ordenadores evitaba el desarrollo, tanto empresarial como personal. A partir de estas necesidades surgen las redes, conjunto de hardware y software que permiten el intercambio de datos (documentos, imágenes, videos, etc.) digítales, el uso de herramientas o recursos entre diferentes estaciones de trabajo.

Con la aparición de las TICs se ha incrementado el interés en las personas de aumentar su nivel cultural, provocando un crecimiento incalculable de personas en la esfera de la informática.

Con el transcurso del tiempo, los requerimientos del negocio en las empresas varían con frecuencia, obligando a que las TICs se encuentren evolucionando constantemente y que cada vez sean más imprescindibles para el desarrollo de la sociedad humana.

Hoy en día, producto de las exigencias de los negocios, las empresas exigen que el desarrollo de las aplicaciones empresariales sean cada vez más complejas, obligando a los desarrolladores a enfrentarse a un conjunto de dificultades como plataformas heterogéneas, manejo de datos complejos, independencia entre aplicaciones, múltiples usuarios simultáneos, entre otras características requeridas por las aplicaciones con el fin de brindar diversos servicios de forma eficiente a los usuarios finales (personal beneficiado).

Introducción

La Universidad de las Ciencias Informáticas (UCI) por sus características se ha convertido en una ciudad digital, en la cual el conocimiento de las diferentes informaciones y el uso de diversos servicios son de vital importancia para el desarrollo de las actividades cotidianas. Como en todas las instituciones, en ella se realizan diversas actividades culturales, políticas, educativas y productivas, respaldadas por diferentes organizaciones de masa como la Unión de Jóvenes Comunistas, el Partido Comunista de Cuba, la Federación de Mujeres Cubanas, etc. Todas las informaciones relacionadas a las actividades y a las organizaciones de masas, incluyendo otras, como noticias, docencias o comunidades son divulgadas mediante un software denominado Intranet, que constituye el principal medio de difusión interno de esta institución.

Con este sistema de gestión de información, el personal vinculado a la UCI puede contar con múltiples informaciones relacionadas con los servicios que se brindan (como servicios telemáticos, directorio personal, etc.), y las diferentes medidas adoptadas por la Dirección de Gestión Tecnológica con vista a contribuir a la informatización de la UCI. Mediante esta aplicación se pone a disposición de los usuarios (personal con acceso a la aplicación) los recursos y espacios que la producción ha ido creando para fomentar e incentivar la producción y la investigación en el entorno universitario.

Desde el surgimiento de esta institución la cantidad de personas vinculadas a ella ha aumentado enormemente, albergando en la actualidad a más de 10.000 personas. Producto de este crecimiento se han extendido, de forma simultánea, las solicitudes de los servicios y las informaciones que se facilitan mediante la Intranet, esto ha provocado que el sistema de gestión de información se encuentre generalmente congestionado. Como esta herramienta constituye el principal medio de divulgación de las informaciones de la institución, requiere de un proceso constante de actualización, que en la actualidad no se puede realizar con eficiencia, pues en la universidad no se cuenta con el personal suficiente que contenga conocimientos del código ni con la documentación necesaria para llevar a cabo esta tarea, originando problemas de

integración de las informaciones. Estos inconvenientes han provocado que los servicios ofrecidos por el sistema de gestión de información no tengan un desempeño eficiente.

El presente trabajo de diploma pretende dar solución a la **situación problémica** anteriormente expuesta. Por tanto, nuestro **problema** consiste en *¿cómo desarrollar el sistema Intranet de la UCI de forma tal que pueda satisfacer las necesidades de sus usuarios?*

Atendiendo a las necesidades de la UCI el **objeto de estudio** del presente trabajo está enmarcado en *las diferentes arquitecturas de software y las herramientas más utilizadas en el desarrollo de aplicaciones empresariales* y el **campo de acción** en *la arquitectura orientada a servicios*.

El **objetivo general** del trabajo es *diseñar una arquitectura de software para un sistema de gestión de información que permita la interoperabilidad entre los servicios y que guíe la planificación y organización del mismo, con la consiguiente mejora en la obtención de información del personal adjunto a la Universidad de las Ciencias Informáticas*.

Para darle cumplimiento al objetivo trazado se desarrollaron las siguientes **tareas**:

1. Consulta de bibliografía vinculada en las temática en estudio para la fundamentación del problema.
2. Fundamentación del tema.
3. Proposición y descripción de la alternativa de solución al problema.
4. Evaluación de la arquitectura propuesta.

Para la realización de estas tareas se utilizarán los diferentes **métodos de investigación.**

1. **Métodos Teóricos.**

1.1.1. **Análisis y síntesis**: para conocer, reflexionar y aumentar los conocimientos acerca de la línea de investigación a partir de la consulta de la literatura científica correspondiente, y luego, haciendo uso de la síntesis para confeccionar un cuerpo coherente en el cual se resuman los resultados obtenidos del análisis.

1.1.2. **La inducción y la deducción**: se empleó para pasar al conocimiento de casos particulares y generales que sirvieron de facilitadores en la formulación teórica de algunas conclusiones.

1.1.3. **Modelación**: con el objetivo de modelar el diseño propuesto.

2. **Métodos Empíricos**

2.1.1. **Observación**: se utilizó para comprobar el estado real de todo el personal de La Universidad de Ciencias Informáticas (UCI) en consecuencia con las necesidades de obtención de información precisa y concreta a través de una aplicación Web.

2.1.2. **Entrevistas**: a todo el personal (estudiantes, trabajadores, profesores) que de una forma labora en la UCI y que requieren de informaciones precisas y confiables sobre aspectos específicos inherentes a la solución del problema de investigación.

2.1.3. **Cuestionarios**: a estudiantes, trabajadores, profesores, con el objetivo de constatar el grado de acercamiento al tema.

El presente trabajo se encuentra estructurado de la siguiente forma:

Introducción

Capitulo 1. Fundamentación teórica.

En ésta sección se realiza un estudio de las arquitecturas más comunes en la actualidad y de las tecnologías que más se utilizan en el desarrollo de intranets, sitios Web, etc.

Capitulo 2. Propuesta y Descripción de la Arquitectura de Software.

En está sección se reflejará la propuesta de la arquitectura de la intranet 2 y la descripción correspondiente, utilizando como guía de desarrollo la metodología Racional Unified Process (RUP).

Capitulo 3. Evaluando la Arquitectura de Software.

En esta sección se realiza una evaluación de la arquitectura propuesta mediante el uso de la técnica de evaluación basada en escenarios y el método Active Reviews Intermediate Designs (ARID).

Conclusiones.

Recomendaciones.

Referencia Bibliográfica

Bibliografía

Anexos

Glosario de Términos

Capitulo 1. Fundamentación teórica

Introducción

El desarrollo de aplicaciones en el entorno empresarial requiere cada vez más alcanzar un conjunto de requisitos comunes tales como integración, seguridad, confiabilidad, escalabilidad y disponibilidad. Sin el cumplimiento de éstos, no se puede hablar de calidad del producto desarrollado. Todos estos requisitos se deciden en el proceso de diseño de la arquitectura de la aplicación, y los servicios Web tienen gran impacto en la implementación de estos requisitos.

La arquitectura del Software tiene que ver con la planificación y mantenimiento en la estructura del sistema de los conceptos iniciales a través del desarrollo y operación. Una buena arquitectura es una estructura del sistema estable que puede acomodarse a los cambios de requisitos y de tecnologías.

Uno de los principios de las metodologías de desarrollo de software consiste en priorizar la definición, el diseño, la implementación y evaluación de la arquitectura (esqueleto del sistema). La esencia de este principio es dedicar los mínimos esfuerzos a implementar un prototipo estable de arquitectura que garantice la viabilidad del proyecto en las fechas más tempranas posibles. La arquitectura implementa los requisitos más críticos y encierra los mayores riesgos del desarrollo, la clave del éxito está en definir qué es y qué no es la arquitectura.

La mayor parte de las metodologías de desarrollo de software expresan que la especificación de la arquitectura de software a edades tempranas es muy ventajosa, a pesar de esto, actualmente no existe un estándar para establecer la definición de una arquitectura. La gran ventaja de usar lenguajes de definición de arquitectura o ADLs (Architecture Description Languages) es que permiten una mejor comunicación entre los diseñadores, implementadores y lectores. Las ventajas de un cierto ADL está dada por su poder expresivo de especificar la estructura y el comportamiento del sistema.

Capitulo 1. Fundamentación Teórica

1.1. Arquitectura de Software

La arquitectura de software (AS) es la parte de la ingeniería de software que se encarga de la descripción de la estructura de un sistema como una serie de componentes, con el objetivo de organizar los distintos subsistemas adecuadamente que conforman el sistema y permitir la integración de diversos grupos de desarrollo en mismo proyecto.

1.1.1 Definición de Arquitectura de Software

Una definición reconocida es la de Clements [4]: La AS es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión del detalle inherente a la mayor parte de las abstracciones.

A pesar de que existen numerosas definiciones de arquitectura, las cuales ninguna de ellas está respaldada unánimente por la totalidad de los arquitectos, existe un acuerdo de que ella se refiere a la estructura a grandes rasgos del sistema, estructura consistente en componentes y relaciones entre ellos.

Teniendo en cuenta la variedad de definiciones de arquitectura de software, Mary Shaw y David Garlan [8] proporcionaron una sistematización iluminadora, explicando las diferencias entre definiciones en función de distintas clases de modelos. Destilando las definiciones y los puntos de vista implícitos o explícitos, los autores clasifican los modelos de esta forma:

Modelos estructurales: Mantienen que la arquitectura de software está compuesta por componentes, conexiones entre ellos y (usualmente) otros aspectos tales como configuración, estilos, restricciones, semántica, análisis, propiedades, racionalizaciones,

requerimientos, necesidades de los participantes. El trabajo en esta área está caracterizado por el desarrollo de lenguajes de descripción arquitectónica (ADLs).

*Modelos de **frameworks***: Son similares a la vista estructural, pero su énfasis primario radica en la (usualmente una sola) estructura coherente del sistema completo, en vez de concentrase en su composición. Estos modelos frecuentemente se refieren a dominios o clases de problemas específicos. El trabajo que ejemplifica esta variante incluye arquitecturas de software específicas de dominios, como CORBA, o modelos basados en CORBA, o repositorios de componentes específicos, como PRISM.

Modelos dinámicos: Enfatizan la cualidad conductual de los sistemas. “Dinámico” puede referirse a los cambios en la configuración del sistema, o la dinámica involucrada en el proceso de la computación, tales como valores cambiantes de datos.

Modelos de procesos: Se concentran en la construcción de la arquitectura, y en los pasos o procesos involucrados en esa construcción. En esta perspectiva, la arquitectura es el resultado de seguir un argumento (script) de procesos. Esta vista se ejemplifica con el actual trabajo sobre programación de procesos para derivar arquitecturas.

Modelos funcionales: Una minoría considera la arquitectura como un conjunto de componentes funcionales, organizados en capas que proporcionan servicios hacia arriba. Es tal vez útil pensar en esta visión como un framework particular.

Ninguna de estas vistas excluye a las otras, ni representa un conflicto fundamental sobre lo que es o debe ser la AS. Por el contrario, representan un espectro en la comunidad de investigación sobre distintos énfasis que pueden aplicarse a la arquitectura: sobre sus partes constituyentes, su totalidad, la forma en que se comporta una vez construida, o el proceso de su construcción. Tomadas en su conjunto, destacan más bien un consenso.

1.1.2 Estilos y Patrones Arquitectónicos

Estilo Arquitectónico

Shaw y Garlan [14] definen *estilo arquitectónico* como una familia de sistemas de software en términos de un patrón de organización estructural, que define un vocabulario de componentes y tipos de conectores y un conjunto de restricciones de cómo pueden ser combinadas.

Buschmann [3] definen *estilo arquitectónico* como una familia de sistemas de software en términos de su organización estructural. Expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como también las reglas para su construcción.

A simple vista, ambas definiciones parecen expresar la misma idea. La diferencia entre los planteamientos de Shaw y Garlan [14] y Buschmann [3] viene dada por la amplitud de la noción de *componente* en cada una de las 20 definiciones. Buschmann [3] asume como componentes a subsistemas conformados por otros componentes más sencillos, mientras que Shaw y Garlan [14] utilizan la noción de componente como elementos simples, ya sean de dato o de proceso.

La siguiente tabla resume los principales estilos arquitectónicos, los atributos de calidad que propician y los atributos que se ven afectados negativamente (atributos en conflicto), de acuerdo a Bass [2].

Estilos	**Descripción**	**Atributos Asociados**	**Atributos en Conflictos**
Datos Centralizados	Sistemas en los cuales cierto números de clientes acceden y	Integrabilidad Escalabilidad Modificabilidad	Desempeño

	actualizan datos compartidos de un repositorio de manera frecuente		
Flujo de Datos	El sistema es visto como una serie de transformaciones sobre piezas sucesivas de datos de entrada. El dato ingresa en el sistema, y fluye entre los componentes, de uno en uno, hasta que se le asigne un destino final (salida o repositorio).	Reusabilidad Modificabilidad Mantenibilidad	Desempeño
Máquinas Virtuales	Simulan alguna funcionalidad que no es nativa al hardware o software sobre el que está implementado	Portabilidad	Desempeño
Llamada y Retorno	El sistema se constituye de un programa principal que el control del sistema y varios subprogramas que se comunican con éste mediante el uso de llamadas.	Modificabilidad Escalabilidad	Desempeño Integrabilidad
Componentes	Consiste en un número	Modificabilidad	Desempeño

Independientes	de procesos u objetos independientes que se comunican a través de mensajes.	Escalabilidad	Integrabilidad

Tabla 1 Principales Estilos Arquitectónicos

Patrón Arquitectónico

Buschmann [2] define *patrón* como una regla que consta de tres partes, la cual expresa una relación entre un contexto, un problema y una solución.

En líneas generales, un patrón sigue el siguiente esquema:

- *Contexto.* Es una situación de diseño en la que aparece un problema de diseño
- *Problema*. Es un conjunto de fuerzas que aparecen repetidamente en el contexto
- *Solución*. Es una configuración que equilibra estas fuerzas. Ésta abarca:
    1. Estructura con componentes y relaciones
    2. Comportamiento a tiempo de ejecución: aspectos dinámicos de la solución, como la colaboración entre componentes, la comunicación entre ellos, etc.

Partiendo de esta definición, propone los *patrones arquitectónicos* como descripción de un problema particular, que aparece en contextos de diseño específico, y presenta un esquema genérico demostrado con éxito para su solución.

Así mismo, Buschmann [3] plantean que los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de software. Provee un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos.

La siguiente tabla presenta algunos patrones arquitectónicos, además de los atributos que propician y los atributos en conflicto, de acuerdo a Buschmann [3].

Patrón Arquitectónico	Descripción	Atributos Asociados	Atributos en conflicto
Layers	Consisten en estructurar aplicaciones que puedan ser descompuestas en grupos de sub-tareas, las cuales se clasifican de acuerdo a un nivel particular de abstracción.	Reusabilidad Portabilidad Facilidad de Prueba	Desempeño Mantenibilidad
Pipes and Filtres	Provee una estructura para Los sistemas que procesan un flujo de datos. Cada paso de procesamiento está encapsulado en un componente filtro (filter). El dato pasa a través de conexiones (pipes), entre filtros adyacentes.	Reusabilidad Mantenibilidad	Desempeño
Blackboard	Se aplica para problemas cuya solución utiliza estrategia no determinísticas. Varios subsistemas ensamblan su conocimiento para construir una posible solución parcial ó aproximada.	Modificabilidad Mantenibilidad Reusabilidad Integridad	Desempeño Facilidad de prueba
Broker	Puede ser usado para estructurar sistemas de	Modificabilidad Portabilidad	Desempeño

	software distribuidos con componentes desacoplados que interactúan por invocaciones a servicios remotos. Un componente broker es responsable de coordinar la comunicación, como el reenvío de solicitudes, así como también la transmisión de de resultados y excepciones.	Reusabilidad Escalabilidad Interoperabilidad	
Model-View-Controler	Divide una aplicación interactiva en tres componentes. El modelo (Model) contiene la información central y los datos. Las vistas (View) despliegan informaciones al usuario. Los controladores (Controlers) capturan la entrada del usuario. Las vistas y los controladores constituyen la interfaz del usuario.	Funcionalidad Mantenibilidad	Desempeño Portabilidad

Tabla 2 Patrones Arquitectónicos

Con la intención de hacer una comparación clara entre estilo arquitectónico y patrón arquitectónico, la siguiente tabla presenta las diferencias entre estos conceptos, construida a partir del planteamiento de Buschmann et al. [3].

Estilo Arquitectónico	**Patrón Arquitectónico**
Sólo describe el esqueleto estructural general para aplicaciones	Existen en varios rangos de escala, comenzando con patrones que definen la estructura básica de una aplicación
Son independientes del contexto al que puedan ser aplicados	Partiendo de la definición de *patrón*, requieren de la especificación de un contexto del problema
Cada estilo es independiente de los otros	Depende de patrones más pequeños que contiene, patrones con los que interactúa, o de patrones que lo contengan
Expresan técnicas de diseño desde una perspectiva que es independiente de la situación actual de diseño	Expresa un problema recurrente de diseño muy específico, y presenta una solución para él, desde el punto de vista del contexto en el que se presenta
Son una categorización de sistemas	Son soluciones generales a problemas comunes

Tabla 3 Diferencias entre Estilos y Patrones Arquitectónicos

La imposición de ciertos estilos arquitectónicos mejora o disminuye las posibilidades de satisfacción de ciertos atributos de calidad del sistema. Con esto afirman que cada estilo propicia atributos de calidad, y la decisión de implementar alguno de los

existentes depende de los requerimientos de calidad del sistema. De manera similar, plantean el uso de los patrones arquitectónicos para mejorar la calidad del sistema. Al respecto, Buschmann [3] afirman que un criterio importante del éxito de los patrones arquitectónicos, constituye la forma en que estos alcanzan de manera satisfactoria los objetivos de la ingeniería de software. Los patrones soportan el desarrollo, mantenimiento y evolución de sistemas complejos y de gran escala. Buschmann [3] establece diferencias sutiles entre ambos conceptos. De cualquier forma, los estilos y los patrones establecen un vocabulario común, y brindan soporte a los ingenieros para conseguir una solución que haya sido aplicada con éxito anteriormente, ante ciertas situaciones de diseño. Además, su aplicación en el diseño de la arquitectura del sistema es determinante para la satisfacción de los requerimientos de calidad.

1.1.3 Lenguajes de Descripción de Arquitectura

Un lenguaje de descripción de arquitectura (ADL) sirve para describir una arquitectura de software. La definición de la arquitectura usando un ADL incluye la posibilidad de especificar la estructura y el comportamiento.

Bass [2] proponen que el establecimiento de una notación estándar para la representación de arquitecturas, a través de un lenguaje de descripción arquitectónica, permite mejorar la comunicación entre el autor y el lector, logrando tener un medio de entendimiento común, ahorrando tiempo en indagar el significado de los gráficos que representan la arquitectura y sus componentes.

Un ADL también deberá ser capaz de facilitar el modelamiento de sistemas de software que sigan distintos patrones de arquitectura.

Un ADL debe hacerse cargo de todas estas características, aunque existen ciertas características deseables en un ADL según Medvidovic [12], las cuales se ilustran en la siguiente taba.

Elemento	Características
Componentes	Interfaces Tipos Semántica Evolución Propiedades no funcionales
Conectores	Interfaces Tipos Semántica Evolución Propiedades no funcionales
Configuraciones	Composición jerárquica Refinamiento y seguimiento Heterogeneidad Escalabilidad Evolucionabilidad Dinamismo Propiedades no funcionales

Tabla 4 Características de los ADLs

Para realizar la descripción de la arquitectura, los ADLs utilizan esencialmente componentes, que son las piezas físicas de la implementación de un sistema, y los conectores para modelar la interacción. Estos conectores pueden ser implícitos en algunos ADLs o bien modelarse como elementos de primera clase [6], dependiendo del ADL y del patrón de arquitectura que se esté modelando.

Componentes

Se entiende por componentes los bloques de construcción que conforman las partes de un sistema de software. A nivel de lenguajes de programación, pueden ser representados como módulos, clases, objetos o un conjunto de funciones relacionadas

Las características de una componente que un ADL debe ser capaz de modelar son las siguientes:

Interfaces. Es el conjunto de puntos de interacción entre una componente y su entorno. La interfaz especifica los servicios que provee un componente (operaciones, mensajes y variables) y también los servicios que la componente requiere de otras componentes del sistema. Es deseable que los ADLs permitan definir los tipos de las interfaces de las componentes de modo de maximizar la flexibilidad y la reutilización de sus definiciones.

Tipos. Los tipos son abstracciones que encapsulan funcionalidades en bloques reutilizables. Un tipo de componente puede ser instanciado múltiples veces en una sola arquitectura o ser reutilizado en diferentes arquitecturas. Los tipos de componentes pueden ser parametrizados facilitando así aún más su reutilización.

Semántica. La semántica de conectores se define como un modelo de alto nivel del comportamiento de un conector. La semántica expresa funcionalidad a nivel de la aplicación y la especificación de protocolos de interacción.

Evolución. Los ADLs pueden permitir la evolución de componentes definiendo tipos de componentes y permitiendo el refinamiento mediante subtipazo (subtyping).

Propiedades no funcionales. Estas propiedades no tienen relación con la funcionalidad sino con características de calidad de la componente tales como seguridad, rendimiento y portabilidad. La mayoría de los ADLs tiene a las componentes como sus principales elementos de especificación; todos modelan interfaces y distinguen entre tipos e

instancias de componentes. Por otro lado, casi ningún ADL soporta evolución ni especificación de propiedades no funcionales.

Conectores

Los conectores definen la interacción entre los componentes. Cada conector provee de una forma para que una colección de puertos esté en contacto y define lógicamente el protocolo a través del cual un conjunto de componentes puede interactuar. Los puertos definen los puntos de interacción de los conectores.

Las características principales que debe tener un conector son las siguientes:

Interfaz. Es el conjunto de puntos de interacción entre un conector y una componente u otro conector. Las interfaces de conectores permiten una conectividad apropiada entre componentes y su interacción en una arquitectura. En general, cuando un ADL soporta conectores como entidades de primera clase [6], entonces soporta explícitamente también la especificación de las interfaces de estos conectores.

Tipos. Los tipos de conectores son abstracciones que encapsulan la comunicación, coordinación y decisiones de mediación de componentes. Interacciones de nivel de arquitecturas son caracterizadas por protocolos complejos identificados para diferentes patrones de arquitectura [10].

Semántica. La semántica de conectores se define como un modelo de alto nivel del comportamiento de un conector. La semántica expresa funcionalidad a nivel de la aplicación y la especificación de protocolos de interacción.

Evolución. La evolución de un conector se refiere a la modificación de las propiedades de un conector, o sea, de la modificación de su interfaz, semántica, o restricciones entre las dos.

Propiedades no funcionales. Las propiedades no funcionales de los conectores no se pueden derivar completamente de la especificación de su semántica. Representan requerimientos para implementar correctamente conectores. Permiten simulaciones de comportamiento en tiempo de ejecución, reforzamiento de restricciones, etc.

Configuraciones

Una configuración o topología es un grafo de componentes y conectores conectados que describen la estructura de la arquitectura. Las características de nivel de configuración se agrupan en tres categorías generales:

- Calidad de la descripción de la configuración: entendimiento, composición, refinamiento y seguimiento, heterogeneidad.
- Calidad de la descripción del sistema: heterogeneidad, escalabilidad, evolución y dinamismo.
- Propiedades de la descripción del sistema: dinamismo, restricciones y propiedades no funcionales.

Las características principales que debe tener una configuración son:

Composición jerárquica. Es un mecanismo que permite que una arquitectura sea definida con distintos niveles de detalle. Estructuras y comportamiento complejos pueden ser representados explícitamente o abstractamente como una composición de componentes y conectores simples.

Refinamiento y seguimiento. Los ADLs deben proveer refinamiento de arquitecturas en sistemas ejecutables y seguimiento de los cambios a través de los distintos niveles.

Heterogeneidad. Las arquitecturas de software deben facilitar el desarrollo de sistemas de gran escala mediante la existencia de componentes y conectores de distinta granularidad, posiblemente especificadas en diferentes lenguajes de modelamiento e implementadas en diferentes lenguajes de programación.

Escalabilidad. Los ADLs deben soportar la especificación y el desarrollo de sistemas de gran escala que pueden crecer en el futuro.

Evolucionabilidad. Una arquitectura evoluciona para reflejar y permitir la evolución de familias de sistemas de software.

Dinamismo. Se refiere a modificar la arquitectura y reflejar esas modificaciones en el sistema mientras éste se ejecuta. Es importante el soporte de dinamismo en sistema como control de tráfico, en donde la disponibilidad y seguridad son críticas.

Propiedades no funcionales. Algunas de las propiedades no funcionales están a nivel del sistema más que a nivel de componentes y conectores individuales. Propiedades no funcionales a nivel de sistema son necesarias para seleccionar apropiadamente componentes y conectores, realizar análisis y reforzar restricciones, entre otras.

1.1.4 Beneficios de Priorizar la Arquitectura

Evaluación de la solución mediante demostraciones tangibles de sus capacidades desde fases muy tempranas de desarrollo.

La principal razón de priorizar la arquitectura es demostrar lo más pronto posible que la solución (diseño) es correcta para resolver los objetivos (los requisitos). La arquitectura debe dar garantías de que la solución es realizable dentro de las restricciones de tiempo, personal y presupuesto, es decir, que el proyecto es viable.

Atención temprana a riesgos relacionados con la arquitectura que, generalmente, coinciden con aquellos que pueden conducir a mayores daños.

La gestión de riesgos relacionados con el proceso de desarrollo es otro de los principios fundamentales de los métodos actuales de desarrollo de software. Significa identificar las dudas e incertidumbres sobre qué hacer o cómo hacerlo y trazar planes para investigarlas y resolverlas.

Centrarse en la creación de un prototipo de arquitectura implica gestionar un conjunto de riesgo y resolverlos. De esta manera se eliminan las mayores causas potenciales de errores.

Propicia la construcción incremental del software (integración temprana) y las correspondientes actividades de verificación.

Otro beneficio de priorizar la arquitectura es que propicia una estrategia incremental de construcción del software, como forma de aplicación del principio “divide y vencerás”. La integración temprana permite dosificar los esfuerzos de realización de pruebas de integración.

Propicia el desarrollo orientado a demostraciones periódicas de productos funcionales.

Este procedimiento garantiza que en cada instante del proceso de construcción del software exista un prototipo funcional validado. Así, cada prototipo parcial permitirá realizar demostraciones periódicas a clientes, usuarios finales con el objetivo de detectar las desviaciones entre el software implementado y el resultado que esperan los clientes o usuarios finales.

Interfaces correctas permiten una cooperación eficiente entre diseñadores e implementadores.

La interfaz de un módulo de software se refiere a cómo interactúa este módulo con el resto del sistema (cómo se identifica, cómo se activa, qué datos necesita, y qué resultados devuelve.). Especificar correctamente estas interfaces propicia la cooperación y el paralelismo en la implementación.

1.1.5 Arquitecturas Más Comunes

Hoy en día, por lo general basta seleccionar una arquitectura conocida atendiendo a las ventajas e inconvenientes que ésta pueda proporcionar a la escena de trabajo.

Las arquitecturas más universales son:

- Cliente/Servidor: el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.
- Arquitectura de tres niveles: Especialización de la arquitectura cliente-servidor donde la carga se divide en tres partes con un reparto claro de funciones: una capa para la presentación, otra para el cálculo y otra para el almacenamiento. Una capa solamente tiene relación con la siguiente.

Otras arquitecturas menos conocidas son:

En pipeline.

Entre pares

En pizarra

Orientada a eventos.

Máquinas virtuales.

Arquitectura Cliente/Servidor

Esta arquitectura es definida por IBM como una tecnología que proporciona al usuario el acceso a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo.

Es una arquitectura de procesamientos cooperativa donde uno de los componentes solicita servicios a otro. Consiste en que un programa (cliente) realiza peticiones a otro (servidor), que le proporciona la respuesta. En ella se encuentra distribuida la capacidad de proceso entre el cliente y el servidor, aunque las ventajas de tipo organizativo son más importantes, debido a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita el diseño del sistema.

Qué es un Cliente

El cliente es el que realiza una petición solicitando un servicio, esta solicitud puede convertirse en varias solicitudes de trabajo mediante redes LAN o WAN. La ubicación de los datos o las aplicaciones solicitadas es totalmente transparente para el cliente.

Qué es un Servidor

Un servidor es uno o varios recursos dedicados a responder las solicitudes de los clientes. Pueden estar conectados a los clientes mediante redes LAN o WAN con el fin de proveer múltiples servicios a los clientes como acceso a una base de datos, procesamiento de imágenes, fax, etc.

La separación lógica entre el cliente y el servidor permite que el servidor no se ejecute necesariamente sobre una misma máquina ni sea necesariamente un sólo programa.

Ventajas de la Arquitectura Cliente / Servidor**.**

1. Centralización del control: los recursos, la integridad y el acceso a los datos son controlados por el servidor, para prevenir que un programa cliente no autorizado o defectuoso pueda afectar al sistema.
2. Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado.
3. Se reduce el tráfico de red considerablemente, el cliente se conecta al servidor utilizando un protocolo de alto nivel.

Arquitectura de tres niveles

Esta arquitectura consiste en una programación por capa que tiene como objetivo separar la lógica de negocio de la lógica de diseño, su ventaja principal es que el desarrollo se puede realizar en varios niveles, permitiendo que los cambios se hagan en un nivel requerido no afecte a los otros niveles. Esta arquitectura permite distribuir el trabajo en niveles, así, cada grupo de trabajo estará aislado de los demás niveles.

En la actualidad se suele utilizar éste tipo de arquitectura para el diseño de sistemas informáticos, donde a cada nivel se le asigna una misión sencilla, facilitando que el diseño de la arquitectura escalable (puedan ampliarse con facilidad).

Capas

Capa de presentación:

Es la intermediaria entre el usuario y el sistema, es una interfaz o ventana que captura la información del usuario realizando un filtrado previo para comprobar que no existen errores de formato y le comunica la información solicitada. Esta capa se comunica únicamente con la capa de negocio.

Capa de negocio:

En esta capa es donde se encuentran los programas que se ejecutan y se establecen las reglas que el sistema debe cumplir. Esta capa se comunica con la capa de presentación para recibir las solicitudes y mostrar los resultados, y con la capa datos, para solicitar al gestor de base de datos acciones cómo almacenar o recuperar datos de él.

Capa de datos:

Está compuesta por uno o varios gestores de bases de datos que realiza el almacenamiento de toda la información, recibe solicitudes de almacenamiento o recuperación de datos desde la capa de negocio.

Físicamente estas capas pueden encontrarse en un mismo ordenador, aunque lo más usual es que existan varios ordenadores clientes, que es donde reside la capa de presentación. La capa de negocio y de datos puede separarse en múltiples ordenadores, permitiendo que la base de datos se pueda establecer en varios ordenadores en caso de que su tamaño y complejidad aumente, y recibirán las

peticiones del ordenador en que se encuentre la capa de negocio. De igual forma ocurre en caso de que la complejidad sea en la capa de negocio.

Cuando los sistemas son muy complejos, por lo general se establece una serie de ordenadores sobre los cuales corre la capa de datos, y otra serie de ordenadores sobre los cuales corre la base de datos.

Es necesario tener en cuenta que en una arquitectura de tres niveles los términos "***capas***" y "***niveles***" no tienen el mismo significado.

Cuando se utiliza el término "***capa***" se hace referencia a cómo una solución se encuentra dividida desde el punto de vista lógica.

Presentación/ Lógica de Negocio/ Datos.

El término "***nivel***", corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física, por ejemplo:

- Cuando una solución de tres capas (presentación, lógica, datos) se encuentra en un mismo ordenador (presentación+lógica+datos), se dice, que la arquitectura de solución es de tres capas y un nivel.
- Cuando la solución de tres capas (presentación, lógica, datos) se encuentran en dos ordenadores (presentación+lógica, lógica+datos), se dice, que la arquitectura de solución es de tres capas y dos niveles.
- Cuando la solución de tres capas (presentación, lógica, datos) se encuentran en ordenadores diferentes (presentación, lógica, datos), la arquitectura de solución es de tres capas y tres niveles.

Cuando se habla de una arquitectura en tres capas, o una arquitectura cliente-servidor, u orientada a servicios, implícitamente se hace referencia a un campo de posibilidades.

1.2 Metodologías y Tecnologías Actuales

La creciente informatización de los procesos productivos y sociales ha traído consigo que las organizaciones y empresas requieran cada vez más de software confiable y con alta calidad, tanto en su desarrollo como en su mantenimiento. Es por ello que en los últimos años se han venido publicando estándares, notaciones y metodologías que establecen buenas prácticas para los procesos de desarrollo de software.

1.2.1 Metodología de Desarrollo de Software

Todo de desarrollo de software es riesgoso y difícil de controlar, pero si este proceso no es guiado por una metodología, obtenemos como resultado clientes insatisfechos con el producto.

La metodología de desarrollo es "un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a realizar nuevo software".

Sintetizando lo anterior, una metodología "representa el camino para desarrollar software de una manera sistemática". Las metodologías persiguen tres necesidades principales:

- Mejores aplicaciones, conducentes a una mejor calidad.
- Un proceso de desarrollo controlado.
- Un proceso normalizado en una organización, no dependiente del personal.

Los procesos se descomponen hasta el nivel de tareas o actividades elementales, donde cada tarea está identificada por un procedimiento que define la forma de llevarla a cabo.

Capitulo 1. Fundamentación Teórica

En ocasiones, el diseño de un software se realiza de manera rígida con los requerimientos que el cliente solicitó, provocando un difícil mantenimiento si el usuario final solicita un cambio durante la fase de prueba. Esto se debe a no usar o aplicar una metodología adecuada, debido a la falta de conocimiento sobre este tema. Algunas de las metodologías que se pueden utilizar para guiar un proceso de desarrollo de software son:

- La Metodología Rational Unified Process (**RUP**) es más adaptable para proyectos de largo plazo.
- La Metodología Extreme Programing (**XP**) en cambio, se recomienda para proyectos de corto plazo.
- La Microsoft Solution Framework (**MSF**) se adapta a proyectos de cualquier dimensión y de cualquier tecnología.

La metodología **RUP** divide en 4 fases el desarrollo del software:

Inicio, El objetivo en esta etapa es determinar la visión del proyecto.

Elaboración, En esta etapa el objetivo es determinar la arquitectura óptima.

Construcción, En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.

Transición, El objetivo es llegar a obtener una versión beta del proyecto.

Cada una de estas etapas fases es desarrollada mediante el ciclo de iteraciones.

Cada ciclo de vida que se desarrolla por iteraciones es llevado bajo dos disciplinas.

Disciplina de Desarrollo

- Ingeniería de Negocios: Entendiendo las necesidades del negocio.
- Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.

- Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
- Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todos los solicitados está presente.

Disciplina de Soporte

Configuración y administración del cambio: Guardando todas las versiones del proyecto.

- Administrando el proyecto: Administrando horarios y recursos.
- Ambiente: Administrando el ambiente de desarrollo.
- Distribución: Hacer todo lo necesario para la salida del proyecto.

Los *elementos* del RUP son:

Actividades, Son los procesos que se llegan a determinar en cada iteración.

Trabajadores, Vienen hacer las personas o gentes involucradas en cada proceso.

Artefactos, Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

La metodología **XP** es una de la más exitosa para el desarrollo de proyectos a corto plazo y consisten en una implementación rápida

Características de XP, la metodología se basa en:

Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir.

Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

Programación en pares: consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

Lo fundamental en este tipo de metodología es:

La comunicación, entre los usuarios y los desarrolladores.

La simplicidad, al desarrollar y codificar los módulos del sistema.

La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

La metodología **MSF** es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

Esta metodología tiene las siguientes características:

- **Adaptable:** es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- **Escalable:** puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.

- **Flexible:** es utilizada en el ambiente de desarrollo de cualquier cliente.
- **Tecnología Agnóstica:** porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

1.2.2 Tecnologías para el Desarrollo de Aplicaciones

Sistemas Gestores de Contenido (**CMS**)

Los sistemas gestión de contenidos (Content Management Systems o CMS) son software que principalmente se utilizan para facilitar la gestión de Web, ya sea en Internet o una Intranet, por lo que también son conocidos como gestores de contenido Web (Web Content Management o WCM) [11].

La ***funcionalidad*** de los CMS puede agruparse en las siguientes categorías [11].

Creación de contenido: Un CMS proporciona herramientas que permiten la creación de páginas Web mediante un editor WYSIWYG (What You See Is What You Get), donde el usuario puede ver los resultando mientras escribe. También cuenta con otras herramientas como, editores de documentos XML, aplicaciones ofimáticas, etc.

Gestión de contenido: Los documentos y las informaciones referentes a la Web, como los datos relativos a los documentos (versiones, fecha de publicación, autor, etc.) son almacenados en una base de datos central.

Publicación: Una página aprobada es publicada automáticamente en la fecha de publicación establecida, y cuando caduca se archiva para futuras referencias. En su publicación se aplica el patrón de diseño definido para toda la Web o para la sección a la que pertenece, de forma que el resultado final es un sitio Web con un aspecto consistente en todas sus páginas.

Presentación: Un CMS puede gestionar automáticamente la accesibilidad del sitio Web con soporte de normas internacionales de accesibilidad, y adaptarse a las necesidades o preferencias de los usuarios. Proporciona compatibilidad con los diferentes navegadores disponibles en todas las plataformas (Windows, Linux, Mac, Palm, etc.) y su capacidad de internacionalización le permite adaptarse al idioma.

Algunas razones para usar CMS

Control de Acceso: Controlar el acceso a un sitio Web no solo consiste en permitir la entrada, sino en comprobar los diferentes permisos asignados a cada grupo e individuo en cada área de la Web.

Consistencia de la Web: La consistencia en un sitio Web no quiere decir que todas las páginas son iguales, sino que existe un orden (visual). Un usuario se percata fácilmente cuando las páginas de un sitio son diferentes por su aspecto, la disposición de los objetos o por los cambios en la forma de navegar. Estas diferencias desorientan al usuario y dan a entender que el sitio no fue diseñado por profesionales. Para evitar esto, los CMS pueden aplicar un mismo estilo a todas las páginas haciendo uso de los CSS y aplicar una misma estructura a las páginas mediante patrones.

Reutilización de objetos o componentes: Un CMS permite recuperar y reutilizar las páginas, documentos y en general cualquier objeto publicado o almacenado.

Servidores de Aplicaciones

Se denomina servidor de aplicaciones a un servidor en una red de ordenadores que ejecuta varios servicios de software. Es el encargado de gestionar la mayoría o la totalidad de las funciones lógicas del negocio y de acceso a los datos de la aplicación y su utilización permite la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

La mayoría de los servidores de aplicaciones son desarrollados sobre el lenguaje de programación Java debido al éxito alcanzado por este lenguaje. En la actualidad, generalmente el término de *servidor de aplicación* hace referencia a un servidor de aplicación J2EE.

Algunos de ellos son, WebSphere (IBM), Oracle Application Server (Oracle Corporation) y WebLogic (BEA), estos se encuentran entre los servidores privativos más conocidos. También tenemos al JOnAS, primer servidor libre que logró una certificación oficial de compatibilidad con J2EE y fue desarrollado por el consorcio ObjectWeb.

Por los beneficios que aportan estos tipos servidores, su aplicación se ha ampliado a otros productos que no pertenecen a J2EE, por ejemplo el Internet Information Server en .NET de Microsoft. Además se puede obtener servidores de aplicación de código abierto perteneciente a otros proveedores como el Base4 Server y Zope.

Sistemas Gestores de Base de datos (**SGBD**)

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que permiten crear y mantener una Base de Datos, asegurando su integridad, confidencialidad y seguridad. Presenta una interfaz, mediante la cual el usuario puede comunicarse con el sistema físico y realizar operaciones como almacenar o recuperar datos de ella. Las principales funciones que debe cumplir un SGBD son la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias.

Producto a la fácil manipulación y los beneficios que estos sistemas permiten se han desarrollados diversos tipos de estos sistemas, los cuales se han clasificado en grupo como los SGBD Libres (PostgreSQL, MySQL, Firebrid, SQLite, Sybase ASE (edición gratuita para Linux)) y los SGBD Comerciales.

Servidor Web

Un servidor Web es un programa que implementa el protocolo HTTP (*hypertext transfer protocol*), protocolo diseñado para transferir lo que llamamos hipertextos, páginas Web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

Un servidor Web se encarga de mantenerse a la espera de *peticiones HTTP* llevada a cabo por un cliente HTTP que solemos conocer como navegador. El navegador realiza una petición al servidor y éste le responde con el contenido que el cliente solicita. A modo de ejemplo, al teclear *www.google.es* en nuestro navegador, éste realiza una petición HTTP al servidor de dicha dirección. El servidor responde al cliente enviando el código HTML de la página; el cliente, una vez recibido el código, lo interpreta y lo muestra en pantalla.

En la actualidad el mundo está dominado por dos grupos de servidores Web, por un lado tenemos a Internet Information Server (IIE) perteneciente a Microsoft, y por el otro a Apache, un proyecto libre, gratuito y de código abierto de la Fundación Apache.

Apache es el servidor Web más utilizado en la esfera de la informática, su escala alcanza casi el 65% de los servicios Web usados en el mundo por sus características, es un proyecto libre, gratuito, que ha demostrado su estabilidad, rendimiento y solidez. Es respaldado por una amplia comunidad de usuarios disponibles para su mantenimiento.

1.3 Conclusiones

En este capitulo se han abordados temas relacionados con el objeto de investigación, tendencias, metodologías y las tecnologías más utilizadas en el mundo informático, y que son examinadas durante el diseño de la arquitectura de software de aplicaciones Web, con el objetivo de proporcionar una solución que permita optimizar la gestión de información y la interoperatividad entre los servicios

Capitulo 2. Representando la Arquitectura de Software.

2.1 Arquitectura propuesta

Para el desarrollo del sistema de gestión de información Intranet 2, se propone el uso de una Arquitectura Orientada a Servicios (SOA) y las siguientes tecnologías

2.1.1Tecnologías para el desarrollo del sistema

CMS Plone: Plone es un sistema de gestión de contenido (SGC o CMS) construido sobre el servidor de aplicaciones Zope. Actualmente es uno de los CMS más utilizados en el mundo, debido a su extrema flexibilidad, facilidad de instalación y configuración. Su arquitectura permite que el sistema sea fácilmente mejorado y se le adiciones nuevos tipos de contenidos, característica que sitúa a Plone en otro nivel en lo que concierne a sus competidores. Cuenta actualmente con varios desarrolladores, distribuidos por todo el mundo que se encargan de mantener las actualizaciones.

2.1.1.1 Servidor de Aplicación Zope:

Zope es un servidor de aplicaciones Web escrito en el lenguaje de programación Python que puede ser manejado casi totalmente usando una interfaz de usuario basada en páginas Web.

Un sitio Web de Zope está compuesto de objetos en lugar de archivos, como es usual con la mayoría de los otros sistemas de servidores Web. Las ventajas de usar objetos en lugar de archivos son [15]:

- Combinan el comportamiento y los datos en una forma más natural que los archivos de texto plano.

- Alientan el uso de componentes estándares que se ocupan de una parte particular de las que forman una aplicación Web, permitiendo flexibilidad y buena descomposición.
- Posibilitan procesos automáticos de gestión de información.

Lo más característico de Zope es su base de datos orientada a objetos, llamada ZODB o Zope Object Database, aunque se reemplazará por Apache. Esa capacidad de trabajar en Linux, Windows y otros sistemas operacionales proviene del hecho de que tanto el Zope como el Plone son escritos en Python que, como el Java, es un lenguaje que usa un interpretador específico para cada combinación de procesador y sistema operacional para trabajar [15].

¿Por qué usar el servidor de aplicaciones Zope?

Zope puede ayudarle potencialmente a confeccionar sitios Web con menor costo y tiempo de desarrollo que se puede obtener utilizando otro servidor de aplicación. Esta reclamación es sostenida por varias características de Zope:

- Zope está libre de costos y es distribuido bajo una licencia de código abierto. Existen muchos servidores de aplicación comerciales que son relativamente caros. [1].
- Zope permite que equipos de desarrolladores colaboren con eficacia. Los ambientes de colaboración requieren de herramientas que permitan a los usuarios trabajar sin interferir el uno con el otro, además Zope tiene Versiones, Historia y otras herramientas para ayudar las personas a trabajar sin peligro juntos, permitiendo la recuperación de errores. Muchos otros servidores de aplicación no proporcionan estos tipos de características. [1].
- Zope corre en las plataformas de sistemas operativos más populares: Linux, Windows NT/2000/XP, Solaris, FreeBSD, NetBSD, OpenBSD, y Mac OS X. Zope incluso corre en Windows 98/ME (recomendado sólo para objetivos de desarrollo).

Muchas otras plataformas de servidores de aplicación requieren que se escoja un sistema operativo según la elección de su licenciador. [1].

- Puede ser ampliado usando el lenguaje interpretado Python. El Python es popular y fácil para aprender, y ello promueve el desarrollo rápido. Muchas bibliotecas están disponibles para Python, las cuales se pueden usar para crear su propia aplicación. Muchos otros servidores de aplicación deben ser ampliados usando lenguajes compilados como Java, que disminuyen la velocidad de desarrollo [1].

2.1.1.2 Sistema Gestor de Base de Datos PostgreSQL:

PostgreSQL Proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales como DB2 u Oracle. Corre en la mayoría de los Sistemas Operativos más utilizados incluyendo, Linux, varias versiones de UNIX, BeOS y Windows. Es un SGBD que brinda la posibilidad de herencia de tablas y acceso concurrente multiversión (no se bloquean las tablas, ni siquiera las filas, cuando un procesador escribe). [7].

Algunas de estás características son:

- **DBMS Objeto-Relacional:** PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multiversión, soporte multiusuario, transacciones, optimización de consultas, herencia, y arreglos.
- **Altamente_Extensible:** PostgreSQL soporta operadores funcionales, métodos de acceso y tipos de datos definidos por el usuario.
- **Soporte_SQL_Comprensivo:** PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.
- **Integridad Referencial:** PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

- **API Flexible:** La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.
- **Lenguajes Procedurales:** PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.
- **MVCC:** MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios, que permite realizar varias acciones a la base de datos. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.
- **Cliente/Servidor:** PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.
- **Write Ahead Logging (WAL):** La característica de PostgreSQL conocida como *Write Ahead Logging* incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos se caiga, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos. Esto puede ser enormemente beneficioso en el caso de caída, ya que cualesquiera cambios que no fueron escritos en la base de datos pueden ser recuperados usando el dato que fue previamente registrado. Una vez el sistema ha quedado restaurado, un usuario puede continuar trabajando desde el punto en que lo dejó cuando cayó la base de datos.

2.1.1.3 Servidor Web Apache: Es un servidor Web, que por su robustez, configurabilidad y estabilidad hacen que cada vez más millones de servidores reiteren su confianza en este programa y su licencia es descendiente de la BSD, que permite la modificación del código fuente. Ha alcanzado gran popularidad en ámbitos empresariales debido, entre otras características a:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita de código fuente abierta. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto.
- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar sus capacidades mediante la integración o la construcción de módulos.
- Apache te permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.

La utilización de Apache en lugar del servidor Web proporcionado por el propio Zope (Medusa) se basa en razones de peso. Apache es un producto con muchos años de experiencia. La ventaja principal es que desde el servidor web poder servir todo tipo de contenido: desde las páginas estáticas de toda la vida, pasando por cgi's y scripts en php, y por supuesto, contenido de Zope. Gracias al producto VirtualHostMonster, el servidor web Medusa de Zope, y Apache, podemos a mapear las URLs de nuestro dominio.

Una representación visual de esta configuración sería la siguiente:

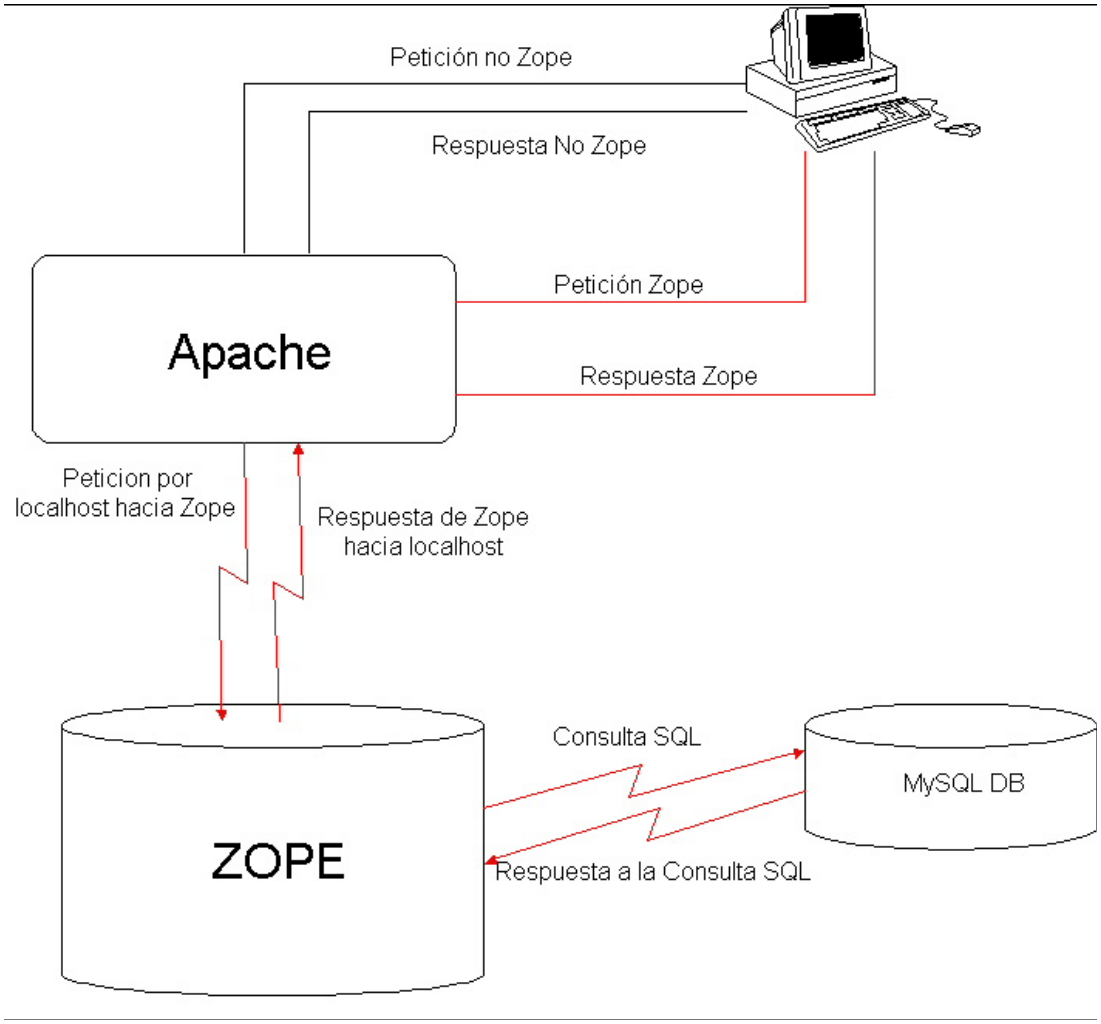

Fig. 1 Representación Visual

2.1.2 Requerimientos de Software

Cliente:

- Cualquier sistema operativo con interfaz gráfica y red.
- Navegador Web

Servidor:

- GNU/ Linux, cualquier distribución (Debian, Ubuntu, Kde, etc.).
- Servidor de Base de Datos PostgreSQL en reemplazo de Zope DB.
- Apache para reemplazar a Zope Web Server.

2.1.3 Requerimientos de Hardware

Para el desarrollo:

Pentium IV o superior.

256 MB de memoria RAM o superior.

Disco duro con capacidad de 80 GB o superior.

Para la explotación:

Clientes:

Pentium IV o superior

240 MB de memoria RAM o superior

Servidor:

Pentium IV o superior

4 GB de memoria RAM

Disco duro con capacidad de 130 GB

2.2 ¿Por qué una Arquitectura Orientada a Servicios? Justificación.

Actualmente, los negocios exigen la creación de aplicaciones más complejas, con menor tiempo y presupuesto de desarrollo. Generalmente, estas construcciones requieren de funcionalidades ya implementadas, dejándoles dos opciones a los arquitectos de soluciones.

1. Tratar de reutilizar las funcionalidades existentes en otros sistemas, lo que constituye una tarea sumamente difícil ya que éstas no fueron diseñadas para integrarse sobre plataformas y/o tecnologías incompatibles entre ellas. Además, en caso de lograr la integración, es muy probable afectar un sistema que está funcionando sin problemas.

2. Reimplementar la funcionalidad requerida (reinventar la rueda). Una opción más fácil y segura, pero que implica mayor tiempo de desarrollo.

La segunda opción es la más usada, aunque no es la más aceptada a largo plazo. Esta decisión trae como resultado:

- Dificultad de migración de lo sistemas internos, porque pueden existir varias conexiones desde sistemas que dependen de éstos para su funcionamiento.
- Al no existir una estrategia de integración de aplicaciones, se generan múltiples puntos de falla que pueden detener fácilmente el funcionamiento de todos los sistemas.
- Generalmente, estos modelos no presentan alta escalabilidad, imposibilitando una rápida respuesta los a cambios.

¿Cómo crear sistemas altamente escalables? Para lograr esto, es necesario crear sistemas donde sus componentes sean débilmente acoplados y que permita la reutilización de los mismos. Principalmente en la industria, la Arquitectura Orientada a Servicios (SOA) ha ocupado un papel fundamental en el desarrollo de aplicaciones, ya que promete ser lo suficientemente flexible para afrontar los cambios tanto en el ambiente del negocio como en la infraestructura tecnológica y proporcionar la agilidad de los negocios que las compañías han anhelado.

SOA ha surgido con la mejor manera de afrontar el desafío de hacer más con menos recursos. Es un estilo que permite la reutilización e integración de componentes mucho más fácil, ayudando a reducir el tiempo de desarrollo.

En un entorno SOA, los clientes pueden acceder a los recursos en forma de servicios que se encuentran disponibles mediante los nodos de una red, estos son independientes y su acceso es estandarizado.

Las SOAs están compuestas por un conjunto de servicios de aplicación débilmente conectados y altamente ínter operables, independientes de la plataforma subyacente y del lenguaje de programación.

En los últimos tiempos se ha hablado mucho de SOA y cada vez más las empresas están concientizadas de la necesidad de incorporar SOA a su infraestructura de tecnologías de información. Esto se debe principalmente a las características que presenta esta arquitectura, y que proporcionan numerosas ventajas en le proceso de satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad.

2.2.1 Arquitectura Orientada a Servicios

Una arquitectura orientada a los servicios (SOA) es un método para construir una infraestructura de tecnología de información (TI) a partir de componentes acoplados de manera flexible, denominados “servicios”, que desempeñan funciones específicas. Las aplicaciones compuestas son un elemento clave de un entorno SOA. Estas aplicaciones se crean invocando y orquestando múltiples servicios, eventos y modelos de tal manera que colectivamente desempeñan una función empresarial de alto nivel. Esta funcionalidad incrementa la agilidad empresarial al permitir a los departamentos de TI reutilizar componentes que ya han sido probados en el proceso de producción y que poseen características de escalabilidad y de calidad de servicio conocidas. La

reutilización de componentes contribuye a reducir el tiempo de lanzamiento al mercado de nuevos productos y los costes de desarrollo del departamento de TI.

2.2.2 Conceptos en SOA

Servicios: Es una función auto contenida, que acepta una o varias llamadas y devuelve una o varias respuestas a través de una interfaz bien definida. Los servicios son independientes de otros procesos y pueden llevar a cabo unidades discretas de trabajo como serían: editar y procesar una transacción.

Orquestación: Coordinar y secuenciar los servicios, además de proveer la lógica adicional para procesar los datos.

Sin estado: En SOA los servicios son totalmente independientes, reciben en la llamada la información necesaria para brindar una respuesta.

Como los servicios son sin estado, éstos pueden ser orquestados en numerosas secuencias para realizar la lógica del negocio.

Proveedor: La función que brinda un servicio en respuesta a una llamada o petición desde un consumidor.

Consumidor: La función que consume el resultado del servicio provisto por un proveedor.

2.2.3 SOA como estilo Arquitectónico

La primera decisión del arquitecto es identificar qué estilo arquitectónico debe utilizar para construir una determinada solución ante un conjunto de requisitos, fundamentalmente no funcionales. El arquitecto debe decidir que relación de componentes, conectores, restricciones y configuraciones va a tener la aplicación; si va

a ser una aplicación basada en flujo de datos, basadas en llamada y retorno, si la arquitectura es en capa o basada en componentes, etc. Existen diferentes nomenclaturas de los estilos arquitectónicos, donde un estilo es un tipo de arquitectura dentro de un repertorio relativamente conocido de un conjunto de estilos básicos reconocidos universales.

Un estilo arquitectónico se define como las famosas cuatro C, Componentes, Conectores, Configuración y Constraint (Restricciones).

Los elementos que corresponden al estilo de arquitectura orienta a servicios son:

Componentes: servicio.

Conectores: En las primeras fases de SOA, incluso de los servicios Web se estimaba que el modelo de conector era una invocación de procedimiento, en la actualidad se estima que es más adecuado describir la forma de entenderse los componentes, como paso de mensaje, el mensaje puede ser una invocación de una función determinada.

Configuración: Este estilo arquitectónico es distribuido, o sea, la plataforma sobre la cual se está estableciendo (SOA) y las máquinas sobre la corre, puede ser una o múltiples con distintas tecnologías, incluso con software de diferentes fabricantes.

Constraint (restricciones): Producto a las libertades de tecnologías y software que SOA proporciona, hace que las restricciones sean de bajo acoplamiento, garantizando la independencia de modelo de implementación de cada una de las piezas participantes, la independencia de plataforma y permitiendo que el transporte y el protocolo que son necesarios para el entendimiento de los distintos componentes a través de sus conectores sean fijados por acuerdo de la industria.

2.2.4 Diseño de SOA

El análisis y diseño orientado a servicios o SOAD (Service-oriented analysis and design) es la identificación de la metodología que se utiliza para modelar y diseñar aplicaciones SOA. Es una metodología para el diseño de aplicaciones muy ágiles en un modelo cliente/servidor, donde se puede aislar la implementación del proceso actual, de manera que un servicio brindado puede ser cambiado sin afectar al cliente.

Para obtener un exitoso proyecto SOA, los desarrolladores deben de auto orientarse en la idea de elaborar servicios comunes, que son secuenciados por los clientes para implementar los procesos de negocio. Los servicios deben ser diseñados para controlar internamente la transaccionalidad de sus propias operaciones. No es recomendable que una transacción traspase los límites de un servicio.

La implementación ideal de un servicio exige resolver algunos inconvenientes técnicos inherentes a su modelo:

- El tiempo de solicitud no son despreciables, gracias a la comunicación de la red, el tamaño de los mensajes, etc. Esto necesariamente implica la utilización de mensajería confiable.
- La respuesta de los servicios se ve afectada directamente por aspectos externos como problemas en la red, configuración, etc. Estos se deben tener en cuenta en el diseño, desarrollándose los mecanismos de contingencia que eviten la para de las aplicaciones y servicios que dependen de él.
- Debe manejar comunicaciones no seguras, mensajes impredecibles, reintentos, mensajes fuera de secuencia, etc.

El servicio debe publicar una interfaz (por ejemplo, utilizando WSDL o proxies) fácilmente localizable en la red. Esta interfaz debe servir como un contrato de servicio, donde se describe cada una de las funciones que provee, e incluso los niveles de

prestación de servicio (SLA). Esta interfaz debe estar claramente documentada de manera que sea muy fácil implementar una conexión.

Un diseño exitoso de una arquitectura orientada a servicios debe estar basado en una plataforma de mensajería confiable, que aísle de la implementación funcional muchos de los problemas anteriormente mencionados. Algunas de las responsabilidades de un mecanismo así incluyen:

- Entrega garantizada de mensajes.
- Enrutamiento de peticiones a un servicio disponible.
- Seguridad del contenido de los mensajes.
- Calidad del Servicio.

Entre más de estas características tenga la tecnología elegida, menos problemas tendrá la solución final en operación.

Una comunicación basada en mensajes por lo general implica que no existen sesiones. Por lo tanto, los clientes no guardan estado en el servicio (mayor escalabilidad), y la autenticación se debe dar a nivel de cada mensaje. Por último, los servicios deben ser diseñados para controlar internamente la transaccionalidad de sus propias operaciones. No es recomendable que una transacción traspase los límites de un servicio.

2.2.5 Componentes en SOA

- Servicios: Entidades lógicas - Contratos definidos por una o más interfaces públicas.
- Proveedor de servicios (service provider): Entidad de software que implementa una especificación de servicio.

- Consumidor de servicios (service consumer): Entidad de software que llama a un proveedor de servicios. Tradicionalmente se le llama "cliente". Puede ser una aplicación final u otro servicio.
- Localizador de servicios (service locator): Tipo específico de proveedor de servicios que actúa como registros y permite buscar interfaces de proveedores de servicios y sus ubicaciones.
- Corredor de servicios (Service broker): Tipo específico de proveedor de servicios que puede pasar requerimientos de servicios a otros proveedores de servicios.

Principales componentes

Servicios

Un servicio de negocio es un componente reutilizable de software, con significado funcional completo, y que está compuesto por:

Contrato: especificación de la finalidad, funcionalidad, forma de uso y restricciones del servicio.

Interfaz: mecanismo de exposición del servicio a los usuarios.

Implementación: debe contener la lógica o el acceso a datos.

Repositorio de servicios

Un repositorio de servicios proporciona facilidades para descubrir servicios y adquirir la información necesaria para su uso, en particular fuera del alcance temporal y funcional del proyecto en el que se crearon. Además de la propia información de contrato, los repositorios pueden proporcionar información acerca de:

- Localización.
- Personas de contacto.
- Restricciones técnicas**.**

Bus de servicios

El bus de servicios es el elemento de las arquitecturas SOA que conecta los servicios con sus consumidores y que proporciona:

- Conectividad: el propósito principal de un bus de servicios es interconectar a los participantes de una arquitectura SOA.
- Soporte a la heterogeneidad de tecnologías: debe ser capaz de conectar a participantes basados en distintos lenguajes de programación, sistemas operativos, entornos de ejecución y protocolos de comunicación.
- Soporte a la heterogeneidad de paradigmas de comunicación: debe ser capaz de mantener distintos modos de comunicación (por ejemplo comunicaciones sincrónicas y asíncronas).

Consumidores de servicios

Definimos consumidores de servicios como aquellos elementos de una arquitectura SOA que:

- Pueden descubrir servicios a través de un repositorio.
- Realizan llamadas a los mismos de acuerdo al contrato y a través de la interfaz definida a tal efecto.

2.2.6 ¿Por qué usar SOA?

Es de conocimiento común que los requisitos de los negocios varían con gran rapidez, lo que obliga que las empresas utilicen sistemas escalables que permitan la interacción entre herramientas incompatibles.

Algunas razones para utilizar SOA son:

- Reutilización: Las funciones de negocio, dentro de una empresa pueden ser expuestas como servicios Web y ser reutilizadas para cubrir nuevas necesidades de negocio.

- Interoperabilidad: Permite que los clientes y servicios se comuniquen independientemente de la plataforma en que residan.
- Escalabilidad: Debido a que servicios están acoplados débilmente, las aplicaciones que usan estos servicios escalan fácilmente.
- Flexibilidad: Es otra característica proporcionada por el acoplamiento débil. Cualquier cambio en la implementación de un servicio no afecta a los otros.
- Eficiencia del coste: Al usar los servicios Web se reutilizan la infraestructura Web existente virtualmente en todas las organizaciones.

El estilo de arquitectura SOA es convidado con el patrón arquitectónico *Layers* (capas), con el fin de estructurar la aplicación en grupos de sub-tareas, las cuales se clasifican de acuerdo a un nivel particular de abstracción.

2.3 Descripción de la Arquitectura de Software propuesta

El binomio RUP/UML será la metodología y lenguaje usados para el proceso de desarrollo del sistema de información Intranet 2. Por demás se cuenta con una potente herramienta para la modelación, la *Herramienta CASE:* Visual Paradigm.

2.3.1 Representando la arquitectura con UML.

La arquitectura, conformada por diferentes visiones del sistema, constituye un modelo de cómo está estructurado dicho sistema, sirviendo de comunicación entre las personas involucradas en el desarrollo y ayudando a realizar diversos análisis que orienten el proceso de toma de decisiones. Para que la arquitectura se convierta en una herramienta útil dentro del desarrollo y mantenimiento de los sistemas de software es necesario que se cuente con una manera precisa de representarla.

Los lenguajes desarrollados hasta el momento para representar la arquitectura presentan diferentes problemas para su utilización, por lo que se decidió utilizar un

lenguaje donde estos inconvenientes pudiesen ser superados. Este lenguaje de modelamiento es UML, conocido en la industria y que además está apoyado por herramientas y metodologías de desarrollo, siendo adoptado por varias empresas como una notación estándar.

UML permite que se represente de manera semi-formal la estructura general del sistema, con la ventaja de que este mismo lenguaje puede ser usado en todas las etapas de desarrollo del sistema y su representación gráfica puede ser usada para comunicarse con los usuarios.

2.3.2 Herramientas de modelado

Tener completa una arquitectura de software, implica obtener un proyecto de calidad. Uno de los aspectos más importantes en la arquitectura, es definir las herramientas a usar para el proyecto. Para el modelado de la arquitectura se decidió utilizar, fundamentalmente dos herramientas:

Herramienta CASE: Visual Paradigm.

Visual Paradigm utiliza UML como lenguaje de modelado. Está disponible en varias ediciones, cada una destinada a unas necesidades: Enterprise, Profesional, Community, Standard, Modeler y Personal. Es una herramienta de gran alcance, fácil de utilizar y que apoya el ciclo de vida completo del software - análisis, diseño, codificación, prueba y despliegue. Es posible dibujar diagramas UML, generar código de diagramas de la clase y viceversa, y generar la documentación. Soporta todos los diagramas de la más reciente notación de UML.

2.3.3 Desarrollo basado en RUP

En este trabajo, para la selección, evaluación, representación y documentación de la Arquitectura de Software, hemos adoptado la propuesta plasmada en [2], como se

expuso en el Capítulo 1; sin embargo, se ha seleccionado RUP (Rational Unified Process) para guiar todo el desarrollo del software.

El Proceso Unificado de Rational (RUP, el original inglés *Rational Unified Process*) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas. [7].

2.3.4 Framework Arquitectónico

La arquitectura sigue el framework "4+1" vista, presentado en [9], este framework define cuatro vista para la arquitectura en conjunto con los escenarios, y es presentado en la siguiente figura.

Framework 4+1	**Arquitectura**
Use – Case View	Vista de Caso de Uso, Restricciones, Calidad
Logical View	Vista Lógica
Process View	Vista de Procesos
Deployment View	Vista de Despliegue
Implementation View	Vista de Implementación

Tabla 5 Vistas utilizadas en el Framework

2.3.5 Vista General del Sistema

La intranet constituye el principal medio de divulgación de la Universidad de las Ciencias Informáticas, es una aplicación Web que tiene como propósito la divulgación de diversas informaciones, dirigidas a cada una de las áreas de la universidad con el objetivo de:

- Reforzar la cultura e identidad organizacional del personal vinculado a la universidad.

- Agilizar la información interna mediante la ampliación del acceso a la misma, la comunicación y el flujo de trabajo entre las personas.
- Ayudar a mejorar los procesos internos y el trabajo diario en el seno de la universidad.
- Potenciar el trabajo reduciendo el tiempo en que se gestiona la información.
- Potenciar el flujo de datos entre los diferentes subsistemas y niveles organizativos.

La aplicación debe permitir:

Publicar noticias: Es un servicio que se encarga de la publicación de las noticias elaboradas por periodistas y colaboradores. Cada noticia se le debe poder adjuntar diferentes materiales multimedia como imágenes, videos, sonidos y su contenido debe ser comprobado, corregido y aprobado antes de su publicación. Las noticias podrán ser organizadas en diferentes clases según sea el interés de los administradores del portal y poder ser relacionadas por medio de las sugerencias del autor y/o por la existencia de de términos en común. Una vez que las noticias cumplan con el plazo de publicación, éstas serán archivadas en la base de datos de la intranet.

Publicar avisos: Es un servicio al cual pueden acceder los responsables de áreas para la publicación de avisos especificando el nivel de prioridad, categoría(s) de usuario a la que va dirigida y periodo de vigencia en la intranet. El contenido del mismo debe ser chequeado y aprobado antes de ser publicado, pueden ser recuperados por autor, por área productora, por palabra de título, por fecha de publicación o de expiración o por la audiencia a la que va dirigida.

Publicar efemérides: Es un servicio que le permite a los responsables de la identificación de efemérides publicar las mismas en el portal, donde se comprobará y se aprobará su contenido. Cada noticia debe estar confeccionada por textos breves, y pueden o no llevar archivos multimedia relacionados con el hecho que describe. La

recuperación de los avisos se podrán llevar a cabo mediante la fecha, palabra de título, categoría. Cuando expire la fecha de publicación, los avisos serán archivados en la base de datos del sistema, permitiendo su recuperación en futuras fechas.

Publicar Sitios Web: El sistema debe ser capaz de publicar los sitios web de las diferentes aéreas de la universidad de acuerdo con la aprobación del consejo editor.

Acceso a los sitios de la Intranet: Los usuarios podrán acceder a los diferentes sitios y recursos disponibles en cada sección del portal.

Acceso a los servicios principales de la comunidad: Los usuario pueden acceder a los principales servicios en línea prestados por otros subsistemas de la universidad.

Servicio de búsqueda de información en la Intranet: Este servicio permitirá la búsqueda y la recopilación de informaciones generales y específicas dentro de la intranet universitaria.

Servicio de información personal: Este servicio recopilará toda la información de interés particular del usuario, atendiendo a sus necesidades y privilegios, integrará recursos disponibles en diferentes subsistemas de información de la universidad (ej.: economía, recursos humanos, servicios técnicos, secretaría docente, formación de postgrados). Solamente se podrá acceder este servicio después de que el usuario se haya autenticado.

Servicio de sindicación: Este servicio permitirá por medio de canales de sindicación de contenidos relacionar las diferentes noticias que se generen en los diferentes subsistemas de información de la universidad y cuyo alcance sea de interés común para todos, las noticia unas vez sindicadas deberán ser aprobadas y corregidas y pasarían a formar parte de la base de datos de noticias del portal de la intranet. Al

mismo tiempo el portal de la intranet debe ser capas de generar sus propios canales de sindicación para que el proceso se pueda realizar a la inversa y de esta forma exista un flujo constante de información entre los diferentes subsistemas.

Debido a la alta importancia el conocimiento de las diversas informaciones para la realización de las tareas cotidianas en la universidad, el sistema debe garantizar las siguientes características:

- Rendimiento aceptable: Hace referencia al tiempo requerido para responder a un o varios eventos por unidad de tiempo.
- Altamente modificable: Es la habilidad de realizar cambios al sistema de forma rápida y a bajo costo.
- Seguro: Hace referencia a la habilidad de evitar una violación en el sistema.
- Altamente disponible: Se refiere al tiempo que el sistema se encuentra funcionando.
- Portable: Es la habilidad del sistema de ejecutarse sobre diferentes software, hardware.

2.3.6 Vista de Caso de Uso

En esta vista se muestra la percepción que tiene el usuario de las funcionalidades del sistema mediante la representación de los actores y casos de usos más importante.

Actores

Un actor representa un conjunto coherente de roles que los usuarios de casos de usos desempeñan cuando interaccionan con estos casos de uso.

Publicador (Web master)

Usuario con permisos de administración, que publica el contenido en el portal.

Revisor

Consejo editorial, compuesto por directivos y periodistas, que se encargan de decidir que contenido será publicado en el portal.

Editor

Usuario encargado de los arreglos y ediciones de las informaciones creadas por el productor.

Productor

Representa a los usuarios del sistema que tienen como responsabilidad actualizar las informaciones que se publican en el sitio, éstos pueden ser periodistas o responsable de áreas.

Casos de Usos Arquitectónicamente Significativos

Los casos de uso describen un conjunto de secuencias de acciones, incluyendo variaciones que un sistema lleva a cabo y que conduce a un resultado observable de interés para un determinado actor.

Diagrama de Casos de Uso Arquitectónicamente Significativo

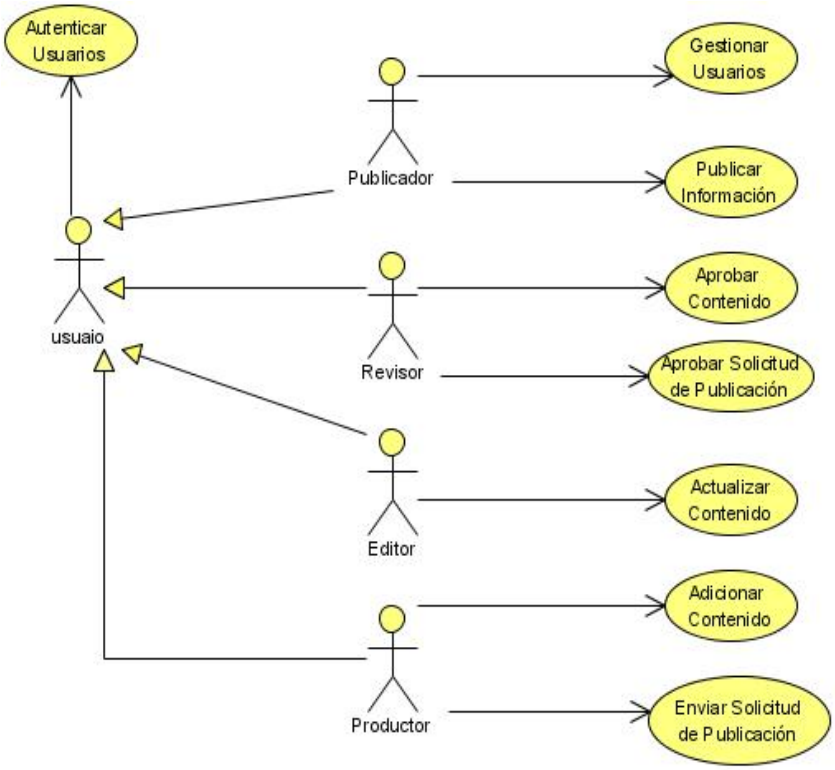

Fig. 2 Diagrama de Caso de Uso Arquitectónico

Breve Descripción de los Casos de Usos Arquitectónicos

- **Autenticar Usuario**

  Descripción

  El caso de uso tiene lugar cuando el usuario accede al sistema para realizar cualquier operación que éste soporta. Para esto debe haberse autenticado correctamente, además debe pertenecer al dominio UCI y/o ser un miembro del sistema.

Pre-Condiciones

El usuario debe ser un miembro del equipo de trabajo del sistema y/o pertenecer al dominio UCI.

- **Gestionar Usuario**

Descripción

Le permite al publicador o administrador asignarle roles a los diferentes usuarios del sistema, así como la edición de los mismos, es decir, mediante este caso de uso el administrador puede adicionar, eliminar o editarles el perfil a los usuarios del sistema.

Pre-Condiciones

Para realizar esta operación el usuario debe haberse autenticado y contar con un perfil que le permita realizar esta operación.

- **Publicar Información**

Descripción

El publicador es el responsable de publicar las distintas informaciones, entiéndase páginas informativas o sitios Web aprobados por la comisión editorial.

Pre-Condiciones

Para comprobar y publicar las solicitudes, el usuario debe tener privilegios para llevar a cabo esta acción y debe haberse autenticado.

- **Aprobar Contenido**

  Descripción

  El Revisor es el responsable aprobar las solicitudes después de chequear que las informaciones, entiéndase noticias, efemérides y avisos cumplen con las normas de publicación establecidas por la universidad.

  Pre-Condiciones

  El usuario, en este caso el Revisor debe tener un perfil que lo habilite para realizar esta actividad y debe haberse autenticado correctamente.

- **Aprobar Solicitud de Publicación**

  Descripción

  El revisor recibe la solicitud de publicación ya sea de sitios Web o de páginas. Informativa, la aprueba según el contenido y finaliza cuando el sistema le envía confirmación de publicación al productor.

  Pre-Condiciones

  El usuario, en este caso el Revisor debe tener un perfil que lo habilite para realizar esta actividad y debe haberse autenticado correctamente.

- **Actualizar Contenido**

  Descripción

  El editor recibe el contenido a publicar, éste es corregido y arreglado según el manual de estilos, y termina cuando el contenido es actualizado y enviado al revisor para su aprobación.

Pre-Condiciones

El usuario debe haberse autenticado y contar con un perfil que le permita realizar estas operaciones.

- **Adicionar Contenido**

Descripción

El productor es el encargado de confeccionar el contenido que será publicado en el portal. Este consta de diferentes categorías: noticias, efemérides y avisos. El CU termina cuando el contenido es enviado al editor.

Pre-Condiciones

El usuario debe haberse autenticado y contar con un perfil que lo habilite para realizar estas operaciones.

- **Enviar Solicitud de Publicación**

Descripción

El caso de se inicia cuando el productor envía una solicitud de publicación a partir de las diferentes opciones de solicitudes: Sitios Web o Páginas Informativas, con la descripción y pequeño resumen del contenido y finaliza con una respuesta del sistema donde se confirma entrega de solicitud.

Pre-Condiciones

El usuario debe haberse autenticado y contar con un perfil que lo habilite para realizar estas operaciones.

2.3.7 Vista de Restricciones

En esta vista se presentan las restricciones a la que está sujeto tanto el proceso de desarrollo como el producto desarrollado.

Restricciones de Estándares

UML: Todos los artefactos confeccionados y utilizados para la comunicación y la documentación del sistema está basado en el lenguaje de modelado UML, con el objetivo de facilitar el entendimiento común entre los miembros del equipo, los clientes y usuarios finales.

Interfaz Web: Todo el contenido debe ser visualizado mediante cualquier navegador Web, por ejemplo: Microsoft Internet Explorer, Mozilla, Firefox, Netscape Navigator, etc.

Servicios Web: La interoperabilidad con los sistemas de la universidad debe estar basada en los servicios Web.

Tecnologías:

- El desarrollo del sistema de información Intranet 2 debe estar realizado en la plataforma GNU/Linux.
- La Base de Datos del sistema será soportada por el gestor de base de datos PostgreSQL como reemplazo a ZopeDB para facilitar una futura migración.
- Se empleará Apache en vez de Zope Web Server como Servidor Web para el manejo de peticiones del cliente.
- Se utilizará el sistema gestor de contenido Plone sobre el servidor de aplicación Zope.

Soporte

El sistema de información Intranet 2 tendrá mantenimiento evolutivo permanente, orientado al mantenimiento de los módulos existentes y principalmente al desarrollo de nuevos módulos para ampliar los servicios del sistema.

2.3.8 Vista de Calidad

Describe los requerimientos no funcionales dentro de las categorías de Usabilidad, Confiabilidad, Portabilidad, Seguridad.

Usabilidad

El diseño de la interfaz del sistema deberá ser sencilla y legible, no puede estar sobrecargada de contenido para lograr un mayor entendimiento a los usuarios (actores humanos), disminuyendo así la necesidad de capacitación de los usuarios. Su contenido deberá ser redactado principalmente en el idioma español, ya que es el idioma común de la audiencia a la que va dirigida, aunque debe permitir la publicación en otros idiomas como inglés.

La documentación de usuario estará anexada a la interfaz, garantizando que en cada lugar que se encuentre ubicado un usuario contará con una opción de ayuda que le indicará en que contexto se encuentra, que información está viendo, que información debe proveer y cuál será la actividad que realizará el sistema una vez provista dicha información. No se proveerá documentación impresa al usuario.

Confiabilidad

El sistema debe garantizar la publicación de todas las informaciones aprobadas, así como el acceso a los diferentes sitios y servicios brindados por la aplicación.

Seguridad

Este sistema de información se regirá por el reglamento de navegación de la universidad. Las peticiones que se le realicen al sistema serán proporcionadas en dependencia de los privilegios con que cuenta el usuario que la solicita. Además las informaciones y servicios contenidos en la aplicación estarán protegidos por la seguridad que brindan las diferentes tecnologías utilizadas para el desarrollo de la misma, incluyendo las ofrecidas por el sistema operativo que lo soportará.

Portabilidad

La intranet podrá ser montada en cualquier plataforma deseada por la institución, y será accedida mediante cualquier navegador Web que utilice el usuario, sin tener en cuenta el sistema operativo que éste emplee.

2.3.9 Vista Lógica

La vista de arquitectura del modelo de diseño o la lógica presenta los subsistemas e interfaces más importantes para la arquitectura.

Descripción de la arquitectura vertical de alto nivel

La Intranet 2 cuenta con la siguiente arquitectura de sistema:

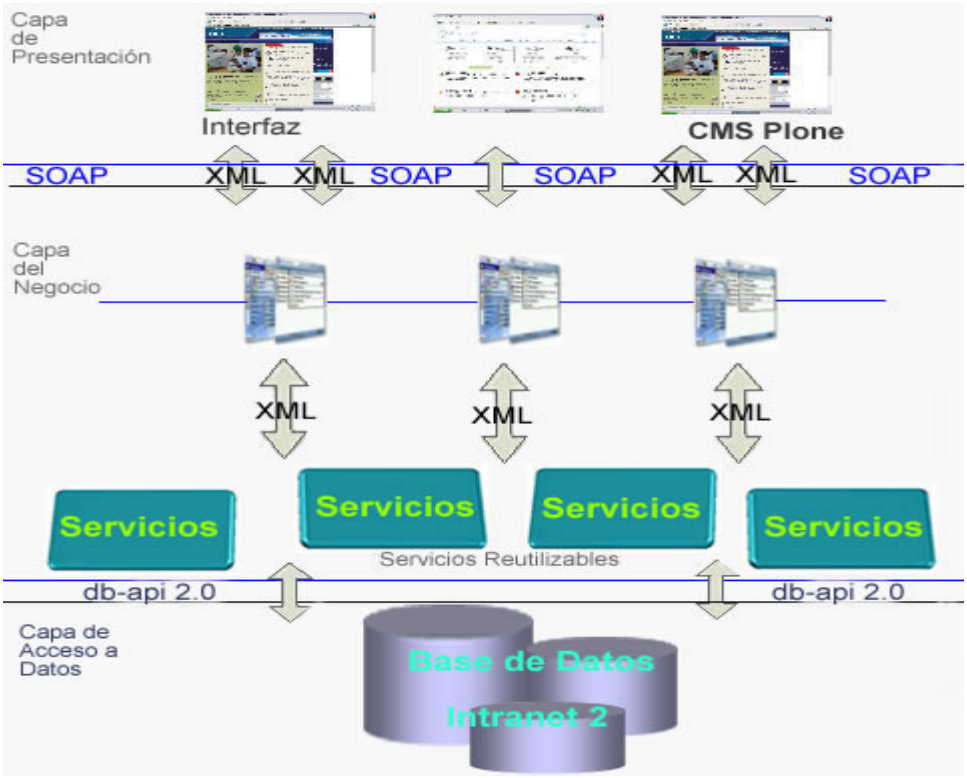

Fig. 3 Estructura de la Arquitectura

El sistema Intranet 2 se encuentra organizado mediante la utilización del ***patrón arquitectónico Layers*** (capas), donde cada capa puede invocar operaciones de la capa inferior. Este patrón ayuda a estructurar aplicaciones que pueden descomponerse en grupos de sub-tareas de forma que las tareas de cada grupo se encuentran en el mismo nivel de abstracción.

La división es en tres capas: *Presentación*, *Negocio* y *Datos*.

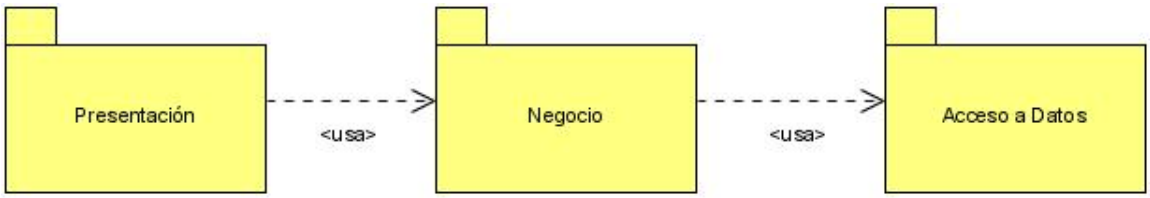

Fig. 4 Capas del Sistema

Breve descripción de las capas.

Capa de Presentación

La capa de Presentación recoge la entrada del usuario y presenta las informaciones solicitadas por el usuario, estas informaciones son mensajes SOAP (basados en XML) que se transmiten mediante el protocolo de transporte *http*. Controla la navegación por las páginas y delega la entrada del usuario a la capa de Negocio, para esta solamente conoce un objeto de la interfaz Entrada de Seguridad, mediante el cual se chequea el acceso a los servicios que brinda dicha capa. La capa de Presentación también puede validar la entrada del usuario y mantener el estado de sesión de la aplicación.

Para desarrollar esta capa se ha elegido Plone, debido a que es un marco de trabajo de componentes de interfase de usuario del lado del servidor para aplicaciones Web, permite manejar los estados, eventos, la validación del lado del servidor, la conversión de datos y definir la navegación entre páginas, además soportar internacionalización y accesibilidad.

Capa de negocio

La capa de Negocio contiene los servicios de negocio de la aplicación. Recibe peticiones de la capa de Presentación, procesa la lógica de negocio basada en las peticiones, y media en los accesos a los recursos de la capa de Acceso a Datos.

Para la capa de Negocio se ha elegido la plataforma de desarrollo Zope, ya que puede organizar de forma efectiva nuestros objetos de la capa central y manejar las conexiones por nosotros, además de equilibrar las peticiones que reciba entre varios ordenadores mediante la uso del producto ZEO. Es una plataforma que puede ser ampliada o modificada fácilmente mediante el lenguaje de programación Python.

Esta capa se conecta con la capa de acceso a datos haciendo uso interfaces de las db-api 2.0 utilizadas por Python.

Capa de Acceso a Datos

La capa de Acceso a Datos recibe peticiones de la capa de negocio, contiene todos los servicios de acceso a datos y es la encargada de acceder a la base de datos de la aplicación.

2.3.10 Descripción de la arquitectura horizontal

Los casos de usos arquitectónicamente significativos se agruparon en los siguientes módulos:

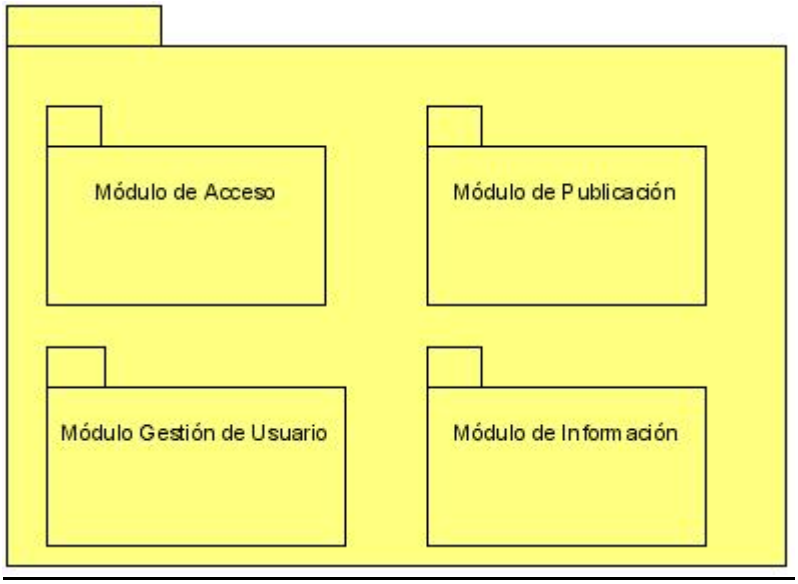

Fig. 5 Principales Módulos del Sistema

Módulo de Acceso: En este módulo se encuentra todo lo referente al acceso al sistema.

Caso de uso Contenido

- Caso de Uso Autenticar Usuario

Módulo Gestión de Usuario: En este módulo se encuentra todo lo referente a la gestión de perfil de los usuarios, entiéndase adicionar usuario, eliminar usuario y editar datos de usuario.

Caso de uso Contenido

- Caso de Uso Gestionar Usuario

Módulo de Publicación: Este módulo engloba todo lo referente a las informaciones o artículos que se publicaran en el portal. Estas informaciones pueden ser noticias, avisos, efemérides, sitios Web, servicios, etc. Estos artículos pueden traer adjunto imágenes, sonidos, videos.

Caso de uso Contenido

- Publicación Información

Módulo de Información: En este módulo se encapsula todo lo referente al proceso involucrado de las informaciones antes de su publicación, es decir, adicionar un artículo, editar un artículo y aprobar un artículo. Estos artículos pueden ser noticias, avisos, efemérides y sitios Web, etc.

Caso de uso Contenido

- Aprobar Contenido.
- Aprobar Solicitud de Publicación.
- Actualizar Contenido.
- Adicionar Contenido.
- Enviar Solicitud de Publicación.

2.3.11 Vista de refinada de la arquitectura del sistema

En el siguiente diagrama se muestra la arquitectura lógica del sistema, en la cual se representa como se interconectan mediante una interfaz los diferentes y principales subsistemas y paquetes del sistema.

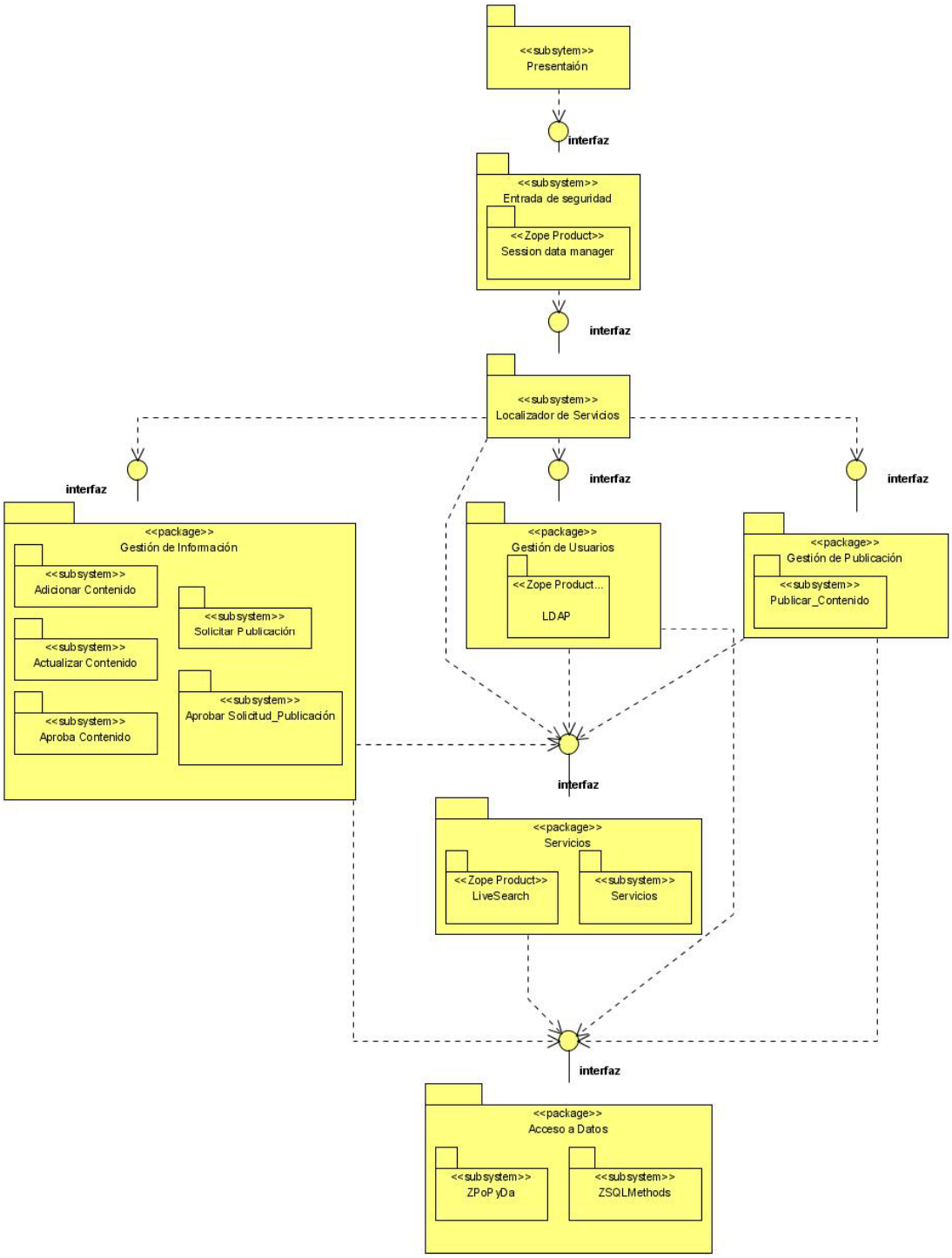

Fig. 6 Vista Refinada de la Arquitectura del Sistema

Capa de Presentación

Contiene a:

- Subsistema Presentación.

Capa de Negocio

Contiene a:

- Subsistema Entrada de Seguridad.
- Subsistema Localizador de Servicios.
- Paquete Gestión de Información.
- Paquete Gestión de Usuarios.
- Paquete Gestión de Publicación.
- Paquete Servicios.

Capa de Datos

Contiene a:

- Paquete Acceso a Datos.

Subsistema de Presentación

Este subsistema contiene todas las páginas dinámicas con las que el usuario interactúa para solicitar y ejecutar los servicios prestados por el sistema, verificando que los datos entrados son los solicitados por los servicios, estas páginas son construidas por Plone, que es el sistema gestor de contenido utilizado.

Subsistema Entrada de Seguridad

Este subsistema tiene como objetivo controlar el acceso de los usuarios a los diferentes servicios brindado por el sistema.

Subsistema Localizador de Servicios

Este subsistema tiene la funcionalidad de localizar los servicios y productos del sistema y que son solicitados por el usuario, en él se almacenan todas las informaciones referentes a cada uno de ellos.

Paquete Gestión de Usuarios

En este paquete se engloba todo lo referente a la autenticación y gestión de los usuarios. Cuenta con productos como el *LDAPMultiPlugins* que contiene Plugins para gestionar usuario de un servidor LDAP,

Paquete Gestión de Publicación

En este paquete se engloba todo lo referente a la publicación de contenido, por ejemplo avisos, sitios Web, servicios, etc. Este paquete permite la publicación de información automáticamente, para esto, el productor debe especificar la fecha de publicación y la de expiración.

Paquete Gestión de Información

En este paquete se encuentra todo lo referente al proceso de almacenamiento de contenidos en el sistema, es decir, un usuario con un determinado perfil habilitado puede adicionar, aprobar, actualizar contenidos como noticias, efemérides, avisos y/o solicitar la publicación de un determinado servicio o sitio Web.

Paquete Servicios

En este paquete se englobarán varios servicios que el sistema deberá permitir, como la búsqueda general o específica de contenido mediante el uso del producto LiveSearch, el servicio de estadísticas del portal entre otros.

Paquete Acceso a Datos

Es el encargado e interactuar con la base de datos de la aplicación, haciendo uso de ZSQLMethods y ZPoPyDa. El producto ZSQLMethods, son los métodos que almacenan las consultas estáticas o dinámicas de acceso a la base de datos relacional (mysql, postgresql, oracle, etc.). Abstraen el motor de base de datos que vaya por debajo a través de un adaptador, creando una capa transparente de consultas SQL. El ZPoPyDa es un adaptador de base de datos que permite conectar los ZSQLMethods contra una base de datos PostgreSQL. Al igual que el producto anterior, requiere que los módulos de python de acceso a base de datos PostgreSQL estén instalados.

2.3.12 Paquetes Arquitectónicamente Significativos del Sistema

Los subsistemas están compuestos por clases, las cuales se encuentran agrupadas en paquetes para lograr un mejor entendimiento y comunicación entre los diferentes miembros de desarrollo y los usuarios finales. En el siguiente diagrama se muestra como se van a empaquetar las diferentes clases del sistema.

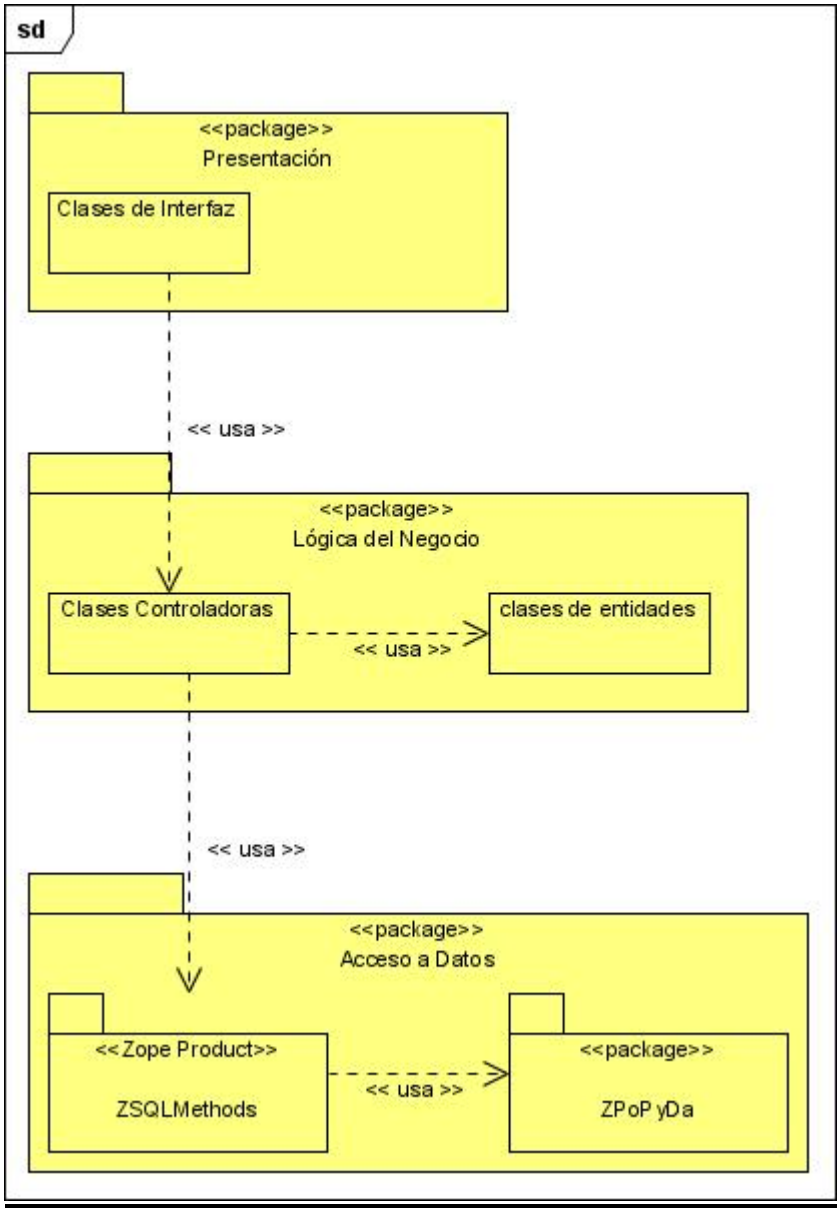

Fig. 7 Paquetes Arquitectónicos del Sistema

2.3.13 Diagrama de Clases del Caso de Uso Arquitectónicamente Significativo (CUAS) Adicionar Contenido.

En esta sección se ha seleccionado el caso de uso Adicionar Contenido, debido a que es una de las principales funcionalidades del sistema Intranet 2 lo que lo convierte en

un caso de uso crítico, y, por tanto, arquitectónicamente significativo. En la siguiente figura, se muestra el diagrama de clases del caso del diseño de uso Adicionar Contenido.

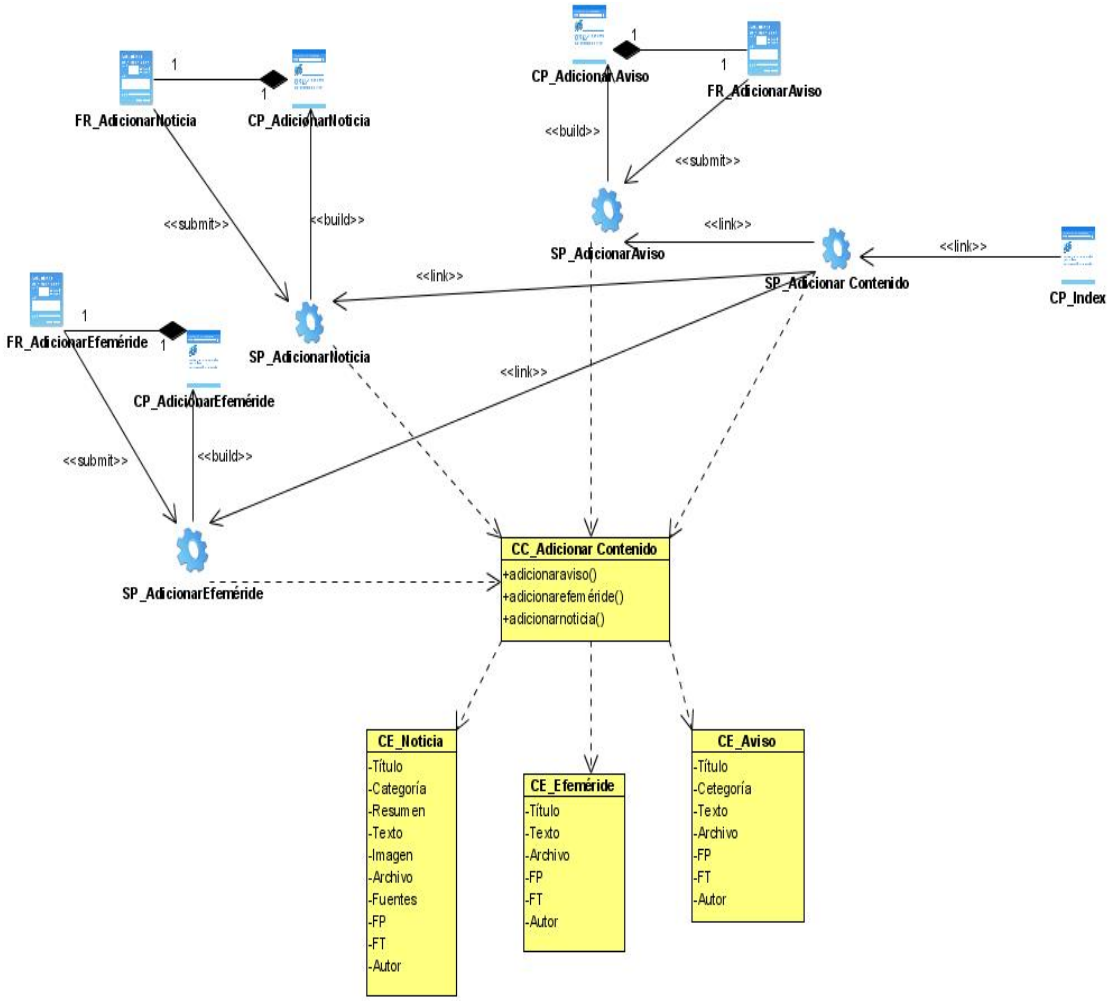

Fig. 8 Diagrama de Clases del CUAS Adicionar Contenido

Paquetes Arquitectónicamente Significativos + Clases del CUAS Adicionar Contenido

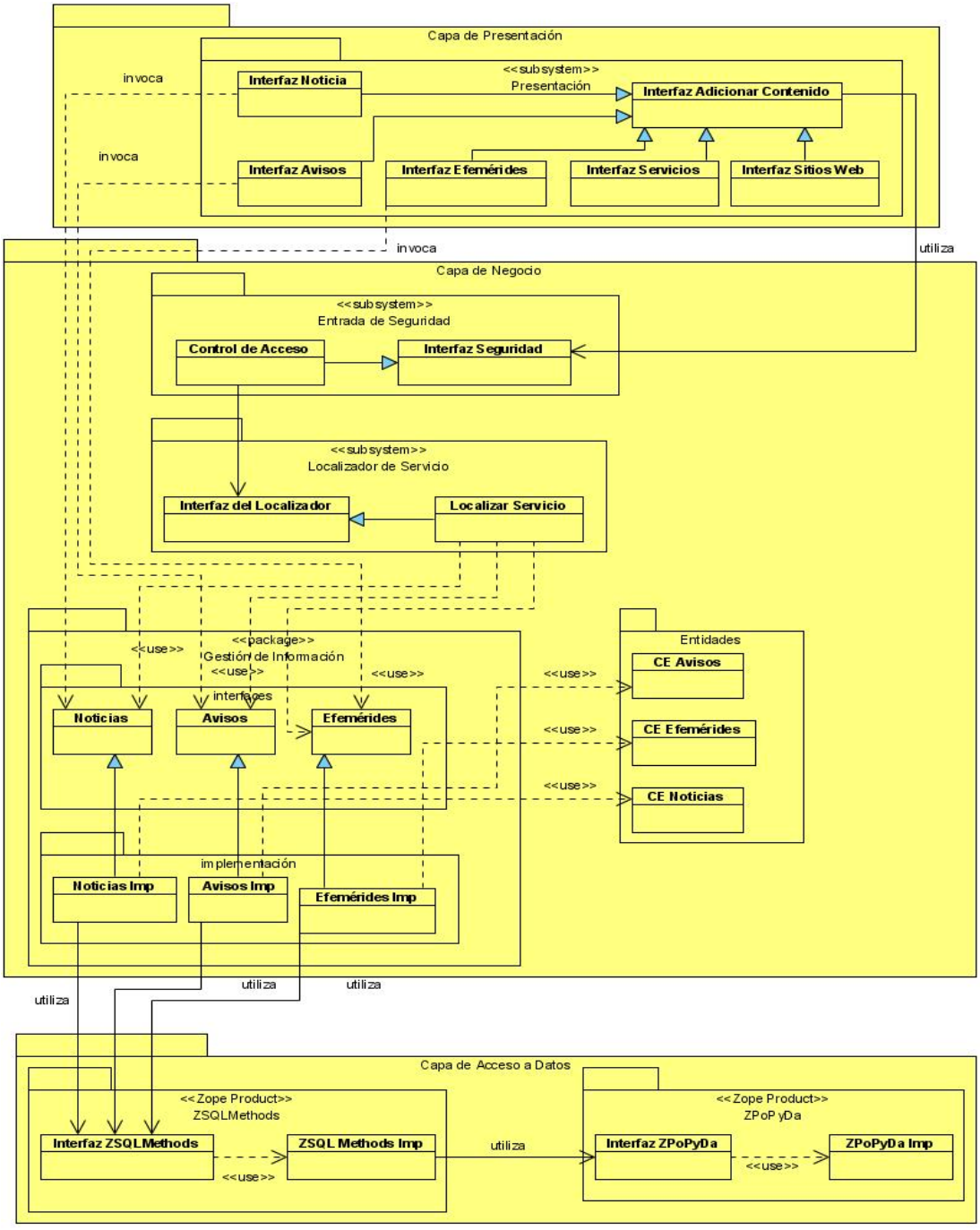

Fig. 9 Diagrama de Paquetes + CUAS Adicionar Contenido

2.3.14 Vista de Procesos

La vista de procesos muestra los principales procesos que el sistema debe tener integrado. Estos procesos se encuentran agrupados por módulos en dependencia de su funcionalidad.

Módulo de Acceso: Este módulo tiene como objetivo mantener el control de acceso al sistema.

Comprobar Nivel de Acceso: Permite chequear los privilegios de los usuarios, evitando que estos accedan a áreas o servicios a los cuales no tienen autorización.

Módulo Gestión de Usuario: En este módulo se realizarán todos los procesos involucrados en la edición y agregación de usuarios, es decir, mediante este módulo el administrador o Web master puede Adicionar usuarios, Editar perfiles de usuarios, Eliminar usuarios. Para poder realizar los procesos de editar y eliminar el Web master tiene que buscar los usuarios, para esto, se incluye un proceso que permitirá obtener el usuario deseado, denominado Seleccionar Usuarios.

Adicionar Usuarios: Este proceso le permite al administrador adicionar usuarios a la base de datos de la aplicación, especificándole el nivel de acceso.

Seleccionar Usuarios: Es un proceso utilizado para seleccionar un usuario de la base de datos.

Editar Usuarios: Mediante este proceso se editan los perfiles del usuario, es decir, sus datos. El Web-master (administrador) es el único que puede cambiar el nivel de acceso al usuario.

Eliminar Usuarios: Por medio de este proceso el administrador elimina de la base de datos a los usuarios.

Módulo de Publicación: Mediante este módulo el Productor o Web-master (administrador) puede publicar los contenidos aprobados por la comisión editorial, para realizar esta operación, este usuario debe realizar primeramente una búsqueda para seleccionar aquellas informaciones que aparecerán publicadas en el portal.

Seleccionar Contenido: Es un proceso que permite seleccionar de la base de datos de la aplicación aquellos contenidos aprobados por la comisión editorial y que aún no se han publicado.

Módulo Información: Este módulo permite realizar la agregación de noticias, avisos, efemérides, sitios Web, servicios, etc. Operaciones necesarias para la publicación de artículos. Los procesos son agrupados por categorías.

Categoría Noticias

Agregar Noticias: Los productores hacen uso de este proceso para adicionar las noticias que se publicaran en el portal.

Buscar Noticias: Es un proceso que permite buscar las noticias almacenadas en la base de datos

Editar Noticias: Una vez seleccionada la noticia el editor se encarga de realizar los arreglos atendiendo a las normas de publicación establecidas por el sistema.

Aprobar Noticias: La comisión editorial selecciona las noticias pendientes y comprueba que éstas cumplan con las normas establecidas para su aprobación.

Eliminar Noticias: Es un proceso que utiliza el administrador en caso de que se desee eliminar una noticia almacenada en la base de datos.

Categoría Efemérides

Editar Efemérides: Al igual que las noticias, las efemérides pasan por un proceso de edición antes de su publicación, este proceso lo lleva a cabo el editor.

Agregar Efemérides: Al igual que las noticias, los productores se encargan de adicionar las efemérides mediante este proceso.

Aprobar Efemérides: Una vez editadas las efemérides, la comisión editorial chequea y aprueba las propuestas.

Buscar Efemérides: El propósito de este objetivo es mostrar las efemérides almacenadas en la base de datos, esta selección depende del criterio de búsqueda del actor.

Eliminar Efemérides: Es llevado a cabo por el administrador cuando se desea eliminar una efeméride de la base de datos.

Categoría Avisos

Agregar Avisos: Es un proceso que permite que los responsables de áreas puedan adicionar avisos y especificar a quienes van dirigidos.

Buscar Avisos: Proceso que se utiliza para obtener los avisos almacenados.

Eliminar Avisos: Proceso ejecutado por el administrador para eliminar un aviso de la base de datos.

Categoría Sitios Web

Agregar Sitios Web: Es un proceso utilizado por los responsables de áreas, mediante el cual solicitan la publicación de los sitios Web pertenecientes a las diferentes áreas de la universidad.

Buscar Sitios Web: Se utiliza para observar cuales son los sitios que se encuentran almacenados en la base de datos.

Editar Sitios Web: Proceso que permite actualizar las informaciones relacionadas con los sitios Web.

Eliminar Sitios Web: Es un proceso que permite eliminar los sitios Web de la base de datos.

Categoría Servicios

Agregar Servicios: El administrador se el responsable de incluir diversos servicios prestados por otros sistemas.

Buscar Servicios: El administrador hace uso de este servicio para obtener los servicios almacenados.

Editar Servicios: Mediante este proceso se actualizan las informaciones referentes a los servicios almacenados.

Eliminar Servicios: Se emplea para quitar un servicio de la base de datos.

Módulo de Publicación: Este módulo permite publicar los artículos pendientes. Esta publicación la realiza el Web master, aunque el sistema puede realizar la publicación de

informaciones automáticamente, introduciéndole una fecha de publicación, al igual que para su archivamiento. Estos artículos aparecerán en la página de acceso del portal en los SimplePortlet correspondientes.

Publicar de Avisos: El Web master selecciona los avisos que desea publicar y específica en que área del portal se mostrarán.

Publicar Sitios Web: El Web master selecciona los sitios que se publicará en el portal y el lugar donde se mostrarán.

Publicar Servicios: El Web master selecciona los servicios que se publicará en el portal y el lugar donde se mostrarán.

2.3.15 Vista de Despliegue

El modelo de despliegue define la arquitectura física del sistema mediante la representación de nodos interconectados, en los que se muestran la asignación de los componentes ejecutables a los nodos. Estos nodos son elementos hardware sobre los cuales pueden ejecutarse los elementos software.

Diagrama de Despliegue

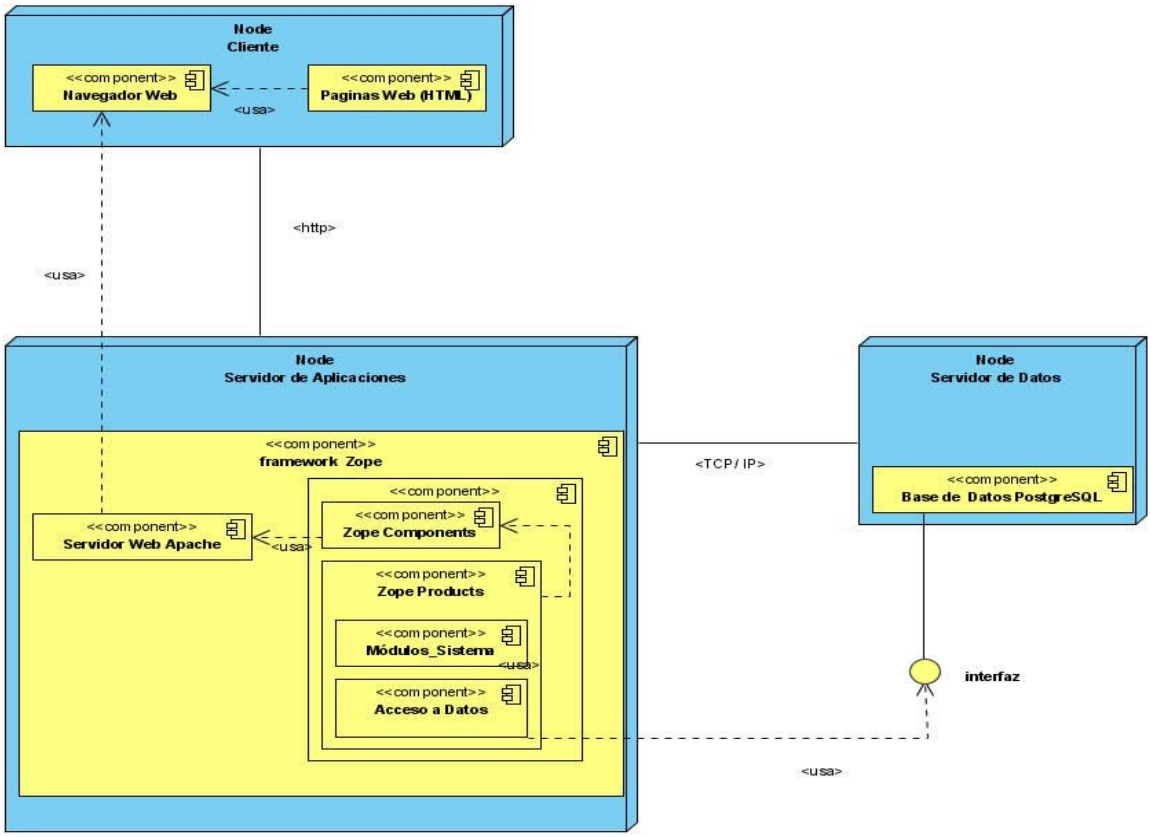

Fig. 10 Diagrama de Despliegue

Nodo Cliente

Las estaciones de trabajo de los clientes deben poseer un navegador (browser) capaz de visualizar html. Dicho nodo debe ser un computador personal que tenga Netscape, Internet Explorer o Mozilla, que posea por lo menos un procesador Pentium, 32 MB de memoria RAM, un monitor VGA.

Nodo Servidor de Aplicaciones

En este nodo se encuentra el servidor Web Apache, es donde se va a encontrar todas las páginas servidoras de la aplicación. Aquí es donde se publicará el sitio de la Intranet

2. Contendrá todos los productos y servicios que el sistema ofrecerá que permitirán la realización toda la lógica del negocio.

Nodo Servidor de Datos

En este nodo se situará los mecanismos de almacenamiento.

Conexiones

Conexión entre el Nodo Cliente y el Nodo Servidor de Aplicaciones

La conexión entre los nodos cliente y el servidor de aplicaciones se realizará mediante los protocolos *http*.

Conexión entre el Nodo Servidor de Aplicaciones y el Nodo Servidor de Datos

La conexión entre estos dos nodos se producirá mediante los protocolos TCP / IP.

Protocolos

- ***http*** *(* ***HyperText Transfer Protocol***): Es un protocolo de transferencia de hipertexto es el protocolo usado en cada transacción de la Web (WWW). El hipertexto es el contenido de las páginas Web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido. También sirve el protocolo para enviar 'información adicional en ambos sentidos, como formularios con campos de texto. HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores.

2.3.16 Vista de Implementación

La vista de implementación describe cómo se implementan los componentes físicos mostrados en vista de distribución agrupándolos en subsistemas organizados en capas, ilustra, además las dependencias entre éstos. Básicamente, se describe el mapeo

desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos.

Producto a que no se encuentra definida completamente la Arquitectura de Información de la Intranet 2, y que se va a utilizar a Zope / Plone para la construcción del sistema, esta vista queda omitida, pues, las funcionalidades básicas antes descriptas se realizarán mediante la utilización de los diferentes productos disponibles en Internet para Plone.

2.4 Conclusiones

El propósito de desarrollar la arquitectura de software es, primeramente crear la estructura o el esqueleto del sistema que cohesiones las funcionalidades más criticas y relevantes, y que sirva de soporte a al resto de las funcionalidades finales. Luego realizar el ensamblamiento de las distintas partes. Todo esto hace del diseño de un software particular una tarea generalmente única. La arquitectura de software constituye un documento guía, que se define a partir de un conjunto de requisitos críticos funcionales, de rendimiento o de calidad, donde se especifica detalladamente los principales componentes que integran el sistema y como se relacionan entre si, facilitando el entendimiento de cada uno de los miembros del equipo de desarrollo.

Capitulo 3 Evaluando la Arquitectura de Software

Introducción

Por lo regular, al final del desarrollo del software se conoce si éste cumplió o no con al menos un atributo de calidad que se especificaron en los requerimientos no funcionales, lo que implica tomar demasiados riesgos innecesarios. Para evitar estos riesgos es recomendable realizar evaluaciones a la arquitectura durante su diseño.

Realizar un buen diseño de arquitectura de software garantiza que el sistema cumpla con uno o varios atributos de calidad, este diseño puede determinar el éxito o el fracaso de un sistema de software

3.1 Evaluando una arquitectura

Uno de los factores que determina el éxito o el fracaso de software es su arquitectura, es decir, la estructura del sistema. Si la arquitectura ha sido bien diseñada, entonces, garantiza que el sistema cumpla con uno o varios atributos de calidad, como por ejemplo confiabilidad, seguridad, etc.

3.1.1 ¿Por qué es necesario evaluar una arquitectura de software?

Evaluar una arquitectura de software sirve para prevenir todos los posibles desastres de un diseño que no cumple con los requerimientos de calidad y para saber que tan adecuada es la AS diseñada para el sistema.

Una evaluación de una arquitectura no da un SI o un NO, si es buena o mala, o una calificación, expresa donde está el riesgo, es decir, fortalezas y debilidades identificadas de la AS.

Después de una evaluación de una arquitectura de software, se pueden tomar algunas decisiones como: si se puede seguir el proyecto con las áreas de debilidad dadas en la evaluación, si hay que reforzar la AS o si hay que comenzar de nuevo toda la AS.

3.1.2 ¿Cuándo es recomendable evaluar la arquitectura?

La evaluación clásica de la arquitectura se realiza cuando ésta se encuentra especificada totalmente y no se ha iniciado su implementación. La arquitectura puede ser evaluada en cualquier momento de desarrollo.

Existen dos variaciones útiles para realizar esta evaluación, la evaluación temprana y la evaluación tardía [14].

La evaluación temprana

Para realizar esta evaluación no es necesario que la arquitectura se encuentre completamente especificada. Esta evaluación permite efectuar decisiones sobre la arquitectura en cualquier nivel, puesto que se pueden imponer cambios arquitectónicos producto de una evaluación en función de los atributos de calidad esperados.

La evaluación tardía

Se realiza cuando la arquitectura del sistema se encuentra establecida y se ha terminado su implementación, es decir, en el momento de adquisición de un sistema ya terminado. Se considera por parte de los autores que la evaluación en este punto es muy importante y útil, puesto que puede observarse el cumplimiento de los atributos de calidad asociados al sistema y cómo será su comportamiento general.

La evaluación de la arquitectura de software debe realizarse cuando esta contiene suficientes elementos como para justificarla. Una buen momento para determinar cuando realizar la evaluación es cuando el equipo de desarrollo comienza a tomar

decisiones que dependen de la arquitectura y que el costo de no tenerlas en cuenta son mayores que el costo de realizar una evaluación

3.1.3 ¿Quiénes participan en la evaluación?

Generalmente, las evaluaciones son realizadas por miembros del equipo de desarrollo, por ejemplo el arquitecto, el diseñador y el administrador de proyecto. Aunque pueden existir excepciones, donde las pruebas son realizadas por un grupo de personas especialistas.

El cliente también se interesa por las evaluaciones, pues en dependencia de los resultados obtenidos durante las pruebas puede decidir si se continua o no con el proyecto.

3.1.4 Resultado de la evaluación

La evaluación de la arquitectura produce un informe, el cual varía en dependencia del método utilizado. Este informe produce respuesta a dos tipos de preguntas.

- ¿Es esta arquitectura adecuada para el sistema para el cual fue diseñada?
- ¿Cual de las arquitecturas propuestas es la más adecuada para el sistema?

Una arquitectura es adecuada cuando cumple dos criterios

- El sistema resultante cumple con los objetivos de calidad.
- El sistema puede ser construido con los recursos disponibles, es decir, es construible.

La evaluación de la arquitectura no produce resultados cuantitativos, no es de interés conocer la cantidad de transacciones por segundos debido a que el sistema aún no

está implementado. Lo que interesa es aprender como un atributo de calidad es afectado por una decisión de diseño arquitectónico.

3.1.5 ¿Por qué cualidades puede ser evaluada una arquitectura?

Los atributos de calidad son requerimientos adicionales del sistema que hacen referencia a características que éste debe satisfacer.

Algunos de los atributos por los cuales puede ser evaluada una arquitectura son:

Disponibilidad (Availability). Es la medida de disponibilidad del sistema para el uso.

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).

Confiabilidad (Reliability). Es la mediada de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo

Seguridad (Security). Es la mediada de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos.

Portabilidad (Portability). Es la habilidad del sistema de ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos.

3.2 Técnicas de evaluación

Existen varias técnicas que permiten realizar evaluaciones a la arquitectura, estas se clasifican en cualitativas y cuantitativas.

En las técnicas de evaluación cualitativas se pueden utilizar escenarios, cuestionarios o listas de verificación. Mientras que en las técnicas de evaluación cuantitativas se pueden emplear métricas, simulaciones, prototipos, experimentos o modelos matemáticos.

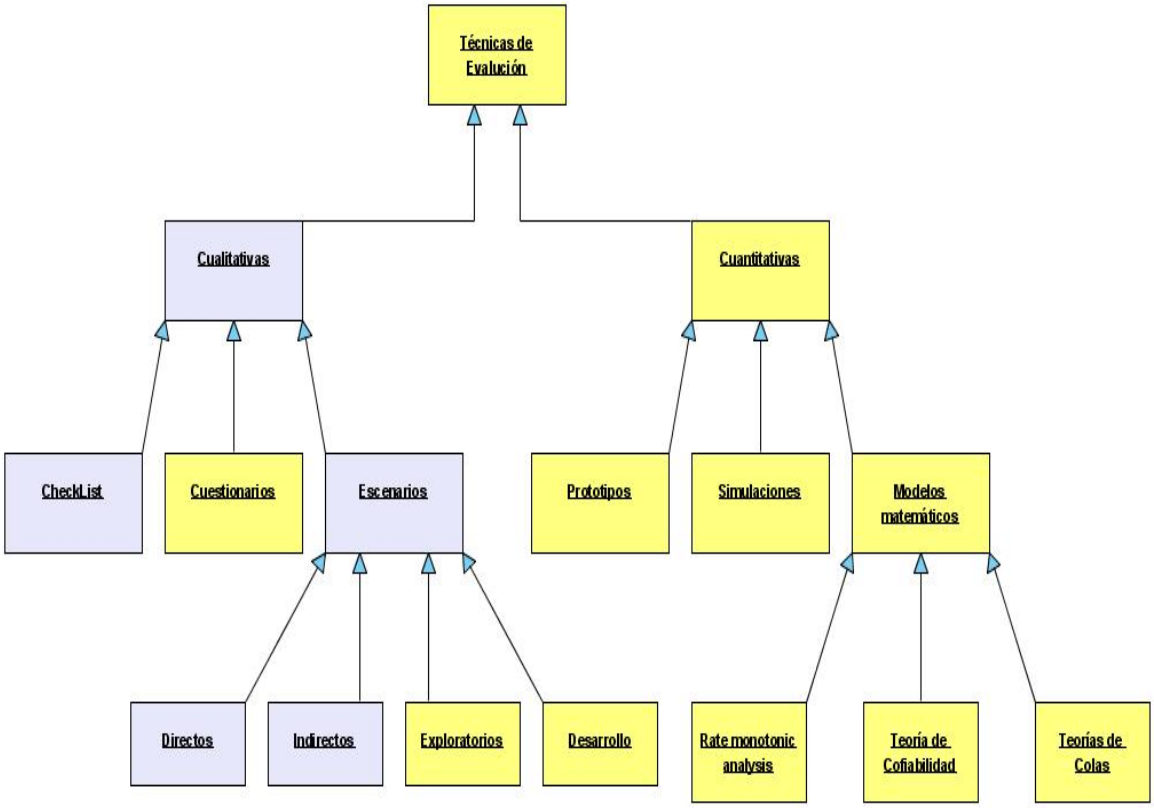

Fig. 11 Técnicas de Evaluación

La mayoría de los métodos de evaluación utilizan escenarios, que son secuencias específicas de pasos que involucran el uso o la modificación del sistema. Por lo regular, las técnicas de evaluación cualitativas son usadas cuando la arquitectura se encuentra en construcción, mientras que las técnicas de evaluación cuantitativas, se usan cuando la arquitectura ya ha sido implantada.

3.3 Evaluando la arquitectura de software propuesta

La evaluación para la arquitectura de software propuesta en este trabajo se realizará mediante el uso del método ARID (Active Review Intermediate Designs), que es un híbrido del método ARD (Active Design Review) y ATAM (Architecture Trade-Off Method). ARID constituye un método conveniente para la evolución de diseños parciales en las etapas tempranas del desarrollo y la técnica de evaluación basada en escenario.

Del método ATAM tomaremos la idea de usar escenarios generados por los involucrados con el sistema, y del ARD no quedamos con la participación activas de los empleados, ya que resulta conveniente la fidelidad de las respuestas que se obtienen de los involucrados en el desarrollo.

La solución de arquitectura que se propone en este trabajo se encuentra especificada a alto nivel, es decir, un diseño poco detallado. Esta especificación a alto nivel impide la selección de escenarios que permitan validar si la arquitectura diseñada satisface a requisitos específicos. Como resultado de esto se realizará la evaluación atendiendo a como responden las vistas seleccionadas antes los principales atributos de calidad.

En esta evaluación se analizarán los siguientes escenarios:

- Seguridad del sistema
- Portabilidad
- Modificabilidad
- Rendimiento del sistema

Seguridad del sistema. En la vista de despliegue se muestra los puntos de contactos con el exterior, así como el software encargado de establecer la interacción. También

se muestra donde los objetos de autenticación son manejados, tarea que realiza el servidor de aplicaciones Zope.

Portabilidad. Para la construcción del sistema se empleará la plataforma de desarrollo Zope, plataforma que puede ejecutarse en las plataformas de sistemas operativos más utilizados como GNU/Linux, FreeBSD, Windows NT/2000/XP, etc.

Modificabilidad. Para analizar el impacto que puede tener un cambio en el sistema se puede consultar la vista lógica, donde el uso de las capas muestra como un cambio en una capa más baja puede ser ocultado detrás de sus interfaces y no impactará las capas encima de ella.

Rendimiento del sistema. Se debe consultar la vista de procesos para analizar la concurrencia en el sistema. También se puede observar la vista de despliegue.

3.4 Conclusiones

Producto a que la arquitectura de sistema esta débilmente detallada, en este capitulo se realizó una pre-evaluación de la arquitectura propuesta y descrita en el Capitulo 2, mostrando como se puede analizar el cumplimiento los atributos de calidad atendiendo a cada una de las vistas desarrolladas.

Conclusiones

La investigación realizada muestra que en el campo de la arquitectura de software no hay consenso generalizado en cuanto a los conceptos más importantes, ni a la forma de documentar la arquitectura. La descripción de la arquitectura no incluye información que sea sólo necesaria para validar o verificar la arquitectura. Por tanto no tiene casos o procedimientos de pruebas, y no incluye una vista del modelo de prueba. Ésta debe mantenerse actualizada a lo largo de la vida del sistema para reflejar los cambios y las adiciones que son relevantes para la arquitectura. Estos cambios son normalmente secundarios y pueden incluir:

- La identificación de nuevas clases abstractas e interfaces.
- La adición de nueva funcionalidad a los subsistemas existentes.
- La actualización a nuevas versiones de los componentes reutilizables.
- La reordenación de la estructura de procesos.

La propuesta de solución presentada en este trabajo ha permitido realizar una documentación adecuada de la arquitectura de software del sistema que se desea construir, la cual es comprendida por los miembros del equipo de desarrollo. La evaluación de la arquitectura muestra el cumplimiento del objetivo del presente trabajo, ya que permite la elaboración de un sistema portable, con una interfaz sencilla y bien estructurada, haciéndola más usable y comprensible para el usuario, garantiza la accesibilidad y seguridad de las informaciones y servicios contenidos en el mismo.

Recomendaciones.

Recomendaciones

Con el objetivo de ampliar y de lograr una mejor comunicación entre los miembros de desarrollo y los usuarios finales sobre la arquitectura de software propuesta en este trabajo se recomienda:

- Incluir en etapas posteriores del proyecto nuevos estilos que resulten de interés y que permitan un mejor entendimiento de cómo serán transformado los datos en el sistema.
- Completar la vista lógica realizando un diagrama de colaboración en donde se muestre la interacción del usuario y las principales clases del sistema.
- Desarrollar las vista de implementación en etapas posteriores del proyecto.
- Crear un equipo como miembro del grupo de trabajo que se encargue de estudiar y aplicar los métodos de evaluación de arquitectura de software más importantes.
- Elaborar un documento en el que se especifique de forma detallada las interfaces de los elementos representados en las vista de la arquitectura propuesta.
- Aplicar la arquitectura de software propuesta en otras instituciones, por ejemplo Instituto Politécnico de Informática (IPI), Mini UCI que requieran el uso de una intranet semejante a la de la UCI.

Referencias bibliográficas.

Referencia Bibliográfica

- [1] AMOS LATTEIER, M. P., CHRIS MCDONOUGH, PETER SABAINI. The Zope Book (2.6 Edition), 2001.
- [2] BASS, L., CLEMENTS, P., & KAZMAN, R. Software Architecture in practice. Addison-Wesley., 1998.
- [3] BUSCHMANN, F., MEUNIER, R., ROHNERT, H., SOMMERLAD, P., & STAL, M. Pattern – Oriented Software Architecture. A System of Patterns. John Wiley & Sons, Inglaterra, 1996.
- [4] CLEMENTS, P. "A Survey of Architecture Description Languages". Proceedings of the International Workshop on Software Specification and Design, Alemania, 1996.
- [5] CORPORATION, R. S. Rational Unified Process: Best Practices for Software Development Teams". 2002.
- [6] G.D. ABOWD, R. A. A. D. G. "Formalizing style to understand descriptions of software architecture". ACM Transactions on Software Engineering and Methodology, 4(4):319-364, 1995.
- [7] JCASANO PostgreSQL, robusto como un elefante. 2005. Disponible en http://www.openecuador.org/modules/news/article.php?storyid=31
- [8] GARLAN, M. S. Y. D. Software Architecture: Perspectives on an emerging discipline. Upper Saddle River, Prentice Hall, 1996.
- [9] KRUCHTEN., P. "The 4+1 View Model of Architecture." IEEE Software, Noviembre de 1995.
- [10] M. C. BASTARRICA, S. F. O. A. P. O. R. "Integrated Notation for Software Architecture Specification". In Proceedings of the XXIV International Conference of the Chilean Computer Science Society (SCCC), Arica, Chile, November 2004.
- [11] ROBERTSON, J. So, what is a content management system? Step Two, 3 junio 2003.
- [12] TUTTLE, N. L. A. M. "An Introduction to Input/ Output Automata". CWI Quart, 2(3):219-246, 1989.

Referencias bibliográficas.

- [13] Proceso Unificado de Racional. Disponible en http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational. Mayo 2007
- [14]Shaw y Garlan http://www.fing.edu.uy/inco/cursos/gestsoft/Presentaciones/Evaluacion%20de%20Arquitecturas%20-%20G10/Evaluacion%20de%20Arquitecturas.doc.
- [15] ZOPETECA Buscas Información sobre Zope y Plone, 2006. Disponible en HYPERLINK "http://www.zopeteca.com/" http://www.zopeteca.com/

Bibliografía.

Bibliografía

- ALBERTO MOLPECERE TOURIS, M. P. M. Arquitectura empresarial y software libre, JE22.
- ANA M. MORENO, M. I. S. Patrones de Usabilidad: Mejora de la Usabilidad del Software desde el momento Arquitectónico.
- BASTARRICA, M. C. Atributos de Calidad y Arquitectura del Software.
- CÁRDENAS, S. Á. Diseño de la arquitectura y los servicios web.
- ERIKA CAMACHO, F. C., GABRIEL NÚÑEZ Arquitecturas de Software, Abril 2004. Disponible en http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.pdf
- JOHN WORSLEY, J. D. and M. H. EDITADO POR ANDREW BROOKINS PostgreSQL Práctico, 2001. Disponible en http://www.sobl.org/traducciones/practical-postgres/node19.html
- JOSEPH F. MARANZANO, S. A. R. A. G. H. Z. Architecture Reviews: Practice and Experience.
- LASSO, A. Arquitectura de Software. Disponibilidad http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art110.asp
- MARÍA L. RODRÍGUEZ, M. N., MIGUEL J. HORNOS, PATRICIA PADEREWSKI, JOÉ LUIS GARRIDO Hacia la satisfación de requisitos de la calidad de los sistemas colaborativos mediante un diseño arquitectónico.
- PARRA, J. D. Hacia una Arquitectura Empresarial basada en Servicios. Disponible en http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art143.asp
- REYNOSO, C. B. Introducción a la Arquitectura de Software, 2004. Disponible en http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.asp
- ROBERTSON, J. So, what is a content management system? , june 2003. Disponible en http://www.steptwo.com.au/papers/kmc_what/index.html
- WADOOA SOA (Service-oriented Architecture) Arquitectura Orientada a Servicios, 2007. Disponible en http://wadooa.com/doku.php/soa

Anexos

Anexo 1: Entrevista

Estimado usuario, a continuación se presentan una serie de cuestionarios que usted bebe responder con el propósito de obtener un Portal con sus preferencias, es importante sea cuidadoso a la hora de responder. Gracias por su colaboración.

1. ¿Cómo desea el diseño de la interfaz?
    a. Sin columnas _____
    b. Divido en dos columnas _____
    c. Divido en tres columnas _____

2. Desea que el portal contenga banner.
    a. Si _____
    b. No _____

¿En que posición desea que aparezcan?

c. Arriba _____
d. Abajo _____
e. Izquierda_____
f. Derecha _____

3. ¿Qué colores desea que tenga el portal?
    a. Frescos _____
    b. Cálidos _____
    c. ¿Cuáles?_____

4. ¿Qué CMS desea que se emplee en el desarrollo de la Intranet?
    a. PLone _____

b. Xoops _____
c. Drupal _____
d. Otro _____

5. ¿Qué idioma desea como primario?
   a. Español _____
   b. Ingles _____
   c. Otro _____

6. ¿Qué leguaje de programación desea?
   a. Python _____
   b. PHP _____
   c. Java _____
   d. Otro _____

7. ¿Qué Sistema Gestor de Bases de Datos desea?
   a. PostgreSql _____
   b. MySql _____
   c. Oracle _____

Anexo 2: Cuestionario

Estimado usuario, con el objetivo de determinar el grado de acercamiento sobre ¿cómo deberá estar confeccionada la intranet de la universidad? para compensar todas sus necesidades se han elaborado un conjunto de preguntas para determinar los principales puntos que el sistema deberá contar. Gracias por su colaboración.

Marque con una X aquellas opciones que usted estime que deben estar presente en la aplicación. En caso de que valore, usted puede realizar proposiciones que contribuyan en el cumplimiento de las metas trazadas.

Cuestionario

¿El portal deberá estar disponible todo el tiempo? _____

¿El portal deberá tener un diseño consistente y uniforme? _____

¿Usar colores agradables y refrescantes para el diseño? _____

¿El diseño del portal ofrece mínimas distracciones? _____

¿El portal deberá informar la ubicación del usuario? _____

¿El portal deberá permitir acceder a la ayuda desde cualquier lugar del sitio?_____

¿El portal deberá mostrar los servicios a los que puede acceder los usuarios?_____

¿El portal deberá permitir el acceso a otros sitios ?_____

¿La estructura del portal deberá ser entendible? _____

¿El portal mantiene en todo momento la misma estructura de interfaz al usuario? _____

¿El portal deberá permitir la personalización del idioma?_____

¿El contenido publicado deberá ser confiable?_____

¿La información contenida en el portal es exacta y relevante?

¿La información contenida en el portal es rica en detalles?_____

¿El portal deberá mostrar mensajes de confirmación?_____

¿La estructura del mensaje deberá ser cortés? _____

¿El tiempo de respuesta deberá ser el mínimo? _____

¿Diferentes caminos conducen al mismo sitio? _____

Glosario de Términos

Session data manager (control de sesiones): A través de este producto podemos llevar un control de sesión de los usuarios que están conectados a nuestra Web, e incluso almacenar datos relevantes de la sesión del usuario, aunque almacenar datos importantes en una sesión no es muy conveniente, debido a la facilidad con la puede acceder a esta información un intruso.

ZSQLMethods: Son los métodos que nos almacenan las consultas estáticas o dinámicas de acceso a la base de datos relacional (mysql, postgresql, oracle, etc.). Abstraen el motor de base de datos que vaya por debajo a través de un adaptador, creando una capa transparente de consultas SQL.

ZPoPyDa: Adaptador de base de datos que nos permite conectas los ZSQLMethods contra una base de datos PostgreSQL. Al igual que el producto anterior, requiere que los módulos de python de acceso a base de datos PostgreSQL estén instalados

Virtual Host Monster: Producto que nos permitirá acceder a Zope de una manera transparente a través de Apache (o con Medusa) mapeando las direcciones URL que nosotros queremos ofrecer al usuario hacia el directorio Zope el cual debe de servirla. Resumiendo, para entendernos, permite servir varios dominios virtuales a partir de una única instalación de Zope; ya sea con la ayuda de Apache o sin ella.

ZEO. Cuando las peticiones a nuestro sitio aumentan en gran medida, de manera que la carga de nuestro procesador es muy alta, podemos utilizar ZEO para distribuir la carga entre otras instancias de Zope instaladas en máquinas diferentes comunicadas entre sí, actuando a modo de cluster.

LDAP: son las siglas de *Lightweight Directory Access Protocol*. Como su propio nombre indica, es un protocolo ligero para acceder al servicio de directorio, especialmente al basado en X.500. LDAP se ejecuta sobre TCP/IP o sobre otros servicios de transferencia orientada a conexión.

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