Universidad de las Ciencias Informáticas

Facultad 3

Propuesta de arquitectura para agilizar el desarrollo de sistemas de gestión web sobre portales empresariales libres.

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

Autor(a):

Yelaine Ruz Pérez

Tutor:

Ing. Alfredo Morales Oliva

Ciudad de la Habana, Cuba

Mayo de 2009

"Año del 50 Aniversario del triunfo de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _2009_.

Yelaine Ruz Pérez

Ing. Alfredo Morales Oliva

_____

Firma de la autora

_____

Firma del tutor

DATOS DE CONTACTO

Alfredo Morales Oliva
Tutor del presente trabajo de diploma
Ingeniero en Ciencias Informáticas (2006). Categoría docente: Profesor Instructor.
amoraleso@uci.cu

Yelaine Ruz Pérez
Autora del presente trabajo de diploma
Estudiante de 5to año de la Universidad de las Ciencias Informáticas
yruz@estudiantes.uci.cu

AGRADECIMIENTOS

Agradezco a todas las personas que de forma directa o indirecta tuvieron alguna relación con el surgimiento de este trabajo, pero esencialmente agradezco a Alfredo Morales Oliva, que no solo ha sido mi tutor en la realización del mismo, me ha enseñado a defender mis principios y ha buscar mi camino, me ha hecho preguntarme si lo que estoy haciendo hoy me acerca al lugar donde quiero estar mañana, me ha impulsado como ser humano, mostrándome nuevas formas de ver la vida y a saber qué debo mantener cerca y que debo alejar de mí.

Me ha enseñado también que basta una idea para cambiar el rumbo de la vida, que los sueños y la perseverancia son una poderosa combinación para lograr lo que uno quiere.

Me ha demostrado que el poder de una persona es grande cuando no está sujeto a la pereza, y cuando confía en el resultado de lo que quiere con todas sus fuerzas.

A él mi eterno agradecimiento por ser una persona tan especial y apoyarme todos estos años.

DEDICATORIA

A mis padres que han tenido los papeles protagónicos en la mujer que soy y seré toda la vida

- A mi madre, sin ti que me has dado todo lo mejor y más especial, esto no lo hubiera podido ni soñar.
- A mi padre gracias por llenarme de argumentos para enfrentar la vida, hoy quiero decirte que acabó la carrera de los 400 metros, ahora comienza una mucho más larga y compleja.

A mis dos hermanos y mi sobrino que se hagan hombres de bien y deseo que la vida siempre los guíe por el camino correcto.

A mi hermana, gracias por darme tu cariño y ayudarme tanto.

A mi abuela Ondina, por ser una fuente de inagotable sabiduría y amor para mí.

A mi abuelo César (mi querido cui-cui), por socorrerme siempre y darme su cariño tan incondicional y transparente.

A mi abuela Juana (mima) por abrirme sus puertas y darme su cariño.

A mi abuelo Jesús que ya no está pero nunca lo olvido, el también tuvo su papel en todo.

A Veylys que más que una amiga ha sido mi hermana todos estos años de la carrera y espero que así sea para el resto de la vida.

A Ray gracias por su apoyo y ayuda en momentos bien difíciles.

A la Revolución, a los que la idealizaron y la consiguieron, a todos esos que desde sus sueños hicieron posible esta realidad.

A todas las personas que han pasado por mi vida y que no están en este instante presentes pero que sin proponérselo tuvieron influencia en este momento.

RESUMEN

Actualmente en el mundo del desarrollo de software se está observando un aumento en la complejidad de los proyectos debido a que las necesidades de los usuarios aumentan en complejidad y cantidad. En la Universidad de las Ciencias Informáticas[1](UCI) existe un aumento en pedidos de software de gestión por lo que se hace necesaria una arquitectura que permita el desarrollo de los mismos de forma ágil, con herramientas que presenten una curva de aprendizaje baja y permitan una alta productividad en los equipos de desarrollo.

Las arquitecturas existentes en la UCI necesitan muchos frameworks integrados para funcionar y aun así no cubren los requisitos más importantes para un dominio de software pues solo se centran en resolver un problema en específico y limitan de esta forma la reutilización.

En el presente trabajo se estudian las tendencias actuales de la arquitectura de software, la tecnología Java y las herramientas base para el desarrollo de sistemas de gestión web. Se hace una propuesta de la arquitectura, brindando además una configuración completa de las herramientas que la conforman y que tributan a la soberanía tecnológica, posteriormente se evalúa la arquitectura propuesta a través del método SAAM[2].

Luego de la evaluación se observa que la arquitectura propuesta posibilita agilizar el desarrollo de los sistemas de gestión web en más de un 50%, eleva la productividad en los equipos de desarrollo, fomenta la reutilización de componentes de alto nivel, no fuerza a los arquitectos a usar una herramienta específica y facilita la implantación de un modelo de fábrica de software.

Palabras claves: Agilidad, arquitectura, componentes, Java, productividad, reutilización, software de gestión.

[1] Universidad cubana donde se estudia la carrera de Ingeniero en Ciencias Informáticas y donde se origina la problemática a la que se le da solución en el presente trabajo.

[2] SAAM (Software Architecture Analysis Method**,** del inglés) primer método surgido para la evaluación de arquitecturas de software, posibilita evaluar de forma rápida varios atributos de calidad enfocándose en la enumeración de un conjunto de escenarios.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En el momento actual de las empresas vinculadas directamente a la producción de software es muy natural que se logren avances orientados a simplificar el desarrollo de proyectos. Con este objetivo son creados nuevos frameworks especializados en tareas específicas dentro de la fase de construcción, también surgen entornos de desarrollo que hacen la programación mucho más cómoda, llegando a lograr entre los desarrolladores una extensión muy alta.

Las aplicaciones para la realización de proyectos web del mismo modo suscitan mejoras: el uso de Gestores de Contenido o CMS es cada vez mayor, pero además de otro tipo de herramientas como son los Gestores Documentales o Repositorios de Documentos y los Gestores de Portales.

El entorno actual de los proyectos y en especial el relacionado con las tecnologías de la información se está caracterizando cada vez más por tener un alto grado de complejidad, necesidad de una aguda creatividad e innovación y complejos escenarios de transferencia tecnológica, además el trabajo en equipo y la toma de decisiones necesitan de una mayor interdisciplinariedad. Todo esto influye en el desarrollo y soporte de las aplicaciones convirtiéndolos en temas más complejos.

En el entorno actual, las grandes empresas del sector de las tecnologías de la información, trabajan constantemente en la actualización y consolidación de sus técnicas, herramientas y metodologías de desarrollo de software, lo que les posibilita afianzar sus mercados y lograr una alta eficiencia operacional en los proyectos que realizan.

Según la empresa consultora Gartner en el informe presentado en el Symposium ITPro del año 2004 [13] predice que los avances tecnológicos determinantes hasta el presente año serán aquellos que resuelvan los siguientes problemas:

- Infraestructura: debe ser robusta, fiable e invisible.
- Desarrollo de aplicaciones o nuevos sistemas de información: desarrollo rápido, menos caro y menos variable.
- Mantenimiento de aplicaciones o sistemas de información: las aplicaciones deben ser lo más flexibles y escalables posible. Se debe pensar que dentro del ciclo de vida de una aplicación su mantenimiento puede suponer entre el 60% y el 80% del gasto total.
- Despliegue de aplicaciones: debe ser seguro, confiable y cubrir un ámbito interempresarial.

Entre los principales problemas que pueden incidir en que las aplicaciones que se desarrollen no estén a la altura que los tiempos exigen se destacan:

- La falta de gestión del conocimiento en la empresa.
- La falta de predictibilidad del tiempo y recursos necesarios para la ejecución del proyecto.
- La escasa planificación y análisis de los requisitos del cliente.
- Poca capacidad para la gestión del cambio
- Alto ciclo para el cambio tecnológico.

La Universidad de las Ciencias Informáticas (UCI) como modelo de universidad que vincula la docencia con la producción de software, a escala industrial, no escapa de muchos de estos problemas y necesita seguir el hilo de los acontecimientos que anteriormente se mencionan y adaptarse rápidamente a dichos cambios para lograr competir con las principales empresas del sector a nivel internacional.

Una de las aristas para resolver los problemas antes mencionados es lograr que el proceso de desarrollo de los sistemas sea cada vez en el menor tiempo posible, sin que esto vaya en el decremento de la calidad del producto final. Para ello las empresas del sector han comenzado a buscar distintos tipos de soluciones tecnológicas o administrativas que conlleven a lograr dicho fin.

La UCI cuenta con un proceso productivo que tiene sus particularidades específicas. Entre las principales se encuentra que su principal fuerza de trabajo son profesores y estudiantes de varios años de la carrera. Esto provoca que no siempre se cuente con especialistas en cada uno de los roles del proyecto y haya que capacitar a los equipos de proyecto cuando se asume un nuevo trabajo. Por tal razón se hace necesario que la curva de aprendizaje de las herramientas que se utilicen para el desarrollo de los productos de software sea lo más leve posible, que la productividad que generen las mismas sea alta y no sea necesario una gran especialización para trabajar con ellas.

Entre las mayores demandas que recibe la UCI de proyectos informáticos están los clasificados en el área de proyectos de gestión, los cuales en la mayoría de los casos definen como uno de sus requerimientos que el mismo sea una aplicación web, sin embargo aún no existen arquitecturas estandarizadas para el desarrollo de este tipo de software, lo que trae consigo que los equipos de cada proyecto tengan que definir su propia arquitectura, implicando que en muchos casos no se puedan reutilizar eficientemente los componentes ya generados en otros proyectos de la propia Universidad o en el peor de los casos que la

arquitectura definida no sea robusta y flexible provocando que el proyecto no pueda cumplir las expectativas del cliente final.

Por otra parte las empresas desarrolladoras de software buscando la manera de ser más eficiente en sus producciones están creando modelos de producción donde la reutilización y el desarrollo ágil sean las principales premisas. El desarrollo de sistemas basado en Java está teniendo un gran auge en el mundo empresarial. Una de las tendencias actuales en este sentido está relacionada con el desarrollo de los portales empresariales o portal server, los cuáles conciben entre otras cosas la fácil integración de aplicaciones ya existentes en la organización, así como la incorporación de nuevos productos de una manera más ágil y estandarizada. Estos portales empresariales están provistos de una arquitectura base que garantiza una alta flexibilidad y escalabilidad en las aplicaciones de gestión web, además de garantizar una seguridad granular de forma eficiente, posibilitando de esta forma un punto de acceso único para todas las aplicaciones que necesitan los empleados, los proveedores o los clientes.

Sin embargo, estas últimas tendencias de las tecnologías y del desarrollo de las arquitecturas de software para aplicaciones de gestión en Java no son muy conocidas en la Universidad y aún no se utilizan sus potencialidades en el desarrollo de los proyectos actuales.

Todo esto además está condicionado por la necesidad en la UCI de tener una completa soberanía tecnológica para todos los productos que sean desarrollados. Por esta razón es preciso que las herramientas que se utilicen en cualquier arquitectura cumplan con las licencias del software libre para que de esta forma puedan ser asimiladas y mantenidas completamente por los equipos de desarrollo.

Por todo lo anteriormente descrito, el problema a resolver es el siguiente:

Problema

¿Con qué arquitectura se puede agilizar el desarrollo de sistemas de gestión web utilizando herramientas libres?

Objeto de estudio

Las arquitecturas de software.

Campo de acción

Las arquitecturas de software para el desarrollo de sistemas de gestión web.

Objetivo general

Proponer una arquitectura que permita agilizar el desarrollo de sistemas de gestión web.

Tareas de investigación

- Analizar el estado del arte de las arquitecturas, estándares y herramientas bases para el desarrollo de sistemas de gestión sobre la web.
- Proponer una arquitectura, con las herramientas, frameworks, entornos de desarrollo y gestor de base de datos libres que sean flexibles, portables, escalables y robustos para el desarrollo de sistemas de gestión web de forma ágil y productiva.
- Evaluar la factibilidad de la arquitectura propuesta.

Hipótesis

Si se utilizan herramientas libres de alta productividad y complejidad de aprendizaje bajo, que permitan una alta flexibilidad y escalabilidad en el desarrollo de los sistemas de gestión web se podrán acortar los tiempos en el desarrollo, se aumentará la productividad de los miembros de los equipos de proyectos de la UCI y se podrá reutilizar de forma más eficiente el conocimiento generado en los proyectos de la Universidad.

Aporte práctico.

Proponer una arquitectura de software, válida para aplicar inmediatamente en los proyectos de la Universidad, usando las últimas tendencias de las tecnologías de Java para el desarrollo de sistemas de gestión web lo que propiciará una mayor productividad de los miembros de los equipos de desarrollo, una mayor agilidad en las producciones, una flexibilidad, portabilidad y escalabilidad en las aplicaciones desarrolladas y una alta reutilización de los componentes desarrollados en los proyectos de la Universidad.

Diseño metodológico

Se emplearon los métodos empíricos de la observación y la experimentación para la investigación. La observación científica fue selectiva, dado que se dirige hacia las últimas tendencias del desarrollo de sistemas web utilizando las tecnologías de Java; sistémica, porque este trabajo ha sido consecuencias del análisis continuo por varios años de los cambios ocurridos en los distintos estilos arquitectónicos; y fue también objetiva, sustentada en la imparcialidad. Además, la observación científica realizada se considera más bien externa, tomando como base las directrices que se encuentran en este ámbito a nivel mundial; pero también vale acotar que se realizó una observación interna, aunque en menor grado, relacionada con los intentos de crear arquitecturas web en Java en la Universidad.

También se utilizó el método experimental el cuál posibilitó probar diferentes combinaciones de técnicas y herramientas posibilitando evaluar de una forma efectiva cada una de las variantes y de esta forma escoger la propuesta final.

El presente trabajo se encuentra estructurado por tres capítulos y varios anexos, esto incluye todo lo relacionado con el trabajo investigativo y práctico realizado:

- **Capítulo 1: Fundamentación teórica.** Definición del marco teórico de la investigación. Se realiza un estudio del estado del arte de las arquitecturas para el desarrollo de software de gestión web y todos los conceptos asociados a las mismas como modelos, estilos y patrones arquitectónicos, también se estudian los avances de la tecnología Java y de las herramientas base para el desarrollo de sistemas de gestión web que han surgido en los últimos tiempos.
- **Capítulo 2: Propuesta de arquitectura para el desarrollo de sistemas de gestión web.** Se propone una arquitectura, con una herramienta base para el desarrollo de sistemas de gestión web, frameworks, entornos de desarrollo y gestor de base de datos libres que logran flexibilidad, portabilidad, escalabilidad y robustez en las aplicaciones que se construyen sobre la arquitectura de forma ágil y productiva. Finalmente se muestra como configurar el entorno de trabajo para el funcionamiento de todas las herramientas que conforman la propuesta.
- **Capítulo 3: Evaluación de la arquitectura propuesta.** En este capítulo se realiza una evaluación de la propuesta arquitectónica con el método SAAM a través de un conjunto de escenarios a los que la arquitectura puede estar expuesta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

La arquitectura de software tiene una responsabilidad directa en el éxito del sistema. En este capítulo se realiza un estudio del estado del arte de las arquitecturas para el desarrollo de software de gestión sobre la web y de todos los conceptos asociados a las mismas, también de la tecnología Java y de las herramientas base para el desarrollo de este tipo de sistemas que han surgido en los últimos tiempos.
Se hace necesario conocer y entender cuál es el significado de una arquitectura de dominio específico para utilizarlo como basamento en el desarrollo del trabajo.
Son revisadas también las diferentes herramientas de desarrollo rápido existentes y se vislumbra en este capítulo a los portal server y portlets como solución a la problemática encontrada en la UCI.

1.2 Necesidad de agilizar las producciones de software.

Una parte de los esfuerzos en el desarrollo de software actualmente consiste en la construcción de herramientas que permitan un rápido desarrollo de aplicaciones informáticas, que además de permitir al desarrollador trabajar en ambientes cómodos olvidando los manejos a bajo nivel, también le permitan reutilizar muchas funcionalidades ya probadas que él tenga en su poder o que las herramientas en sí puedan brindarle.
La vida moderna a nivel mundial se está caracterizando cada vez más por ser automatizada. Surgen con frecuencia nuevas aplicaciones software y artefactos que hacen a los hombres desarrollarse rápidamente. Cada nueva aparición de un producto asegura su mejora o superioridad con respecto a otros, creando a nivel mundial una competencia constante por ganar el mercado internacional, pero solo triunfan en este medio las empresas donde los niveles y procesos de producción sean acelerados pero con una calidad asegurada. [7]
Las organizaciones productoras de software; a cualquier escala; a pesar de tener sus características propias con respecto a otras industrias no quedan excentas de esta competencia, en efecto la informática en estos momentos está siendo una de las ramas con más desarrollo a nivel mundial, muchos de los equipos utilizados por el hombre necesitan de un software para poder funcionar, estos softwares evolucionan en cortos períodos de tiempo, brindando cada vez más funcionalidades o mejorando las que

ya tenían en un inicio, de esta forma la relación entre productores y consumidores se convierte en un ciclo que está muy lejos de desaparecer.

Los grandes movimientos corporativos de esta industria, junto a la rápida evolución de los productos y la aparición de nuevas aplicaciones, dispositivos y sistemas de trabajo, seguirán contribuyendo a mantener un escenario de alto grado de intensidad competitiva en la industria del software.

Las necesidades de los usuarios aumentan en exigencia y con ellas los procesos de la informática entran en un cambio constante y hacen surgir nuevas ideas, es premisa para los productores de software dar respuesta a sus clientes aunque esto conlleve, en muchos casos, imponerse nuevos retos que desembocan en novedosas soluciones. [7]

Debido a lo dicho anteriormente es que las personas y centros dedicados al mundo de la computación necesitan que el tiempo de desarrollo de sus productos sea cada vez más corto. No todas las herramientas existentes actualmente brindan esta posibilidad pero ya existen avances que están ayudando de manera significativa.

1.3 Arquitectura de software

El término arquitectura de software (AS) tiene sus orígenes en los años 60 del siglo XX, pero no es hasta la década de los 90 cuando alcanza gran popularidad.

La necesidad del manejo de la arquitectura de un sistema de software nace con los sistemas de mediana o gran envergadura, que se proponen como solución para un problema determinado. En la medida que los sistemas de software crecen en complejidad, bien sea por número de requerimientos o por el impacto de los mismos, se hace necesario establecer medios para el manejo de esta complejidad [18].

La técnica es descomponer el sistema en piezas que agrupan aspectos específicos del mismo, producto de un proceso de abstracción y que al organizarse de cierta manera constituyen la base de la solución de un problema en particular.

Varias instituciones y autores ha dado su definición de AS y a continuación se relacionan algunas consideradas como las más completas u oficiales en el mundo de las empresas productoras de software.

La definición oficial de arquitectura de software que da la IEEE Std 1471-2000, plantea: La arquitectura de software es la organización fundamental de un sistema formada por sus componentes[3], las relaciones entre ellos y el contexto en el que se implantarán y los principios que sustentan su diseño y evolución. [19]
También Rumbaugh, Jacobson y Booch en su libro El Proceso Unificado de Desarrollo de Software dan una definición sobre la AS y plantean que es el conjunto de decisiones significativas acerca de la organización de un sistema software, la selección de los elementos estructurales a partir de los cuales se compone el sistema y las interfaces entre ellos, junto con su comportamiento, tal y como se especifica en las colaboraciones entre esos elementos, la composición de estos elementos estructurales y de comportamiento en subsistemas progresivamente mayores, y el estilo arquitectónico que guía esta organización (…) [23]
La práctica ha demostrado que resulta importante extender el concepto de arquitectura de software considerando los requerimientos y restricciones del sistema.
La arquitectura de software puede considerarse entonces como el "puente" entre los requerimientos del sistema y la implementación. [18]
La AS tiene como objetivo primario aportar elementos para ayudar a la toma de decisiones y proporciona conceptos y un lenguaje común que permiten la comunicación entre todos los que tienen parte en el proceso del desarrollo del software, también describe diversos aspectos del software de una forma comprensible utilizando modelos o vistas.
Los modelos o vistas de una arquitectura pueden expresarse mediante uno o varios lenguajes, el más obvio es el lenguaje natural, pero existen otros lenguajes, los cuales son apropiados únicamente para el modelado, dentro de estos últimos se encuentra el Lenguaje Unificado de Modelado (UML, de sus siglas en inglés), el mismo ha sido adoptado de forma extendida para la mayoría de los modelos.
Con el surgimiento de un nuevo proyecto para hacer un software no se crea una nueva arquitectura, lo que se realiza es la adopción de una arquitectura conocida valorando sus ventajas e inconvenientes frente al nuevo problema planteado. En caso de tener acumulada cierta experiencia en la construcción de software, entonces según los resultados obtenidos anteriormente, se podrá definir cual es la mejor AS a aplicar en cada caso.
Las arquitecturas más conocidas son:

[3] 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.

- Monolítica: donde el software se estructura en grupos funcionales muy acoplados.
- Cliente-servidor: donde el software reparte su carga de cálculos en dos partes independientes pero sin reparto claro de funciones.
- Arquitectura por niveles o por capas: esta constituye una especialización de la arquitectura cliente-servidor, donde la carga se distribuye en partes con un reparto claro de las funciones y donde cada capa tiene relación con la siguiente.
- Arquitectura orientada a servicios: Define la utilización de servicios para dar soporte a los requisitos del negocio.

La AS representa las bases sobre las que se desarrollará el sistema, brinda una estructura que soporta las soluciones a cada tipo de problema que pueda aparecer en el desarrollo, asegura que los requerimientos más importantes puedan ser evaluados e implementados, permite flexibilidad en el sistema pues facilita la ejecución de futuros cambios y promueve la reutilización de componentes existentes como librerías de clases, sistemas legados y de aplicaciones de terceros.

La AS provee una descripción de alto nivel de los subsistemas que comprenden la arquitectura del sistema. Está atada a comprender cómo las mayores operaciones del negocio son soportadas. Ayuda a la planificación de recursos y la asignación de tareas, debido a que el trabajo de desarrollo puede ser particionado a través de los subsistemas y los esfuerzos de desarrollo individual pueden proceder en paralelo. Cuando los equipos de desarrollo están geográficamente dispersados puede minimizar el número de interacciones requeridas.

Puede observarse que al hablar de arquitectura de software, de forma general, se hace alusión a la especificación de la estructura del sistema, entendida como la organización de componentes y relaciones entre ellos; los requerimientos que debe satisfacer el sistema y las restricciones a las que está sujeto, así como las propiedades no funcionales del sistema y su impacto sobre la calidad del mismo; las reglas y decisiones de diseño que gobiernan esta estructura y los argumentos que justifican las decisiones tomadas. [19]

1.3.1 Estilos arquitectónicos

Cuando se define la arquitectura de software hay que decidir cual o cuales estilos arquitectónicos serán utilizados en la misma. La aplicación de dichos estilos mejora o disminuye la satisfacción de los atributos

de calidad del sistema, es decir, la aplicación de un estilo depende en gran medida de los requisitos del sistema y de como se dará respuesta a los mismos.

Se define 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. Así mismo, se considera como un tipo particular de estructura fundamental para un sistema de software, conjuntamente con un método asociado que especifica cómo construirlo. Éste incluye información acerca de cuándo usar la arquitectura que describe, sus invariantes y especializaciones, así como las consecuencias de su aplicación." [8]

Existe una gran variedad de estilos arquitectónicos las cuales tienen diversas clasificaciones entre las que se encuentran:

- Estilos de flujo de datos
- Estilos centrados en datos
- Estilos de llamada y retorno
- Estilos de código móvil
- Estilos heterogéneos
- Estilos peer-to-peer

1.3.2 Patrones de arquitectura

Se define al 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: [8]

- 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. Esta abarca:
  - Estructura con componentes y relaciones
  - Comportamiento a tiempo de ejecución: aspectos dinámicos de la solución, como la colaboración entre componentes, la comunicación entre ellos, etc.

Los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen

reglas y guías para organizar las relaciones entre ellos. Ayudan a especificar la estructura fundamental de una aplicación. Cada actividad de desarrollo es gobernada por esta estructura; por ejemplo, el diseño detallado de los subsistemas, la comunicación y colaboración entre diferentes partes del sistema.
Cada patrón de arquitectura ayuda a conseguir una propiedad específica en el sistema global. Los patrones que dan soporte a características similares se agrupan en una misma categoría.

1.3.3 Arquitectura cliente-servidor

Esta arquitectura se encuentra dentro de la clasificación de estilo de llamada y retorno. El cliente y el servidor generalmente están localizados en diferentes sistemas, sin embargo pueden encontrarse en el mismo sistema. El cliente es la entidad que hace la petición por un servicio y el servidor es la entidad que provee el servicio correspondiente a la petición. El servicio debe procurar el resultado, el cual es retornado al cliente.
Los clientes pueden tener varias clasificaciones, unos además de la lógica de presentación contienen gran parte de la lógica de negocio de la aplicación, los demás clientes manejan usualmente sólo la lógica de presentación lo que adiciona grandes ventajas como permitir que futuros cambios de negocio en la aplicación no afecten al cliente. [14]

1.4 Estilo de desarrollo por modelos. MDA

La Arquitectura Dirigida por Modelos; en inglés Model Driven Architecture (MDA); es una especificación detallada por el OMG (Object Management Group)[4] que integra diferentes especificaciones y estándares definidos por la misma organización con la finalidad de ofrecer una solución a los problemas relacionados con los cambios en los modelos de negocio, la tecnología y la adaptación de los sistemas de información a los mismos. [10]
MDA permite el despliegue de aplicaciones informáticas, las cuáles pueden ser diseñadas mediante el uso de UML y otros estándares. El diseño con dichos estándares da como resultado un grupo de modelos que son sometidos a varias transformaciones en dependencia del objetivo perseguido con los mismos pero de ellos se puede obtener código desplegable y otros modelos para una tecnología específica.

[4] Object Management Group: Es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA. Es una organización sin ánimo de lucro que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para las mismas. El grupo está formado por compañías y organizaciones de software como: Hewlett-Packard (HP), IBM, Sun Microsystems y Apple Computer.

En un modelo, que es independiente de la plataforma, se expresan las especificaciones y funcionalidades de la aplicación, esta independencia ayuda a olvidar las características específicas de las distintas plataformas.

Al modelo obtenido se le aplican un grupo de transformaciones y trazas para lograr la generación automática de código para una plataforma específica, con esto se logra una independencia entre la capa de negocio y la tecnología que se emplea, además de que se hace mucho más simple introducir nuevas funcionalidades o cambios en los procesos de negocio sin costes adicionales, ni esfuerzo necesario por parte del equipo de trabajo, pues los mismo no repercutirían en todos los niveles, las modificaciones se realizarían a nivel de modelo y estos se propagarían a la aplicación, es decir, no interviene la plataforma en ello. Sin duda esta forma de introducir los cambios o nuevas funcionalidades aumenta la productividad y no atrasa de forma dramática el cronograma de trabajo del personal involucrado en el proyecto, punto este que se hace cada vez más importante debido a las altas demandas de los clientes.

MDA se apoya sobre los siguientes estándares para llevar a cabo su función: UML, MOF, XMI, CWM, SPEM, ASL y QVT.

En MDA el primer modelo generado es el Modelo Independiente de la Computación (CIM), en el se modelan los requisitos del sistema, sirviendo de esta forma para entender el problema y como base a los demás modelos que serán generados. Este modelo puede ser uno solo o la unión de varios modelos de este tipo.

Algunas de las características que presenta este modelo se listan a continuación:

- Se centra en los requerimientos y representa el nivel más alto del modelo de negocios.
- Transciende a los sistemas pues cada proceso de negocio interactúa con trabajadores humanos y/o componente de máquina.
- Describe solamente aquellas interacciones que tienen lugar entre los procesos y las responsabilidades de cada trabajador, sea o no humano.
- Su objetivo fundamental que cualquiera que pueda entender el negocio y los procesos del mismo pueda comprenderlo, ya que éste evita todo tipo de conocimiento especializado o de sistemas.

Los requisitos recogidos en el CIM han de ser trazables a lo largo de los modelos PIM y PSM que los implementan. [10]

Como se ha dicho anteriormente el CIM abre paso a los demás modelos que deberán ser generados. A partir del CIM se crea el Modelo Independiente de la Plataforma (PIM), este modelo ya muestra descripciones del sistema pero no hace ninguna referencia a la plataforma, presenta especificaciones de la empresa, informaciones y la distribución de los procesos en la misma.
Algunas de las características que presenta este modelo se listan a continuación:

- Representa el modelo de procesos de negocio a ser implementado.
- Modela los procesos y estructuras del sistema, sin hacer ninguna referencia a la plataforma en la (o las) que será desplegada la aplicación.
- Ignora los sistemas operativos, los lenguajes de programación, el hardware y la topología de red.

El PIM se ajusta a uno o varios estilos de arquitecturas [10]. Luego de elaborar el PIM y revisar los estilos de arquitectura a los que se acoja, el arquitecto deberá seleccionar la plataforma o las plataformas necesarias para cumplir con todos los requisitos del sistema.
El estándar MDA permite la obtención de un tercer modelo llamado Modelo Específico de la Plataforma (PSM), la forma de obtenerlo es aplicando mapas que contienen reglas de transformación en dependencia de la plataforma con la que se vaya a trabajar.
Los mapas incluyen la transformación de tipos de valores para la conversión desde el PIM al PSM, los metamodelos y sus reglas para la transformación en tipos de valores existentes en las plataformas. [10]
El PSM obtenido especifica la forma en que el sistema usa la plataforma sobre la que se basó el mapa para generarlo, si el modelo proporciona la información necesaria para construir y poner en marcha el sistema entonces será una implementación.
Algunas de las características que presenta este modelo se listan a continuación:

- Representa la proyección de los PIM en una plataforma específica.
- Los modelos PSM deben colaborar entre sí para una solución completa y consistente.
- Tiene que lidiar explícitamente con los sistemas operativos, los lenguajes de programación y las plataformas.

MDA logra una clara separación de responsabilidades. Por un lado, se modelan los PIM, que representan los modelos del negocio, y por otro lado, los PSM con las pautas tecnológicas. Esto permite que ambos modelos puedan evolucionar por separado.

Existen herramientas que tienen automatizado todo el proceso por el que transitan los diferentes modelos. Las herramientas que soportan MDA pueden ofrecer diferentes características, entre las que son deseables al menos las siguientes: [10]

- Creación de modelos de datos entidad-relación partiendo del modelo CIM:
    - Conservando todas las restricciones especificadas en el modelo: claves primarias y claves externas, multiplicidad de las relaciones (n:m, 1:n, 0:n, …), longitud de atributos y precisión de los mismos.
    - Manejo de disparadores.
- Generación y modelado de PIM desde modelos entidad-relación:
    - Modelos de clases con objetos estereotipados.
    - Uso de diagramas de actividad y otros elementos que demuestren comportamiento (diagramas de colaboración, secuencia, estados,...)
    - Uso de OCL[5] para la especificación de restricciones.
    - Soporte para valores etiquetados en las clases de datos, para facilitar la internacionalización del producto.
- Generación de PSM:
    - Elección de la plataforma para la que generar el PSM (EJB/JDO, .NET, Corba…) y elección de la tecnología de cliente (Struts, JSP, JSF, AJAX, XMLC, Swing, SWT,…).
    - División del modelo de datos y la interfaz gráfica de usuario en paquetes diferentes.
    - Generación de ficheros de propiedades (I18n).
    - Generación correcta de los diagramas de actividad apropiados.
    - Generar clases de prueba.
- Adaptación del PSM:
    - Poder introducir código suplementario al generado que permita completar la aplicación. Este código debe ser mantenido en un repositorio para evitar tener que reescribirlo en sucesivas generaciones del modelo.
- Generación de la aplicación:

[5] OCL (*Object Constraint Language)*, es un lenguaje de restricciones de objetos que permite a los desarrolladores de software escribir restricciones sobre modelos de objetos. [9]

- Generación del esquema de base de datos.
- Generación de código.
- Generación de Makefiles, ANT, u otros scripts para la construcción del código.
- Producción de ficheros de configuración.
- Producción de documentación.

En la búsqueda de agilizar las producciones se tiene como opción MDA y es real que aplicándolo es posible reducir drásticamente los esfuerzos en el desarrollo de productos de software pues reduce la carga de trabajo para una buena parte del equipo de desarrollo.
Este adelanto en el desarrollo se puede ver afectado si no se tiene dentro del equipo alguna persona especialista que entienda perfectamente el código generado, esta es una de las desventajas de este modelo, el código generado no es fácilmente mantenible provocando esto que la necesidad de modificaciones en la codificación se convierta en una tarea de complejidad.
Otro problema que se presenta es que todas las herramientas que soportan MDA hacen una interpretación distinta del modelo provocando que el código generado no siga un estándar.

1.5 Arquitectura de software de dominio específico

Un dominio es un área de interés, usualmente representando un espacio del problema que es un subconjunto de una o más disciplinas. [28]
Un dominio es definido por un conjunto de problemas o funciones comunes que las aplicaciones en ese dominio pueden resolver o hacer.
La idea básica de la arquitectura se software de dominio específico (DSSA) es hacer práctica la reutilización de software, enfocándose en los problemas específicos de dominios de software y desarrollando soluciones basadas en componentes para estos dominios de problemas. Estos componentes son genéricos y necesitan ser configurados en el momento que la aplicación es implementada.

DARPA[6] definió 4 pasos claves para las DSSA: [28]

- Un modelo del dominio es desarrollado para establecer una terminología estándar y una semántica común para el dominio del problema.
- Un conjunto de requerimientos de referencia para todas las aplicaciones dentro del dominio del problema que es desarrollado.
- Una arquitectura de referencia es definida para una solución genérica, estandarizada y basada en componentes del sistema para anticipar los requerimientos del cliente.
- Nuevas aplicaciones son desarrolladas para determinar requerimientos específicos de la aplicación, seleccionar o generar componentes particulares y ensamblarlos acorde a la arquitectura de referencia.

La arquitectura de referencia puede incluir componentes concretos que deberían ser embebidos en todas las aplicaciones que derivan de ella. Además es adaptada para resolver los requerimientos particulares de los clientes.
La actividad principal es ajustar a la medida, la arquitectura de referencia, usando los requerimientos de la aplicación para producir la arquitectura específica para la aplicación.
En el presente trabajo el dominio específico considerado incluye todas las aplicaciones de gestión web sobre portales empresariales libres pues cuando se construye una nueva herramienta se está resolviendo una problemática que comparte características comunes con las demás aplicaciones del mismo entorno.

1.6 La arquitectura de software y la calidad

En la actualidad la calidad surge como una necesidad para poder competir, pues da a las empresas; de diferentes sectores; mayores posibilidades de triunfar.

[6] DARPA: Agencia de Investigación de Proyectos Avanzados de Defensa (DARPA por sus siglas en inglés) es una agencia del Departamento de Defensa de los Estados Unidos responsable del desarrollo de nuevas tecnologías para uso militar.

La Calidad de Software para Pressman [40] es “la concordancia con los requisitos funcionales y de rendimiento establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado de forma profesional”.
La ISO/IEC (*Intenational Standart Organitation* u Organización Internacional de Estándares, en español) define a la calidad de software como la totalidad de rasgos y atributos de un producto de software que le apoyan en su capacidad de satisfacer sus necesidades explícitas o implícitas [22].
Las definiciones de calidad expuestas anteriormente concuerdan en un mismo criterio: la calidad es la capacidad del software para satisfacer las necesidades de los usuarios.
Las empresas para lograr la calidad requerida elaboran actividades periódicas que permiten el control de obtención de la misma, a este conjunto de actividades se le ha llamado Aseguramiento de la Calidad.
También Pressman [40] ha dado una definición para el Aseguramiento de la Calidad y plantea que puede tener las siguientes actividades: evaluaciones en las etapas del desarrollo, auditorías y revisiones, estándares que se aplican al proyecto, mecanismos de medida (métricas), métodos y herramientas de análisis, diseño, programación y prueba, y documentación y control de software.
Sin duda el Aseguramiento de la Calidad propicia atributos y características que influyen significativamente en la Calidad de Software.

1.7 Arquitectura para software empresarial

Una arquitectura de desarrollo de software empresarial ha de ofrecer una serie de servicios a los arquitectos y desarrolladores encaminados a facilitar la implementación de aplicaciones empresariales, al tiempo que brinda la mayor cantidad de funcionalidades a los usuarios. Normalmente tiene los siguientes requisitos:

- Escalabilidad: ha de ofrecer una buena escalabilidad tanto vertical como horizontal, de modo que se pueda aumentar la estructura tecnológica sin necesidad de realizar modificaciones.
- Mantenibilidad: ha de permitir la realización de modificaciones a los componentes existentes sin que se modifique el comportamiento del sistema.
- Fiabilidad: Grado en el que un programa se espera que realice su función con una precisión requerida.

- Disponibilidad: debe tener soporte de arquitecturas tolerantes a fallos y sistemas de redundancia y todo aquello que asegure que el sistema empresarial estará siempre disponible.
- Extensibilidad: ha de ser posible la añadidura de nuevos componentes y capacidades al sistema sin que se vean afectados el resto de los componentes.
- Manejabilidad: los sistemas han de ser fácilmente manejables y configurables.
- Seguridad: se han de tener buenos sistemas de seguridad, tanto a nivel de autenticación, como de autorización y de transporte de datos.
- Rendimiento: se han de ofrecer automáticamente soporte general de mecanismos que permitan aumentar el rendimiento de manera transparente al usuario.

La importancia que pueda tener una arquitectura empresarial frente a otra es el cumplimiento de todos estos requisitos de forma automática, logrando que el desarrollo de sistemas sea más productivo.

1.8 Especificaciones de Java para el desarrollo de software

JSR

Es el acrónimo de Java Specification Request. Cuando una persona o entidad cree que es necesaria la presencia de una determinada tecnología dentro de las plataformas basadas en Java, lo que hace es crear un JSR y presentarlo para su aprobación. Dentro de este documento se relata por qué es necesaria dicha tecnología y por qué no se pueden abordar los problemas con las tecnologías existentes. Si dicha petición se aprueba entonces se crea una especificación, un documento en el cual se describe dicha tecnología, sus partes, las relaciones entre las mismas y los roles de las personas que la usarán. Además de la realización de este documento, el equipo encargado del desarrollo de la especificación ha de proporcionar un test de compatibilidad y una implementación de referencia. [46]

JCP

Java, siempre fue criticado por ser exclusivamente de Sun Microsystems. A raíz de todas esas críticas, Sun, el 8 de diciembre de 1998, decidió dar la posibilidad a todo el mundo de participar en la evolución de Java y de todas las plataformas que se basan en el lenguaje, con esa intención se creó el JCP[7].

[7] JCP: un organismo formado por alrededor de 500 empresas, asociaciones y particulares cuyo objetivo es asegurar la evolución de las plataformas basadas en Java

Para entrar a formar parte del JCP las empresas e individuales han de pagar una cuota anual a Sun Microsystems. Cualquiera puede participar en el desarrollo de cualquier parte de la plataforma, previo pago de esa cuota, y por supuesto, si el comité de la especificación en la que quiere participar considera que esa persona o entidad tiene los conocimientos necesarios para aportar algún beneficio.
Una de las labores del JCP es la de controlar la evolución de las diferentes especificaciones que forman las plataformas basadas en Java. Este organismo es el encargado de decidir que especificaciones se aprueban y de controlar las fases por las que pasan. [47]

1.9 La plataforma Java 2 Enterprise Edition

Java 2 Enterprise Edition (J2EE, Java 2 edición empresarial) es una especificación JSR; concretamente el JSR-151; que define una plataforma de desarrollo empresarial. La plataforma J2EE está formada por varios componentes:

- Un conjunto de especificaciones.
- Un test de compatibilidad, el J2EE Compatibility Test Suite (CTS).
- La implementación de referencia de J2EE.
- Un conjunto de guías de desarrollo y de prácticas aconsejadas denominadas J2EE BluePrints.

Debido a la necesidad del mercado de desarrollo de software de contar con medios y herramientas que permitieran construir aplicaciones corporativas es que se diseñó la plataforma abierta y estándar de Java. Se le denomina plataforma porque proporciona técnicas específicas que describen el lenguaje, pero además, provee las herramientas para implementar productos de software basados en dichas especificaciones.
La plataforma J2EE ha sido diseñada para aplicaciones distribuidas con base en componentes o unidades funcionales de software que interactúan entre sí para formar parte de una aplicación empresarial J2EE. Un componente de esta plataforma debe formar parte de una aplicación y ser desplegado en un contenedor, o sea, en la parte del servidor J2EE que le ofrece al componente ciertos servicios de bajo nivel y de sistema, tales como seguridad, manejo de concurrencia, persistencia y transacciones. Como se puede apreciar, J2EE no es solo una plataforma o una tecnología, sino un estándar de desarrollo, construcción y despliegue de aplicaciones. [48]

Esta plataforma ofrece muy buenas perspectivas para la implementación de software empresarial, específicamente para aquellos sistemas informáticos que requieran basar su arquitectura en productos asentados en software libre.
A continuación se muestran algunas de sus principales ventajas:

- Soporte para múltiples sistemas operativos: al ser una plataforma Java, es posible desarrollar arquitecturas basadas en J2EE usando cualquier sistema operativo donde pueda estarse ejecutando una máquina virtual de Java, teniendo la gran ventaja de una independencia total de la arquitectura de hardware.
- Organismo de control: J2EE está controlada por un organismo formado por más de 400 empresas. Entre esas empresas se encuentran muchas de las más importantes del mundo informático, tales como Sun Microsystems, IBM, Oracle, BEA, HP, AOL, etc.
- Competitividad: muchas empresas crean soluciones basadas en J2EE que ofrecen características tales como rendimiento y precio muy diferentes. De este modo, se ha desarrollado a un nivel exponencial la plataforma y los clientes tienen la posibilidad de escoger entre una gran cantidad de opciones.
- Madurez: creada en el año 1997, J2EE ya tiene varios años de vida y una amplia cantidad de proyectos importantes a sus espaldas.
- Soluciones libres: sobre la plataforma J2EE es posible crear arquitecturas basadas por completo en productos de software libre. Además los arquitectos de software disponen de muchas soluciones libres para cada una de las partes de su arquitectura.

Como se ha dicho, la plataforma J2EE está definida por una especificación en formato electrónico, en ella se definen, de manera muy general, las pautas, reglas y servicios que han de seguir y ofrecer los diferentes servidores de aplicaciones que quieran implementar la plataforma J2EE. Además de eso, define también las normas generales que han de cumplir los desarrolladores que quieran crear aplicaciones empresariales compatibles con J2EE y los diferentes roles que éstos pueden asumir.
Los servicios que han de ofrecer los servidores de aplicaciones que implementen la plataforma J2EE están a su vez definidos por diferentes especificaciones. Estas especificaciones definen con mucho más detalle los diferentes componentes de los servidores de aplicaciones, como puedan ser un contenedor

web, un servidor de mensajería, el sistema de seguridad, etc. De algún modo, se puede decir que la especificación J2EE engloba a un gran conjunto de especificaciones.

Cualquiera puede participar en la creación de estas especificaciones y por supuesto cualquiera puede acceder a sus contenidos.

La gran importancia de toda la lista de especificaciones radica en que cuando se utiliza un servidor de aplicaciones que implementa la plataforma J2EE, los desarrolladores, obtienen de manera automática todos estos servicios. Es decir, se dispone de una gran caja de herramientas que se pueden aprovechar para realizar aplicaciones de una manera mucho más eficaz.

Hay que señalar que cada una de estas especificaciones puede utilizarse perfectamente por separado, por otra parte, cada una puede tener diferentes implementaciones que se pueden ofrecer como productos independientes.

La plataforma J2EE define un modelo de programación encaminado a la creación de aplicaciones basadas en n-capas. La lógica de la aplicación se divide en componentes de diferentes funciones que componen una aplicación J2EE y que están distribuidos en dependencia de la capa en el ambiente multicapas J2EE al cual la aplicación pertenece. Aunque puede variar, típicamente una aplicación suele tener cinco capas diferentes:

- Capa cliente: representa la interfaz de usuario que maneja el cliente.
- Capa de presentación: representa el conjunto de componentes que generan la información que se mostrará en la interfaz de usuario del cliente. Normalmente se crea a través de componentes basados en Servlets y JSP.
- Capa de lógica de negocio: contiene los componentes de negocio reutilizables.
- Capa de integración: aquí se encuentran los componentes que permiten hacer más transparente el acceso a la capa de sistemas de información. Este es el lugar idóneo para implementar la lógica de objetos de acceso a datos (DAO, data access object).
- Capa de recursos: engloba los sistemas en los cuales la información se almacena físicamente como bases de datos relacionales, sistemas legacy, bases de datos orientadas a objetos, bancos de ficheros de datos, etc.

Las ventajas de un modelo como este son muy importantes. Al tener las capas separadas se tiene poco acoplamiento entre las mismas, de modo que es mucho más simple hacer modificaciones en ellas sin que

afecten a las demás. Todo esto redunda en la obtención de mejoras en cuanto a mantenibilidad, extensibilidad y reutilización de componentes. Otra ventaja que se obtiene es la de promover la heterogeneidad de los clientes, ya que añadir nuevos tipos de clientes se reduce a añadir nuevas capas de interfaz de usuario y presentación, sin necesidad de modificar el resto de las capas.
El modelo de desarrollo con J2EE está basado en componentes reutilizables, con el objetivo de aumentar la reusabilidad de las aplicaciones, estos componentes; gracias a las especificaciones; son intercambiables entre servidores de aplicaciones, por lo que la portabilidad de las aplicaciones es muy alta. [48]

1.10 Herramientas base para el desarrollo de sistemas de gestión web

1.10.1 Gestores de contenidos o CMS

Cuando se mencionan los Gestores de Contenido o CMS (Content Managment System) se hace referencia a aquellas herramientas que permiten gestionar todo el contenido de un portal, esto implica crear un contenido, editarlo, borrarlo o publicarlo, etc.
Las características principales que presentan estas herramientas son las siguientes:

- Interfaz gráfica fácil de utilizar y lo más completa posible, que permita realizar todas las operaciones sobre el contenido y con la asesoría necesaria para que cualquier persona; sin necesidad de amplios conocimientos y con un tiempo mínimo de aprendizaje; pueda trabajar sobre ella.
- Editores para introducir el contenido. Son muy utilizados los editores WYSIWYG (what you see is what you get) en los que vemos directamente el resultado de la edición.
- Motor de Workflow. La gestión del flujo de trabajo es una parte importante de la gestión del contenido. La posibilidad de separar las tareas a realizar entre distintos roles de personas es muy utilizada en estos casos ya que esta problemática se repite en gran parte de los proyectos web.
- Gestión de permisos de usuarios y grupos. Otro apartado importante de estas herramientas es la gestión de usuarios, grupos de usuarios, y los permisos de los que cada uno dispone. Es muy importante asegurar que un contenido es editado y publicado solo por aquellas personas que tienen permisos para ello.

- Búsqueda del contenido. Ofrecer una herramienta para buscar información según una serie de palabras claves es muy importante en portales, o en sitios donde la cantidad de información es tan elevada que ni siquiera una buena organización en secciones y categorías puede asegurar que se encuentre una información deseada en un momento dado. [41]

1.10.2 Gestor Documental o Repositorio de Documentos

En lo referente a un Gestor Documental o Repositorio de Documentos, se debe tener en cuenta que su función principal es almacenar documentos, el tipo de los documentos puede variar y pueden ser: XML, texto, PDF's, .doc, imágenes, entre otros.

Las funciones principales que debe ofrecer son:

- Garantizar la persistencia de la información, ya sea en Base de Datos, o en un sistema de ficheros.
- Permitir una estructura coherente según el modelo del problema. Suele ser muy útil y flexible usar una estructura en árbol ya que esta permite crear tantos niveles de información como sean necesarios, facilitando que no se dependa de una estructura por defecto.
- Control de versiones. Una función igualmente importante que las anteriores es la necesidad de disponer de un control de las distintas versiones que se han producido en un contenido. Estas versiones e información adicional como quién realizó la modificación, la creación, el borrado y en que momento sucedió, puede ayudar a la hora de recuperar información que se ha modificado por error, o simplemente para temas estadísticos.
- Indexado de documentos. El mantener indexado el contenido que se tiene almacenado en el repositorio puede hacer que las búsquedas sobre el mismo sean bastante más rápidas que si se realizan sobre toda la base de información. [41]

1.10.3 Servidor de portales

Los portales empresariales se están convirtiendo en una poderosa herramienta que aumenta la productividad de los empresarios, debido a que presentan una alta integración de herramientas en un mismo entorno de trabajo, por ello representan un gran paso de avance en la integración de la infraestructura y las aplicaciones existentes en las organizaciones.

Esta nueva tecnología agrupa las características de los gestores de contenidos, los gestores documentales y otras que le han sido añadidas a esta tecnología de reciente creación la cual proporciona

a los usuarios un acceso simple y unificado a las aplicaciones y sus contenidos, además de brindar un espacio de trabajo colaborativo. [41]

A continuación se muestran las características más relevantes de estas herramientas, las cuales se amplían y mejoran como resultado de la evolución de las mismas.

1. Desplegar Portlets: Este es el requisito más básico de un portal, un portal debe permitir a un usuario final agregar y eliminar portlets, sin ningún esfuerzo de programación. Además permiten que las modificaciones se realicen en tiempo de ejecución, por lo tanto cuando un nuevo portlet es instalado el usuario no debe preocuparse por el servidor de portales.
2. Reutilización de los portlets: El tipo más común de portlets que los portales despliegan son los JSR-168; aunque ya integran la nueva versión de este estándar, el JSR-286; al cumplir con un estándar en su confección cualquier sistema basado en esta tecnología puede reutilizar las funcionalidades de un portlet.
3. Consumo y publicación de servicios remotos: Algunos portal server incluyen el estándar WSRP lo que les permite desplegar portlets remotos pero también consumir portlet, widgets, servicios web, páginas web entre otros servicios que estén publicados de forma remota. Un portal server permite a los usuarios finales agregar y administrar portlets remotos de manera transparente.
4. Autentificación y Single Sign-On (SSO): Como uno de los objetivos de un portal server es integrar un conjunto de aplicaciones, este cuenta con un sistema de autenticación que permite a un usuario ingresar a todos los contenidos y aplicaciones mediante una única entrada (single sign-on). Tener SSO permite que los portlet actúen en nombre del usuario al conectarse a otros sistemas, el usuario ingresa solo una vez su usuario y contraseña, el portal almacena sus credenciales y los portlet pueden ingresar siempre a los sistemas identificándose automáticamente en nombre del usuario.

   Puede suceder que en la organización ya posean algún mecanismo de autenticación; como por ejemplo LDAP; un portal puede adaptarse a este tipo de situaciones y por tanto trae mecanismos de integración con estos servidores o servicios.
5. Personalización: El portal server provee facilidades para que los usuarios finales puedan personalizar su ambiente de trabajo, permitiéndoles organizar el estilo del portal, seleccionando los colores, creando páginas, posición de ventanas u otros según sus gustos y necesidades, además permite seleccionar los contenidos que se desean ver. La personalización permite al portal entregar información específica enfocada a un usuario individual o un grupo de usuarios [38].

6. Multilenguaje: Un requisito común en las aplicaciones es que soporten varios lenguajes, los portal server permiten que los portlets sean desplegados en el lenguaje preferido por el usuario.
7. Seguridad: Las organizaciones necesitan saber que su información, datos y aplicaciones están protegidos contra un acceso no autorizado. En el caso de un portal server la autenticación permite definir que usuarios pueden acceder a determinadas páginas, a determinados portlet y permitir o denegar la autorización para modificar preferencias de estos. Un portal server típicamente permite definir roles de seguridad; que son privilegios que se conceden a usuarios específicos o grupos de ellos. Los roles de seguridad permiten conceder o restringir el acceso a los recursos del portal para uno o un conjunto de usuarios a la vez.
8. Herramientas de Colaboración: Un portal server comúnmente provee de herramientas que permiten a los usuarios comunicarse con otros para discutir temas de contenido u otros, esto puede darse mediante un conjunto de herramientas de colaboración. Este tipo de aplicaciones son un componente opcional que los proveedores de portales incluyen comúnmente como parte del portal. Algunas de estas herramientas se describen a continuación:

    - Calendarios Compartidos: permiten conocer la disponibilidad de sus compañeros.
    - Conferencias virtuales: permiten la comunicación entre empleados en tiempo real, este tipo de herramienta evita realizar reuniones en persona ayudando a maximizar el tiempo de producción.
    - Mensajería instantánea: permite a los miembros de un equipo conversar en línea en lugar de tener que hacerlo por teléfono u otros medios.
    - Gestores de Contenidos: permiten modificar e intercambiar documentos a un grupo de empleados en línea.

9. Búsquedas: Debido a que las organizaciones reúnen grandes cantidades de información en diversos formatos, la mayoría de los portales ofrecen servicios de búsqueda que permiten a los usuarios localizar recursos. Las capacidades de búsqueda permiten exploraciones a través de HTML, documentos, textos y datos, búsquedas simples o búsquedas federadas. Un portal server es capaz de realizar búsquedas en todas las aplicaciones y contenidos que integra de forma rápida.
10. Acceso desde múltiples dispositivos: Un portal server permite mostrar contenido en múltiples dispositivos dependiendo de los requerimientos de las empresas, ejemplos de dispositivos pueden ser: teléfonos celulares, PDA’s, etc.

11. Análisis del Portal: En ocasiones es útil conocer quiénes y cómo es accedido el portal server. Algunos proveedores de portales ofrecen medios para reunir y analizar información sobre las tendencias de las visitas al portal, de sus componentes, su uso, y su contenido. Los portal server permiten recolectar la información y usarla para mejorar la efectividad del sitio.
12. Herramientas de Navegación: Las herramientas para navegación son fundamentales en un portal server por que permiten llegar a todos los contenidos y aplicaciones que integran. Los mismos incluyen comúnmente menús de navegación.
13. Desplegar múltiples portales: En ocasiones es útil para una organización tener múltiples portales, por ejemplo para los diferentes departamentos que la componen y así mostrar diferentes contenidos en cada uno de los portales o tener temas o skins que diferencie la actividad de cada departamento.
14. GUI de Administración: Los portal server proveen herramientas de administración enfocadas a un usuario final para administrar el portal, sus componentes y sus operaciones. No solo los administradores de un portal tienen acceso a ellas, algunas de estas operaciones están disponibles para todos los usuarios del portal. La administración consiste principalmente en:

    - Administración de Usuarios: permite agregar o eliminar cuentas de usuario, y modificar perfiles de usuarios.
    - Administraciones de Roles: permite definir roles de usuarios y agregar o eliminar usuarios de un determinado rol.
    - Administración de Permisos: permite definir que roles de usuarios pueden acceder a las páginas, contenidos y aplicaciones del portal.
    - Administración de Páginas: permite crear, modificar o eliminar páginas.
    - Administración de Aplicaciones: permite agregar o eliminar ventanas de portlet y además cambiarlas de ubicación dentro de una página. Además debe permitir ver y modificar preferencias, ver la definición del portlet y crear o eliminar instancias y ventanas de portlet.
    - Administración de Aplicaciones Remotas: en el caso de actuar como consumidor permite agregar o eliminar un portlet remoto y modificar sus propiedades (el cache por ejemplo). En caso de actuar como productor, debe permitir definir que portlets locales serán publicados para que sean utilizados remotamente.

- Administración de Temas y Skins: permite modificar la apariencia del portal cambiando Skins y temas del portal. Un portal permite hacer esto generalmente para cada página o para el portal completo.

Estructura de un portal server

Un portal está compuesto por páginas, cada página de un portal es un conjunto de portlets, cuando un portlet es llamado genera contenido dinámico, este contenido es llamado fragmento. El fragmento no es una página web completa como en el caso de los servlet, solo es un trozo de código (HTML, XHTML, WML). [34]

Cada una de estas páginas es creada por el servidor de portales y se crea a partir de los fragmentos generados por cada portlet. El servidor de portal envía la página al dispositivo del cliente, por ejemplo un browser, donde es desplegada al usuario. El servidor de portales es el encargado de albergar instancias de portales y puede verse como un conjunto de objetos: páginas, portlets y temas.

Una ventana de portlet muestra fragmentos de markup generados por un portlet e incluye un título, botones de control y otras decoraciones. Estos fragmentos junto a las ventanas de portlet son agrupadas por el portal en la página que se devuelve al usuario. Es importante destacar que el contenedor no es responsable de agregar los fragmentos generados por cada portlet, esto es misión del portal. [34]

Un portal debe permitir realizar todas estas operaciones de una manera que facilite su gestión. La gran mayoría de los portales incluyen una interfaz de usuario para estas tareas, típicamente una interfaz web, algunos incluyen una consola donde pueden ejecutar comandos o en algunos se deben realizar directamente en un archivo de configuración.

1.11 Principio de desarrollo con portlets

En los últimos años los portlets han surgido como un nuevo estándar para facilitar una mayor reutilización, flexibilidad y escalabilidad en el desarrollo de aplicaciones Java para entornos web.

Técnicamente, un portlet es un trozo de código, que se ejecuta en el servidor de portal, y cuya función es proporcionar el contenido que se va a integrar en las páginas del portal, el contenedor administra los portlets y procesa peticiones que generan contenido dinámico. El contenido dinámico generado es un fragmento de código que puede aparecer en distintos lenguajes, entre ellos se encuentra: HTML, XHTML, WML, etc.

Los mismos por sí solos no tienen sentidos, ellos son desplegados o ensamblados en los portal server y con la combinación de varios se crean sistemas que cubran múltiples necesidades de un entorno específico.

A través del portal server los portlets muestran sus contenidos e interactúan con los usuarios, los cuales hacen peticiones y reciben resultados.

JSR-168/268 es un estándar donde se recoge la forma en que los componentes de un portal serán desarrollados y como los portlets funcionarán de manera conjunta o interconectados aún cuando estén en portales diferentes, también define entidades y hace comparaciones entre diversas tecnologías.

La JSR-168/268 fue desarrollada por la Java Community Process (JCP), y tiene el respaldo de los principales proveedores de servidores de portal, como por ejemplo Apache, BEA, IBM, Oracle, SAP, Sun Microsystems, y muchos otros. [33]

Cuando se cuenta con un portal server que soporta la especificación JSR-168/268, se le podrá agregar entonces cualquier tipo de portlets que haya sido desarrollado basado en la especificación.

Para que un portal sea escalable para muchos millones de usuarios los portlets usan el patrón flyweight[8] para implementar la personalización. [16]

Los portlets son configurables y personalizables para un usuario o un grupo de usuarios, por lo tanto no todos los usuarios verán la misma información de un mismo portlet.

1.11.1 Personalización

Los productores de portlets al construirlos les agregan varias características que hacen que los usuarios al usarlos puedan configurarlos y personalizarlos de la forma en que les sea más cómodo.

En la especificación además del descriptor de despliegue del portlet se definen tres niveles en donde se almacena distintos datos de personalización: definición del portlet, entidad del portlet, y ventana del portlet [16]. Es importante mencionar que los datos de un nivel superior son heredados a uno inferior creándose un árbol de herencia, donde por ejemplo, un cambio en la definición del portlet será visible en todas las entidades de portlets creadas a partir de esa definición. El árbol se crea porque para utilizar un portlet en el portal a parte de su definición, una nueva instancia del portlet debe ser creada con los datos de la definición. Además como estas instancias pueden aparecer en uno o más páginas del usuario, una o más

[8] Patrón flyweight: Se utiliza para eliminar o reducir la redundancia cuando se tiene gran cantidad de objetos que contienen información idéntica, además de lograr un equilibrio entre flexibilidad y rendimiento (uso de recursos).

ventanas de portlet son creadas [16]. Las instancias permiten que un portlet muestre contenido de acuerdo a las necesidades del usuario, y las ventanas permiten que cada portlet tenga distintos estados de navegación.
Los tres niveles se describen a continuación:

- Definición del portlet: Provee al administrador del portal la habilidad para personalizar la configuración del portlet. Este tipo de preferencias solo pueden ser modificadas por un administrador, algunos proveedores de portales permiten realizar esto a través de un modo de portlet opcional llamado CONFIG que solo es visible para administradores.
- Entidad del portlet: Provee al usuario final la habilidad para personalizar el portlet. Comúnmente un portlet posee preferencias pre-establecidas que luego el usuario modifica según sus necesidades. Para esto la especificación define el modo de portlet EDIT.
- Ventana del Portlet: La ventana del portlet también posee datos adjuntos que permiten al portlet mostrar una vista específica de su contenido. Estos datos de configuración son utilizados para los estados de navegación (modos del portlet, estados de ventana y parámetros de despliegue (render parameters).

Tanto la definición del portlet como la entidad del portlet almacenan los datos de configuración persistentemente, la ventana del portlet solo almacena los datos de manera temporal. Por ejemplo si en una sesión un usuario cambia el portlet del modo VIEW al modo HELP cuando el usuario vuelva a ingresar al portal el portlet volverá al modo VIEW.

1.11.2 Modos de los portlets

Los portlets tienen diferentes modos los cuales indican la función que está realizando el componente, el tipo de tarea que debe hacer y que contenido debe mostrar.
En la especificación JSR-168/268 se encuentran definidos tres modos para los portlets, aunque también existe la opción de definir modos personalizados, los modos definidos son: view, edit y help.
A continuación se describen brevemente los tres modos mencionados anteriormente: [33]

- View: En este modo el portlet despliega su contenido normal basado en la funcionalidad ofrecida por el portlet.
- Edit: Este modo presenta al usuario la posibilidad de personalizar los parámetros del portlet.

- Help: Este modo entrega una descripción de como el portlet debe ser usado.

1.11.3 Estados de ventana de los portlets

Como se ha dicho anteriormente el portlet genera contenido, este contenido necesita de un espacio dentro de la página donde será mostrado, existe un indicador para el tamaño que el portal asigna al contenido generado, este indicador se nombra estado de ventana y se encuentra definido en la especificación JSR-168; cómo mismo sucede con los modos descritos anteriormente los estados también tienen una opción que es para personalizar, es decir, para que el usuario defina el estado de ventana según su necesidad, los estados de ventanas predefinidos son: normal, maximized y minimized.
A continuación se describen brevemente los tres estados de ventana mencionados anteriormente: [33]

- Normal: La ventana del portlet comparte el espacio con otros portlets.
- Maximized: La ventana del portlet ocupa todo el espacio disponible, no comparte el espacio con otros portlets o tiene mayor espacio para colocar su salida que en su estado de ventana normal y que el resto de los portlets.
- Minimized: El portlet produce una salida mínima o no mostrar ningún contenido.

1.11.4 Ciclo de vida de un portlet

Al hablar de un ciclo de vida en la mayoría de los casos lo que se desea es englobar con esta frase a las diferentes fases o cambios por los que cruza un objeto determinado a lo largo de su existencia, el objeto puede ser físico o no.
Los portlets son objetos que no pueden ser palpados de forma física pero tienen “ciclo de vida”, ellos transitan por diversas fases que son controladas por su contenedor, dichas fases han sido enmarcadas en un espacio de tiempo y está perfectamente definido cuando el portlet se encuentra en alguna de ellas.
Cada portlet tiene una interfaz y ella a su vez define el ciclo de vida de este, la interfaz debe estar implementada, en caso contrario el portlet deberá extenderla de una clase que ya la tenga implementada.
Los métodos que se implementan dentro de la interfaz y que definen el ciclo de vida del portlet son: init(), processAction(), render() y destroy().
A continuación se hará una breve explicación del estado del portlet en cada uno de los momentos del ciclo de vida: [33]

- init(): Una vez cargado el portlet este método es llamado una sola vez. Es usado para ejecutar tareas de inicialización.
- processAction(): Es llamado luego de que el usuario envía cambios en el estado del portlet o causan una redirección (por ejemplo, al enviar un formulario). Se utiliza para procesar acciones de los usuarios.
- render(): Es llamado a continuación del processAction() para que el portlet sea redibujado.
- destroy(): Es el último en el ciclo de vida. Es llamado cuando el contenedor de portlets destruye el portlet en cuestión.

La ejecución del portlet se realiza siguiendo un ciclo de vida bien definido. En primer lugar el contenedor de portlets debe cargar el portlet. En segundo lugar, debe ser inicializado con los parámetros del descriptor de despliegue. A continuación, el portlet queda a disposición de las peticiones que los usuarios realicen a través del contenedor. Finalmente, y una vez que se deja de utilizar dicho portlet, debe ser descargado y queda fuera de servicio. Este ciclo se resume a continuación: [16]

1 Inicializar y poner en servicio el portlet (método init).
2 Responder a las peticiones del contenedor de portlets (métodos processAction y render).
3 Destruir el portlet cuando es necesario ponerlo fuera de servicio (método destroy).

El paso número dos descrito anteriormente se divide en dos fases: [16]

- ACTION: El contenedor de portlet llama el método processAction() para notificar que el usuario ha gatillado una acción en este portlet. Solo una acción por cada petición del cliente es gatillada. Durante esta etapa, un portlet puede redireccionar la petición, puede cambiar o modificar su estado de ventana, y puede cambiar de modo.
- RENDER: El contenedor de portlet llama el método render() para solicitar el fragmento generado por el portlet. El método render() es llamado para cada portlet en la página actual, por lo tanto, cada portlet generará el fragmento que le corresponde.

1.11.5 Compilación de las principales características de los portlets

- Son componentes java-web.
- Son administrados por servidores de portales que funcionan como contenedores.

- Generan contenido dinámico.
- Interactúan con clientes web a través del paradigma request/response.
- Solo generan fragmentos de código, ellos no pueden generar código HTML que contenga las etiquetas base: < body >, <iframe>, <frame>, <frameset>, <head>, <HTML> o <title>.
- No están asociados a una url, usan métodos tales como createActionURL() o createRenderURL() para construir una URL que permita al cliente emitir acciones para recuperar visualizaciones a partir del portlet que está actualmente en ejecución.
- Los clientes web no pueden interactuar directamente con un portlet, solo puede ser a través de un servidor de portal.
- Pueden existir varias veces y de forma simultánea en una o más páginas de un portal.
- Tienen un manejo de peticiones más refinado en términos de distinguir entre action request y render request. El ACTION es originado vía url por el cliente u otro portlet, al contrario un RENDER, es llamado por el contenedor como una respuesta para el estado actual del portlet.
- Tienen acceso a la información del perfil de usuario. Esta capacidad permite facilidades para implementaciones de autenticación, personalización, y seguridad.
- Poseen sus propias funciones de escritura de URLs, de esta manera se puede interactuar con un portlet a través de acciones o de enlaces a otras URLs, gestionándolo el servidor del portal.
- Un portlet puede almacenar datos no persistentes en su sesión en dos niveles diferentes: a nivel de aplicación y a nivel del propio portlet.

1.12 Conclusiones

Son múltiples las soluciones arquitectónicas para un proyecto de gestión sobre la web, cada una orientada a resolver un problema en específico, no existe una solución genérica que cubra los requisitos más importantes en un dominio, que pueda ser aplicada inmediatamente y que ayude a conseguir una alta productividad en los equipos de desarrollo integrando conceptos de desarrollo rápido y reutilización.

Sobre el principio de utilizar herramientas libres se realizó un estudio de las últimas tendencias de la tecnología Java y de las herramientas base para el desarrollo de sistemas de gestión sobre la web, encontrando como tecnología novedosa los portal server y portlets, herramientas estas basadas en estándares y que pueden ayudar en gran medida a solucionar los requisitos más comunes del dominio de sistemas definido.

En la Universidad no existe una arquitectura basada en un portal server, esto implica que se desaprovechen las funcionalidades y características de estas poderosas herramientas.

CAPITULO 2 PROPUESTA DE ARQUITECTURA PARA EL DESARROLLO DE SISTEMAS DE GESTIÓN WEB

2.1 Introducción

En el presente capítulo se definen; para el dominio específico; los requerimientos de referencia que deben cumplir la mayoría de las aplicaciones que forman parte de este dominio. Además se realiza un análisis de cada uno de los componentes que integran la arquitectura base. Finalmente se realiza la propuesta de arquitectura para desarrollar las aplicaciones del dominio específico y se provee la forma de configurar el entorno de trabajo, poniendo a disposición de los proyectos que decidan utilizar la arquitectura un material completo para su aplicación de inmediata.

2.2 Definición del dominio

En el área de la construcción de software se define como dominio a un área de aplicación de productos de software que está centrada en torno a un cuerpo de conocimientos y tienen un alcance asociado. El presente trabajo se centra en los sistemas de gestión sobre la web. En dicho dominio se comparten entre todos sus productos aspectos comunes, aspectos opcionales y aspectos variables que establecen diferencias entre los mismos.

La arquitectura que se propone en este trabajo es para aplicarla al dominio de los sistemas de gestión sobre la web, la misma describe la estructura principal de todo el grupo de productos y no solamente la de uno en particular, también incluye la captura de los aspectos o requisitos que son comunes, opcionales y variables en esta familia de productos de software.

En la Universidad se han desarrollado varias aplicaciones de gestión que están montadas sobre la web, dichas aplicaciones muestran áreas de necesidades que son comunes para todas, aún en negocios diferentes. Estas necesidades de los clientes de cada proyecto convergen en requerimientos comunes para todas las aplicaciones dentro de este dominio. A continuación se muestra un listado de dichas necesidades o requerimientos de referencia:

- Generación de reportes: Los reportes en los sistemas de gestión son muy variados y cambian constantemente. Por esta causa las aplicaciones actuales deben contribuir a la generación de reportes de forma dinámica.

- Seguridad: En la mayoría de las aplicaciones es necesario manejar los requerimientos de integridad, confidencialidad de la información, disponibilidad y no repudio.
- Auditoría: Es necesario tener registrado qué pasa en la aplicación constantemente y quién es el responsable de cada acción realizada.
- Almacenamiento de datos en Base de Datos: Se necesita guardar los datos de la aplicación en bases de datos. Es necesario que este almacenamiento sea lo más óptimo posible para posibilitar que la recuperación de dichos datos sea eficiente y fiable.
- Manejo de Transacciones: Se necesitan que las operaciones sean transaccionales y en muchos casos éstas se conforman por más de una petición del cliente.
- Capa de presentación flexible y rica en estilos: La capa de presentación debe cumplir esta característica para soportar las exigencias de presentación para los distintos tipos de clientes que puede presentarse en una misma aplicación, por ello debe ser posible que los usuarios modifiquen esta capa en dependencia de sus necesidades de trabajo.
- Alto rendimiento de procesamiento de las peticiones del cliente: Es necesario que las peticiones del cliente se respondan en el menor tiempo posible por lo que todas las capas de la aplicación deben interactuar de forma eficiente.
- Interacción con aplicaciones externas: Es muy común la interacción con sistemas externos por lo que se hace necesario que existan estándares en el uso de las tecnologías utilizadas.
- Administración de los aspectos configurables de la aplicación: Toda aplicación necesita ser configurable tanto en la etapa de despliegue como en tiempo de desarrollo.
- Nueva entrada de datos y módulos al sistema: Los usuarios en la actualidad necesitan que sus sistemas sean extensibles y permitan la integración de nuevos módulos de forma sencilla.

2.3 Arquitectura base

La arquitectura propuesta debe ser tratada como una arquitectura base pues está enfocada a una línea de productos. Los arquitectos que decidan utilizar este modelo arquitectónico observarán que la arquitectura particular de cualquier producto de la línea será la instanciación a la arquitectura que se propone en el presente trabajo. La arquitectura base definida sirve para guiar el desarrollo de productos planeados así como también ayuda a construir productos aún no anticipados.

En la arquitectura propuesta habrá como núcleo un portal server pues esta nueva tecnología ayuda a resolver la mayoría de los requisitos de referencia de los sistemas de gestión sobre la web. La plataforma a utilizar será J2EE, por tanto las aplicaciones construidas sobre la arquitectura contarán con sus estándares. Se utilizará la máquina virtual de Java para posibilitar que las aplicaciones puedan ser desplegadas en cualquier sistema operativo.

La arquitectura podrá montarse sobre cualquier gestor de base de datos y servidor de aplicaciones pues los portal server permiten la integración con la mayoría de estos productos existentes hoy en día.

Los módulos integrados a los portal server se llaman portlets y estos pueden obtenerse de varias formas:

1. los contenedores de portlets traen algunos en su instalación que pueden ser utilizados
2. podrán ser creados a través de IDES de desarrollo y frameworks
3. consumidos de forma remota a través del estándar WSRP
4. extraídos de un repositorio

Luego de tener todos los portlets necesarios se ensamblará la aplicación sobre el portal server.

En el área del software libre existen muchas variantes diferentes para cada uno de los componentes que integran la arquitectura. Esto trae consigo que sea necesario hacer un análisis para cada uno de los componentes necesarios que contribuya a que el desarrollo e integración de las nuevas funcionalidades para cada sistema sea una tarea que no necesite de un esfuerzo adicional.

La selección de los componentes no se ha restringido, la única condición es que todo esté basado en el software libre. En las próximas secciones se realizará un análisis para seleccionar cada uno de los componentes que deben ser integrados en la arquitectura (ver Figura 1), estos son:

- Sistema operativo
- Máquina virtual de Java
- Gestor de base de datos
- Servidor de aplicaciones
- El portal server.
- El framework que permitirá agilizar el desarrollo de nuevos portlets.
- El entorno de desarrollo (IDE) para utilizar el framework y desarrollar los portlets.

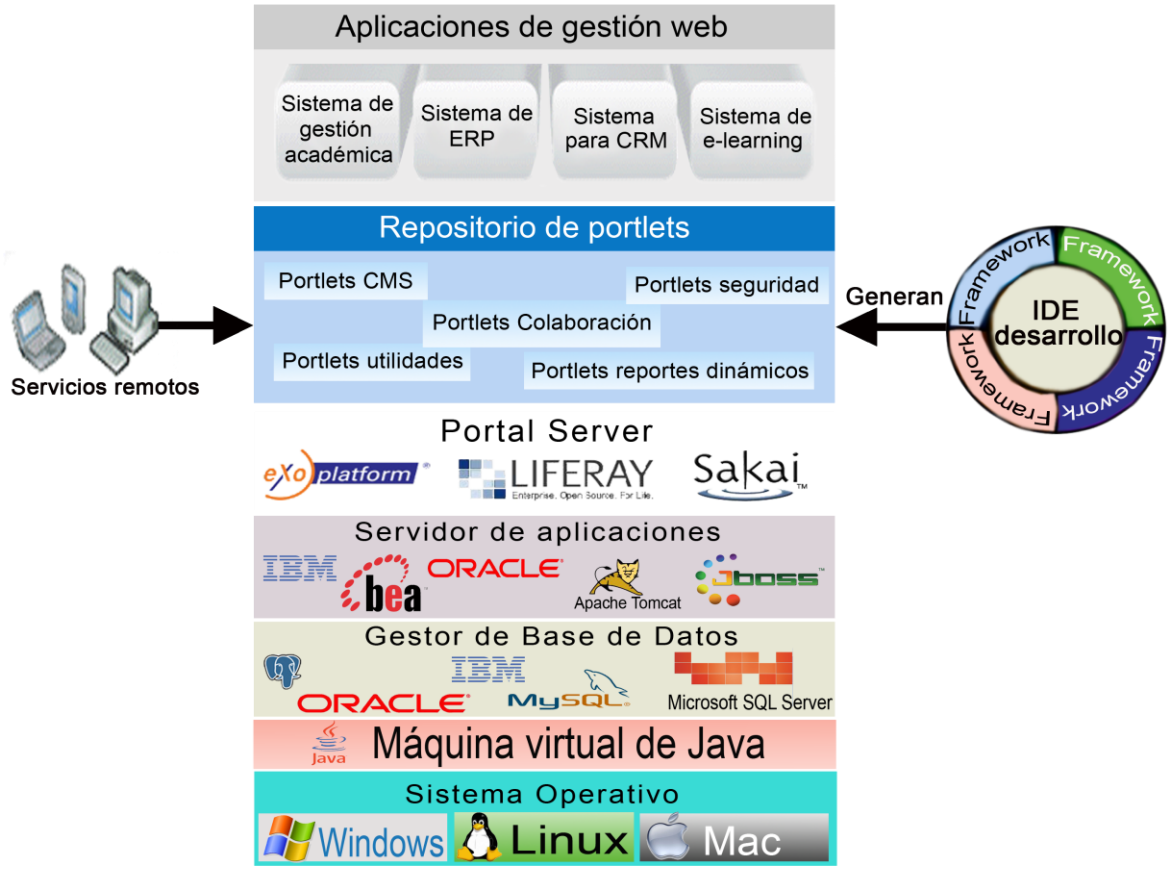


Figura 1. Componentes de la arquitectura base propuesta

Para garantizar cada uno de los requisitos expuestos en la definición del dominio es imprescindible que la arquitectura base tenga las condiciones necesarias para lograr el nivel de seguridad, flexibilidad y escalabilidad que necesitan los sistemas de gestión sobre la web.

2.4 Sistema Operativo

En la Universidad hace algunos cursos se ha decidido comenzar a migrar hacia el software libre, uno de los pasos es la utilización de alguna de las distribuciones del sistema operativo Linux como variante líder para ayudar a lograr la soberanía tecnológica.

Los nuevos productos que se creen en la UCI deben funcionar sobre distribuciones del sistema operativo Linux, independientemente de que funcionen también en otro.

La investigación que se recoge en este trabajo tuvo como una de sus premisas que la arquitectura que se propone funcione sobre cualquier sistema operativo. En este punto ayudó la tecnología Java pues una de sus funcionalidades más importantes es ser multiplataforma, permitiendo que las aplicaciones creadas sobre la arquitectura se puedan ejecutar en varios entornos.

Como otras características del sistema operativo Linux se encuentran las siguientes:

- No vende directamente su software, lo pone a disposición de cualquiera en Internet, aunque permite a personas o empresas distribuirlo comercialmente mientras se respete su licencia.
- Disponibilidad en varias plataformas hardware.
- Una amplia colección de software disponible.
- Un grupo de herramientas para facilitar el proceso de instalación y actualización del software.
- Su compromiso con los principios y valores involucrados en el movimiento del Software Libre.
- No tiene marcado ningún entorno gráfico en especial, ya sea GNOME, KDE u otro.

Para la arquitectura propuesta se decide usar la distribución Debian 5.0, debido a que es un sistema operativo estable, tiene variantes para estaciones clientes y servidores y tiene un gran uso en la Universidad.

2.5 Servidor de aplicaciones

Un servidor de aplicaciones se conecta a una red de computadoras para la ejecución de aplicaciones pero también gestiona la mayor parte o la totalidad de las funciones de la lógica de negocio y de acceso a los datos de la aplicación. Los principales beneficios que se pueden encontrar en este tipo de tecnología son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones, por lo cual es muy utilizada sobre todo en aplicaciones sobre la web.

En la arquitectura propuesta se ha incluido el uso de un servidor de aplicaciones, y se ha concebido que el arquitecto de software escoja dentro de esta gama de productos el que más se acomode a sus necesidades; puede ser un servidor de aplicaciones ligero o pesado.

La selección de forma general depende de varios factores, algunos de los que deciden son:

- la tecnología de servidor que se necesita utilizar
- el presupuesto con el que se cuenta
- la comunicación con variados servicios

- la facilitación a los desarrolladores e una API para eliminar el trabajo con el sistema operativo o con las nuevas interfaces requeridas
- el soporte a estándares como: HTML, XML, IIOP, JDBC, SSL, etc.

En el presente trabajo se propone utilizar específicamente el Apache Tomcat en su versión 5.5.27, para su selección se tuvieron en cuenta las características anteriores y las que se muestran a continuación que son específicas de la herramienta:

- es ligero
- es un servidor HTTP y un contenedor de *servlets*.
- es la implementación de referencia de las especificaciones de *servlets* (2.4) y de *JSP* (2.0).
- es software libre (licencia Apache 2.0) gestionado por la fundación Apache.
- puede funcionar como servidor HTTP o conectado a otro servidor HTTP como Apache HTTP Server o IIS.
- presenta compatibilidad con J2EE
- puede ejecutar servicios web mediante Apache Axis.

2.6 Selección del gestor de base de datos

Para este trabajo es necesario seleccionar un gestor de base de datos, a continuación se hace una comparación entre dos de los más usados gestores de bases de datos existentes que pueden ser utilizados en el desarrollo de sistemas de gestión sobre la web.

PostgreSQL

Las características positivas que posee este gestor son: [39]

1. Posee una gran escalabilidad. Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta (en algunos benchmarks se dice que ha llegado a soportar el triple de carga de lo que soporta MySQL).
2. Implementa el uso de rollback's, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en las que MySQL no podría.

3. Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, como puede ser Oracle.
4. Es un gestor de base de datos liberado bajo la licencia BSD.

Los mayores inconvenientes que se pueden encontrar a este gestor son: [39]

1. Consume gran cantidad de recursos.
2. Tiene un límite de 8K por fila, aunque se puede aumentar a 32K, con una disminución considerable del rendimiento.
3. Es de 2 a 3 veces más lento que MySQL.

MySQL

Las características positivas que posee este gestor son: [39]

1. Sin lugar a duda, lo mejor de MySQL es su velocidad a la hora de realizar las operaciones, lo que le hace uno de los gestores que ofrecen mayor rendimiento.
2. Su bajo consumo lo hacen apto para ser ejecutado en una máquina con escasos recursos sin ningún problema.
3. Las utilidades de administración de este gestor son envidiables para muchos de los gestores comerciales existentes, debido a su gran facilidad de configuración e instalación.
4. Tiene una probabilidad muy reducida de corromper los datos, incluso en los casos en los que los errores no se produzcan en el propio gestor, sino en el sistema en el que está.
5. El conjunto de aplicaciones Apache-PHP-MySQL es uno de los más utilizados en Internet en servicios de foro y en buscadores de aplicaciones.

Gran parte de los inconvenientes se exponen a continuación: [39]

1. Carece de soporte para transacciones, rollback's y subconsultas.
2. El hecho de que no maneje la integridad referencial, hace de este gestor una solución pobre para muchos campos de aplicación (versión 4 de MySql)
3. No es viable para su uso con grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad.
4. En estos momentos sus nuevas versiones son propietarias y por consiguiente ya no se brinda mantenimiento a las versiones anteriores que eran libres.

Los arquitectos están libres de seleccionar el gestor que más se acomode a su negocio, actualmente como gestor de base de datos libre y robusto solo queda la opción de PostgreSQL es el gestor recomendado para usar en la arquitectura propuesta.

2.7 Portal server existentes en el mercado. Comparación y selección de esta herramienta.

Se ha decidido usar como eje principal de la arquitectura un portal server o servidor de portales como también se le conoce.

En la tabla que se muestra a continuación aparece una comparación entre los requisitos más esenciales y complejos de una aplicación de gestión sobre la web y las funcionalidades que brinda el portal server en cada caso.

Requisitos de los sistemas de información web	Funciones que provee un portal server
Entrada de datos y procesamiento	El portal server trae incluido un grupo de portlets que se pueden utilizar en las aplicaciones que se construyan pero si un sistema necesita alguna funcionalidad específica entonces será necesario implementarla con el framework destinado para eso.
Seguridad	La autentificación centralizada permite definir una seguridad granular a nivel de usuarios, grupos de usuarios y roles de cada sistema instalado en el portal server, así como definir los derechos a los diferentes recursos a nivel de servidor, de aplicaciones, de páginas y por ultimo a nivel de cada portlet.
Auditoría	Los portal server posibilitan controlar las preferencias de los usuarios para luego perfeccionar los servicios que se brindan, pero no garantizan el control y seguimiento estricto sobre cada una de las operaciones que se realizan con los datos, estas funciones deben ser

	implementadas a los portlets a través de los frameworks que permiten desarrollarlos.
Almacenamiento de datos en Base de Datos	La elección del portal server influye mucho en este tipo de requisitos. Muchos solo permiten algún tipo de gestor de base de datos en específico. Esta limitante se tendrá en cuenta para la elección del portal server que se utilizará en la arquitectura para garantizar la mayor flexibilidad en este requisito.
Capa de presentación flexible y rica en estilos	Los portal server son en sí la nueva generación de sistemas en la web que posibilitan utilizar las últimas tendencias en lo que se refiere a la interfaz de usuario. Brindan la posibilidad de modificar la presentación y estructura del portal a las condiciones que desee el usuario final.
Alto rendimiento de procesamiento de las peticiones del cliente	Los portal server están concebidos para atender una alta concurrencia de usuarios, de hecho su principal fin es utilizarlos como portales para internet. Hay que señalar que este requisito está muy atado a las condiciones de hardware donde se instale la aplicación final.
Integración con aplicaciones externas	Los portal server son la generación de sistemas que posibilita la integración de las aplicaciones que la arquitectura SOA requiere. Además la creación de portlet que consuman servicios remotos, así como exponer los mismos mediante el estándar WSRP hace a los portal server la herramienta ideal para la integración de sistemas.
Administración de los aspectos configurables del sistema.	La administración que brinda un portal server se aplica a los portlet que ya tiene integrado, así como a la gestión general de los contenidos, páginas y a los nuevos portlet

	que sean instalados. Sin dudas esta gestión de administración es muy útil para la mayoría de las aplicaciones, aunque es necesario aclarar que las configuraciones propias de cada uno de los portlet que se integren deberán ser concebidas en la etapa de desarrollo del mismo.
Escalabilidad del sistema	La instalación, desinstalación y configuración de portlets en tiempo de ejecución hace que los portal sever sean muy escalables. Además el cambio de la propia estructura y presentación del sistema también puede cambiarse cada vez que sea necesario.

Tabla 1. Comparación entre los requisitos de los sistemas de gestión sobre la web y las funciones de un portal server.

Como se puede apreciar en la tabla anterior, los portal server garantizan la mayoría de los requisitos más complejos que pueden aparecer en el desarrollo de un sistema de gestión sobre la web. Los requisitos que no se logran garantizar de forma completa o parcial serán portlet que se desarrollarán e integrarán al portal server.

No es posible evaluar de manera eficaz todos los portal server libres existentes y por lo tanto, es necesarios seleccionar un número sobre la base de criterios de selección definidos, además de tener en cuenta la popularidad actual del proyecto, la actividad de la comunidad que lo mantiene y la experiencia propia de desarrollar y probar dichos proyectos. Esto no significa que los demás que no son analizados no puedan ser útiles. A continuación se listan los portal server escogidos para la evaluación:

- GripShpere: Uno de los primeros en implementar el estándar JSR-168 en Europa.
- ExoPlatform: Por su popularidad.
- Liferay: Por su popularidad, interfaz de usuario y funcionalidades adicionales.
- Jboss Portal: Por su integración con la plataforma completa de Jboss para proyectos SOA.

GridSphere

GridSphere es un portal server muy estable inicialmente financiado por el proyecto GridLab de la Unión Europea. El mismo tiene una interfaz de usuario muy amigable y es muy fácil de usar, además es completamente compatible con JSR-168.

El prototipo funcional del portal server se basa en la API propuesta por IBM WebSphere que es un producto comercial. La personalización de la interfaz de usuario es fácil y no requiere de un trabajo complicado con ficheros de configuración. Además provee un API propio para la creación de nuevos portlets para el portal pero esto trae consigo que los portlet creados con esta variante no cumplan con el estándar JSR-168.

GridSphere puede ser desplegado con Tomcat y cualquier otro servidor de aplicaciones compatible con J2EE pero la documentación relacionada con este tema es muy escasa y sólo se dispone de información de usuarios en las listas de correo.

Este portal server no soporta muchas de las últimas tecnologías, o al menos no se menciona, como por ejemplo Structs [45] o Spring [44] aunque sí permite realizar portlet con JSF. La persistencia de los datos la garantiza utilizando Hibernate [17] lo que significa que puede ser conectado a cualquier gestor de base de datos que soporte conexiones con JDBC sin necesidad de modificar el código, simplemente cambiando algún fichero de configuración.

GridSphere admite varios tipos de mecanismos de autenticación, por defecto utiliza el control por usuario y contraseña. También apoya Función Control de Acceso basado en Roles (RBAC) para separar los usuarios en clientes, usuarios, administradores y súper usuarios. Además posee un mecanismo flexible para la adaptación de presentación al usuario que está basado en descriptores XML que se pueden modificar fácilmente para crear diseños personalizados al portal.

Este proyecto es muy bien gestionado desde su página principal, en la cual se pueden encontrar información y tutoriales actualizados en su wiki. Además el portal tiene incluido un grupo de portlets y servicios públicos que permiten realizar un grupo de operaciones básicas como son la configuración de la seguridad y herramientas colaborativas. [15]

eXo Platform

La plataforma eXo es definida como un portal server y como un gestor de contenidos empresariales (CMS) [12]. eXo Platform proporciona a los usuarios personalizar un punto de acceso único a la empresa para los sistemas de información y los recursos.

A través de un ambiente web, eXo Platform proporciona información comercial de la empresa a los empleados, permite el intercambio y la gestión de sus datos, así como la ejecución de los procesos críticos de negocio. El mismo es compatible con JSR-168 y esta constituido por varios módulos. Además garantiza soporte íntegro a WSRP y a otras diferentes tecnologías a través de diferentes vías de integración.

eXo Platform soporta que se desplieguen portlet cuando está en ejecución y al igual que otros muchos incluye en su distribución un conjunto de portlet para realizar tareas comunes como son la colaboración, la navegación, gestión de usuarios y la gestión de contenido. También este portal server incluye una capa para la integración natural de aplicaciones desarrolladas con el frameworks Structs [45] lo que garantiza una gran flexibilidad para integrar aplicaciones ya desarrolladas en esta tecnología.

En general, eXo Platform es un potente portal server de código abierto que trae muchas tecnologías vanguardias incluidas o compatibles. También viene con plug-in de Eclipse, que es muy conveniente para los desarrolladores y puede ampliarse y adaptarse para dar cabida a las necesidades personalizadas de cada aplicación.

Liferay

Liferay portal server es mucho más que un contenedor de portlet [30]. El mismo incluye muchas características útiles como por ejemplo Gestor de Contenidos (CMS), compatibilidad para consumir y exponer servicios en WSRP, Autentificación centralizada (SSO) con posibilidad de integrarse fácilmente con tecnologías como LDAP, CAS u OpenID además de soporte para programación orientada a aspectos (AOP) y muchas otras de las últimas tecnologías.

Liferay tiene un buen diseño arquitectónico basado en las mejores prácticas de J2EE que le permite ser utilizado con una gran variedad de servidores de aplicación tanto los más ligeros como el Tomcat y Jetty, así como más potentes como Borland ES, JBoss, Glassfish, JOnAS, JRun, Oracle9iAS, Orion, Pramati,

RexIP, Sun JSAS, WebLogic, y WebSphere entre otros. De hecho Liferay es el único portal server de código abierto que soporta cualquier tipo de despliegue tanto en servidores propietarios como libres.
La flexibilidad en su diseño permite garantizar las ventajas de tecnologías como Structs [45], Spring [44], EJB [11], JMS [24], Java Mail y Web Services. Liferay es el primer portal server en introducir el concepto de WebOS, además posee más de sesenta portlet ya integrados entre los que se incluye una suite completa para la colaboración y la gestión de redes sociales, la gestión de contenidos, herramientas para realizar encuestas, recoger información mediante formularios, entre otros. También garantiza soporte para 22 lenguajes y al utilizar Hibernate para la persistencia puede ser usado en cualquier gestor de base de datos que posibilite JDBC. La interfaz de usuarios es muy amigable pues para ello utiliza la tecnología AJAX. Tiene además concebida una alta integración con otros proyectos de software libre para la gestión de procesos, la inteligencia de negocio, etc. Cumple con los estándares JSR-168, JSR-268 y WSRP. Su comunidad es una de las más activas en estos temas con más de un millón de descargas, sesenta mil por mes, así como más de cinco mil usuarios activos en la comunidad para contribuir en el proyecto. La documentación de este portal server es muy actualizada, completa y detallada, la misma se puede encontrar en libros completos, tutoriales y los recursos de información que brindan en la web como son la wiki, el foro, el blog y las listas de correos de discusión del portal server.

Jboss Portal

JBoss Portal Edition proporciona un entorno de código abierto, basado en estándares, para alojar y servir aplicaciones e información en una interfaz web de portal, publicar y gestionar su contenido y personalizar la experiencia del usuario. Simplifica el acceso a las aplicaciones y la información proporcionando una única fuente de interacción con información corporativa y basada en Internet [26]. JBoss Portal soporta servicios Web para portlets remotos (WSRP) completamente. Además es compatible con cualquier gestor de base de datos que soporte JDBC. Entre las principales desventajas se encuentra el poco uso que tiene en la actualidad la versión libre de este portal, pues la versión comercial si tiene una gran difusión a nivel mundial, debido a que se incluye como parte de un paquete completo de herramientas para SOA que provee Jboss. Otra de las dificultades es que puede ser desplegado solo en servidores de aplicación de Jboss.

2.7.1 Análisis de los portal server

Definición

Las herramientas a evaluar son portal server, específicamente Liferay, eXo Platform, JBoss Portal y Gridsphere.

	Liferay	eXo Platform	JBoss Portal	Gridsphere
Manufactura	Liferay, Inc.	The eXo platform SARL	Red Hat Middleware, LLC	GridLab
Sitio web en Internet	liferay.com	exoplatform.com	portals.apache.org	gridsphere.org
Versión actual	5.2.2	2.5	2.7.5	3.0.8

Tabla 2. Información general de los portal server.

La evaluación para la selección se basará en cinco criterios, los mismos son: Información general, Contenido, Integración, Seguridad e Interfaz de usuario.

Primeramente se muestran los criterios de evaluación sujetos a una tabla de valores que asignará una puntuación a cada aspecto, luego se califica teniendo en cuenta las herramientas de forma concreta y posteriormente se hará la selección final de la herramienta que se propone para utilizar en la arquitectura.

Evaluación

Para evaluar cada funcionalidad se tendrá en cuenta la siguiente notación:

Funcionalidad	Puntos
No cubierta	0
Parcialmente cubierta	1
Completamente cubierta	2

Tabla 3. Notación para evaluar los diferentes criterios.

A continuación se muestran los diferentes criterios de evaluación y se clasificarán según la puntuación anterior.

Información general	Puntuación		
	0	1	2
Licencia	Que no tenga una licencia de software libre.	Cuenta con una licencia bajo los principios del software libre pero no deja libertad al usuario para introducir nuevas restricciones o normativas a sus resultados pues debe estar estrechamente vinculado a los planteamientos de la licencia.	Cuenta con una licencia bajo los principios del software libre y permite heredar los planteamientos de la licencia y es posible aplicar restricciones adicionales a la misma.
Documentación	No tiene ninguna documentación.	La documentación es muy restringida y se presenta en solo 3 lenguajes y no incluye el español.	La documentación siempre está actualizada, traducida a más de 3 lenguajes incluyendo el español y adaptada para diferentes lectores como pueden ser usuarios finales, administradores, entre otros.
Tecnologías	No usa los estándares JSR 168/286, JSR 170, WSRP, tampoco AJAX. Puede integrarse con una	Solo utiliza el estándar JSR 168/286, 170 y WSRP. Puede integrarse con dos tecnologías de creación de	Utiliza los estándares JSR 168/286, JSR 170, WSRP, además de AJAX. Permite integración con

	sola tecnología de creación de interfaces.	interfaces.	SOA y con tres o más tecnologías de creación de interfaces.
Comunidad	No tiene comunidad o solo tienen un medio de comunicación.	Tiene solo dos medios de comunicación.	Tiene más de dos medios de comunicación.

Tabla 4. Evaluación de los criterios pertenecientes a la Información General.

Contenido	**Puntuación**		
	0	1	2
Estándares	No presentan ningún estándar.	Presentan entre 1 y 3 estándares.	Presentan más de tres estándares
Cantidad de portlets	Integran entre 0 y 20 portlets en su instalación.	Integran entre 20 y 40 portlets en su instalación.	Integran más de 40 portlets en su instalación.
Instalación de nuevos portlets	No permite la instalación de nuevas funcionalidades.	Para instalar nuevos portlets es necesario detener el portal server.	Para instalar nuevos portlets no es necesario detener el portal server, esta operación es transparente al usuario.
Soluciones que se generan a partir de las funcionalidades que integran en su instalación.	No incluye ninguna solución.	Incluye menos de 3 soluciones.	Incluye más de 3 soluciones

Tabla 5. Evaluación de los criterios pertenecientes a los Contenidos.

Integración	Puntuación		
	0	1	2
Estándares de integración	No soporta ningún estándar.	Soporta solo los estándares JSR 168 y WSRP.	Soporta los estándares JSR 168/286 y WSRP.
Sistemas Operativos	Se puede ejecutar solo en un sistema operativo.	Se puede ejecutar entre 1 y 3 sistemas operativos.	Se puede ejecutar en más de 3 sistemas operativos.
Servidores de aplicación	Se despliega en solo 1 servidor de aplicación.	Se despliega entre 2 y 5 servidores de aplicación.	Se despliega en más de 5 servidores de aplicación.
Gestores de base de datos	No usa PostgreSQL como gestor de base de datos.	Utiliza PostgreSQL y 5 gestores de base de datos más.	Utiliza PostgreSQL y más de 5 gestores de base de datos.

Tabla 6. Evaluación de los criterios pertenecientes a la Integración.

Seguridad	Puntuación		
	0	1	2
LDAP	No permite la integración con esta tecnología.	-	Permite la integración con esta tecnología.
Active Directory	No permite la integración con esta tecnología.	-	Permite la integración con esta tecnología.
JAAS	No permite la integración con esta tecnología.	-	Permite la integración con esta tecnología.

Single-Sign-On	No permite la integración con esta tecnología.	-	Permite la integración con esta tecnología.
SSL	No permite la integración con esta tecnología.	-	Permite la integración con esta tecnología.
Role Management	No permite la integración con esta tecnología.	-	Permite la integración con esta tecnología.
OpenID	No permite la integración con esta tecnología.	-	Permite la integración con esta tecnología.

Tabla 7. Evaluación de los criterios pertenecientes a la Seguridad.

Interfaz de usuario	**Puntuación**		
	0	1	2
Tecnología para la creación de temas	No permite crear nuevos temas, pues las interfaces son estáticas.	Solo permite usar una tecnología específica para la creación de temas.	Posibilita el desarrollo de temas con varias tecnologías.
Lenguajes	No incluye el español y permite la generación en solo 3 lenguajes.	Incluye el español y permite generación en 5 lenguajes.	Incluye el español y permite generación en más de 5 lenguajes.
Personalización	No admite la creación de páginas personales.	Solo permite crear páginas privadas que no pueden ser vistas por el resto de la comunidad.	Permite crear páginas privadas y públicas con múltiples contenidos que pueden ser compartidas por toda la comunidad.

Tabla 8. Evaluación de los criterios pertenecientes a la Interfaz de usuario.

Calificación

En las siguientes tablas se especifican las tecnologías y características de cada portal server evaluado, así como la evaluación de cada aspecto; mostrándose la misma encerrada en paréntesis.
Al final de cada tabla se muestra el promedio correspondiente a cada herramienta el cual será utilizado en la construcción de una gráfica de red que mostrará cuál es la herramienta prevaleciente para la arquitectura propuesta.

Información general

	Liferay	eXo Platform	JBoss	GridSphere
Licencia	MIT (2)	GPL (2)	LGPL - GNU Lesser GPL (1)	GridSphere Software (1)
Documentación	Contiene documentación actualizada para desarrolladores, usuarios finales, administradores, libros completos, videos, en varios idiomas incluyendo el español. (2)	Solo contiene documentación para usuarios finales, administradores y videos, solo en idioma inglés y francés. (1)	Contiene una guía rápida para usuarios finales, manuales de usuarios, JBoss Portal Javadoc, la documentación está en varios idiomas incluyendo el español. (2)	Contiene guías de inicio rápido, documentación para el desarrollo, aunque presenta deficiencia en la actualización de la documentación, la mayoría esta disponible solo en inglés. (1)
Tecnologías	JSR 168, JSR 286, JSR 170, WSRP, Struts, JSP, JSF, Tiles, AJAX, EJBs, Spring, AOP, SOAP, Hibernate, RMI,	JSR 168/286, JSR 170, WSRP, JSF, WebDAV, Velocity, AJAX. (1)	JSR 168, JSR 170, WSRP, Struts, JSP, JSF, AJAX, JMX 1.2, J2EE 1.4, Spring, XML. (1)	Struts, JSP, JSF, PHP, Perl, Spring, Velocity, XML, KOSMOS. (0)

	Velocity, Lucene, Terracota. (2)			
Comunidad	Comunidad activa con blogs, wikis, foros, seguimiento de problemas. (2)	Comunidad con lista de correo. (1)	Comunidad activa con foros y wikis. (1)	Comunidad con lista de correo y wikis. (1)
Promedio	***2***	***1.25***	***1.25***	***0.75***

Tabla 9. Evaluación de los criterios pertenecientes a la interfaz de usuario en los diferentes portal server.

Contenido

	Liferay	eXo Platform	JBoss	GridSphere
Estándares	JSR 170, JSR 220, JSR 127, JSF, AJAX (2)	JSR 170, JSF (1)	JSR 170, JSF, JMX 1.2 (2)	JSR 170, JSF, AJAX (2)
Cantidad de portlets	Más de 80 (2)	Más de 40 (2)	Más de 20 (1)	Más de 30 (1)
Instalación de nuevos portlets	Despliegue sin necesidad de detener el portal server (2)	Es necesario de detener el portal server (1)	Despliegue sin necesidad de detener el portal server (2)	Es necesario de detener el portal server (1)
Soluciones que se generan a partir de las funcionalidades que integran en su	CMS, calendarios, foros, libro de direcciones, chat, tiendas, wiki, Google Maps, flujo de trabajo rápido para	Documentación de administración, BPM, foros, correo, calendarios, wiki, chat, buscadores.	CMS, business intelligence, wiki. (1)	Herramientas de administración, noticias, perfiles personalizados, buscadores. (2)

instalación.	notas, entre otras. (2)	(2)		
Promedio	***2***	***1.5***	***1.5***	***1.5***

Tabla 10. Evaluación de los criterios pertenecientes a los contenidos en los diferentes portal server.

Integración

	Liferay	eXo Platform	JBoss	GridSphere
Estándares de integración	JSR 168, JSR 286, WSRP (2)	JSR 168, WSRP (1)	JSR 168, WSRP (1)	JSR 168, WSRP (1)
Sistemas Operativos	Windows, Linux, Mac OS X, Solaris, AIX, BSD (2)	Windows, Linux, UNIX (1)	Windows, UNIX, Linux (1)	Windows, Linux, Unix, Solaris, AIX (2)
Servidores de aplicación	Borland ES, JBoss, JOnAS, JRun 4 Updater 3, OracleAS, Orion, Pramati, RexIP, Sun JSAS, WebLogic, SP4, WebSphere, Glassfish, Geronimo, entre otros. (2)	WebLogic, Jboss, JOnAS, WebSphere, Oracle AS. (1)	JBoss. (0)	IBM WebSphere, JBoss, Tomcat 5.x / 5.5.x. (1)
Gestores de base de datos	DB2, Firebird, Hypersonic, InterBase, JDataStore, MySQL, Oracle, PostgreSQL, SAP, SQL Server, entre otros. (2)	Hypersonic, MySQL, Oracle, PostgreSQL DB2, MS SQL (2)	IBM DB2, MySQL, Oracle, Microsoft SQL Server, MaxDB. (0)	IBM DB2, MySQL, Oracle, Microsoft SQL Server, MaxDB (0)
Promedio	***2***	***1.25***	***0.5***	***1***

Tabla 11. Evaluación de los criterios pertenecientes a la integración en los diferentes portal server.

Seguridad

	Liferay	eXo Platform	JBoss	GridSphere

LDAP	Sí, soporta GUI (2)	Sí (2)	Sí (2)	Sí (2)
Active Directory	Sí, soporta GUI (2)	No (0)	Sí (2)	No (0)
JAAS	Sí (2)	Sí (2)	Sí (2)	Sí (2)
Single-Sign-On	Sí – CAS (2)	Sí – CAS (2)	Sí (2)	Sí (2)
SSL	Sí (2)	Sí (2)	Sí (2)	Sí (2)
Role Management	Sí (2)	Sí (2)	Sí (2)	Sí (2)
OpenID	Sí (2)	No (0)	No (0)	No (0)
Promedio	***2***	***0.71***	***0.85***	***0.71***

Tabla 12. Evaluación de los criterios pertenecientes a la seguridad en los diferentes portal server.

Interfaz de usuario

	Liferay	eXo Platform	JBoss	GridSphere
Tecnología para la creación de temas	Velocity, JSP, Tiles (2)	Eye-Candy (Mac OS, Vista) (1)	JSP, Servlet (2)	XML-based (1)
Lenguajes	Inglés, Alemán, Francés, Español, Japonés, y 10 lenguajes más. (2)	Inglés, Francés, Portugués. (0)	Inglés, Alemán, Francés. (0)	Francés, Inglés, Español, Alemán y 8 lenguajes más. (2)
Personalización	Permite crear	Permite crear	Permite crear	Permite crear

	páginas privadas y públicas con múltiples contenidos que pueden ser compartidas por toda la comunidad. (2)	páginas privadas y públicas con múltiples contenidos que pueden ser compartidas por toda la comunidad. (2)	páginas privadas y públicas con múltiples contenidos que pueden ser compartidas por toda la comunidad. (2)	páginas privadas y públicas con múltiples contenidos que pueden ser compartidas por toda la comunidad. (2)
Promedio	***2***	***1***	***1.3***	***1.6***

Tabla 13. Evaluación de los criterios pertenecientes a la interfaz de usuario en los diferentes portal server.

Selección

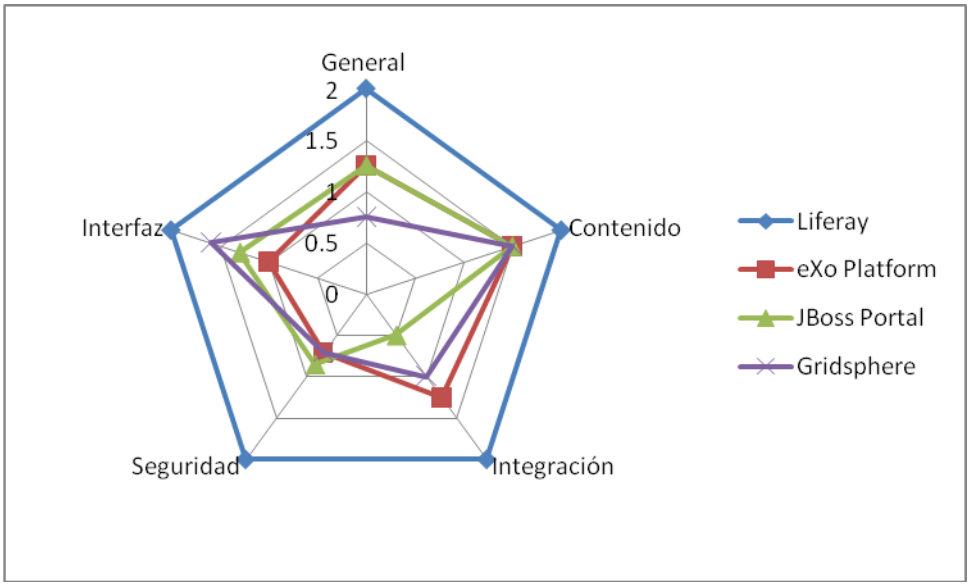


Figura 2. Comparación de los diferentes portal server respecto a los criterios: Información general, interfaz de usuario, seguridad, integración y contenidos.

Como se puede apreciar en la figura 2 Liferay es el más completo en los cinco criterios que se evaluaron, es el que mejor cumple con los requisitos necesarios para formar parte de la arquitectura propuesta.

2.8 Frameworks para el desarrollo de portlets

Como el resto de los componentes propuestos para utilizar en la arquitectura existen gran variedad de frameworks para el desarrollo de los portlets, entre ellos difieren en varias características y esencialmente en el marco de trabajo utilizado, a continuación se revisan las diferencias de los marcos de trabajo que utilizan estas herramientas y se escogen varios frameworks para ser analizados y dejar plasmada una propuesta sólida.

2.8.1 Modelo Vista Controlador (MVC) vs Componentes de negocio

Un componente de negocio contiene toda la información necesaria sobre un concepto de negocio para poder crear aplicaciones sobre eso. En un componente de negocio se define:

- La estructura de datos.
- Las validaciones, cálculos y en general toda la lógica de negocio asociada a ese concepto.
- Las posibles vista, esto es, la configuración de todos las posibles interfaces gráficas para este componente.
- Se define las posibilidades para la presentación tabular de los datos.
- Mapeo objeto-relacional, lo que incluye información sobre las tablas de la base de datos y la forma de convertir a objetos la información que en ellas hay.

Esta forma de dividir es ideal para el trabajo en grupo, y permite desarrollar un conjunto de componentes interproyecto.

Los componentes de negocio permiten definir toda la información sobre un concepto de negocio en un único sitio o fichero.

En un marco de trabajo MVC la lógica de negocio (el modelo), la interfaz de usuario (la vista) y el comportamiento (el controlador) se definen separadamente. Este tipo de marco de trabajo es útil si la frecuencia de cambios en la lógica y estructura de datos es baja y la posibilidad de cambiar la tecnología de interfaz de usuario o acceso a datos es alta.

Añadir un nuevo campo en un componente de negocio; para un desarrollador; es sencillo pues solo es necesario modificar un solo fichero y el cambio se transmite al resto de las estructuras involucradas dentro del componente.

Pero, los marcos de trabajo MVC son malos cuando los cambios a la estructura y los datos son muy frecuentes (como en caso de las aplicaciones de gestión). Si se aplica el mismo ejemplo de añadir un nuevo campo en un marco MVC, el desarrollador tiene que cambiar la interfaz de usuario, la clase del modelo y la tabla de la base de datos.

Otra ventaja que brindan los marcos de trabajo orientados a componentes de negocio es la distribución del trabajo en los equipos que desarrollan la aplicación. Es fácil hacer una distribución orientada a la lógica de negocio y no por capa tecnológica como se hace en la mayoría de los proyectos en la Universidad.

2.8.2 Selección del framework para el desarrollo de portlets

Debido a la diversidad de frameworks que existen para la tarea de crear nuevos portlets es necesario hacer una comparación y determinar cual es el que más ayuda con respecto a la reutilización de lo portlets que se crean y a la productividad del equipo de desarrollo lo que se revertirá en un gran avance del trabajo.

Portal Pack

Portal Pack es un conjunto de extensiones para NetBeans IDE que soportan el ciclo de vida completo de una aplicación de portlets dentro del IDE. Con esta herramienta los desarrolladores pueden crear, empaquetar, desplegar y testear portlets dentro de NetBeans. El pack incluye la generación automática de código y de los descriptores de despliegue, con lo que se pueden desarrollar portlets muy rápidamente. Además se integra con varios servidores de portal, permitiendo realizar despliegues de portlets en servidores remotos y locales [35].

- Las características principales de Portal Pack 3.0 son:
    - Soporte para la especificación JSR 286.
    - Storyboard de eventos para los eventos de JSR 286.
    - Constructor visual de portlets para crear portlets JSF usando un editor WYSIWYG.
    - Portlets con el framework Spring MVC.
    - Portlets con lenguajes diferentes como Ruby/PHP/Groovy que pueden desplegarse en Sun GlassFish Web Space Server y en Liferay Portal Server.
    - Soporte para Liferay Service Builder.

    - Soporte para despliegue en directorios: no hay necesidad de volver a desplegar la aplicación de portlets una y otra vez cuando se realizan cambios en los archivos JSP/HTML/JS.
- Servidores soportados por Portal Pack 3.0
    - Sun GlassFish Web Space Server 10.0 (antes conocido como Project WebSynergy)
    - Liferay Portal Server 5.1.x/5.2.x
    - Open Portal Portlet Container 2.x
    - Sun Java System Portal Server 7.x

Lomboz

Es un plugin gratuito y abierto de Eclipse para el entorno de desarrollo J2EE. Tiene medios para desarrollar, probar, perfilar y desplegar aplicaciones Web, servicios Web, Java, J2EE y EJB. Admite la mayoría de los runtimes de servidores de aplicaciones J2EE estándar, y admite la mayoría de los runtimes populares de código abierto tales como JOnAS. Está distribuido bajo LGPL. [32]
Está empaquetado, integrado y es dependiente de muchos otros paquetes de Eclipse open source para desarrollo en J2EE.
Lomboz suministra:

- Asistentes para crear y ensamblar módulos J2EE.
- Editor JSP y asistente para el código.
- Compatibilidad con JBoss, WebLogic, Apache Tomcat, JOnAS y JRun.
- Generadores de código EJB basados en XDoclet.
- Generadores de Servicios Web basados en Apache Axis.
- Incluye la plataforma Eclipse WebTools, Web Services, JSF y Herramientas JPA.
- Incluye el editor BPEL y soporte para Apache ODE

OpenXava

OpenXava es un marco de trabajo para desarrollar aplicaciones JavaEE/J2EE rápida y fácilmente. La filosofía subyacente es definir con anotaciones de Java o con XML y programar con Java. El objetivo principal es hacer que las cosas más típicas en una aplicación de gestión sean fáciles de hacer, mientras que ofrece la flexibilidad suficiente para desarrollar las funciones más avanzadas y específicas [37].

Las piezas fundamentales para crear una aplicación OpenXava son los componentes de negocio, en el contexto de OpenXava un componente de negocio es una clase Java (aunque existe también una versión XML) que contiene toda la información necesaria sobre un concepto de negocio para poder crear aplicaciones sobre eso.

Algunas característistica de OpenXava son:

- Alta productividad para aplicaciones de gestión.
- Curva de aprendizaje corta y sencillez de uso.
- Suficientemente flexible como para crear aplicaciones sofisticadas.
- Es posible insertar una funcionalidad en cualquier punto.
- Basado en el concepto de componente de negocio.
- Genera una aplicación J2EE completa, incluyendo la interfaz de usuario (AJAX).
- Soporta cualquier servidor de aplicaciones (Tomcat, JBoss, WebSphere).
- Soporta JSR-168/286: Todos los módulos OpenXava también son portlets estándar.
- Soporte completo de EJB3 y JPA.
- Está probado con los portales: JetSpeed 2, WebSphere Portal, Liferay y Stringbeans.
- Fácil integración de informes hechos con JasperReports.
- Licencia LGPL.
- Todas las etiquetas y mensajes están en inglés, español, alemán, polaco, indonesio, francés, italiano, chino y catalán.

Naked Objects

Se trata de un framework Java en el que se busca que los objetos de negocio sean responsables no sólo de la lógica de negocio sino también del control de las acciones y de mostrar su estado visual, es decir, que cada objeto sea modelo, vista y controlador. [52]

Básicamente lo que se hace es modelar y a partir de este modelo generar las diferentes capas que componen la aplicación: presentación, ventanas y controles incluidos. El modelo lo controla todo, incluyendo las modificaciones, la teoría que guía es que los requerimientos son expresados y volcados en el modelo, cuando un requerimiento cambia, por ende el modelo ha de cambiar y esto disparará la serie de cambios en las diferentes capas de la aplicación. La licencia bajo la que está registrado es Apache.

JMatter

Este es un framework que pretende multiplicar la productividad de los programadores al construir aplicaciones de escritorio que accedan a una base de datos. Para ello se basa en un conjunto de convenciones de nomenclatura y opciones de configuración por defecto, unido a la posibilidad de construir el esqueleto de una aplicación que acceda a una base de datos simplemente con un comando de ant. Está orientado a un tipo muy particular de aplicación (aquélla orientada a realizar operaciones CRUD en una base de datos) por tanto no tiene sentido para muchos otros problemas. Otras características esenciales se listan a continuación: [27]

- Toda la lógica del negocio debe encapsularse en los objetos del dominio.
- La interfaz de usuario debe ser una representación directa de los objetos del dominio.
- La interfaz de usuario debe ser creada 100% automáticamente de la definición de los objetos del dominio.
- Se distribuyen bajo una doble licencia GPL/comercial.
- Esta basado en Naked Objects
- Basado en componentes de negocio.
- La principal ventaja de desarrollar con este framework es que reduce el tiempo de desarrollo ayudando al aumento de la productividad del equipo de desarrollo.
- Se basa en Hibernate para la persistencia.
- Garantiza el acceso y la gestión de usuarios.
- Apoyo a la construcción de asistentes, agenda (calendario de los componentes), y más.
- Soporta la construcción e invocación de consultas para la persistencia desde la interfaz.
- Construcción de interfaces ricas utilizando los componentes de Swing y facilitando las operaciones CRUD. La interfaz se construye dinámicamente, en tiempo de ejecución, a partir del modelo.
- La esencia es concentrarse en la implementación de los aspectos genéricos del negocio.
- No es necesario escribir ninguna línea de código para obtener la interfaz de la aplicación.

Luego de revisar las características de varios frameworks se confrontan por los siguientes criterios de comparación:

	JMatter	Naked Objects	Portal Pack	OpenXava	Lomboz

Licencia de distribución	GPL / comercial	Apache	LGPL	LGPL	LGPL
Marco de trabajo (MVC o Componente de negocio)	Componente de negocio	Componente de negocio	MVC	Componente de negocio	MVC
IDE de desarrollo que soporta	Netbeans Eclipse	Eclipse	NetBeans IDE	Eclipse	Eclipse
Portal Server para desplegar los portlets	Ninguno	Ninguno	Sun GlassFish Web Space Server 10.0 Liferay Portal Server 5.1.x / 5.2.x Open Portal Portlet Container 2.x Sun Java System Portal Server 7.x	JetSpeed 2 WebSphere Portal Liferay Stringbeans	Liferay eXo Portal
Curva de aprendizaje	Media	Media	Media	Baja	Alta
Multilenguaje	Sí	Sí	Sí	Sí	Sí
Tecnología de interfaces generadas	Swing	Java Server Pages	Java Server Faces	Ajax	Java Server Pages

Tabla 14. Comparación de frameworks para construcción de portlets.

El framework seleccionado para la arquitectura es OpenXava, primeramente porque es un framework basado en componentes de negocio, ganando con esta característica muchas ventajas como la de tener

toda la atención centrada en la lógica de negocio, haciendo sencilla la distribución del trabajo dentro de un proyecto productivo y un aumento considerable de la eficacia en el momento de la implementación y conformación del sistema.
De los frameworks presentes en la tabla que están basados en componentes de negocio, solo OpenXava genera portlets y como requisito adicional lo hace bajo los estándares JSR-168/286, aportando con esto la posibilidad de reutilizar todos los portlets desarrollados para las distintas aplicaciones. Genera interfaces dinámicamente con tecnología AJAX a partir de clases Java y anotaciones JPA.

2.9 Selección de un entorno integrado de desarrollo

Como entornos de desarrollo para proponer en la arquitectura se tuvieron en cuenta el Eclipse y se estudió además la posibilidad de emplear NetBeans.
El proyecto NetBeans ha hecho que el desarrollo de aplicaciones Java de tipo empresarial sea bastante rápido y sencillo, su facilidad de uso, su cumplimiento de regulaciones, sus perfiles de rendimiento, además de su flexibilidad entre plataformas. Permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. NetBeans es un proyecto de código abierto con un alto número de usuarios y una comunidad en constante crecimiento.
Eclipse es un entorno integrado de desarrollo de código abierto concebido principalmente para Java pero cuyas funcionalidades pueden extenderse mediante la adición de plug-ins que lo dotan de nuevas funcionalidades o de soporte para otros lenguajes de programación. Es multiplataforma (Windows, Linux y Mac OS X) y la dirección del proyecto es llevada a cabo por la Fundación Eclipse, una entidad en la que tienen participación importantes empresas del sector informático cómo IBM, SAP, Borland, Intel, Oracle, Nokia o Motorola.
Presenta una forma de instalación sencilla y una gran comunidad que extiende constantemente las áreas de aplicación cubiertas.
La versión actual de Eclipse dispone de varias características que se mencionan a continuación:

- Editor de texto
- Resaltado de sintaxis
- Compilación en tiempo real
- Pruebas unitarias con JUnit
- Control de versiones con CVS

- Integración con Ant
- Asistentes (*wizards*): para creación de proyectos, clases, tests, etc.
- Refactorización

En cuanto a la adición de plugins que permite se pueden mencionar los siguientes:

- Control de versiones con Subversion
- Integración con Hibernate

NetBeans, a pesar de ser de código abierto y libre no tiene las facilidades brindadas por Eclipse para integrar elementos que faciliten el desarrollo de aplicaciones web.

La elección fue Eclipse pues presenta mejor relación calidad-facilidad y se puede usar para otros tipos de aplicaciones; no solo para las web.

Otra razón que impulsó la selección de Eclipse fue que de las dos herramientas analizadas este es el entorno de desarrollo más difundido en la UCI.

2.10 Propuesta de herramientas para la arquitectura

Luego de todo el análisis realizado en el presente capítulo para la selección de las herramientas a utilizar se muestra de forma concreta en la Figura 3 como quedará la arquitectura que se propone en el presente trabajo.

El núcleo de la arquitectura será el portal server Liferay 5.2.2, debido a que dicha herramienta ayuda a resolver una gran cantidad de requisitos de los sistemas de gestión web sin la necesidad de que el equipo de desarrollo lleve a cabo ningún esfuerzo.

La arquitectura no obliga a los arquitectos de software a utilizar las herramientas que se proponen y aunque el trabajo estuvo enfocado todo el tiempo en realizar una propuesta de herramientas basadas en los pilares del software libre el portal server Liferay 5.2.2 tiene compatibilidad con dieciséis servidores de aplicaciones y trece gestores de base de datos de los existentes en el mercado, ver Figura 2, aunque se propone Apache Tomcat 5.5.27 y PostgreSQL 8.3.

La versión libre de OpenJDK 1.6 (ver Figura 2) permitirá que las aplicaciones sean multiplataformas, permitiendo al arquitecto de software utilizar el sistema operativo necesario.

Luego de agrupar todas estas herramientas novedosas el equipo de desarrollo solo tendrá que centrarse en el negocio de su aplicación específicamente en desarrollar con el IDE Eclipse 3.4 y como único framework OpenXava 3.1.1 los portlets que el Liferay 5.2.2 no tenga integrados y que no puedan ser

obtenidos de forma remota. Luego de completar la cantidad de portlets necesarios se colocarán en el contenedor (Liferay 5.2.2) y se tendrá una aplicación de gestión web lista para ser utilizada, Ver Figura 2.

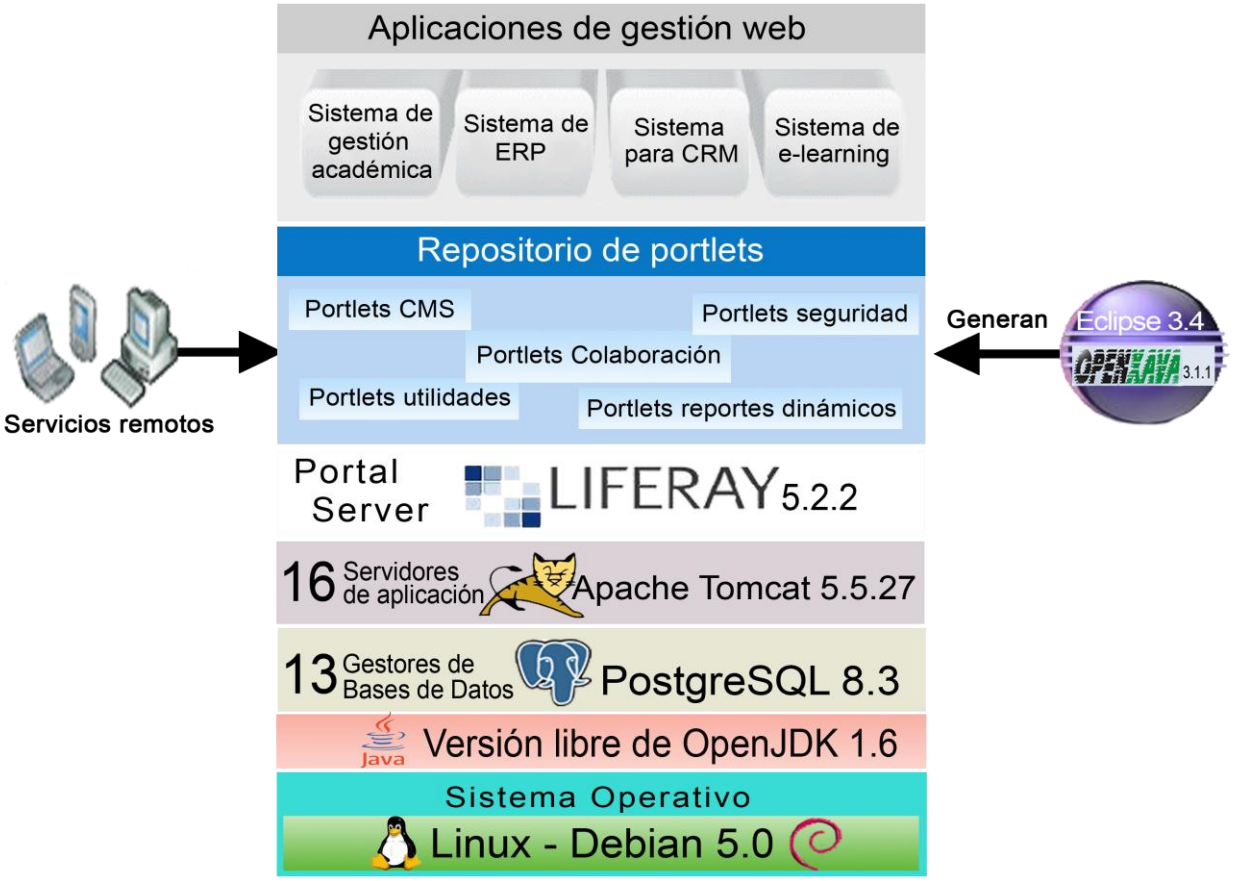


Figura 3. Propuesta de herramientas de la arquitectura.

La propuesta de herramientas se resume de la siguiente manera:

- Sistema operativo: Debian 5.0
- Máquina virtual de Java: Versión libre de OpenJDK 1.6
- Gestor de base de datos: PostgreSQL 8.3
- Servidor de aplicaciones: Apache Tomcat 5.5.27

- El portal server: Liferay 5.2.2
- El framework que permitirán agilizar el desarrollo de nuevos portlets: OpenXava 3.1.1
- El entorno de desarrollo (IDE) para utilizar el framework y desarrollar los portlets: Eclipse 3.4

2.11 Configuración del entorno de trabajo

A continuación se describe paso a paso como configurar el entorno de trabajo para construir portlets utilizando OpenXava 3.1.1 y como desplegarlo en el Liferay 5.2.2. Para este ejemplo se utilizará el gestor de bases de datos PostgreSQL 8.3.

1. Configurar la máquina virtual de Java: En el sistema operativo seleccionado para el entorno de desarrollo. En este caso se aconseja utilizar la versión libre OpenJDK 1.6 sobre el sistema operativo Linux-Debian 5.0.
   - Instalar el paquete OpenJDK
     - Abrir el gestor de paquetes de Debian Synaptic.
     - Seleccionar el paquete *openjdk-6-jdk* para instalar.
     - Pulsar en Aplicar para instalar el paquete.
   - Configurar las variables de entorno de Java
     - Abrir un terminal.
     - Ejecutar el comando *sudo su –* para realizar operaciones como administrador del sistema.
     - Editar el fichero */etc/profile.* Ejemplo ejecutar comando *mcedit /etc/profile*
     - Copiar al final del fichero la siguiente configuración:
       - export JAVA_HOME=/usr/lib/jvm/java-6-openjdk
       - export JRE_HOME=/usr/lib/jvm/java-6-openjdk/jre
     - Salvar los cambios y cerrar
2. Instalar el IDE de desarrollo Eclipse y el compilador Ant.

- Abrir el gestor de paquetes de Debian Synaptic
- Seleccionar los paquetes *eclipse y ant* para instalar
- Pulsar en Aplicar para instalar los paquetes

3. Configurar el framework OpenXava para el trabajo con Eclipse
   - Descargar la última versión del framework del sitio oficial (http://www.openxava.org)
   - Descompactar el archivo en una carpeta local del sistema. Ejemplo: */home/desarrollo/openxava3.1.1*
   - Abrir el Eclipse
   - Abrir desde Eclipse el workspace de OpenXava
     - Ir al menú *File/ Switch_workspace*
     - Poner la siguiente dirección: */home/desarrollo/openxava-3.1.1/workspace* y presionar Aceptar
4. Crear un nuevo proyecto OpenXava
   - Crear un nuevo proyecto vacio en Eclipse
     - Ir a *File/New/Project*
     - Seleccionar *Next*
     - Poner nombre del proyecto, ejemplo: *Universidad*
     - Seleccionar *Finish*
   - Crear la plantilla base para un proyecto OpenXava
     - Abrir el fichero *CrearNuevoProyecto.xml* ubicado dentro del proyecto OpenXavaPlantilla
     - Ejecutar la tarea *crearNuevoProyecto* con el Ant

- Pedirá el nombre del proyecto a crear. Poner el mismo que se creo vacío con el Eclipse, en este caso *Universidad.*
- Refrescar el workspace (F5)
- En el proyecto Universidad se crea la siguiente estructura de carpetas:
  - [raiz]: En la raíz del proyecto donde se encuentra el fichero el build.xml (con las tareas Ant).
  - [src]: Carpeta fuente que contendrá el código fuente Java escrito para el proyecto.
  - [xava]: Los archivos XML para configurar las aplicaciones OpenXava. El principal es aplicacion.xml.
  - [i18n]: Archivos de recursos con las etiquetas y mensajes en varios idiomas.
  - [properties]: Archivos de propiedades para configurar la aplicación.
  - [data] Útil para guardar los scripts para crear las tablas de la aplicación, si aplicara.
  - [web]: Contenido de la parte web. Normalmente archivos JSP, lib y classes. La mayoría del contenido es puesto automáticamente, pero es posible poner aquí los propios archivos JSP.

5. Configurar la comunicación con el gestor de base de datos que se utilizará en el proyecto. En este caso se decidió por PostgreSQL 8.3.
   - Instalar el PostgreSQL 8.3
     - Abrir el gestor de paquetes *Synaptic*
     - Seleccionar el paquete *postgresql* y *pgadmin3*
     - Pulsar *Aplicar* para instalar los paquetes
   - Configurar PostgreSQL 8.3

- Abrir el pgadmin
- Iniciar el servicio de PostgreSQL. Para ello realizar lo siguiente:
  - Abrir un terminal
  - Ejecutar el comando *sudo su –* para realizar operaciones como administrador del sistema
  - Ejecutar el comando */etc/init.d/postgresql-8-3 start*
- Crear la base de datos correspondiente al proyecto. En este caso *universidad*. Con el usuario propietario de la base de datos llamado *postgres* y la contraseña *postgres*.

➢ Configurar el proyecto para el uso de PostgreSQL

- Crear una carpeta en el proyecto, en este caso *lib*
- Copiar para la carpeta *lib* (creada en paso anterior) y en */home/desarrollo/openxava3.1.1/tomcat/common/lib* el driver JDBC de Java que corresponde al gestor de base de datos. En este caso *postgresql8.3.jar*
- Actualizar la tarea ant *ActualizarEsquema* dentro del fichero build del proyecto *Universidad* con la siguiente configuración:

```
<target name="actualizarEsquema">
<ant antfile="../OpenXava/build.xml" target="updateSchemaJPA">
<property name="persistence.unit" value="junit"/>
<property name="schema.path" value="lib/postgresql.jar"/>
</ant>
</target>
```

- En la carpeta *persistence* del proyecto, abrir el fichero *hibernate-cfg.xml* y dentro de la sección *<session-factory>* poner la siguiente configuración:

```
<property name="hibernate.connection.datasource">java:comp/env/jdbc/UniversidadDS
</property>
```

<property name="hibernate.connection.driver_class">org.postgresql.Driver
</property>
<property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
<property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/Universidad
</property>
<property name="hibernate.connection.username">postgres</property>
<property name="hibernate.connection.password">postgres</property>
<property name="hibernate.show_sql">false</property>

- En la carpeta *persistence/META-INF* del proyecto, abrir el fichero *persistence.xml* y dentro de la sección *<persistence>* poner la siguiente configuración:

<!-- Tomcat + Postgres -->
<persistence-unit name="default">
<non-jta-data-source>java:comp/env/jdbc/UniversidadDS</non-jta-data-source>
<class>org.openxava.session.GalleryImage</class>
<properties>
<property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect"/>
</properties>
</persistence-unit>

- En la sección *<persistence-unit name="junit">* poner la siguiente configuración:

<!-- JUnit Postgres -->
<persistence-unit name="junit">
<properties>
<property name="hibernate.connection.driver_class" value="org.postgresql.Driver"/>
<property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect"/>
<property name="hibernate.connection.url"
value="jdbc:postgresql://localhost:5432/universidad"/>
<property name="hibernate.connection.username" value="postgres"/>
<property name="hibernate.connection.password" value="postgres"/>
<property name="hibernate.show_sql" value="false"/>
</properties>

- Editar el fichero *context.xml* ubicado en */home/desarrollo/openxava3.1.1/tomcat/conf* y poner al final la siguiente configuración:

<Resource name="jdbc/UniversidadDS" auth="Container" type="javax.sql.DataSource"
maxActive="20" maxIdle="5" maxWait="10000" username="postgres" password="postgres"
driverClassName="org.postgresql.Driver"
url="jdbc:postgresql://localhost:5432/universidad"/>

6. Desplegar los portlets generados en el proyecto OpenXava en el Liferay
   - Una vez terminado de desarrollar y probar el proyecto OpenXava (para más información de como desarrollar con este framework revisar la documentación ubicada en */home/desarrollo/openxava3.1.1/doc*)
   - Generar los portlet correspondientes al proyecto
     - Abrir el fichero *build.xml* y ejecutar la tarea ant *generarPortlets*
     - Los portlets *generados se ubican en la carpteta* */home/desarrollo/openxava3.1.1/workspace.dist/Universidad.dist/Universidad.war*
   - Descargar el Liferay del sitio oficial (http://www.liferay.com). Utilizar para este caso la distribución embebida en el servidor de aplicaciones *Tomcat 5.5.27* y descomprimir el dichero en la siguiente dirección */home/desarrollo/liferay5.2.2.*
   - Copiar el fichero *Universidad.war* para la carpeta */home/desarrollo/liferay5.2.2/deploy*
   - Copiar las librerías de OpenXava */home/desarrollo/openxava3.1.1/tomcat/common/lib/ejb.jar* y */home/desarrollo/openxava3.1.1/tomcat/common/lib/jta.jar* para la capeta del Liferay */home/desarollo/liferay5.2.2/tomcat-5.5.27/common/lib*
   - Editar el fichero *context.xml* del Liferay ubicado en */home/desarrollo/liferay5.2.2/tomcat5.5.27/conf* con la fuente de datos correspondiente a los portlets que se instalarán, en este caso agregar:

     <Resource name="jdbc/UniversidadDS" auth="Container" type="javax.sql.DataSource"
     maxActive="20" maxIdle="5" maxWait="10000" username="postgres" password="postgres"
     driverClassName="org.postgresql.Driver"
     url="jdbc:postgresql://localhost:5432/universidad"/>
   - Iniciar el Tomcat 5.5.27 del Liferay 5.2.2

- Abrir un terminal
- Ejecutar el *comando sudo su –*
- Ubicarse en la carpeta bin del tomcat, para ello ejecutar el siguiente comando: *cd /home/desarrollo/liferay5.2.2/tomcat5.5.27/bin*
- Ejecutar el comando *./startup.sh* y si desea ver los logs del tomcat ejecutar *./catalina.sh run*

Una vez autenticado en el Liferay con los permisos correspondientes se podrán agregar a las páginas los portlets instalados los cuales estarán ubicados en la categoría Universidad. Para más detalles revisar la guía de administración de Liferay (http://www.liferay.com).

2.12 Conclusiones

En el presente capítulo se realizó una propuesta de arquitectura de software basada en herramientas que le permitieran a la misma tener una alta flexibilidad, portabilidad, escalabilidad y robustez para el desarrollo de forma productiva de los sistemas de gestión web.

Se brinda también una configuración del entorno de trabajo con las herramientas de la propuesta.

Es necesario dejar plasmado que al establecer varios proyectos en este modelo de arquitectura cada vez será menor la necesidad de desarrollo de portlets pues el factor reutilización aumentará de forma considerable permitiendo obtener productos en menores rangos de tiempo y con la calidad requerida.

CAPITULO 3 EVALUACIÓN DE LA ARQUITECTURA PROPUESTA

3.1 Introducción

En este capítulo primeramente se hace un estudio en relación con los atributos de calidad y su relación en la evaluación de las arquitecturas de software. Posteriormente se revisan las principales técnicas y métodos existentes para la evaluación, se realiza la selección, se explica el procedimiento y se procede a aplicarlo en la arquitectura propuesta.

3.2 Relación entre la arquitectura de software y los atributos de calidad

El desarrollo de software ha demostrado que para alcanzar un atributo de calidad específico, es necesario tomar decisiones arquitectónicas que requieren un pequeño conocimiento de funcionalidad.

Cada decisión arquitectónica tiene un efecto cercano sobre los atributos de calidad y el arquitecto puede argumentar cómo la decisión tomada permite alcanzar algún objetivo, con frecuencia el objetivo es un atributo de calidad en particular.

La relación entre arquitectura de software y atributos de calidad tiene diversos beneficios que se mencionan a continuación:

- Se realza en gran medida el proceso de análisis y diseño arquitectónico, puesto que el arquitecto puede reutilizar análisis existentes y determinar acuerdos explícitamente.
- Una vez que el arquitecto entiende el impacto de los componentes arquitectónicos sobre uno o varios atributos de calidad, estará en condiciones de reemplazar un conjunto de componentes por otro cuando lo considere necesario.
- Una vez que se codifica la relación entre arquitectura y atributos de calidad, es posible construir un protocolo de pruebas que habilitará la certificación del sistema.

La arquitectura de un software tiene gran importancia ya que es la base de diseño del mismo además de convertirse en un artefacto que determina atributos de calidad.

3.2.1 Atributos de Calidad

Según Barbacci [4] la calidad de software se define como el grado en el cual el software posee una combinación deseada de atributos.

Los atributos a los que se hace referencia son requerimientos o características que el sistema debe tener, diferentes a los requerimientos funcionales.
Estos requerimientos o características son los atributos de calidad, los cuales Barbacci [4] define como las propiedades de un servicio que presta el sistema a sus usuarios.
Los atributos de calidad se recogen en dos categorías diferentes:
Los observables vía ejecución: Aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución. (Ver Tabla 21)
Los no observables vía ejecución: Aquellos atributos que se establecen durante el desarrollo del sistema. (Ver Tabla 22)
Es importante tener en cuenta dos aspectos, el primero es que todos los atributos de calidad no se logran de manera aislada, lograr uno puede tener efectos negativos para alcanzar otro que también se desee alcanzar y el segundo es que tener conocimiento de los atributos observables, no implica que se satisfacen los atributos no observables vía ejecución.

3.3 Evaluación de arquitecturas de software

El primer paso para la evaluación de una arquitectura es conocer qué es lo que se quiere evaluar, según Kazman [29], esto permite establecer una base para la evaluación.
Para evaluar una arquitectura hay que tener en cuenta si las decisiones que se han tomado con respecto a la misma están basadas en el logro de los atributos de calidad del sistema, de ser así entonces la evaluación se centra en revisar que tipo de impacto tiene la arquitectura elegida sobre los atributos de calidad que deben estar presentes en el sistema.
Los arquitectos de software deben tener claridad en las tecnologías existentes y en las características que se quieren lograr en el sistema, esto permitirá tomar decisiones acertadas en diferentes momentos, entre los cuales están:

- Comparación de alternativas de arquitecturas de software similares
- Comparación de la arquitectura original y la modificada, producto de nuevas necesidades
- Comparación de la arquitectura de software con respecto a los requerimientos del sistema
- Comparación de una arquitectura de software con una propuesta teórica
- Valoración de una arquitectura en base a escalas específicas

El autor Kazman [29] establece que no es necesario que la arquitectura esté completamente especificada para efectuar la evaluación, y esto abarca desde las fases tempranas de diseño y a lo largo del desarrollo. Bosch [6] por su parte plantea que es posible efectuar decisiones sobre la arquitectura a cualquier nivel, puesto que se pueden imponer distintos tipos de cambios arquitectónicos, producto de una evaluación en función de los atributos de calidad esperados. Pero los dos autores concuerdan en que mientras mayor es el nivel de especificación, mejores son los resultados que produce la evaluación.

El propósito de forma general es evaluar el potencial de la arquitectura de software para obtener los atributos de calidad requeridos. Según el objetivo perseguido con la evaluación será el resultado que se obtenga, no siempre se obtienen resultados cuantitativos que permitan tomar decisiones de forma directa, generalmente lo que se recopilan son riesgos que se corren al asumir la arquitectura propuesta, dichos riesgos pueden sucederse o no, pero el arquitecto de software debe tener conocimiento para saber qué hacer en cada uno de los casos. Uno de los puntos para el éxito con una arquitectura está en que la misma presente el menor número de riesgos posibles.

La evaluación de las arquitecturas de software puede ser realizada mediante el uso de diversas técnicas y métodos. A continuación se hace un breve estudio de algunas técnicas y métodos que existen en la actualidad para llevar a cabo esta tarea.

3.3.1 Técnicas de evaluación de arquitecturas de software

Las técnicas utilizadas para la evaluación de atributos de calidad requieren información, del sistema a desarrollar, que no está disponible durante el diseño arquitectónico, debido a ello el arquitecto de software debe tener conocimientos y hacer un gran esfuerzo para crear; con poca información detallada; especificaciones y predicciones que ayuden a evaluar y que conduzcan a resultados bastante exactos.

Las técnicas de evaluación de arquitecturas de software son: evaluación basada en escenarios, evaluación basada en simulación, evaluación basada en modelos matemáticos y evaluación basada en experiencia.

En este trabajo la variante escogida es la evaluación basada en escenarios, en la cual se define que un escenario es una breve descripción de la interacción de alguno de los involucrados en el desarrollo del sistema con éste.

Los escenarios proveen un medio para concretar y entender atributos de calidad. Entre las ventajas de su uso están:

- son simples de crear y entender
- son poco costosos y no requieren mucho entrenamiento
- son efectivos

Actualmente las técnicas basadas en escenarios cuentan con dos instrumentos de evaluación relevantes, el “Utility Tree” propuesto por Kazman [29] y los “Profiles”, propuestos por Bosch [6].
De los dos instrumentos se va a utilizar el “Utility Tree” (Árbol de utilidad, del español) (Ver Figura 5), el mismo es un esquema en forma de árbol que presenta los atributos de calidad más importantes de un sistema de software en particular, estos atributos son refinados hasta el establecimiento de escenarios que especifican con suficiente detalle el nivel de prioridad de cada uno.
No existe un conjunto preestablecido de atributos, sino que son definidos por los involucrados en el desarrollo del sistema al momento de la construcción del árbol.

3.3.2 Métodos de evaluación de arquitecturas de software. Selección del método a aplicar en la arquitectura propuesta

Ha sido planteado por Kazman [29] que hasta hace poco no existían métodos de utilidad general para evaluar arquitecturas de software. Si alguno existía, sus enfoques eran incompletos, específicos, y no repetibles, lo que no brindaba mucha confianza.
Debido a esto han sido propuestos múltiples métodos de evaluación. A continuación se explican algunos de los más importantes.

Método de Análisis de Arquitecturas de Software (SAAM)

El Método de Análisis de Arquitecturas de Software (Software Architecture Analysis Method, SAAM) es el primero que fue ampliamente promulgado y documentado. Desde su creación ha demostrado ser muy útil para evaluar de forma rápida distintos atributos de calidad.
El método de evaluación SAAM se enfoca en la enumeración de un conjunto de escenarios que representan los cambios probables a los que estará sometido el sistema en el futuro. Como entrada principal, es necesaria alguna forma de descripción de la arquitectura a ser evaluada.
En su trabajo Kazman [29] plantea que las salidas de la evaluación del método SAAM son las siguientes:

- Una proyección sobre la arquitectura de los escenarios que representan los cambios posibles ante los que puede estar expuesto el sistema.
- Entendimiento de la funcionalidad del sistema, e incluso una comparación de múltiples arquitecturas con respecto al nivel de funcionalidad que cada una soporta sin modificación.

El método se puede aplicar para evaluar una arquitectura o para evaluar varias, si el objetivo de la evaluación es una sola arquitectura se obtienen los lugares en los que la misma puede fallar, pero si el objetivo son varias arquitecturas el método produce una escala relativa que permite observar qué opción satisface mejor los requerimientos de calidad con la menor cantidad de modificaciones.

Método de Análisis de Acuerdos de Arquitectura (ATAM)

El nombre del método ATAM surge del hecho de que provee una visión de cómo los atributos de calidad interactúan con otros y revelan una arquitectura que satisface las necesidades para el problema en cuestión.
Según Kazman [29], el Método de Análisis de Acuerdos de Arquitectura (Architecture Trade-off Analysis Method, ATAM) está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM, explicado anteriormente.
El método se centra en la búsqueda de enfoques arquitectónicos que son los medios empleados por la arquitectura para alcanzar los atributos de calidad.

Active Reviews for Intermediate Designs (ARID)

El método ARID fue creado para evaluar diseños parciales de la arquitectura en las etapas tempranas del desarrollo. En ARID confluyen los métodos ATAM; visto anteriormente; y Active Design Review (ADR).
ADR es utilizado para la evaluación de la calidad y la completitud en la documentación de diseños detallados de unidades del software como los componentes o módulos, a esto se le une la idea del uso de escenarios.

Método de análisis de costos y beneficios (CBAM)

Según Kazman [21] el método CBAM es un marco de referencia que no toma decisiones por los involucrados en el desarrollo del sistema, ayuda en la búsqueda y documentación de los costos,

beneficios y riesgos para ayudar a la toma de decisiones. Uno de los elementos que introduce el método son las llamadas estrategias arquitectónicas, que consisten en posibles opciones para la resolución de conflictos entre atributos de calidad presentes en una arquitectura.

3.3.3 Comparación entre métodos de evaluación

La Tabla 15 presenta una comparación entre los métodos de evaluación Software Architecture Analysis Method (SAAM), Architecture Trade-off Analysis Method (ATAM), Active Reviews for Intermediate Designs (ARID) y Cost-Benefit Analysis Method (CBAM).

	SAAM	ATAM	ARID	CBAM
Atributos de calidad contemplados	➢ Modificabilidad ➢ Funcionalidad	➢ Modificabilidad ➢ Seguridad ➢ Confiabilidad ➢ Desempeño	➢ Conveniencia del diseño evaluado	➢ Funcionalidad ➢ Modificabilidad
Objetos analizados	➢ Documentación ➢ Vistas arquitectónicas	➢ Estilos arquitectónicos ➢ Documentación ➢ Flujo de datos ➢ Vistas Arquitectónicas	➢ Especificación de los componentes	➢ Estilos arquitectónicos ➢ Documentación
Etapas del proyecto en las que se aplica	➢ Luego de que la arquitectura cuenta con funcionalidad ubicada en módulos	➢ Luego de que el diseño de la arquitectura ha sido establecido	➢ A lo largo del diseño de la arquitectura	➢ Luego de que el diseño de la arquitectura ha sido establecido
Enfoques utilizados	➢ Lluvia de ideas para escenarios y articular los requerimientos de	➢ Utility Tree y lluvia de ideas para articular los requerimientos de	➢ Revisiones de diseños, lluvia de ideas para obtener	➢ Teoría “W” ➢ Análisis de

	calidad ➢ Análisis de los escenarios para verificar funcionalidad o estimar el costo de los cambios	calidad ➢ Análisis arquitectónico que detecta puntos sensibles, puntos de balance y riesgos	escenarios	escenarios

Tabla 15. Comparación entre métodos de evaluación.

Al revisar y hacer un estudio de la Tabla 15 se puede observar que los métodos con más profundidad de evaluación son el ATAM y el SAAM, ambos pertenecen al grupo de técnicas de evaluación basadas en escenarios y son métodos de análisis para hacer evaluaciones de forma cualitativa.

En los dos métodos los requerimientos de calidad son evaluados y aceptados a través de la evolución, así como las arquitecturas son diseñadas y evaluadas. No están diseñados para ninguna clase de atributos de calidad o métricas de software específicas. Cualquier atributo de calidad puede ser analizado con estos métodos.

El método ATAM le da gran importancia a los *tradeoffs*[9] y plantea que una arquitectura por sí sola no alcanza todos los atributos de calidad que se necesitan.

Muchos atributos de calidad presentan una característica: si se mejora uno se perjudica otro por tanto; con ATAM; se busca encontrar un punto admisible entre las distintas arquitecturas tomando no solo en cuenta los aspectos técnicos.

Se ha decidido utilizar el método SAAM pues en el presente trabajo no se busca analizar varias arquitecturas para llegar a un conjunto de soluciones deseables sino evaluar la arquitectura que se propone.

Es el método SAAM el que brinda una metodología basada en escenarios para evaluar las propiedades de calidad de una arquitectura, también se acomoda mucho a la arquitectura propuesta, pues este método permite ser aplicado en el momento que se diseña la arquitectura y después que la misma ha sido aplicada en un software, en el caso específico del trabajo será aplicado en la fase de diseño de la arquitectura.

[9] Un tradeoffs es la decisión que se tiene que tomar entre arquitecturas alternativas para llegar a un conjunto de soluciones deseables

El resultado de SAAM es una arquitectura superior a otras con respecto a varios escenarios. Es muy importante que los escenarios que se utilicen para evaluar la arquitectura se basen en futuros escenarios reales, posibilitando obtener resultados fiables.

3.4 Aplicación del método SAAM a la arquitectura de software definida

3.4.1 Propósito

El método tiene como propósito fundamental basarse en escenarios que se definan y que tengan una repercusión negativa en el sistema.

3.4.2 Contexto y escenarios

Luego de tener claridad en cuanto al contexto del sistema entonces se busca evaluar a través de escenarios los atributos de calidad que se muestren más complejos e imprecisos de lograr con la arquitectura.
Los escenarios representan la base del método, son técnicas cualitativas de evaluación de arquitecturas que simulan la interacción entre un interesado y el sistema. Los mismos toman dos clasificaciones, directos o indirectos:

- Un escenario directo es el que puede satisfacerse sin la necesidad de modificaciones en la arquitectura.
- Un escenario indirecto es aquel que requiere modificaciones en la arquitectura para poder satisfacerse.

Entre los escenarios indirectos es posible ver interacciones, las mismas se producen cuando dos o más escenarios indirectos requieren cambios sobre el mismo componente o en las conexiones entre estos.

3.4.3 Roles

Existen varios roles involucrados en la aplicación del método SAAM, los mismos se muestran en la tabla 5. En el presente trabajo se usa una muestra de estos roles para aplicar el método SAAM.

Roles

Interesados externos	➢ Organización cliente ➢ Usuarios finales ➢ Administradores del sistema
Interesados internos	➢ Arquitectos de Software ➢ Analistas de requerimientos
Equipo SAAM	➢ Líder ➢ Expertos en el dominio de la aplicación ➢ Expertos externos en arquitectura ➢ Secretario

Tabla 16. Roles involucrados en la aplicación del método SAAM

3.4.4 Pasos para la aplicación del método SAAM

El método seleccionado consta de seis pasos para la evaluación, a continuación se explicará en que consiste cada uno para luego proceder a aplicarlos a la arquitectura propuesta en el capítulo anterior.

1. Descripción de la arquitectura: La arquitectura (o las candidatas) debe ser descrita haciendo uso de alguna notación arquitectónica que sea común a todas las partes involucradas en el análisis. Deben incluirse los componentes de datos y conexiones relevantes, así como la descripción del comportamiento general del sistema.
2. Desarrollo de escenarios: Esta tarea conlleva ilustrar los tipos de actividades, el sistema de apoyo y los tipos de cambio que pueda tener el sistema en el tiempo. Es importante capturar todos los usos importantes de un sistema, pues representan las tareas pertinentes a varias personas que tienen alguna interacción con el sistema: usuario final, administrador del sistema, arquitecto de software, programadores, entre otras.
3. Clasificación de los escenarios: Los escenarios toman dos clasificaciones: directos e indirectos, a estos últimos se les presta mayor atención dentro del método. La clasificación es la primera

indicación de la idoneidad de una arquitectura con respecto a la satisfacción de un conjunto de escenarios.

4. Evaluación individual de los escenarios indirectos: Para cada escenario indirecto, se listan los cambios necesarios sobre la arquitectura y se plantea su prioridad, para ello se tiene en cuenta el grado de dificultad de la modificación necesaria, la unidad de medida del grado de dificultad varía en dependencia del escenario en cuestión, puede ser: horas-hombre, líneas de código afectado, etc. Las clasificaciones para asignar la prioridad son las siguientes: mínima, moderada y considerable.
5. Evaluación de la interacción entre escenarios indirectos: En este punto hay que evaluar cuando dos o más escenarios indirectos proponen cambios sobre un mismo componente o en las conexiones entre estos.
6. Resultados de la evaluación: Este punto del proceso es subjetivo, en él se concluye el resultado de la arquitectura en dependencia de las evaluaciones realizadas con respecto a los escenarios.

Hasta este punto se ha hecho una revisión de en que consiste cada paso del método SAAM, a continuación se comienzan a aplicar cada uno de ellos a la arquitectura propuesta.

Descripción de la arquitectura: En este primer paso se orienta leer el capítulo anterior o capítulo 2, titulado "Propuesta de arquitectura para el desarrollo de sistemas de gestión web" donde se define el dominio y requerimientos de las aplicaciones que se pretenden construir con esta arquitectura que se propone, específicamente las secciones 2.2 y 2.3.

En las tablas 6, 7, 8 y 9 que se muestran a continuación se desarrollan varios pasos del método que se aplica, los mismos son: desarrollo de escenarios, clasificación de los escenarios y evaluación individual de los escenarios indirectos. Para este proceso se revisaron los escenarios más generales de proyectos de gestión sobre la web que se han recogidos de los usuarios finales, administradores, arquitectos y desarrolladores de varios proyectos de la Universidad.

Usuarios finales	
Escenario # 1: El sistema permite cancelar operaciones que ya han sido comenzadas.	
Atributo de calidad: Usabilidad	Característica: Cancelación de operaciones

Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: El sistema en cada interfaz incluye por defecto acciones de cancelación que le permiten al usuario final realizar dicha operación. Además el uso de JPA permite hacer rollback a operaciones que se estén realizando con la base de datos.	
Escenario # 2: El sistema incluye puntos de prueba y recopilación de datos para facilitar la evaluación de su robustez, corrección y usabilidad de manera sistemática.	
Atributo de calidad: Usabilidad	Característica: Evaluación del sistema
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: Es posible con solo configurar el portal server (Liferay). Para ello se activa el modo de chequeo de las preferencias del usuario y se puede obtener toda la información relativa a las páginas visitas, contenidos revisados, enlaces consultados, etc.	
Escenario # 3: Los usuarios cuentan con los medios para reducir la cantidad de trabajo perdido debido a fallos del sistema.	
Atributo de calidad: Usabilidad	Característica: Recuperación ante fallos
Clasificación (Directo/Indirecto): Indirecto	Prioridad (Escenario indirecto): Moderada
Solución: Para este escenario es necesaria una implementación de los portlet del sistema de manera tal que cuando ocurra un fallo se pueda guardar el estado de las operaciones hasta ese momento de forma persistente y recuperarlo una vez que se restablezca el servicio. Esto no es una tarea simple en aplicaciones web donde por lo general solo se utilizan las bondades del navegador para guardar información del usuario.	
Escenario # 4: El sistema brinda alternativas y mecanismos para otorgar acceso a los usuarios que hayan olvidado o perdido su clave de acceso.	
Atributo de calidad: Usabilidad	Característica: Recuperar contraseñas olvidadas

Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: El portal server Liferay permite obtener la contraseña olvidada a partir de la pregunta/respuesta definida y la envía por correo al usuario.	
Escenario # 5: El sistema es fácilmente configurable para el despliegue en múltiples lenguajes.	
Atributo de calidad: Usabilidad	Característica: Mostrar el contenido en varios idiomas
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: Todos los portlets que se desarrollan, así como las funcionalidades básicas del portal server pueden ser desplegados en más de 20 idiomas. Con tan solo configurar en el sistema los posibles idiomas disponibles, el usuario podrá seleccionar en tiempo de ejecución el preferido.	
Escenario # 6: El sistema asegura a sus usuarios poder modificar fácilmente sus interfaces.	
Atributo de calidad: Usabilidad	Característica: Modificar las interfaces
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: El portal server Liferay, permite configurar las preferencias del usuario y da la posibilidad de que cada usuario construya su entorno de trabajo como lo desee. El usuario puede tener la posibilidad de definir la estructura de sus páginas, los portlets que incluye en las mismas, los colores, fuentes y demás recursos que necesite modificar para una vista personalizada.	
Escenario # 7: El sistema permite al usuario cambiar rápidamente entre estas tareas y aplicaciones con las que trabaja.	
Atributo de calidad: Usabilidad	Característica: Apoyar a las actividades múltiples
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):

Solución: En primer lugar las aplicaciones desarrolladas sobre esta arquitectura funcionarán en el cliente con un navegador web, por tanto el usuario podrá cambiar entre aplicaciones sin ninguna dificultad. Por otra parte y lo más importante de esta arquitectura es la posibilidad que brinda para integrar varias aplicaciones en un mismo entorno de trabajo, evitando así que el usuario tenga que necesariamente cambiar de entorno de trabajo.	
Escenario # 8: El sistema es capaz de mostrar al usuario cuanto demora la tarea que está realizando para de esta forma ayudar en el avance de su trabajo permitiéndole trabajar en otra tarea mientras la anterior se completa.	
Atributo de calidad: Usabilidad	Característica: Mostrar el tiempo de duración de las tareas
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: Todos los portlets desarrollados con el framewoks OpenXava, pueden resolver este escenario sin dificultad. Tan solo es necesario definir en las acciones que se definan para el portlet si este demora o no, en caso afirmativo el sistema automáticamente mostrará una barra de estado con el tiempo estimado de duración de la acción que se esta realizando.	
Escenario # 9: El sistema permite a los usuarios buscar datos de forma rápida; en un área amplia y coherente; basándose en los criterios introducidos.	
Atributo de calidad: Usabilidad	Característica: Apoyar las búsquedas
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: El portal server Liferay, incluye un motor de búsqueda basado en Lucene que permite indexar toda la información y encontrar de manera rápida la misma en cualquiera de los portlets que lo permitan. Ejemplo en los portlets de wiki, foro, blog, librería, etc.	
Escenario # 10: El sistema permite que los usuarios accedan a los datos desde distintos puntos de vista.	
Atributo de calidad: Usabilidad	Característica: Permitir accesibilidad a cada una de las operaciones del sistema.

Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: Esta tarea se puede resolver de manera muy fácil utilizando el framework OpenXava. Pues el mismo permite en cada acción que se define para el usuario, agregarle la combinación de teclas que permitirá ejecutar dicha acción y el framework automáticamente garantiza esta funcionalidad.	

Tabla 17. Escenarios definidos para los usuarios finales.

Arquitecto de software	
Escenario # 1: El sistema posee componentes capaces de leer datos provenientes de otros sistemas. El sistema posee componentes capaces de producir datos para otro sistema.	
Atributo de calidad: Funcionalidad	Característica: Interoperabilidad
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: Cada portlet que se desarrolla y que se incluye en el portal server puede estar accediendo a una fuente de datos diferente. Para ello solo es necesario configurar el servidor de aplicaciones para que permita la conexión con la fuente de datos. Además los portets desarrollados con OpenXava permiten utilizar multiesquema. Por otra parte cada portlet es expuesto como un servicio mediante el estándar WSRP, lo que permite que puedan ser accedidos remotamente.	
Escenario # 2: El sistema detecta la actuación de un intruso e impide acceso a los componentes que manejen información sensible. El sistema asegura que los componentes no pierdan datos ante un ataque (interno o externo).	
Atributo de calidad: Funcionalidad	Característica: Seguridad
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: La seguridad es un tema donde nunca se puede estar conforme con las medidas tomadas, en dependencia de la sensibilidad de la información que se maneje. En esta arquitectura el mecanismo de seguridad del sistema es controlado por el portal server. Este garantiza que la información sea accedida solo por los usuarios con los privilegios requeridos utilizando para ello estándares como SSL 2. Además la validación de la seguridad y los datos de entrada a nivel de cliente, de servidor y de	

gestor de base de datos evita los posibles ataques de intrusos y evita de esta forma la perdida de información del sistema.
Escenario # 3: Los componentes del sistema manejan entradas de datos incorrectas.
Atributo de calidad: Fiabilidad / Característica: Madurez
Clasificación (Directo/Indirecto): Directo / Prioridad (Escenario indirecto):
Solución: La validación de los datos introducidos es garantizada a nivel de cliente, servidor y gestor de base de datos. Con tan solo anotar; utilizando JPA; cada uno de los atributos de los componentes de negocio el framework OpenXava genera todas las validaciones necesarias.
Escenario # 4: Ante problemas con el ambiente un subconjunto determinado de los componentes puede continuar prestando sus servicios.
Atributo de calidad: Fiabilidad / Característica: Capacidad de restablecimiento o recuperación.
Clasificación (Directo/Indirecto): Directo / Prioridad (Escenario indirecto):
Solución: Debido a que el portal server es quien garantiza el ciclo de vida de los portlets y son estos los que tienen las funcionalidades específicas, en caso que un portlet este corrupto o su funcionamiento no sea el adecuado, el portal server lo trata como un ente aislado y de esta forma no interfiere en la presentación del servicio de otros portlet, es decir es posible que en una misma pagina existan portlets funcionando y portlets defectuosos y el sistema continuará funcionando sin dificultad.
Escenario # 5: El sistema debe recibir los servicios de sus componentes en el transcurso de un tiempo indicado.
Atributo de calidad: Eficiencia / Característica: Tiempo de comportamiento
Clasificación (Directo/Indirecto): Indirecto / Prioridad (Escenario indirecto): Moderada
Solución: Esta funcionalidad debe ser desarrollada. La cual debe establecer los tiempos de respuesta para componentes y

controlar los mismos para alertar sobre posibles ineficiencias del sistema.	
Escenario # 6: Los componentes pueden compartir recursos adecuadamente. El sistema controla que ningún componente se quede sin recursos cuando los necesita.	
Atributo de calidad: Eficiencia	Característica: Recursos utilizados
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: La arquitectura propuesta garantiza una alta concurrencia a los recursos. Utilizando para ello definiciones de JPA que permiten obtener los recursos bajo demanda y permitir alta concurrencia de diferentes portlet al mismo recurso. Para ello solo es necesario anotar los componentes de negocio con @version y FetchType.LAZY; anotaciones definidas en JPA.	
Escenario # 7: Es posible verificar el estado de los componentes del sistema. El sistema brinda facilidad para adaptar un componente. El sistema debe facilitar la sustitución/adaptación de un componente.	
Atributo de calidad: Mantenibilidad	Característica: Habilidad de cambio, estabilidad, prueba.
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: Para verificar el estado de un componente solo es necesario ubicarlo en una página y probar sus funcionalidades. Para adaptar un portlet es necesario editar las configuraciones del mismo. Las cuales pueden ser accedidas por lo general por usuarios con mayores privilegios. Por otra parte la sustitución/adaptación de portlets es una funcionalidad que garantiza el portal server. En el mismo instalar y desinstalar portlet es una tarea común.	
Escenario # 8: El sistema debe continuar funcionando correctamente aún cuando los servicios de los componentes provistos por el ambiente varíen.	
Atributo de calidad: Portabilidad	Característica: Adaptabilidad
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):

Solución: Cuando los portlets consumen servicios y estos cambian, solo se necesita cambiar las configuraciones de los mismos. Mientas esta tarea no se realice el portlet no estará disponible, sin embargo el resto de los portlets del sistema seguirán brindando los servicios.	
Escenario # 9: Los componentes pueden instalarse fácilmente en todos los ambientes donde debe funcionar.	
Atributo de calidad: Portabilidad	Característica: Capacidad de instalación
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: La arquitectura propuesta puede instalarse en múltiples escenarios, por ejemplo, permite utilizar 14 gestores de bases datos diferentes, 16 servidores de aplicaciones, utilizando para ello cualquier sistema operativo que soporte la máquina virtual de java (Windows, Linux, Unix, Mac).	

Tabla 18. Escenarios definidos para el arquitecto de software.

Desarrolladores	
Escenario # 1: Se desea cambiar la interfaz del usuario de forma tal que esta modificación no influya en las funcionalidades del sistema.	
Atributo de calidad: Modificabilidad	Característica: Modificar las interfaces
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: Un usuario con derechos de administración en el portal server puede cambiar el esquema de las páginas, el tema visual y la navegación sin necesidad de programar. Solo configurando el portal server.	
Escenario # 2: Se desea controlar que usuarios han creado, modificado o eliminado cada uno de los registros del sistema.	
Atributo de calidad: Auditoría	Característica: Control de accesos

Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: El framework OpenXava incluye un proyecto llamado AccesTracking que permite integrarse con los nuevos proyectos desarrollados y garantiza el registro en cada una de las operaciones CRUD que realizan los usuarios. De esta forma se garantiza la auditoría del sistema.	
Escenario # 3: Integración de nuevos reportes para el usuario.	
Atributo de calidad: Modificabilidad	Característica: Flexibilidad
Clasificación (Directo/Indirecto): Indirecto	Prioridad (Escenario indirecto): Moderada
Solución: Los modelos de OpenXava permiten realizar reportes simples, pero cuando se necesita de reportes más complejos es necesario integrar herramientas más potentes y dedicadas a esta tarea. En los proyectos OpenXava se pueden integrar a nivel de código de manera muy natural los reportes generados con Jasper Report.	
Escenario # 4: Se desean realizar pruebas automáticamente a cada uno de los módulos desarrollados.	
Atributo de calidad: Eficiencia	Característica: Gestión de pruebas automáticas
Clasificación (Directo/Indirecto): Indirecto	Prioridad (Escenario indirecto): Moderada
Solución: OpenXava provee integración completa con Junit. Esto posibilita que el desarrollador pueda programar casos de prueba que cubran la mayoría de los escenarios del portlet y ejecutar las mismas de manera automática, simulándose como si el usuario estuviese dando click en la interfaz. De esta forma se minimizan los tiempos necesarios para probar cada uno de los portlets desarrollados.	
Escenario # 5: Se desea integrar un dispositivo de entrada/salida.	
Atributo de calidad: Modificabilidad	Característica: Flexibilidad

Clasificación (Directo/Indirecto): Indirecto	Prioridad (Escenario indirecto): Considerable
Solución: En el caso que se deseen integrar al proyecto dispositivos externos como impresoras, escáner u otros, si es necesario implementar las interfaces correspondientes para integrarse con los dispositivos.	
Escenario # 6: Se desea cambiar una parte o un proceso completo contenido en uno de los módulos del sistema.	
Atributo de calidad: Modificabilidad	Característica: Flexibilidad y modularidad
Clasificación (Directo/Indirecto): Indirecto	Prioridad (Escenario indirecto): Moderada
Solución: Esto depende de como se haya desarrollado el portlet. Si se utilizó un motor de BPM integrado a Liferay (jBPM o Intalio) solo es necesario editar el nuevo proceso y desplegarlo en el motor de proceso. Esta variante no tiene ningún costo en codificación asociado. En caso que el proceso se haya codificado completamente dentro del portlet es necesario rehacer el portlet por completo y reinstalarlo en el portal server.	
Escenario # 7: Modificar la distribución del sistema.	
Atributo de calidad: Modificabilidad	Característica: Variabilidad
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: El portal server Liferay se integra con el proyecto Terracota, permitiendo así realizar un despliegue en un clúster de servidores. Además los servidores de aplicaciones y gestores de bases de datos pueden cambiar de estructura y ruta y solo se necesita configurar las conexiones en el fichero destinado a este afecto y el sistema continúa su funcionamiento.	

Tabla 19. Escenarios definidos para los desarrolladores.

Administrador
Escenario # 1: Se desea crear nuevos usuarios, roles y grupos en el sistema.

Atributo de calidad: Seguridad	Característica: Gestión de usuarios
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: El portal server Liferay posee un panel de control para el administrador del sistema que le permite gestionar todos los usuarios, roles y grupos de usuarios del mismo sin necesidad de implementación alguna.	
Escenario # 2: Se desea integrar el sistema a servicios de autenticación de usuarios centralizados.	
Atributo de calidad: Seguridad	Característica: Integración con servicios especializados en gestión de acceso.
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: El portal server Liferay se puede integrar solo haciendo algunas configuraciones con sistemas de autenticación centralizada como LDAP, CAS u OpenID.	
Escenario # 3: Se desean instalar, desinstalar y/o configurar portlets.	
Atributo de calidad: Extensibilidad	Característica: Instalar, desinstalar y configurar nuevos módulos en el sistema.
Clasificación (Directo/Indirecto): Directo	Prioridad (Escenario indirecto):
Solución: Una vez que el sistema ya está en plena producción pueden ser integrados nuevos porlets al sistema sin dificultad. El portal server posee una herramienta que permite instalar y desinstalar portlet en tiempo de ejecución, así como configurar las preferencias de cada uno.	

Tabla 20. Escenarios definidos para el administrador.

Evaluación de la interacción entre escenarios indirectos

Los escenarios indirectos identificados en la evaluación realizada son los siguientes:

- Los usuarios cuentan con los medios para reducir la cantidad de trabajo perdido debido a fallos del sistema.
- El sistema debe recibir los servicios de sus componentes en el transcurso de un tiempo indicado.
- Integración de nuevos reportes para el usuario.
- Se desean realizar pruebas automáticamente a cada uno de los módulos desarrollados.
- Se desea cambiar un dispositivo de entrada/salida.
- Se desea cambiar una parte o un proceso completo contenido en uno de los módulos del sistema.

Cada uno de los escenarios indirectos identificados tiene incidencia sobre los nuevos portlet que se desarrollen y están asociados a necesidades puntuales en el desarrollo de los sistemas.

Los mismos pueden encontrarse de manera aislada o pueden interactuar varios en un mismo portlet, por ejemplo pudiera ser necesario en un portlet integrar un reporte dinámico, realizar pruebas automáticas y conectarlo a un dispositivo de entrada y salida. Esta interacción complejiza el desarrollo del portlet en cuestión, pero la interacción entre los mismos no implica ineficiencias en el sistema, el único inconveniente en este caso sería el costo en tiempo y recursos para el desarrollo.

Realizando un análisis de estos escenarios indirectos se puede observar que los mismos están diseñados para requisitos específicos y por tanto es muy difícil que una arquitectura base genérica como la propuesta cubra de manera directa este tipo de escenario.

Resultados de la evaluación

Después de haber realizado las evaluaciones a la arquitectura propuesta se han observado los beneficios que brinda el proceso, primeramente ayuda a centrar la atención en los detalles importantes de la arquitectura y permite al equipo de desarrollo hacer caso omiso de las áreas menos críticas. Sin duda la aplicación de este método evita muchos de los problemas a los que se puede enfrentar después el equipo de desarrollo.

Se evaluaron en total 29 escenarios, de ellos 23 resultaron directos y 6 indirectos. De los escenarios indirectos 5 tienen una prioridad moderada y uno solo muestra una prioridad considerable, en este caso se debe prestar una especial atención y priorizarlo si en la aplicación que se desee desarrollar está presente este escenario, el mismo consiste en lograr la conexión de dispositivos de entrada/salida al sistema.

La arquitectura resuelve; sin necesidad de hacer transformaciones; la mayoría de los escenarios evaluados, demostrando con esto que la arquitectura es altamente escalable.

Hay que tener en cuenta que la mayoría de los gastos de un software se producen en la fase de mantenimiento y mejoras de funcionalidades, por ello se considera muy importante las funcionalidades que brinda esta arquitectura para lograr estos factores.

El rol que más escenarios indirectos presenta es el desarrollador, pues el atributo de calidad que más se presenta a este rol es la modificabilidad, son estas las personas que se vinculan directamente en las modificaciones o funcionalidades que se agreguen o se necesiten en el sistema.

Según los basamentos seguidos por el método SAAM a mayor cantidad de escenarios evaluados de directos, mayor será la completitud de la arquitectura propuesta, en el caso del presente trabajo la arquitectura resuelve de forma directa el 80% de los escenarios de forma directa (ver Figura 4) esto demuestra que el desarrollo se agiliza en más de un 50%.

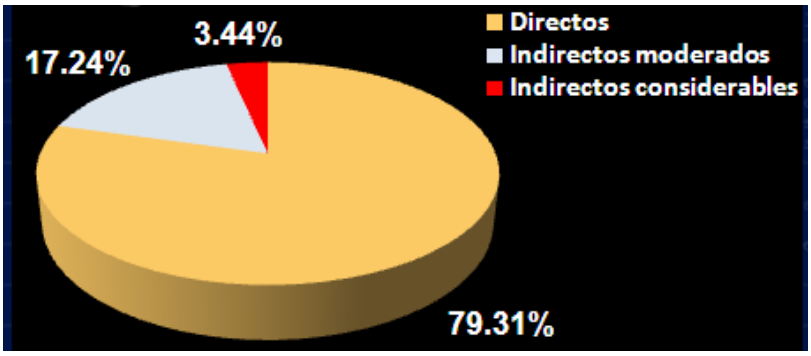


Figura 4. Gráfica con resultado de escenarios evaluados

Para finalizar la evaluación de la arquitectura propuesta es preciso resaltar que la misma esta siendo aplicada en el desarrollo de dos proyectos de gestión en la Universidad. Los proyectos son: Sistema para la Gestión del Ingreso a la UCI y Sistema para la Gestión de la Producción y los Recursos Humanos del Polo de Informática Jurídica de la Facultad 3. La arquitectura ha contribuido al desarrollo rápido de estas herramientas, garantizando una alta flexibilidad y escalabilidad en las mismas. Lo anterior puede ser corroborado con los avales que se muestran en el anexo 4.

3.5 Conclusiones

En este capítulo luego de realizar varias búsquedas a nivel nacional e internacional se obtuvo como aporte sustancial un conjunto de escenarios definidos desde cuatro roles diferentes, los mismos pueden ser aplicados a cualquier arquitectura usada o por usar en un sistema de gestión sobre la web.

Se demostró; a través de la evaluación utilizando el método SAAM; que la arquitectura propuesta reduce el tiempo, esfuerzo, costo y complejidad de crear y mantener los productos que se hagan sobre ella pues recoge los aspectos más comunes de esta línea de productos a través de la consolidación y la reutilización que brinda.

CONCLUSIONES GENERALES

Con la culminación de la presente investigación se han arribado a varias conclusiones de forma general:

- Se definieron un conjunto de requisitos genéricos de referencia para sistemas de gestión sobre la web que puede ser incluidos por los analistas en cualquier software de esta línea.
- Se realizó un análisis de cada una de las herramientas candidatas; todas basadas en software libre; para cubrir las necesidades de los componentes de la arquitectura.
- Se provee una configuración completa del entorno de trabajo para utilizar la arquitectura propuesta.
- Se recopilaron desde los roles de administrador del sistema, arquitecto de software, desarrollador y usuarios finales 29 escenarios de los más importantes que pueden surgir en los proyectos de gestión web, estos escenarios están listos para con ellos evaluar cualquier arquitectura pensada para los sistemas de esta línea.
- La arquitectura propuesta resuelve de manera directa el 80% de los escenarios que le fueron aplicados permitiendo agilizar el desarrollo en más de un 50%.
- La arquitectura garantiza alta productividad en los equipos de desarrollo de software.
- La arquitectura fomenta la reutilización de componentes de alto nivel, este factor irá en aumento cada vez que se concluya un nuevo proyecto.
- La arquitectura facilita el desarrollo rápido basado en un modelo de fábrica de software.
- La cantidad de frameworks para desarrollar se reduce considerablemente, hasta el momento los proyectos necesitan como mínimo tres frameworks diferentes para desarrollar una aplicación, en la arquitectura que se propone solo es necesario el framework OpenXava, que además presenta una curva de aprendizaje baja.
- La arquitectura propuesta no fuerza a los arquitectos a utilizar una herramienta en específico pues es una arquitectura genérica.

RECOMENDACIONES

Luego de analizar los resultados del trabajo resulta factible llegar a las siguientes recomendaciones para trabajos futuros:

- Evaluar la arquitectura propuesta después de aplicada en el desarrollo de algún proyecto.
- Realizar un estudio de herramientas BPMS que se puedan integrar a la arquitectura como JBoss JBPM o Intalio.
- Sugerir a las entidades correspondientes (Dirección de Producción de la Facultad 3, Dirección Técnica de la Infraestructura Productiva de la UCI) que valoren el posible uso de la arquitectura propuesta y/o que consideren los principios de reutilización de componentes que esta posibilita para que puedan generalizarla.

BIBLIOGRAFÍA

1. *A Service Oriented Architecture for Portals Using Portlets.* **Akram, Asif. 2007.** UK : CCLRC Daresbury Laboratory, 2007.
2. Asynchronous JavaScript And XML(AJAX). [Online] [Cited: febrero 5, 2009.] http://es.wikipedia.org/wiki/AJAX.
3. **autores, Colectivo de. 2003.** *MDA Guide Version 1.0.1.* 2003.
4. **Barbacci, M, et al. 1995.** *Quality Attributes.* s.l. : Carnegie Mellon University, 1995.
5. **BEA. 2006.** *BEA WebLogic Portal: Portlet Development Guide.* 2006.
6. **Bosch, J. 2000.** *Design & Use of Software Architectures.* s.l. : Addison Wesley, 2000.
7. **Bravo, Juan. 2007.** *Estudio Sectores de DBK: "Software" .* Madrid : s.n., 2007.
8. **Buschmann, F, et al. 1996.** *Pattern Oriented Software Architecture. A system of patterns.* Inglaterra : John Wiley & Sons, 1996.
9. **Casas Cuevas, Juan and Arenas Fernández, Mercedes. 2003.** *OCL (Object Constraint Language).* [digital] Valencia : Universidad Politécnica de Valencia, 2003.
10. **Corredera de Colsa, Luis Enrique. 2005.** Arquitectura dirigida por modelos para J2ME. *Arquitectura dirigida por modelos para J2ME.* [Online] mayo 2005. [Cited: 2 14, 2009.] http://personal.telefonica.terra.es /web/lencorredera/mda_j2me.pdf.
11. Enterprise Java Bean (EJB). [Online] [Cited: febrero 1, 2009.] http://java.sun.com/products/ejb.
12. eXo Platform. [Online] [Cited: febrero 4, 2009.] http://www.exoplatform.com.
13. **Gartner. 2004.** *Gartner Predicts: The future of it Gartner symposium.* Barcelona, España : s.n., 2004.
14. **González Aller, Javier. 2009.** Microsoft. *Los nuevos retos en las aplicaciones de gestión y administración empresarial.* [Online] 2009. [Cited: 3 19, 2009.] http://www.microsoft.com/spain/empresas/soluciones/nuevos_retos.mspx.
15. **GridSphere.** GridSphere. [Online] [Cited: enero 28, 2009.] http://www.gridsphere.org.
16. **Hepper, Stefan, et al. 2005.** *Portlets and Apache Portals.* s.l. : Manning Publications, 2005.
17. Hibernate. [Online] [Cited: febrero 6, 2009.] http://www.hibernate.org/.
18. **Hofmeister, C, Nord, R and Soni, D. 2000.** *Applied Software Architecture.* s.l. : Addison, 2000.

19. **IEEE. 2004.** IEEE Standards Association. *IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.* [Online] 2004. [Cited: 12 2, 2009.] http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html.
20. IFrame. [Online] [Cited: febrero 6, 2009.] http://es.wikipedia.org/wiki/Iframe.
21. **In, H, Kazman, R and Olson, D. 2001.** *From Requirements Negotiation to Software Architectural Decisions.* s.l. : Software Engineering Institute, Carnegie Mellon University, 2001.
22. **ISO/IEC 9126. 1998.** *Information Technology – Software Product Quality – Part 1.* 1998.
23. **Jacobson, Ivar, Booch, Grady and Rumbaugh, James. 2000.** *El proceso unificado de desarrollo de software.* Madrid : Addison Wesley, 2000.
24. Java Message Service(JMS). [Online] [Cited: febrero 1, 2009.] http://java.sun.com/products/jms/.
25. Java Server Page (JSF). [Online] [Cited: febrero 6, 2009.] http://java.sun.com/javaee/javaserverfaces/.
26. JBoss Portal. [Online] [Cited: febrero 3, 2009.] http://jboss.org/jbossportal/.
27. **JMatter. 2008.** JMatter. *JMatter.* [Online] 2008. [Cited: 3 25, 2009.] http://www.jmatter.org/.
28. **Kaiser, S H. 2005.** *Software Paradigms.* 2005.
29. **Kazman, R, Clements, P and Klein, M. 2001.** *Evaluating Software Architectures. Methods and case studies.* s.l. : Addison Wesley, 2001.
30. Liferay. [Online] [Cited: enero 15, 2009.] http://www.liferay.com.
31. *Liferay Portal overview.* **Shum, Joseph. 2008.** 2008.
32. **Lomboz. 2008.** Lomboz. *OW2 Lomboz.* [Online] 2008. [Cited: 1 14, 2009.] http://lomboz.ow2.org/.
33. **Madrid Vicencio, Alejandro. 2008.** Java Specification Request 168 (JSR-168). *j-portals.* [Online] 10 2, 2008. [Cited: 2 6, 2009.] www.j-portals.com.
34. **Madrid Vicencio, Alejandro. 2008.** Portales Empresariales basados en Estándares. *j-portals.* [Online] 10 2, 2008. [Cited: 2 5, 2009.] www.j-portals.com.
35. **NetBeans. 2009.** NetBeans. *Portal Pack Project.* [Online] 2 10, 2009. [Cited: 1 11, 2009.] http://portalpack.netbeans.org/.
36. **Oracle. 2008.** *Oracle WebLogic Portal: Portal Development Guide.* 2008.
37. **Paniza, Javier. 2008.** *OpenXava: Guía de referencia versión 3.1.* 2008.

38. **Polgar, Jana. 2005.** *Building and Managing Enterprise-Wide Portals.* s.l. : Idea Group Publishing, 2005.
39. *PostgreSQL.* **Aliaga Ibarra, Antonio and Miani Flores, Marcos Agustin. 2008.** 2008.
40. **Pressman, Roger S. 2002.** *Ingeniería del Software: Un enfoque práctico.* 2002.
41. **Raposo Vargas, Sergio. 2007.** Open CMS Hispano: La comunidad de Open CMS en castellano. *CMS, Repositorios y Gestores de Portales.* [Online] 9 13, 2007. [Cited: 3 2009, 4.] http://www.opencmshispano.com.
42. Sakai Project. [Online] [Cited: febrero 6, 2009.] http://sakaiproject.org/portal.
43. Servicio de Autentificación Centralizada (CAS). [Online] [Cited: febrero 2, 2009.] http://www.ja-sig.org/products/cas/.
44. Spring. [Online] [Cited: febrero 6, 2009.] http://www.springsource.org/.
45. Structs. [Online] [Cited: enero 15, 2009.] http://struts.apache.org.
46. **Sun Microsystems.** Java Community Process. *JSRs: Java Specification Requests.* [Online] [Cited: 1 16, 2009.] http://jcp.org/en/jsr/overview.
47. **Sun Microsystems.** Java Community Process. *Java Community Process.* [Online] [Cited: 1 13, 2009.] http://jcp.org/en/home/index.
48. **Sun Microsystems.** Sun Developer Network (SDN). *Java 2 Platform, Enterprise Edition (J2EE) Overview.* [Online] [Cited: 2 20, 2009.] http://java.sun.com/j2ee/overview.html.
49. uPortal. [Online] [Cited: febrero 6, 2009.] http://www.uportal.org/.
50. WebOS. [Online] http://es.wikipedia.org/wiki/WebOS.
51. **WebSphere, IBM. 2006.** *Portlet Development Workbook Using the Standard API.* 2006.
52. **Wiseman, Geoffrey. 2007.** InfoQ. *Naked Objects adds Java 1.5, Injection, Hibernate.* [Online] 11 14, 2007. [Cited: 3 10, 2009.] http://www.infoq.com/news/2007/11/naked-objects-3.

ANEXOS

Anexo 1

Tabla 21. Descripción de atributos de calidad observables vía ejecución.

Atributo de Calidad	Descripción
Disponibilidad (Availability)	Es la medida de disponibilidad del sistema para el uso.
Confidencialidad (Confidentiality)	Es la ausencia de acceso no autorizado a la información.
Funcionalidad (Functionality)	Habilidad del sistema para realizar el trabajo para el cual fue concebido.
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.
Confiabilidad (Reliability)	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo.
Seguridad externa (Safety)	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información.
Seguridad interna (Security)	Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos.

Anexo 2

Tabla 22. Descripción de atributos de calidad no observables vía ejecución

Atributo de Calidad	Descripción

Configurabilidad (Configurability)	Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema.
Integrabilidad (Integrability)	Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente al ser integrados.
Integridad (Integrity)	Es la ausencia de alteraciones inapropiadas de la información.
Interoperabilidad (Interoperability)	Es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema.
Modificabilidad (Modifiability)	Es la habilidad de realizar cambios futuros al sistema.
Mantenibilidad (Maintainability)	Es la capacidad de someter a un sistema a reparaciones y evolución, además de lograr que sea de forma rápida y a bajo costo.
Portabilidad (Portability)	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos.
Reusabilidad (Reusability)	Es la capacidad de diseñar un sistema de forma tal que su estructura o parte de sus componentes puedan ser reutilizados en futuras aplicaciones.
Escalabilidad (Scalability)	Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental.
Capacidad de Prueba (Testability)	Es la medida de la facilidad con la que el software, al ser sometido a una serie de pruebas, puede demostrar sus fallas.

Anexo 3

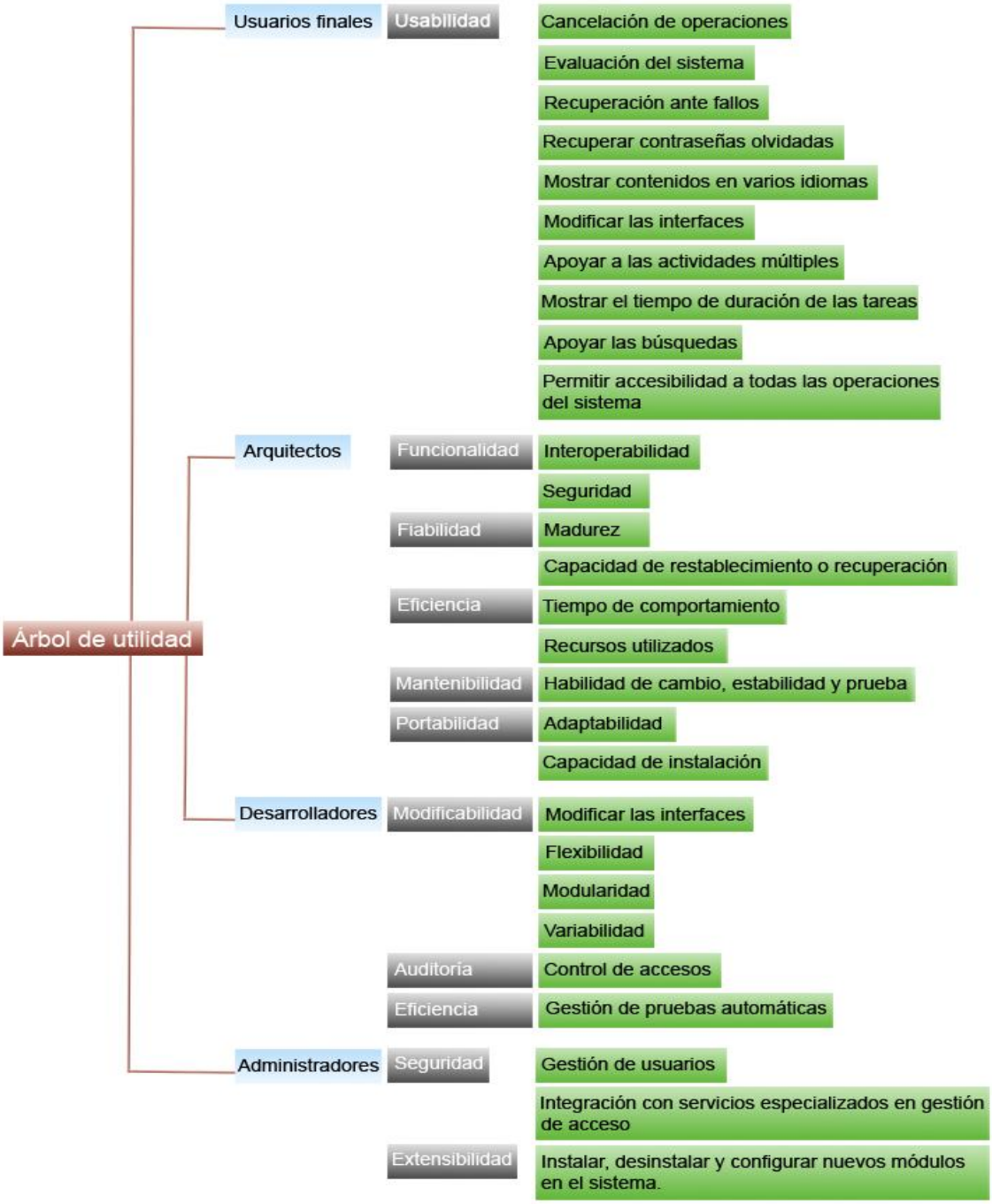
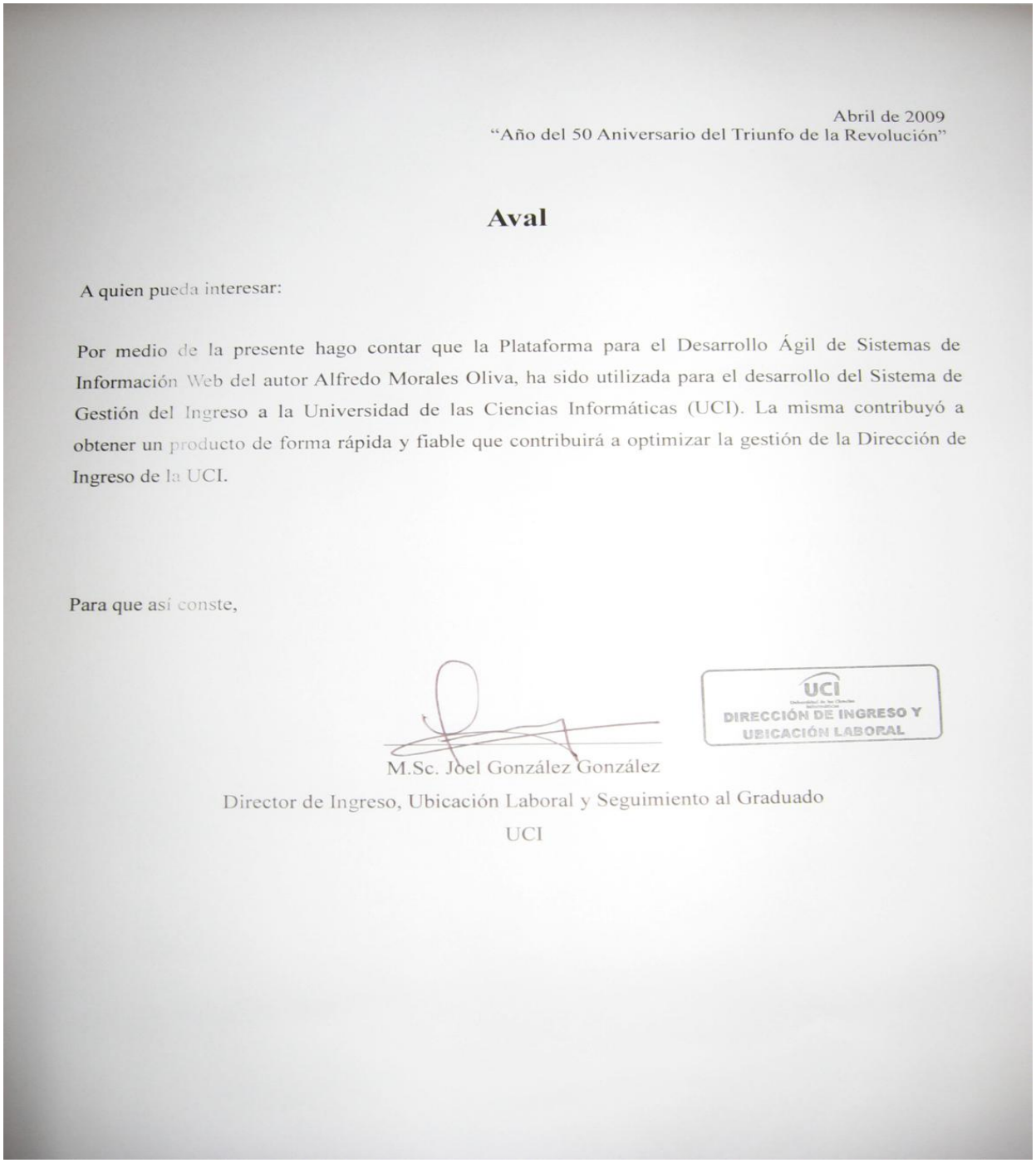


Figura 5. Árbol de utilidad con escenarios para el método SAAM

Anexo 4



Abril de 2009
"Año del 50 Aniversario del Triunfo de la Revolución"

Aval

A quien pueda interesar:

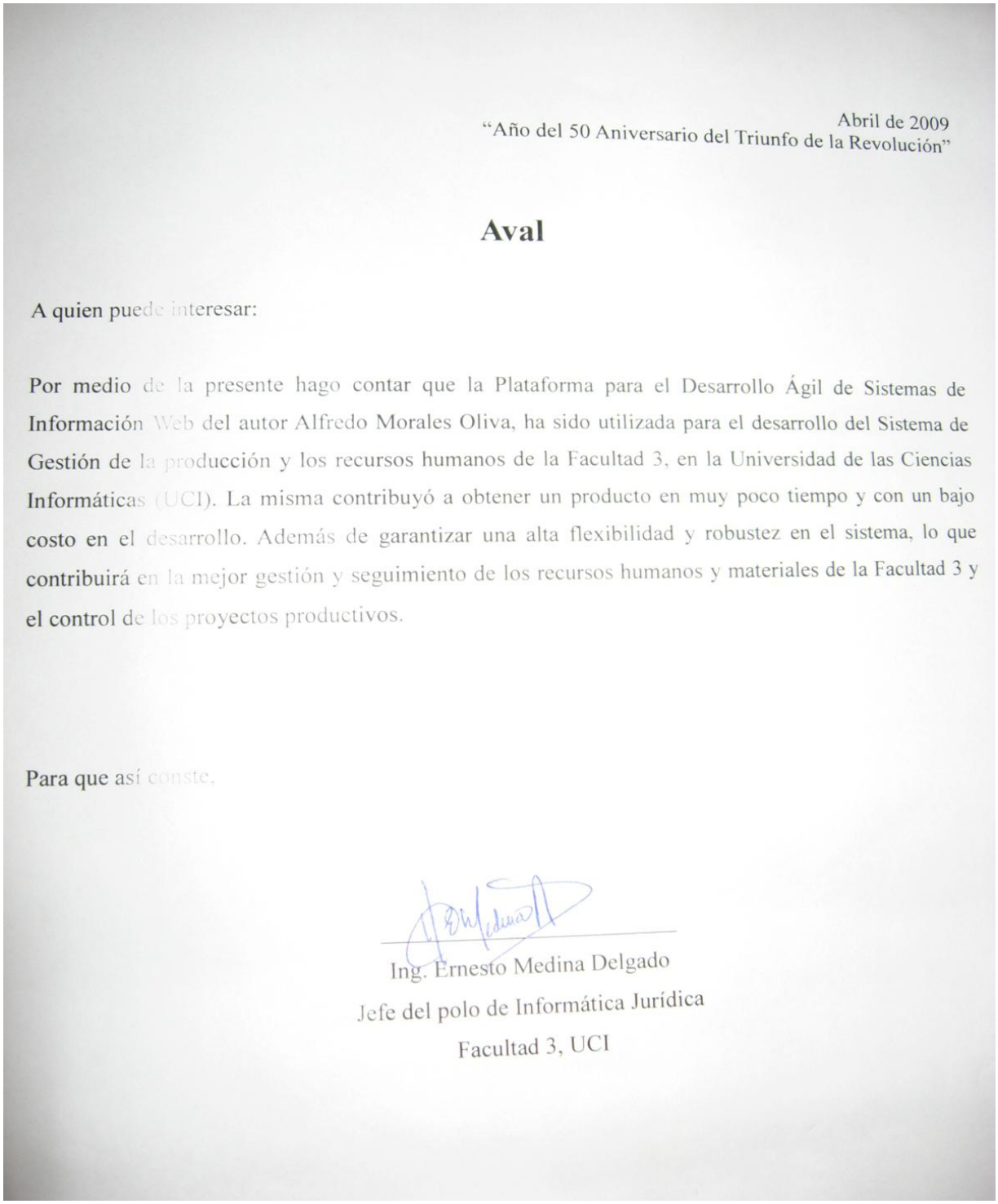
Por medio de la presente hago contar que la Plataforma para el Desarrollo Ágil de Sistemas de Información Web del autor Alfredo Morales Oliva, ha sido utilizada para el desarrollo del Sistema de Gestión del Ingreso a la Universidad de las Ciencias Informáticas (UCI). La misma contribuyó a obtener un producto de forma rápida y fiable que contribuirá a optimizar la gestión de la Dirección de Ingreso de la UCI.

Para que así conste,

UCI
DIRECCIÓN DE INGRESO Y UBICACIÓN LABORAL

M.Sc. Joel González González
Director de Ingreso, Ubicación Laboral y Seguimiento al Graduado
UCI

Figura 6. Aval del Sistema de Gestión de Ingreso a la UCI.



Abril de 2009
"Año del 50 Aniversario del Triunfo de la Revolución"

Aval

A quien puede interesar:

Por medio de la presente hago contar que la Plataforma para el Desarrollo Ágil de Sistemas de Información Web del autor Alfredo Morales Oliva, ha sido utilizada para el desarrollo del Sistema de Gestión de la producción y los recursos humanos de la Facultad 3, en la Universidad de las Ciencias Informáticas (UCI). La misma contribuyó a obtener un producto en muy poco tiempo y con un bajo costo en el desarrollo. Además de garantizar una alta flexibilidad y robustez en el sistema, lo que contribuirá en la mejor gestión y seguimiento de los recursos humanos y materiales de la Facultad 3 y el control de los proyectos productivos.

Para que así conste,

Ing. Ernesto Medina Delgado
Jefe del polo de Informática Jurídica
Facultad 3, UCI

Figura 7. Aval del Sistema de gestión de la producción y lo recursos humanos de la Facultad 3.