UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Portal de PDVSA

Trabajo de diploma para optar por el título de ingeniaría en informática.

Autor: Maykell Frómeta Flores

Tutor.Ing: William Ascuy Morales.

Ciudad de la Habana

Junio de 2005

DECLARACIÓN DE AUTORÍA

Yo: ***Maykell Frómeta Flores*** me declaro como único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) para que hagan el uso de este trabajo de la manera que estimen conveniente.

Y para que así conste firmo la presente a los ___ días del mes de _____ del 2006.

_____ _____

Firma del Autor Firma del Tutor

Opinión del usuario del Trabajo de Diploma.

El Trabajo de Diploma, titulado: **"Portal de PDVSA"**, fue realizado en la Universidad de Ciencias Informáticas. Este centro considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface:

- ☐ Totalmente
- ☐ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta Universidad los beneficios siguientes:

_____
_____
_____
_____
_____
_____
_____
_____

Y para que así conste, se firma la presente a los ____ días del mes de _____ del año _____

_____ _____

Representante de la entidad Cargo

_____ _____

Firma Cuño

Opinión del Tutor del Trabajo de Diploma

Título: "Portal de PDVSA"

Autor: Maykell Frómeta Flores

Tutor: Ing. William Azcuy Morales

El trabajo que se ha presentado forma parte del proyecto de desarrollo de un portal para PDVSA. El tema tratado por el aspirante representa un paso importante para la construcción del sistema y contribuye a lograr los objetivos de calidad que se esperan.

Durante el desarrollo de la investigación el estudiante ha demostrado un alto nivel de independencia y laboriosidad, expresado en los aspectos que ha tenido que estudiar para elaborar la propuesta de arquitectura del portal de PDVSA, de vital importancia para la obtención de la aplicación.

Las soluciones a los diferentes obstáculos que aparecieron demostraron una elevada creatividad que se reflejó en el trabajo individual arduo y continuo que demostró la gran responsabilidad asumida por el estudiante ante las tareas que le impuso el desarrollo de la tesis.

En la obtención de esta solución el autor mostró una gran receptividad ante las críticas y sugerencias efectuadas así como capacidad para apropiarse de los cocimientos que necesitaba para enfrentar las dificultades surgidas en el proceso de creación. Entre los conocimientos que ya dispone se encuentra el dominio de un conjunto de herramientas de última generación, tanto de modelación, como entornos de desarrollo de sistemas, entre los que podemos citar la metodología RUP, Rational XDE, la plataforma J2EE y la tecnología de portlets. El documento realizado cumple con los requisitos planteados para este tipo de trabajos.

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático.

_____ _____

Firma Fecha

Con infinito amor…

A mi abuela y a mi hermana,
por estar siempre a mi lado y sentir con el mismo corazón
cada momento de mi vida:
Eladia y Marelis, las quiero con toda mi alma.

Agradezco…

A mis padres que siempre han estado a mi lado en estos largos años de estudio, que me han brindado su apoyo incondicional y han depositado su confianza en mí. Ellos son el motivo por el que cada día trato de ser mejor como persona, ellos me han inculcado valores y principios que siempre llevaré presente, me han mostrado los sentimientos más puros que debe tener un hombre y me han impulsado con su sacrificio y su aliento hasta aquí.

Mamá y Papá, ¡ los Quiero !.

A mi Rosita, por apoyarme, por quererme, por crear luz a mi alrededor cuando las cosas se enturbian, por soñar junto a mí.

A mi tía Máni, por recibirme siempre con todo su amor y su alegría.
A mi tío Chango, a mi prima Vilniela que no deje de escribirme por correo, a toda mi familia que siempre me ha brindado su cariño y su apoyo.

A mis amigos Eduardito, Lester y Laidel.
A mis compañeros de aula de la UCI, que son personas maravillosas y ha sido una gran experiencia compartir con ellos.

A mis amigos Henry y Fals.

Resumen

El portal de PDVSA está concebido para que sea la nueva imagen de PDVSA, y exponga con claridad al mundo los lineamientos de la nueva empresa petrolera venezolana totalmente comprometida con su pueblo. Además tiene entre sus objetivos solucionar problemas que presentaba el antiguo portal, entre los que se encuentran problemas relacionados con la organización y actualización de la información, interactividad con el usuario y flexibilidad para adaptarse a distintas plataformas. Para la solución se decidió desarrollar una aplicación web que lograra una integración eficiente con un sistema publicador de contenidos que posibilitara mantener fácilmente actualizado el portal, lo que aquí se expone es el resultado parcial de un proyecto más abarcador.

En el presente trabajo se hace un análisis y descripción de la arquitectura de software que debe tener el portal para centrar su desarrollo y lograr responder a las expectativas del cliente. Se hace un análisis de los principales tipos de arquitectura, de las nuevas tecnologías relacionadas con el desarrollo de portales y como debe ser la arquitectura del portal para lograr aprovechar todas las ventajas de estas.

Se pretende hacer una propuesta de arquitectura que deje el proyecto listo para ser abordado en la fase de construcción, fundamentar las herramientas y estándares a utilizar y lograr utilizar el sistema de publicación de contenidos QuipusNews para actualizar el portal.

Como resultado se obtuvo una propuesta de arquitectura que responde a las necesidades planteadas.

ÍNDICE

INTRODUCCIÓN

Venezuela está llamada a ser uno de los puntales de la integración latinoamericana, por esto en este hermano país se vienen efectuando cambios muy profundos tanto en lo político lo social y lo económico; el pueblo venezolano encabezado por su mandatario Hugo Chávez Frías está dando pasos muy firmes en pos de lograr una Venezuela mejor, una nación mucho más fuerte no solo para sí misma sino para toda América Latina, es por eso que se hacen grandes esfuerzos para impulsar su economía sobre todo en su rama más poderosa, la rama petrolera, y en su empresa más importante PDVSA.

PDVSA es la empresa insigne de Venezuela y por tanto le corresponde más que a ninguna difundir la nueva imagen de la empresa bolivariana; además de revindicar la fortaleza de PDVSA que no ha mermado en nada como quieren hacer ver los opositores al proyecto bolivariano.

PDVSA cuenta en la actualidad con un sitio web de cara a Internet, pero adolece de problemas que conspiran con su posibilidad de desempeño como punto de entrada para personal que labora fuera de la Empresa incluso fuera del país, y que necesita manejar información en dependencia de su rango y actividad; además no cuenta con una forma eficiente de gestionar y organizar la personalización de la información que se publica en el portal.

Ante esta situación y debido a la creciente popularización de los portales corporativos como solución a la presencia efectiva de grandes empresas en la web, PDVSA decidió encargar el desarrollo de un portal corporativo usando un servidor de portales, debido a que estas herramientas están siendo crecientemente usadas para este propósito.

Existía un obstáculo para encarar el proyecto, y era que en nuestra universidad no se tenían referencias de esta naciente tecnología, que cuenta con pocos años de uso a nivel empresarial y cuyas herramientas de desarrollo están todavía por consolidarse y presentan continuos cambios. Por tal motivo no se contaban con elementos para tener una visión global del sistema que se necesita desarrollar.

En el presente trabajo se hace un análisis de la arquitectura de software, disciplina derivada de la ingeniería de software que se ha consolidado como elemento clave para el desarrollo de software en la actualidad. Se acerca a la arquitectura de software desde varias aristas, su papel para dar una visión global de un sistema software, su importancia en la obtención de software de calidad y que cumplan con requisitos del cliente, los diferentes tipos de arquitectura de software así como sus ventajas y desventajas. La capacidad de un sistema para adaptarse a restricciones tecnológicas así como aprovechar las ventajas de estas, lograr productos reutilizables y con buen rendimiento es el resultado de definir una buena arquitectura.

Se defiende la idea de que al definir y describir una arquitectura de software para un sistema se logra tener una visión global sobre el mismo y se logra profundizar en las tecnologías y estándares necesarios lo que permite encarar el proyecto con mayores probabilidades de éxito.

Los objetivos del trabajo consisten en define el diseño de una arquitectura base que permita tener una visión general del portal que se pretende desarrollar, hacer una descripción de la arquitectura basándose en la tecnología de portlets, donde se definan los principales componentes para desarrollar el portal de PDVSA y lograr usar QuipuisNews como sistema de publicación de contenido para Exoplatform.

Se ha distribuido el contenido lo largo de 5 capítulos, el primero aborda los principales conceptos asociados al dominio del problema, el segundo trata acerca de las tendencias y tecnologías actuales sobre las que se apoya la propuesta, el tercero describe los procesos del negocio, el cuarto es acerca de la construcción de la solución propuesta y el quinto es el estudio de factibilidad.

CAPÍTULO I

FUNDAMENTACIÓN DEL TEMA

1.1 INTRODUCCIÓN

En este capítulo se brinda una aproximación general y detallada de los temas relacionados con las aplicaciones web, un primer acercamiento al concepto de arquitectura de software, el concepto y principales características de los servidores de portales, las características de cada tecnología propuesta así como una descripción de los principales aspectos relacionados con el dominio del problema que son necesarios para entender el modelo de negocio y la propuesta de solución.

En este capítulo se muestran los problemas que fundamentan la propuesta de solución, se exponen los objetivos generales y específicos, y se describen los procesos de negocio relacionados con el objeto de estudio de este trabajo.

1.2 SOBRE LAS TRES W.

1.2.1 Historia y propósitos de la World Wide Web.

El World Wide Web data desde Marzo de 1989. En ese mes, Tim Berners-Lee del Laboratorio Europeo de Física de Partículas en Génova, (CERN) propuso desarrollar un "sistema de hipertexto" con el propósito de permitir eficiencia y compartir información fácilmente entre equipos de investigadores de la comunidad de High Energy Physics separados geográficamente.

Los tres componentes importantes del sistema propuesto eran los siguientes:

Una interfase consistente.

La habilidad de incorporar un extenso rango de tecnologías y diferentes tipos de documentos.

Un instrumento para leer los documentos en forma universal; esto es, cualquier persona en cualquier lugar y que esté conectada a la red, podrá leer el mismo documento al mismo tiempo que otra persona, y podrá hacerlo de forma fácil. [1]

Los orígenes de Internet se remontan al año 1961. Desde entonces, se han desarrollado varios lenguajes y protocolos evolucionados dentro de la estructura creciente de Internet. Es importante recordar que la Web es solo una parte de todo Internet. Mucha gente piensa que la WEB e Internet es la misma cosa, pero no están en lo cierto. La World Wide Web es un conjunto de servicios basados en hipermedios, ofrecidos en todo el mundo a través de Internet, se lo llama WWW (World Wide Web - Telaraña de Cobertura Mundial). No existe un centro que administre esta red de información, sino más bien está constituida por muchos servicios distintos que se conectan entre sí a través de referencias en los distintos documentos, por ejemplo, un documento contenido en un computador en Canadá, puede tener referencias a otro documento en Japón, o a un archivo en Inglaterra, o a una imagen en Suecia. [2]

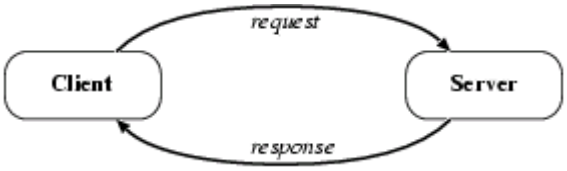
La web es el equivalente de la oficina de una empresa, donde puede poner a disposición pública productos, servicios, áreas de soporte al cliente, y en general cualquier información de la empresa. La información que las empresas y personas ponen a disposición pública a través de la web, está en un lenguaje especialmente diseñado para esto -el lenguaje HTML (HyperText Markup Language)- este lenguaje está siendo adoptado como el estándar universal para la creación y distribución de información no sólo en ambientes públicos, sino también en los entornos privados al interior de las compañías y las corporaciones. Actualmente tenemos una variedad de programas gratuitos y comerciales capaces de crear documentos en HTML de una manera muy fácil y rápida.

Hoy, la Web es algo cotidiano para una gran parte de los más de 600 millones de usuarios de Internet que hay en todo el mundo. Sus utilidades son diversas, su impacto en la economía mundial es apreciable. No sólo hay documentos de texto: hay imágenes, vídeos, música, se pueden comprar cosas, se pueden hacer reservaciones...

1.2.2 Modelo Cliente – Servidor

La arquitectura cliente-servidor llamado modelo cliente-servidor o servidor-cliente es una forma de dividir y especializar programas y equipos de cómputo a fin de que la tarea que cada uno de ellos realiza se efectúe con la mayor eficiencia, y permita simplificar las actualizaciones y mantenimiento del sistema.

El escenario de un modelo cliente/servidor ocurre cuando un proceso desea un servicio que proporciona otro proceso, así el primero le envía un mensaje solicitando ese servicio al segundo: una petición. El proceso que cumple el servicio se llama servidor y el solicitante se llama cliente.



Los procesos clientes y servidores han de seguir un protocolo de comunicaciones que define: a) Como se codifican las peticiones; y b) cómo se sincronizan entre sí los procesos.

Los clientes y servidores han de estar de acuerdo en cómo se escriben los mensajes: en qué orden van los posibles parámetros de la petición, cuántos bytes ocupan, etc.

La forma de sincronización nos dice si el cliente puede seguir adelante justo después de enviar la petición (no bloqueante), o por el contrario tiene que esperar a que el servidor le envíe una respuesta (bloqueante). Si la comunicación es no bloqueante, habrá que definir un mecanismo para que el cliente pueda saber si la respuesta del cliente está disponible. En esta práctica se adoptará una comunicación bloqueante: el cliente siempre esperará hasta recibir una respuesta del cliente.

El diálogo cliente/servidor es casi siempre bidireccional. Por un lado, el cliente envía información al servidor (el tipo de servicio solicitado más los parámetros); por otro, el servidor devuelve información al cliente (los resultados del servicio, códigos de error en caso de producirse, etc.

1.2.3 La Web y el modelo Cliente - Servidor

La Web funciona siguiendo el denominado modelo cliente-servidor, habitual en las aplicaciones que funcionan en una red. Existe un servidor web que es quien tiene las páginas web, y un cliente web, que es quien las pide.

Cliente web

El cliente web es un programa con el que el usuario interacciona para solicitar a un servidor web el envío de páginas de información. Estas páginas se transfieren mediante el protocolo HTTP.

Las páginas que se reciben son documentos de texto codificados en lenguaje HTML. El cliente web debe interpretar estos documentos para mostrárselos al usuario en el formato adecuado.

Además, cuando lo que se recibe no es un documento de texto, sino un objeto multimedia (vídeo, sonido, etc.) no reconocido por el cliente web, éste debe activar una aplicación externa capaz de gestionarlo.

Entre los clientes web (también conocidos como visualizadores o navegadores) más usuales están el Netscape Navigator y el Microsoft Internet Explorer. La mayoría de ellos soportan también otros protocolos, como el FTP (File TransferProtocol), para la transferencia de ficheros, y el SMTP (Single Mali Transfer Protocol), para el envío y la recepción de correo electrónico.

Servidor web

El servidor web es un programa que está permanentemente escuchando las peticiones de conexión de los clientes mediante el protocolo HTTP.

El servidor funciona de la siguiente manera: si encuentra en su sistema de ficheros el documento HTML solicitado por el cliente, lo envía y cierra la conexión; en caso contrario, envía un código de error que cierra la conexión. El servidor web también se ocupa de controlar los aspectos de seguridad, comprobando si el usuario tiene acceso a los documentos.

1.2.4 El modelo Cliente – Servidor y las Aplicaciones Distribuidas

Una aplicación que siga el modelo cliente servidor puede fácilmente convertirse en una aplicación distribuida solo con que el cliente y el servidor se encuentren en máquinas distintas.

En esta arquitectura la capacidad de proceso está repartida entre el servidor y los clientes. En la funcionalidad de un programa distribuido se pueden distinguir 3 capas o niveles:

Manejador de Base de Datos (Nivel de almacenamiento),

Procesador de aplicaciones o reglas del negocio (Nivel lógico) y

Interfaz del usuario (Nivel de presentación)

En una arquitectura monolítica no existe tal distribución, sin embargo en el modelo cliente-servidor, en cambio, el trabajo puede estar repartido entre uno o varios ordenadores.

En la actualidad se suele hablar de arquitectura de tres niveles, donde la capa de almacenamiento y la de aplicación se ubican en (al menos) dos servidores diferentes, conocidos como servidores de datos y servidores de aplicaciones.

1.2.5 Aplicaciones Web

Una aplicación Web es un sitio Web donde la navegación a través del sitio, y la entrada de datos por parte de un usuario, afectan el estado de la lógica del negocio. En esencia, una aplicación Web usa un sitio Web como entrada (front-end) a una aplicación típica. Si no existe lógica del negocio en el servidor, el sistema no puede ser llamado aplicación Web. [3]

Las aplicaciones Web están en todas partes, ejemplos comunes de aplicaciones Web son aquellas que nos permiten buscar como Google, colaborar en proyectos como Sourceforge, comprar artículos como en eBay, comunicarnos con otras personas a través del e-mail como Hotmail o ver las últimas noticias en CNN.com.

Las aplicaciones Cliente - Servidor han encontrado su forma mas difundida en las aplicaciones Web, donde toda la aplicación reside en un servidor. Al igual que en

las aplicaciones Cliente – Servidor distribuidas en las aplicaciones Web se distinguen tres niveles: Interfaz de usuario, lógica de negocios y datos.

El nivel de interfaz de usuario está compuesto por las páginas HTML que el usuario solicita a un servidor web y que visualiza en un cliente web (normalmente, un navegador web).

El nivel de lógica de negocio está compuesto por los módulos que implementan la lógica de la aplicación y que se ejecutan en un servidor de aplicaciones.

El nivel de datos está compuesto por los datos, normalmente gestionados por un sistema de gestión de bases de datos (servidor de datos), que maneja la aplicación web.

1.3 ARQUITECTURA DE SOFTWARE.

1.3.6 ¿Qué es la Arquitectura del Software?

Existen muchas definiciones de Arquitectura del Software y no parece que ninguna de ellas haya sido totalmente aceptada. En un sentido amplio se podría estar de acuerdo en que la Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de:

Definir los módulos principales

Definir las responsabilidades que tendrá cada uno de estos módulos

Definir la interacción que existirá entre dichos módulos:

Control y flujo de datos

Secuenciación de la información

Protocolos de interacción y comunicación

Ubicación en el hardware

La Arquitectura del Software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño. La definición oficial de Arquitectura del Software es la *IEEE Std 1471-2000* que reza así: “La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución”. [17]

El objetivo principal de la Arquitectura del Software es aportar elementos que ayuden a la toma de decisiones y, al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto. Para conseguirlo, la Arquitectura del Software construye abstracciones, materializándolas en forma de diagramas comentados. [17]

1.3.7 Representación de la arquitectura

Para representar adecuadamente la arquitectura de un sistema es necesario contar con varios diagramas o vistas. Dada la cantidad de características y de elementos que tiene un sistema de software no es posible incluirlos todos en un solo diagrama y que sirva, además, para todas las personas que participan en el desarrollo. Cada una de estas vistas es una estructura de la arquitectura del sistema, que muestran una parte del sistema como un conjunto de componentes, conectores y restricciones sobre sus tipos y relaciones. Además, cada estructura puede relacionarse con las demás para complementar la visión integral del sistema. La arquitectura, conformada por diferentes visiones del sistema, constituye un modelo de cómo está estructurado dicho sistema, sirviendo de comunicación entre las personas involucradas en el desarrollo y ayudando a realizar diversos análisis que orienten el proceso de toma de decisiones. [18]

Si seguimos al pie de la letra la definición de la IEEE sobre arquitectura de software: “La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución”, entonces sería conveniente hacer un breve acercamiento a lo que entendemos por componentes y conceptos relacionados.

1.3.8 Ingeniería de software basada en componentes

La Ingeniería del Software basada en Componentes (*Component-Based Software Engineering*, CBSE) es una disciplina reciente, que se encuadra dentro de la Ingeniería del Software, y a la que se está prestando una atención cada vez

mayor. Este interés se debe a la concurrencia de diversos factores, entre los que debemos citar la aparición y progresiva utilización de las plataformas de componentes, que facilitan el uso de la llamada tecnología de componentes y hacen posible el desarrollo de verdaderos sistemas abiertos y de aplicaciones *plug-and-play.*

Por otro lado, no es tampoco ajena a este fenómeno la continua expansión de la WWW (*World-Wide Web*) como marco global para el intercambio de información y servicios, incluidos los comerciales, con la consiguiente necesidad de aplicaciones capaces de realizar este intercambio.

Su objeto es la construcción de aplicaciones mediante el ensamblado de componentes ya existentes, en lugar de partir de cero. En este contexto, el desarrollo de sistemas de software implica la reutilización de componentes prefabricados, posiblemente desarrollados por terceras partes y no de forma específica para una aplicación concreta, sino pensando en su integración en diversos sistemas, de manera que se cree un verdadero *mercado de componentes*. Uno de los sueños que siempre ha tenido la Ingeniería del Software es el de contar con un mercado global de componentes, al igual que ocurre con otras ingenierías, o incluso con el hardware.

1.3.9 Componente

A pesar de que existen discrepancias en cuanto a una definición unívoca de la noción de componente, una cosa parece estar clara: *los componentes son unidades de composición*, y su existencia solo tiene sentido como parte de un entorno o plataforma de componentes. En efecto, si tomamos como ejemplo un equipo musical o un mueble modular, los elementos que forman estos sistemas son componentes, debido exclusivamente a que han sido diseñados para ser compuestos unos con otros, permitiendo múltiples combinaciones y resultados. Lo mismo ocurre con el software. Considérense, por ejemplo, un componente *botón* de una interfaz gráfica de usuario. Su valor reside no tanto en la funcionalidad que proporciona como en el hecho de que ha sido diseñado para colaborar con otros

componentes de forma consistente para construir interfaces de usuario complejas. [19]

Disponer de componentes no es suficiente, a menos que seamos capaces de ensamblarlos y reutilizarlos. Y reutilizar un componente no significa usarlo más de una vez, sino que implica la capacidad del componente para ser utilizado en contextos distintos a aquellos para los que fue diseñado. El uso de componentes de software y de los mecanismos de composición adecuados proporcionan los medios para la reutilización sistemática del software. Además, la composición de componentes de software reutilizables y configurables nos permite abordar el problema de la evolución de los requisitos mediante la sustitución y reconfiguración solo de las partes del sistema afectadas.

Los componentes no pueden existir fuera de un entorno de composición definido, sino como parte de un marco de trabajo composicional que brinde: una biblioteca de componentes, una arquitectura de software reutilizable en la que encajen estos componentes y un mecanismo de conexión que permita su composición. [19]

Hacia los marcos de trabajo es hacia donde se esta dirigiendo gran parte del desarrollo de software actual, más específicamente hacia los marcos de trabajo para el desarrollo de aplicaciones web como portales. Basándonos en lo antes mencionado podemos identificar los servidores de portales como marcos de trabajo ideales para el desarrollo de aplicaciones web de tipo portales, y los portles como los componentes que se gestionan dentro del entorno de composición de los servidores de portales. Muchos autores ya nombran este tipo de ingeniería de componentes como “arquitectura de portlets” y la identifican como un estilo al mismo nivel de otros estilos arquitectónicos como el MVC (Modelo Vista Controlador) o el de Capas. Afirmaciones mas que fundadas si nos atenemos al concepto de patrón de arquitectura.

1.3.10 Patrones o Estilos de arquitectura.

Los patrones de arquitectura expresan una organización estructural para un sistema software. Proveen un conjunto de subsistemas predefinidos e incluyen reglas y lineamientos para conectarlos. [20]

1.3.11 Patrón de arquitectura. Modelo Vista Controlador

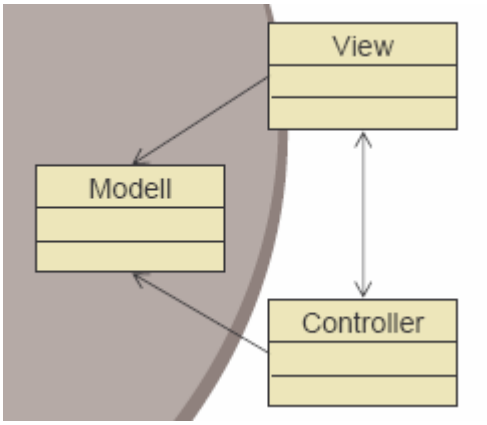
El **Model-View-Controller** (Modelo-Vista-Controlador, en adelante MVC) fue introducido inicialmente en la comunidad de desarrolladores de Smalltalk-80. MVC divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

Modelo (Model): Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.

Vista (View): Muestra la información al usuario. Obtiene los datos del modelo. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.

Controlador (Controller): Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio ("**service requests**" en el texto original) para el modelo o la vista. El usuario interactúa con el sistema a través de los controladores.

Las Vistas y los Controladores conforman la interfaz de usuario. Un mecanismo de propagación de cambios asegura la consistencia entre la interfaz y el modelo. La separación del modelo de los componentes vista y del controlador permite tener múltiples vistas del mismo modelo. Si el usuario cambia el modelo a través del controlador de una vista, todas las otras vistas dependientes deben reflejar los cambios. Por lo tanto, el modelo notifica a todas las vistas siempre que sus datos cambien. Las vistas, en cambio, recuperan los nuevos datos del modelo y actualizan la información que muestran al usuario.



Este patrón es muy popular y ha sido portado a una gran cantidad de entornos y frameworks entre los que se encuentran WinForms, ASP .Net, J2EE. Las herramientas de programación visual como Visual Basic, Visual Studio .Net, etc., siguen también alguna variante de este esquema. El MVC es un patrón ampliamente utilizado en múltiples plataformas y lenguajes.

Algunos de sus principales beneficios son:

Menor acoplamiento.

Desacopla las vistas de los modelos.

Desacopla los modelos de la forma en que se muestran e ingresan los datos.

Mayor cohesión.

Cada elemento del patrón esta altamente especializado en su tarea (la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio).

Las vistas proveen mayor flexibilidad y agilidad.

Se puede crear múltiples vistas de un modelo.

Se puede crear, añadir, modificar y eliminar nuevas vistas dinámicamente.

Las vistas pueden anidarse.

Se puede cambiar el modo en que una vista responde al usuario sin cambiar su representación visual.

Se pueden sincronizar las vistas.

Las vistas pueden concentrarse en diferentes aspectos del modelo.

Mayor facilidad para el desarrollo de clientes ricos en múltiples dispositivos y canales.

Una vista para cada dispositivo que puede variar según sus capacidades.

Una vista para la Web y otra para aplicaciones de escritorio.

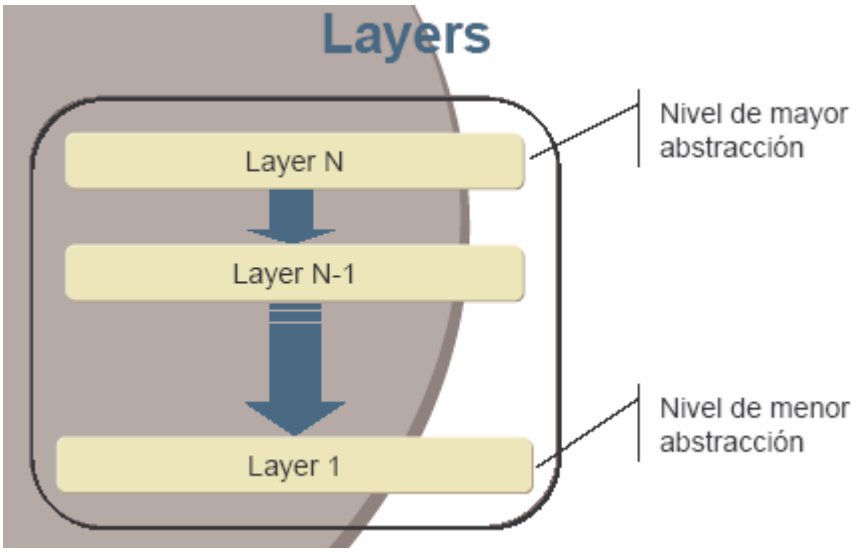
Más claridad de diseño.

Facilita el mantenimiento.

Mayor escalabilidad. [21]

1.3.12 Patrón Layers

Descompone la aplicación en un conjunto de capas independientes y ordenadas jerárquicamente. Cada nivel o capa utiliza los servicios de la capa inmediatamente inferior y ofrece servicios a la capa inmediatamente superior. [20]



Algunas de sus ventajas

Reutilización de un mismo nivel en varias aplicaciones.

Permite la estandarización.

El cambio de un nivel no afecta al resto.

Desventajas

Trabajo innecesario de paso de argumentos.

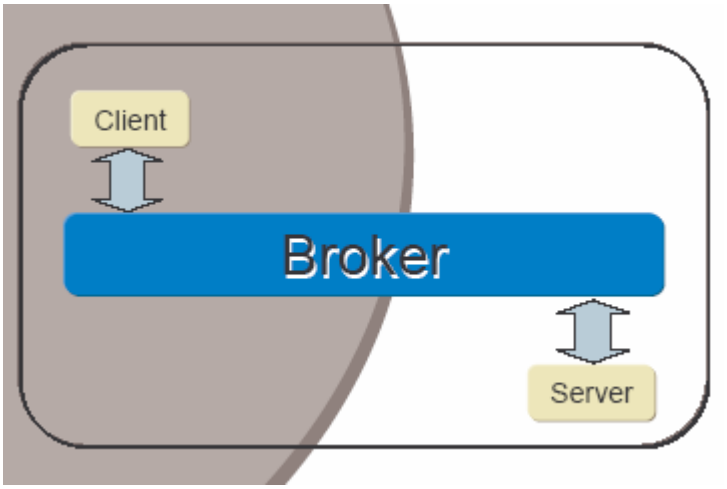
Si el número de niveles es excesivo puede ser muy ineficiente.

Si hay pocos niveles tenemos el diseño poco organizado.

Un mal diseño o un cambio importante de funcionalidad pueden forzar cambios que se transmitan en cascada de un nivel a otro.

1.3.13 Patrón Broker

Este patrón de arquitectura se usa para organizar sistemas distribuidos con componentes débilmente acoplados que interactúan entre sí invocando servicios remotos.



Ventajas

Transparencia de ubicación.

Facilidad para modificar un componente sin afectar al resto.

Portabilidad

Reusabilidad

Desventajas

Eficiencia restringida

Menor tolerancia frente a fallos: si falla el broker todo el sistema se viene abajo.

Prueba y depuración costosas, pues existen muchos componentes en el sistema.

1.3.14 Rol del arquitecto

El arquitecto obtiene información sobre el problema y diseña una solución que satisfaga los requisitos funcionales y no funcionales manteniendo flexibilidad cuando los requisitos cambien.

Se apoya en patrones modelos y frameworks, participa activamente en todas las fases del desarrollo y establece lineamientos generales que deben ser tenidos en cuenta para el desarrollo de nuevos proyectos.

1.3.15 Importancia de la arquitectura de software

Los equipos de desarrollo que diseñen una buena arquitectura podrán producir no solo productos de mayor calidad, sino a un menor coste y en menos tiempo. Esto se debe a que los riesgos arquitectónicos del proyecto son menores y están mucho mas controlados, y que al poder integrar una visión orientada a componentes, las posibilidades de reutilizar software ya desarrollado son mucho mayores, con las ventajas que ello implica.

1.4 PORTALES

1.4.16 ¿Que es un portal?

La manera en la que instituciones y empresas desarrollaban su presencia en Internet ha sufrido una evolución en los últimos años. Al principio de la popularización de Internet se tendía a aparecer en la red de alguna manera, lo que hizo surgir infinidad de pequeñas páginas corporativas en las que se ofrecía una información básica sobre la empresa o institución en cuestión, los datos fundamentales de contacto, alguna información general y, en pocos casos se aportaba información más profunda.

Posteriormente este modelo de proyecto web dejó de resultar efectivo para sus responsables. La simple presencia en Internet se tornó insuficiente, y algunos de estos sitios fueron incorporando algunos servicios de valor añadido y mejorando los contenidos, mientras que otros conservaron ese primer modelo de presencia mínima. Para los primeros, ahora no basta con "estar" sino que es preciso "hacer" (1). Estos ya no plantean estrategias pasivas hacia la clientela, sino que emplean

métodos más agresivos, más activos, en dura competencia por la captación de usuarios y, sobre todo, por la fidelización de éstos respecto a su producto o institución.

Se puede ofrecer una primera definición básica diciendo que un portal es una aplicación web que constituye un único punto de entrada a información personalizada proveniente de diferentes fuentes. [4]

1.4.17 Evolución de los portales

En 1994 surge Yahoo!, con el objetivo de indexar las páginas web existentes. Un equipo humano recopilaba información y la clasificaba dentro de su índice de categorías, cubriendo alrededor de 200.000 páginas (aproximadamente el 20 % del 1.000.000 páginas existentes por entonces), ofreciendo la posibilidad de recuperar información a través de su índice de categorías y, accesoriamente, mediante un motor de búsqueda. La puesta en marcha de este sistema propició el auge de los directorios, motores de búsqueda y metabuscadores, que hemos estudiado en el capítulo anterior. Esta evolución se vio altamente favorecida por el desarrollo de las posibilidades técnicas y las necesidades de los usuarios con lo que estos lugares fueron incorporando algunos de los servicios de valor añadido que ahora es habitual encontrar en los portales (mensajes sms, e-mail, espacio web...).

Otra vía, por la que se evolucionó hasta lo que ahora conocemos como portales, fue a través de las páginas web de los grandes proveedores de servicios Internet, como AOL (http://www.aol.com) o la páginas principales de Microsoft (http://www.microsoft.com) o de Netscape (http://www.netscape.com), que estaban configuradas por defecto como páginas de inicio en sus navegadores. Muchos usuarios iniciaban a diario la navegación desde esta página al no haber modificado la configuración inicial del navegador, con lo que estas páginas recibían gran cantidad de visitas cada día, lo que rentabilizaba la publicidad albergada en ella (que habitualmente se paga por número de clics recibidos). Estos sitios implantaron contenidos atractivos y servicios de valor añadido, con los

que se trataba de potenciar el tráfico recibido y, como objetivo último, fidelizar al usuario.

1.4.18 Portales Corporativos

Un portal corporativo es una intranet que provee de información de la empresa a los empleados, así como de acceso a una selección de sitios web públicos y sitios web de mercado vertical (proveedores, vendedores, etc.) Incluye un motor de búsqueda para documentos internos y la posibilidad de personalizar el portal para diferentes grupos de usuarios y particulares. Sería el equivalente interno a los portales de carácter general. Los portales corporativos tienden a ser una prolongación natural de las intranet corporativas, en las que se ha cuidado la organización de la información y la navegación, donde se permite, y sobre todo, se potencia el acceso a información de la propia institución, la edición de material de trabajo propio, el contacto con clientes y proveedores, etc. En ellos se distingue la parte intramuros, o del cortafuegos hacia adentro, y la parte extramuros o externa, dependiendo de que el destinatario de esa información sea miembro de la institución o bien un elemento externo a ésta. [4]

1.5 SERVIDORES DE PORTALES

1.5.19 Portlets

A medida que las empresas han ido haciéndose presentes en el creciente mercado de Internet, la demanda de estas hacia los desarrolladores de portales ha ido creciendo, y como en esta área de negocios los avances son vertiginosos dado que el número de consumidores que gestionan sus necesidades a través de Internet y el monto de las operaciones entre empresas canalizadas por la web aumenta rápidamente, los tiempos de desarrollo han tenido que abreviarse, así los servidores de portales no han surgido como un trabajo de algún programador aislado sino como una respuesta a las exigencias de las empresas que en los últimos años han comenzado el asalto a Internet.

Sin embargo es conveniente para una total comprensión de que se entiende por Servidores de Portales, ver que se entiende por Portlets. Según la Java Specification Request 168 y la WSRP (Web Services for Remote Portals), que tratan de definir los estándares para el desarrollo de portlets y su interoperabilidad, son componentes web basados en Java, y gestionados por un contenedor de portlets que procesa peticiones y genera contenido dinámico. Los portales usan portlets como componentes de interfaz de usuario que proveen de una capa de presentación a los sistemas de información. El portlet permite la personalización, la presentación, y la gestión de la seguridad. Los portlets además pueden ser vistos como aplicaciones web que funcionan como componentes de aplicaciones web. Un portlet puede encapsular toda una lógica de negocio, una capa de presentación compuesta por varias páginas de interfaz de usuario y además manejar su seguridad, su persistencia, exactamente igual que una aplicación web. Por esta característica es que se han popularizado tanto, ya los principales servidores de portales del mercado vienen con un conjunto de portlets por defecto listos para ser usados, los más comunes son, calendarios, banners, encuestas, foros, chats, menús.

1.5.20 Contenedor de Portlets

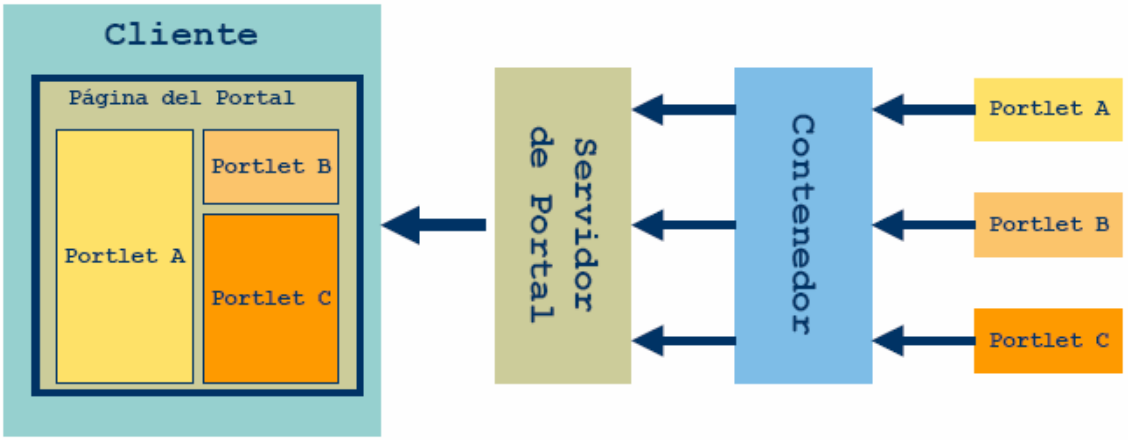
Un contenedor de portlets básicamente es un programa que corre en el servidor de portales y que se encarga de cuatro tareas fundamentales:

Ofrece un entorno de ejecución para los portlets.

Maneja su ciclo de vida.

Facilita un mecanismo de persistencia.

Proporciona servicios comunes a todos los portlets.



1.5.21 Sobre Servidores de Portales

Se puede decir que un servidor de portales es un servidor de aplicaciones corriendo una aplicación tipo portal, un servidor de portales, lo que hace es crear un entorno que sirve la información al usuario proveniente de varias fuentes, y puede además del contenido, integrar el acceso a aplicaciones de trabajo, herramientas de colaboración, contenido proveniente de fuentes externas. EL servidor de portales sirve para crear entornos de trabajo, acceso a información que además puede estar personalizada para usuarios y grupos de usuarios.

En los últimos años han surgido un gran número de Servidores de portales. Ejemplos: BEA WebLogic Portal, IBM Webphere Portal, Sun One Portal Server, Oracle Application Portal Server, SAP Portal, Vignette Application Portal, Microsoft SharePoint Portal Server, Red Hat Enterprise Portal, Jakarta Jeetspeed, eXo Platform. Suelen proporcionar un portal pre-construido en el que es posible instalar portlets.

Las primeras versiones de los servidores de portales presentaban dos problemas:

Problema 1: sólo permiten desarrollar portlets locales

Problema 2: los portlets desarrollados con un determinado portal no se pueden instalar en otro portal.

Esto traía aparejado varias consecuencias no deseables, cualquier portal que integre los servicios de la BBC por ejemplo debe re-implementar la interfaz gráfica

de los portlets que esta utiliza, por otro lado, si se desea construir otro portal que use portlets previamente desarrollados es preciso volver a instalarlos en el nuevo portal (si está construido con el mismo servidor de portales) o en el peor de los casos desarrollarlos completamente. [6]

1.5.22 WSRP

Para solucionar esto se desarrolló en septiembre de 2003 por la OASIS (Organization of the Advancement of Structured Information Standards) la primera versión del estándar WSRP (Web Services for Remote Portlets), este estándar especifica el conjunto de interfaces que debe desarrollar un productor de portlets, productor y consumidor pueden usar diferentes tecnologías. [5]

1.5.23 JSR 168

Sin embargo el WSRP no resolvía el segundo problema, la necesidad de que portlets desarrollados en un servidor de portales puedan ser instalados en otro diferente, puesto que el WSRP define un API para exportar portlets de un productor a consumidores remotos no para desarrollar portlets locales. Para resolver este problema se publicó en Octubre del 2003 la primera versión del JSR 168, un API estándar para desarrollar portlets en java con compatibilidad con WSRP.

1.5.24 Ventajas de los Servidores de Portales

Integración de aplicaciones heredadas: La mayoría de las empresas cuentan con viejas aplicaciones las cuales son imprescindibles dentro de la organización las cuales han sido desarrolladas con diferentes tecnologías (J2EE, .NET) y generalmente totalmente independientes unas de otras, los servidores de portales brindan la posibilidad de integrar estas aplicaciones en el portal.

Único punto de entrada: Una vez que un usuario se autentifica en el portal, no tiene que volver a hacerlo para acceder a ninguna de las aplicaciones heredadas integradas en el portal.

Personalización: Los usuarios pueden personalizar el contenido disponible en el portal así como cambiar el look and feel de las páginas que él ve.
Seguridad basada en Roles: Las aplicaciones y contenido disponibles en el portal pueden ser accedidos de acuerdo al rol y cargo del usuario en la organización.

Facilita Herramientas de Colaboración: La mayoría de los servidores de portales provee una seria de herramientas de colaboración como salas de Chat, Foros de discusión y sistemas de mensajería ya desplegados, lo que supone una gran ayuda a los desarrolladores puesto que no tienen que programarlo, solo configurarlo.

1.6 OBJETO DE ESTUDIO

El concepto de arquitectura se usa de forma amplia y en campos muy distintos, por lo que su significado es algo difuso. En el campo del software, la arquitectura nos identifica los elementos más importantes de un sistema así como sus relaciones. Es decir nos da una visión global del sistema.
¿Por qué es esto importante? Porque necesitamos arquitectura para entender el sistema, organizar su desarrollo, plantear la reutilización del software y hacerlo evolucionar.
En los últimos años se ha evidenciado que para desarrollar sistemas software eficientes y que cumplan con las necesidades de los usuarios es necesario diseñar una arquitectura de software adecuada.
PDVSA, dadas las características de su actividad empresarial y de negocios, por su valor estratégico para el desempeño de la economía nacional y por su ubicación como una de las principales empresas del mundo en el importantísimo sector petrolero, se propone gestionar de forma más eficiente su portal en Internet, que tantos beneficios puede llegar a aportar en términos de imagen, y para el logro de acciones de negociación efectivas, que contribuyan al desarrollo corporativo y de la nación venezolana.

Las arquitecturas software no responden únicamente a requisitos estructurales, sino que están relacionadas con aspectos de rendimiento, usabilidad, reutilización, restricciones económicas y tecnológicas, e incluso cuestiones estéticas, por lo que definiendo una buena arquitectura de software se estará asegurando en fases muy tempranas del proyecto que muchos de los objetivos que se plantea PDVSA con su portal se cumplan.

1.6.25 Descripción del proceso de negocio actual

La corporación Petróleos de Venezuela, PDVSA, tiene actualmente presencia en la web a través de su sitio www.pdvsa.com, desde el cual brinda información relacionada con su actividad empresarial, su proyección social y su estructura organizativa.

Sin embargo, este sitio no cubre las expectativas comunicacionales e informativas que demandan los públicos tanto nacionales como extranjeros, externos como internos, ni con las posibilidades que las Gerencias de AAPP de la corporación y de las filiales están en condiciones de satisfacer.

1.6.26 Situación Problémica

Más allá de los valores positivos que deben reconocérseles al sitio actual, presenta una serie de características que conspiran con su posibilidad de desempeño como punto de entrada para personal que labora fuera de la empresa incluso fuera del país, y que necesita manejar información en dependencia de su rango y actividad; por lo que PDVSA decidió el uso de un sistema servidor de portales en este caso Exoplatform para montar su portal y explotar las ventajas que los portales corporativos ofrecen.

1.6.27 Problema

El problema está dado en que no existe experiencia en el desarrollo de aplicaciones usando la tecnología de portlets y no se conocen las características del servidor de portales Exoplatform; por lo que no se tienen elementos para

plantearse una visión general del sistema que se quiere desarrollar usando esta tecnología. Por otro lado Exoplatform no cuenta con un sistema de publicación de contenidos, elemento imprescindible cuando hablamos de portales empresariales cuyo contenido cambia constantemente en el tiempo y se necesita de un mecanismo que facilite la tarea de mantener actualizado el portal.

1.7 PROPUESTA DE SOLUCIÓN

Por cuanto se necesitan elementos que permitan tener una visión general del sistema al equipo de desarrollo para acometer el proyecto utilizando una tecnología novedosa y poco conocida como es la tecnología de portlets y para permitir diseñar la estructura del sistema, se propone definir una arquitectura que deje el proyecto listo para ser acometido a partir de la fase de construcción. Esta propuesta esta basada en la investigación sobre arquitecturas de software, la tecnología de portlets y los servidores de portales principalmente Exoplatform.

Por cuanto nuestra universidad cuenta con sistema de publicación de contenidos llamado QuipusNews y Exoplatform carece de una herramienta similar siendo esta es imprescindible para desarrollar un portal corporativo eficiente, se propone un mecanismo para permitir la comunicación del sistema de publicaciones QuipusNews con el servidor de portales Exoplatform y la implementación de este.

1.8 FUNDAMENTACIÓN DE LOS OBJETIVOS DEL TRABAJO

Para llevar a cabo la solución propuesta se plantean un conjunto de objetivos.

1.8.28 Objetivo General

Como objetivo general se define el diseño de una arquitectura base que permita tener una visión general del portal que se pretende desarrollar.

1.8.29 Objetivos Específicos

Los objetivos específicos de la propuesta son:

Hacer una descripción de la arquitectura basándose en la tecnología de portlets, donde se definan los principales componentes para desarrollar el portal de PDVSA.

Lograr usar QuipuisNews como sistema de publicación de contenido para Exoplatform.

1.8.30 Tareas a cumplir

Investigar acerca de los principales conceptos acerca de arquitectura de software.

Estudiar la documentación acerca de la tecnología de portlets y el servidor de portales Exoplatform.

Dominar la herramienta de modelado Rational XDE y su integración con Eclipse.

Estudiar la documentación sobre el sistema de publicaciones QuipusNews.

Implementar un mecanismo para usar QuipusNews como sistema de publicación de contenidos de Exoplatform.

1.9 CONCLUSIONES

En este capítulo se realizó una caracterización de las tecnologías actuales, así como un análisis de las mejores prácticas en la actualidad que se acomodan al objeto de estudio y la solución propuesta. Se definieron además los objetivos generales y específicos, según un análisis previo de la situación problémica y el problema originado en cuestión.

CAPÍTULO II

TENDENCIAS Y TECNOLOGÍAS ACTUALES A CONSIDERAR

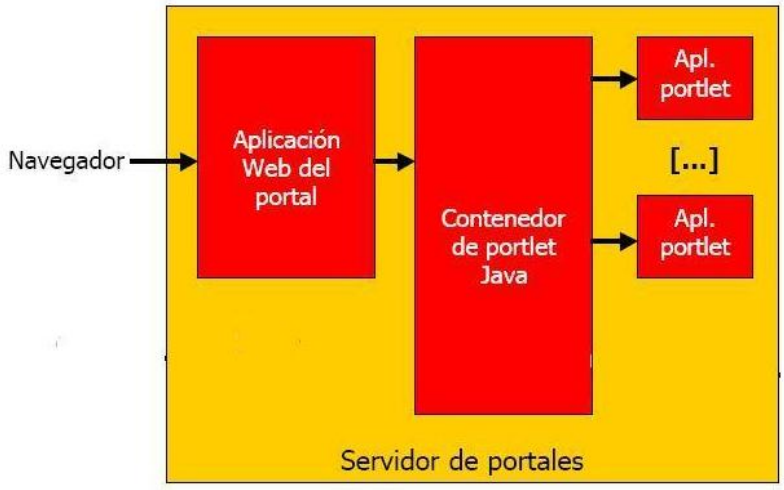
2.1 INTRODUCCIÓN

En el presente capítulo, se hace un análisis de las principales tendencias de la arquitectura de software así como los patrones de arquitectura de software y del estado actual de las tecnologías que pudieran ser adecuadas para la construcción del sistema que se pretende desarrollar.

2.2 ARQUITECTURAS DE SOFTWARE

Con la aparición de los servidores de portales ha ido emergiendo la llamada arquitectura de portlets. Esta arquitectura no se puede enmarcar completamente dentro de los patrones de arquitectura analizados anteriormente, como el MVC, el patrón Layers o el patrón Broker.

La arquitectura típica de un portal desarrollado con la tecnología de portlets es como se ilustra en la figura que sigue.



Básicamente estará presente la aplicación web del portal que se encargará de labores como la personalización, la autenticación, el registro de nuevos usuarios.

Un contenedor de portlets que controla los accesos, tiempo de vida e integración de un portlet simple. Proporciona el contenido retornado desde un portlet al portal

para agruparlo con el contenido de otros portlets. Los contenedores de portlets son proporcionados por el fabricante y no existe ninguna forma en la que los desarrolladores puedan interactuar con el.

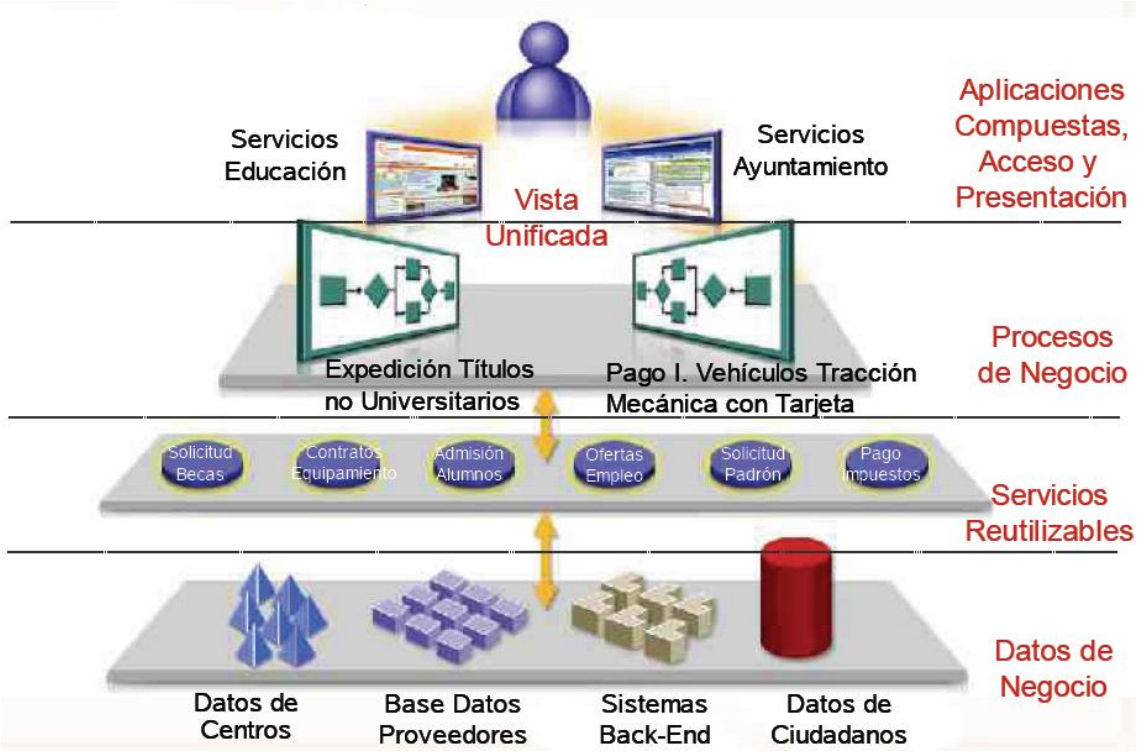
Los portlets, que proveen contenido a su contenedor de portal invocante, para efectos de despliegue de la información en una página de portal.

En una aplicación de estas características los desarrolladores deben tomar las decisiones basados en las características de los portlets pensando en ellos como los componentes principales de su arquitectura.

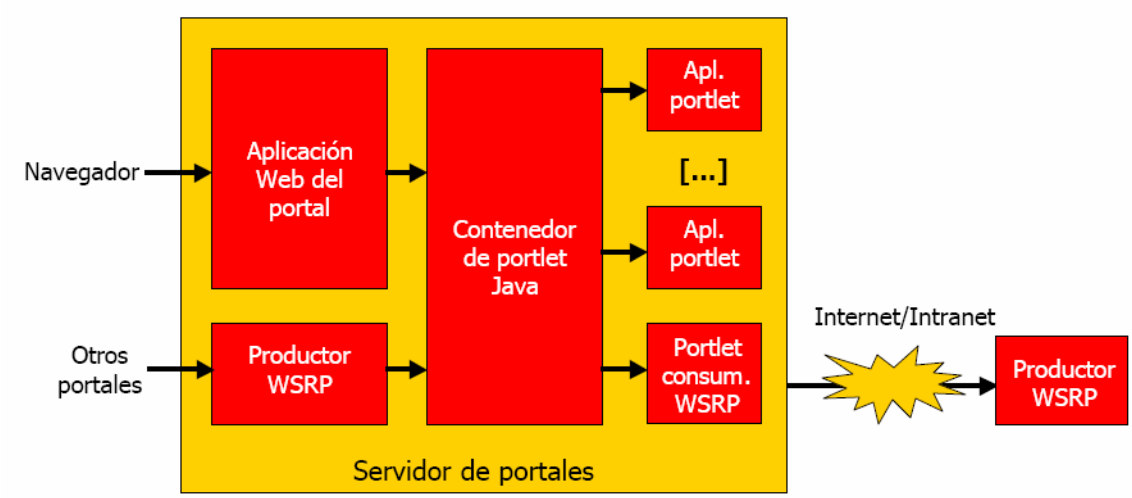
Así podemos elegir entre 2 aproximaciones fundamentales a la hora de diseñar portales con la tecnología de portlets. La primera una arquitectura de portlets orientada a servicios y la segunda una arquitectura de portlets funcional.

2.2.1 Arquitectura de portlets orientada a servicios (SOA)

Esta aproximación se basa en el estándar WSRP mediante el cual los portlets pueden consumir contenido exportado por otros y exponer a su vez contenido para ser consumido. La Orientación a Servicios es una forma de diseñar sistemas de software, una arquitectura orientada a servicios (SOA por sus siglas en Inglés) es un estilo de arquitectura orientado a maximizar la interoperabilidad, compartición y colaboración de aplicaciones en un entorno distribuido. Se basa en componentes modulares de software a los cuales se accede y se utiliza de forma estandarizada y que son independientes de cualquier plataforma o tecnologías.



Una arquitectura SOA contará con 4 capas, una de presentación, otra de procesos del negocio, otra de servicios y la capa de datos. Esta perspectiva se aplicará principalmente a soluciones de un alcance amplio, por ejemplo a empresas grandes con muchas dependencias distribuidas a distancias grandes o que sus dependencias tengan sus propios sistemas de gestión y bases de datos, principalmente constituye una referencia a entornos de intranet.



Hay dos tendencias en la actualidad en cuanto a la capa de servicios una consiste en agrupar todos los portlets en un servidor que haga función de UDDI (Universal description, Discovery, and Integration)) Por otra parte, los Portlets pueden ser muy útiles a la hora de crear Portales Federados, como IBM describe en 'Guide to Webphere Portal'. Si una corporación tiene diferentes portales, es posible publicar alguno de los portlets como web services, que a su vez serán usados por otros portales. Podemos decir, por tanto, que un Portal Federado es un portal, donde "los portlets puedes estar ejecutándose en cualquier localización en la red de portales". [10]

2.2.2 Arquitectura de portlets funcional.

El elemento sobre el que se basa esta aproximación es el estándar JSR168. Gracias a este estándar un portlet hecho para un servidor de portales puede ser desplegado en otro diferente y sin que halla que cambiar nada en su código. Aprovechando esto los arquitectos pueden pensar y diseñar sus portales bajo la óptica de encapsular en los portlet las funcionalidades básicas del portal que generarán contenido al usuario. Estos portlets por lo general serán diseñados para ejecutar tareas de generación de contenido poco complejas, por ejemplo un portlet que se encargue de mostrar los resultados deportivos, otro para manejar

mensajería, otro que maneje un foro. No debe haber un portlet que a la vez que muestre los resultados deportivos contenga un foro de discusión sobre temas variados, puesto que esto conspiraría contra la reusabilidad del portlet como componente en soluciones futuras, debemos tratar que nuestros portlets sean concretos y reusables, es mucho más probable que en el futuro tengamos que desarrollar un portal que necesite de un portlet que muestre los resultados deportivos que de un portlet que a la vez muestre los resultados deportivos y contenga el foro del portal.

Esta aproximación resulta relativamente sencilla e intuitiva, debe ser la variante ante portales no muy complejos donde las funcionalidades de generación de contenido al usuario sean típicas.

Debido a las limitaciones del JSR 168 los portlets deberán ser componentes con un nivel de interacción mínima entre ellos. Por ejemplo no deberíamos diseñar un portlet que se encargue de matricular a un nuevo estudiante en la universidad y cuando termine este proceso entonces llame a otro portlet para que se encargue de darle de alta en el registro de militantes de la UJC. Esto no se debería hacer (aunque existen mecanismos que lo permiten) puesto que la especificación de portlets actual no proporciona un mecanismo estándar de interacción entre portlets y se corre el riesgo que un mecanismo forzado no funcione en otros servidores de portales.

Por otro lado los portlets que se diseñen bajo esta perspectiva deben utilizar solamente funciones y servicios del API que define el JSR 168 y no funciones o servicios pertenecientes al servidor de portal para el cual fueron concebidos originalmente, una vez más se piensa en la usabilidad.

2.3 SERVIDORES DE PORTALES

2.3.3 Principales prestaciones de un Servidor de Portales

Los portales y la tecnología que permite que se puedan construir y administrar están ganando popularidad como una forma de darle a los usuarios acceso rápido

a la información que necesitan, y cuando la necesitan, a través de un visor web. Es un punto de partida intuitivo para acceder al contenido y las aplicaciones de muchas fuentes diversas, internas y externas, que por lo general requerían interfaces múltiples.

Las empresas en todas las industrias están utilizando portales para mejorar las comunicaciones entre negocios y entre la empresa y sus clientes. También ven a los portales como una herramienta clave en la entrega a tiempo de contenido que es relevante para sus empleados en la toma de decisiones. Un portal también puede aumentar la productividad y facilitar el compartir del conocimiento, además de proveer una vista única y consistente de la empresa hacia los clientes, proveedores, inversionistas, socios y visitantes. Estos beneficios son los que están impulsando la popularidad de los servidores de portales, adicionados por el alivio que traen a las empresas de mucho tiempo y esfuerzo que se requeriría para construir un portal desde cero.

En términos generales, los portales les permiten a las empresas combinar información dinámica en tiempo real con servicios y aplicaciones en un formato que está personalizado para cada usuario. Un buen servidor de portales debe proveer una funcionalidad clave, funcionalidad que se analiza a continuación.

Interfaces intuitivas y personalizables. Los portales deben proveer una interfaz fácil de manejar que puedan ser diseñadas simulando los diseños de las aplicaciones actuales. Detrás de bambalinas, estas interfaces deben también operar para el administrador del sistema como las puertas de seguridad y control de acceso a la información y a servicios como los de búsqueda de información. Las interfaces de portales pueden variar ampliamente en la cantidad de personalización y la flexibilidad que permiten.

Presentación de contenido personalizado. Un portal deberá poderse apalancar en la información de cada usuario almacenada en su perfil para entregar contenido personalizado. Cada usuario puede tener una vista del negocio que está "hecha a

la medida" para sus intereses o sus requerimientos, o su nivel de privilegios. Los empleados podrían acceder la información de los empleados o grupos de trabajo mientras que los clientes, socios y proveedores solo pueden ver la información de inventarios, órdenes de compra e información de proyectos, por ejemplo.

Seguridad. Muchos de los proveedores de portales dicen que su sistema provee ambientes más seguros y que controlan el acceso a los sistemas empresariales y al software integrando todo en un solo punto de ingreso. Sin embargo, así como el nivel de integración varía de producto a producto, también varía el nivel de seguridad. La mayoría de proveedores puede integrarse con directorios LDAP (Lightweight Directory Access Protocol), pero pocos se integran con los productos que administran políticas, dejando que sea el administrador del sistema que defina los papeles que juega cada usuario y los niveles de acceso permitidos. Mientras que un solo juego de usuario-clave es conveniente para el usuario, la falta de una política administrativa centralizada para el manejo de la seguridad, podría dejar al descubierto algunas aplicaciones de la empresa. Se debe establecer si el portal corporativo que se implementará también aumentará la seguridad para los sistemas críticos y los datos o si solamente simplifica la administración de la seguridad.

Integración de aplicaciones. Muchos de los servidores de portales anuncian su producto como uno que puede apalancarse en la tecnología actual del negocio e integrarla al portal, proporcionando una vista única del negocio y de aplicaciones heredadas. Sin embargo algunos portales presentan solo alguna funcionalidad básica de las aplicaciones, mientras que otros literalmente traen las aplicaciones heredadas hacia el PC del usuario. Idealmente, el portal deberá permitirle a la empresa apalancar su sistema transaccional, como un ERP o el sistema de recursos humanos, al igual que aplicaciones más antiguas heredadas de ciclos tecnológicos anteriores.

Comunicación y Colaboración. Esta funcionalidad incluye chat, correo electrónico, calendarios compartidos, tableros de discusiones, foros, control de versiones,

reuniones por Web y asignación de recursos. Algunos productos proveen la facilidad de incorporar las aplicaciones existentes de correo o workflow, mientras que otros más sofisticados permiten la creación de proyectos o espacios de comunidades.

2.3.4 Servidores de Portales Comerciales y de código abierto

Esta es una clasificación entre otras que se puede hacer en cuanto a los servidores de portales. Por un lado están los Servidores de portales comerciales los cuales son desarrollados por empresas que consideran el código un producto mas y no permiten que sea visto ni modificado por otros. Entre estos se encuentran Webphere portal Server, Microsoft SharePoint portal Server, Bea System portal Server entre otros. Por otra parte tenemos los casos de servidores de portales de código abierto los cuales exponen el código que puede de esta manera ser consultado y modificado por cualquiera. Entre estos últimos encontramos varios exponentes, Exo Platform, Liferay, Pluto,…

La disponibilidad del código fuente posibilita que se hagan personalizaciones del producto, correcciones de errores y desarrollo de nuevas funciones. Así se asegura que el producto pueda evolucionar incluso después que la empresa creadora desaparezca.

Se podría pensar que al estar desarrollados y coordinados por un solo grupo o empresa los servidores de portales de código abierto son mucho más estable y coherentes sin embargo esto no resulta ser cierto puesto que los servidores de portales de código abierto generalmente también están coordinados por un solo grupo o empresa.

2.3.5 Criterios de evaluación

En el mercado existen varios servidores de portales quizás mas de los que se desearía muchas veces sin embargo las empresas deben elegir siempre pensando en sus necesidades y realidades algunos criterios de evaluación que pueden ser útiles en la mayoría de los casos se listan a continuación:

Cumplen con el estándar JSR 168.

Recursos y soporte

Foros de opinión y artículos en Internet

Experiencia y facilidad de uso.

Soporte e integración con frameworks como Struts, Spring, Tapestry e IDEs como Eclipse.

Soporte de servicios web y sistemas de mensajería conforme a la especificación WSRP.

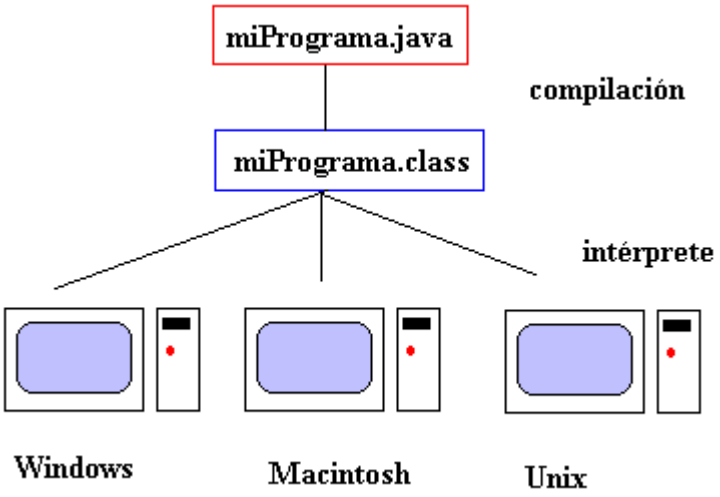
2.4 FUNDAMENTACIÓN DE LAS TECNOLOGÍAS EN QUE SE BASA LA PROPUESTA

2.4.6 JAVA

Java es un lenguaje de programación orientado a objetos puro, en el sentido de que no hay ninguna variable, función o constante que no esté dentro de una clase. Se accede a los miembros dato y las funciones miembro a través de los objetos y de las clases. Por razones de eficiencia, se han conservado los tipos básicos de datos, int, float, double, char, etc, similares a los del lenguaje C/C++.

El lenguaje Java es a la vez compilado e interpretado. Con el compilador se convierte el código fuente que reside en archivos cuya extensión es .java, a un conjunto de instrucciones que recibe el nombre de *bytecodes* que se guardan en un archivo cuya extensión es .class. Estas instrucciones son independientes del tipo de ordenador. El intérprete ejecuta cada una de estas instrucciones en un ordenador específico (Windows, Macintosh, etc). Solamente es necesario, por

tanto, compilar una vez el programa, pero se interpreta cada vez que se ejecuta en un ordenador.



Cada intérprete Java es una implementación de la Máquina Virtual Java (JVM). Los *bytecodes* posibilitan el objetivo de "write once, run anywhere", de escribir el programa una vez y que se pueda correr en cualquier plataforma que disponga de una implementación de la JVM. Por ejemplo, el mismo programa Java puede correr en Windows 98, Solaris, Macintosh, etc.

Las características más notables de este lenguaje y que lo ha hecho atractivo para desarrollar una amplia gama de proyectos por parte de desarrolladores de todo el mundo se resumen a continuación:

Java es orientado a objetos. Mucho se ha hablado en los últimos años sobre esta tecnología y varios han sido los artículos al respecto en esta revista.

Arquitectura neutral, portátil y robusta. Es neutral, al adoptar un sistema de código binario que es independiente de arquitecturas hardware, sistemas operativos y sistemas de ventanas. Es portátil, al definir de forma precisa los tipos y tamaños de los datos. Y es robusta, al poseer un chequeo del código tanto en tiempo de compilación como de ejecución. Y la mayor diferencia con C y C++, el modelo de memoria de Java elimina la posibilidad de sobrescribirla y la corrupción de los datos.

Simple. Posee las estructuras mínimas de un lenguaje de programación tradicional, sin añadir ninguna estructura más.

Independiente de la plataforma. Java se compila a un formato de código de byte que puede ser leído e interpretado por muchas plataformas, incluyendo Windows 95, Windows NT, Solaris 2.3, Linux, Mac OS, etc.

Seguro. El código de Java puede ser ejecutado en un entorno que prohíbe la introducción de virus, borrar y modificar ficheros o la ejecución de operaciones que provoquen la caída del ordenador y la pérdida de datos.

Alta prestación. Todavía no, (pero falta poco) para que el Java compilado pueda rivalizar en velocidad al C++.

MultiHilos. En Java, todo transcurre de forma paralela, con varias tareas de forma simultánea. Un único programa Java puede procesar diferentes cosas de forma independiente y continua. [8]

Cada día que pasa uno de los requerimientos que se esta haciendo fijo a la hora de desarrollar una aplicación empresarial es que esta sea multiplataforma. Para una empresa, el desarrollo de una misma aplicación entre distintas plataformas lleva muchísimo tiempo y dinero. Hasta ahora existía la solución de comprar compiladores compatibles o de la misma marca y desarrollar todo el código (casi siempre en C++) utilizando una batería de clases comunes a las dos plataformas. Sin embargo utilizando Java, el código sólo hay que escribirlo una sola vez, instalar el intérprete de Java y nada más.

2.4.7 Servidores de Portales

Los portales tradicionales adolecen de limitaciones técnicas que dificultan su adaptación a las nuevas necesidades de un mundo web en proceso de cambio. A un nivel muy general, podemos decir que la tendencia actual de los entornos web, tanto intranet como extranet está encaminándose a sustituir la información como eje central y valor clave de los portales en favor de la colaboración. El portal de PDVSA en su primera etapa y así lo cubre este trabajo está concebido para funcionar en un entorno de extranet, sin embargo este proyecto forma parte de un

esfuerzo aún más ambicioso que pretende transformar la realidad caótica que hoy identifica la intranet de PDVSA y fusionar en una sola solución el portal de PDVSA para que de la cara tanto para su intranet como para su extranet, las aspiraciones y necesidades que hoy se plantea PDVSA no están lejos de lo que esta ocurriendo en el mundo del portal corporativo actual. Los entornos educativos (universidades, colegios y comunidades educativas en general), las intranets de empresas medianas y grandes, las redes de investigación, las asociaciones de profesionales, y en general cualquier organización que conste de una comunidad social de usuarios demanda hoy día un nuevo tipo de portal muy centrado en el usuario que no sólo sea un escaparate informativo, sino un entorno de trabajo, información y colaboración adaptado a su realidad, organización y actividad donde el usuario reciba el protagonismo y la interactividad que poco a poco se está acostumbrado a tener en Internet.

Como consecuencia directa y natural de lo anteriormente expuesto, las empresas y organizaciones de todo el mundo están imponiendo una serie de requerimientos a los desarrolladores de portales entre las más importantes se encuentran las siguientes:

Personalización: los portales colaborativos requieren personalización a nivel de grupo y a nivel de usuario. Cada grupo y cada usuario debe tener una usabilidad e incluso un interfaz gráfico diferente al resto. Hoy día los portales soportan muy pocas variaciones en este sentido.

Gestión de contenidos: no sólo los administradores del portal deben contribuir a sus contenidos. Cualquier usuario debe ser capaz de hacerlo, eso sí, con facilidades de control de visibilidad en base a su grupo, departamento o puesto dentro de la comunidad o jerarquía de usuarios global. Los portales tradicionales carecen de las facilidades necesarias para que un usuario no especializado cree contenidos, y de los mecanismos de control necesarios para impedir que dichos contenidos sean visibles sólo para ciertos usuarios o grupos de usuarios.

Gestión de maquetación web: los usuarios y grupos de usuarios no deben limitarse a colaborar y crear sus propios contenidos, sino que deben ser capaces de expresarse en la web manteniendo su propio espacio dentro del portal, con su propia estructura de secciones, subsecciones, columnas, ventanas, aspecto gráfico, etc. Hoy día los portales son maquetados exclusivamente por el administrador de sistemas, o por el proveedor que desarrolla el portal. Es normalmente ya muy costoso incluso para un técnico especializado modificar la estructura de un sólo sitio web, y esto limita obviamente la posibilidad de que haya múltiples portales gestionados por los propios usuarios, sin conocimientos especializados, uno por cada usuario, grupo, departamento, centro, o entidad dentro de la comunidad, que es un requisito de los portales colaborativos.

Colaboración: los portales colaborativos no sólo requieren mostrar contenidos, sino que como su propia denominación indica deben incluir herramientas de colaboración, que permitan a sus usuarios sentirse parte de la comunidad a través de su participación y comunicación con otros usuarios en salas de conversación, tablones de anuncios, blogs, encuestas, foros, etc.

Tecnología abierta: el salto hacia los propios usuarios como creadores de contenidos y subportales descoloca a muchos productos de gestión de contenidos y portales cuyas licencias se miden por el número de usuarios que realizan este trabajo de gestión, y que están dimensionadas a nivel de coste para un orden máximo de decenas de administradores en grandes clientes. Si hablamos de cientos o miles de usuarios potenciales, los costes se disparan. Esto, unido a la propia evolución de la tecnología web que se renueva cada poco tiempo completamente, hace que sea una demanda cada vez mayor el no tener coste asociado a número de usuarios, ni atarse a tecnologías propietarias. Por otra parte, la evolución imparable del software libre en torno a estos estándares hace plantearse como alternativa la integración de soluciones open, que eliminan el coste de licencia asociada a usuarios. [9]

Como respuesta a estas demandas están empezando a establecerse en el mercado entornos de desarrollo y soluciones que facilitan la implantación de este tipo de portales, como los servidores de portal (portal servers) como las plataformas más especializadas
en colaboración, que aúnan herramientas ya conocidas como blogs, chats, foros, agendas o tablones de anuncios en soluciones completas. Incluyen grandes avances tecnológicos como la inclusión de la tecnología de portlets para permitir la gestión de maquetación.

2.4.8 Portlets

La tecnología de portlets enamora tanto por sus características técnicas como por sus ventajas en cuanto a costo en tiempo de desarrollo de un portal. De forma similar a lo que hace un servlet, un Portlet procesa una petición y responde con la salida apropiada a cada caso según ciertas condiciones de proceso. Obviamente, toda página dinámica realiza este proceso. En el caso de un portal con varios componentes modulares, la API de los Portlets gestiona y controla la ejecución, comunicación y respuesta de todos los portlets activos.

Cada portlet contribuye a la salida global con su parte correspondiente. Además, esta salida se produce de forma personalizada y en el idioma correspondiente. Desde el punto de vista de un desarrollador, un portlet es muy similar a un servlet, excepcionales algunas restricciones como los redirects y los forwards. En todo momento, el nivel de presentación se abstrae del nivel de lógica o contenido. Por otra parte, para integrar contenido externo, no es necesario codificar una línea de Java. Algunos productos proporcionan la facilidad de integración de portlets a través de los canales de Rich Site Summary (RSS) y Open Content Syndication (OCS). Naturalmente, también es posible procesar peticiones SOAP dentro de un Portlet a través de aplicaciones de terceros. De esta forma, se abre el camino para los Web Services a través del estándar Universal Description Discovery and Integration (UDDI) e invocar los servicios descritos en el Web Service Description Language (WSDL).

Desde el punto de vista del usuario final, los Portlets pueden ayudar de forma muy importante a mejorar la usabilidad de un portal, ya que cada usuario podrá ajustar a su antojo la distribución y la apariencia de los componentes con los que desea trabajar. Además, el proveedor de un portal podrá decidir qué puede ver y qué no, cada usuario. Esta característica puede lograrse, evidentemente, sin la necesidad del uso de Portlets, pero de cara al desarrollador el uso de portlets lo facilita muchísimo, sin tener en cuenta que además, se estará siguiendo un estándar abierto.

2.5 FUNDAMENTACIÓN DE LA METODOLOGÍA A UTILIZAR

Cada año que pasa vemos como programas ya comunes para nosotros nos sorprenden con nuevas funcionalidades y grandes mejoras y otros sistemas desconocidos surgen con excepcionales prestaciones, sin embargo increíblemente a lo sumo en 1 año y medio estos avances se tornan insuficientes para satisfacer las necesidades de sus usuarios. Esto se debe a que los usuarios necesitan un software que este cada vez más adaptado a sus necesidades pero esto trae como consecuencia que el software se haga cada vez más complejo.

Aunque en los últimos años se han establecido muy fuertemente varias metodologías que ayudan a desarrollar software de una forma coherente y organizada, todavía constituyen mayoría los desarrollos que al estar desprovistos de estas prácticas terminan en fracaso y no se acercan a la demanda de software potente y complejo que impera hoy en el mundo.

La razón principal por la que enfrentar un proyecto de software de escala mediana o grande es tan difícil estriba en la complejidad que representa para los desarrolladores coordinar las diferentes cadenas de trabajo que se presentan, por lo que se hace necesario de un proceso que integre las múltiples facetas del desarrollo y permita trabajar de manera coordinada a los desarrolladores.

2.5.9 El Proceso Unificado de Desarrollo (RUP)

El Proceso Unificado de Desarrollo no es una metodología surgida de la nada sino que es el resultado de la experiencia acumulada a lo largo de 30 años de desarrollo de software. Ha ido madurando y esta fuertemente marcada por varios métodos de desarrollo de software anteriores fusionando las mejores prácticas de cada uno de ellos y enriquecida por la experiencia de muchísimas otras personas que contribuyeron a que sea lo que hoy es.

Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo el Proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentas tipos de organizaciones, diferentas niveles de aptitud y diferentes tamaños de proyecto. [11]

El Proceso Unificado utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language, UML) para preparar todos los esquemas de un sistema software. De hecho UML es un pilar esencial del Proceso Unificado, sus desarrollos fueron paralelos.

No obstante, los verdaderos aspectos definitorios del Proceso Unificado se resumen en tres frases claves, dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental. Esto es lo que hace único al Proceso Unificado. [11]

2.5.10 UML(Unified Modeling Language)

El Lenguaje de Modelado Unificado (UML - Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales, como lo son procesos de negocio y funciones de sistema, además de cosas concretas como clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa. UML ha puesto fin a las llamadas “guerras de métodos” que se han mantenido a lo largo de los años 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos. [35]

2.5.11 Por que es necesario UML.

Uno de los objetivos principales de la creación de UML era posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado. Para ello era necesario definir una notación y semántica común.

Conforme aumenta la complejidad del mundo, los sistemas informáticos también deberán crecer en complejidad. En ellos se encuentran diversas piezas de hardware y software que se comunican a grandes distancias mediante una red, la misma que está vinculada a bases de datos que, a su vez, contienen enormes cantidades de información. Si desea crear sistemas que lo involucren con este nuevo milenio ¿cómo manejará tanta complejidad? La clave está en organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan y convengan con él. El UML proporciona tal organización.

La necesidad de diseños sólidos ha traído consigo la creación de una notación de diseño que los analistas, desarrolladores y clientes acepten como pauta (tal como la notación en los diagramas esquemáticos sirve como pauta para los trabajadores especializados en electrónica). El UML es esa notación.

La arquitectura, conformada por diferentes visiones del sistema, constituye un modelo de cómo está estructurado dicho sistema, sirviendo de comunicación entre las personas involucradas en el desarrollo y ayudando a realizar diversos análisis

que orienten el proceso de toma de decisiones. Para que la arquitectura se convierta en una herramienta útil dentro del desarrollo y mantenimiento de los sistemas de software es necesario que se cuente con una manera precisa de representarla. Las herramientas que se han elaborado para representar una arquitectura de software son los Lenguajes de Definición de Arquitecturas o ADL (Architecture Description Language). Sin embargo, los lenguajes desarrollados hasta el momento presentan diferentes problemas para su utilización en una empresa, como:

Requieren una extensa capacitación.

No son amigables para presentar la arquitectura a personas ajenas a la construcción del software.

No tienen herramientas ni metodologías de apoyo.

Algunos se encuentran especializados solo en un tipo particular de sistemas.

Sólo tienen en cuenta una sola estructura del sistema.

Las desventajas que se presentan en estos lenguajes pueden ser superadas si se utiliza un lenguaje de modelado que sea reconocido en la industria y que además esté apoyado por herramientas y metodologías de desarrollo, este lenguaje de modelado es UML, que se ha convertido en una notación estándar de hecho en las empresas. UML permite que se represente de manera semi-formal la estructura general del sistema, con la ventaja de que este mismo lenguaje puede ser usado en todas las etapas de desarrollo del sistema y su representación gráfica puede ser usada para comunicarse con los usuarios. [18]

2.6 HERRAMIENTAS A UTILIZAR

2.6.12 Racional XDE

Características y Beneficios

Rational XDE le ofrece a los diseñadores y desarrolladores de software un rico conjunto de capacidades de desarrollo y análisis runtime orientadas al modelado para construir aplicaciones de software de calidad. Ofrece ambientes completos de diseño y desarrollo visual que satisfacen las necesidades de toda organización

enfocada tanto a sistemas basados en J2EE como en .NET. Esta solución permite que los usuarios trabajen dentro de la IDE Eclipse incluida ya que brinda total integración con el marco de trabajo Eclipse, lo cual permite tener un único ambiente de trabajo para todas las actividades de desarrollo. Rational XDE está preparada también para integrarse con IBM® Webphere™ Studio Application Developer y otras IDEs importantes de la plataforma Java™. El Rational XDE Modeler permite que los arquitectos y diseñadores practiquen el desarrollo orientado al modelado utilizando Unified Modeling Language™ (UML). El soporte de UML, que es el estándar de la industria, y un poderoso motor de patrones permiten a los diseñadores y arquitectos crear una arquitectura de aplicación semánticamente rica que satisface las necesidades del negocio y es fácilmente comprendida por el resto del equipo. Los arquitectos y diseñadores pueden utilizar el soporte multimodelo del Rational XDE Modeler para separar incumbencias del análisis, arquitectura, diseño e implementación. Los desarrolladores pueden usar los modelos y patrones de arquitectura como base para la implementación, acelerando en consecuencia el desarrollo de las aplicaciones en coincidencia con la arquitectura.

El Rational XDE también hace posible que los arquitectos de datos y los diseñadores de bases de datos puedan crear modelos de datos lógicos y físicos para bases de datos DB2, Oracle, Sybase, y SQL Server. Los arquitectos pueden seguir un acercamiento top-down, creando un modelo lógico de requerimientos de datos, transformándolos en un diseño físico de base de datos, y luego implantar ese diseño en una base de datos. O partir de una base de datos existente y hacer ingeniería inversa del esquema generando el modelo físico de datos. La sofisticada capacidad de "Comparar y Sincronizar" permite comparar un modelo físico de datos con una base de datos, y reconciliar las diferencias. Es más, dado que la capacidad de modelado de datos comparte el mismo ambiente que el de modelado de la aplicación, es fácil conservar los modelos de datos y de la aplicación sincronizados.

2.6.13 ExoPlatform

Exoplatform es un poderoso servidor de portal corporativo de código abierto, como el código de toda la plataforma esta disponible permite que el portal pueda ser adaptado para responder a requerimientos específicos.

Exoplatform está disponible bajo la licencia GNU/GPL lo cual es una gran ventaja puesto que el software de código abierto en general ofrece un estándar alto de calidad ya que el código es revisado y mejorado por una gran comunidad de desarrolladores, y por otra parte no se está amarrado a una licencia comercial con el costo y riesgo que esto puede traer.

ExoPlatform 1.0 cuenta con herramientas fuertemente orientadas al desarrollador, incluye integración con Eclipse y un framework basado en la tecnología Java Server Fases la cual es mas cómoda a la hora de trabajar con portlets que la tecnología JSP aunque con ambas se logran buenos resultados.

La interfase basada en contenedores y portlets es atractiva e intuitiva para los diseñadores, resulta muy fácil adicionar nuevos portlets y colocarlos en el lugar deseado. Entre otras características interesantes sobresale el hecho que Exo soporta web services para portlets remotos (WSRP), resulta relativamente poco complicado cambiar de idiomas así como configurar nuevos idiomas, además brinda un mecanismo para el control del look and feel de nuestros portales aunque en la práctica resulte mucho menos efectivo que el brindado por otros servidores de portales.

2.6.14 Eclipse

Eclipse es un entorno independiente de la plataforma, de código abierto, para crear aplicaciones de cualquier tipo. Eclipse fue creado originalmente por IBM. Ahora lo desarrolla la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

El entorno integrado de desarrollo (IDE) de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad, a diferencia de otros entornos

monolíticos donde las funcionalidades están generalmente prefijadas, las necesite el usuario o no. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes además de Java. Por ejemplo, existe un módulo para dar soporte a C/C++. Existen módulos para añadir un poco de todo, desde Telnet hasta soporte a bases de datos.

Los componentes gráficos (widget) de Eclipse están basados en un juego de herramientas de tercera generación para Java de IBM llamado SWT que mejora los de primera y segunda generación de Sun (AWT y Swing, respectivamente). La interfaz de usuario de Eclipse cuenta con una capa intermedia de interfaz gráfica (GUI) llamada JFace, lo que simplifica la creación de aplicaciones basadas en SWT.

La definición que da el proyecto Eclipse acerca de su software es: "una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular".

Una de sus grandes ventajas es que basa su funcionamiento en plugins con lo que es ampliable para que haga prácticamente cualquier cosa, desde edición de XML a control del Tomcat, pasando por plugins para otros lenguajes como Perl o *Shell Script*.

Aprovechando esta característica de Eclipse los desarrolladores de Exoplatform crearon un módulo (plug in) para la integración con Exo, por lo que es muy cómodo ahora programar portlets en Eclipse y posteriormente desplegarlos directamente en el servidor. [14]

2.6.15 PostgreSQL

El manejador de bases de datos PostgreSQL es cada vez más popular y suele ser una alternativa muy atractiva ante las opciones comerciales, que provienen principalmente de Oracle e IBM.

Al igual que todo el software libre, cuenta con dos ventajas claras: un código fuente optimizado que puede ser modificado y adaptado, y una baja inversión por implementación, ya que no existen costos por licencia.

PostgreSQL ofrece la mayoría de las ventajas que otros programas comerciales tienen. Es un manejador de base de datos relacionales orientado a objeto e incorpora casi todas las funcionalidades de SQL, incluyendo tipos de datos definidos por el usuario, subselecciones y gran variedad de transacciones.

Su uso en proyectos de Internet es sumamente sencillo. La interfaz más usada suele ser PHP, sin embargo se puede acceder a sus funciones desde lenguajes como C, C++, ODBC, Phyton, TCL, .NET y JAVA que es en este caso el que nos interesa.

Para el desarrollo de bases de datos se cuenta con la herramienta Pgaccess, que brinda una poderosa interfaz gráfica que acorta los tiempos de desarrollo. Además, esta herramienta facilita la generación de reportes, una tarea fundamental cuando se administran bases de datos.

El soporte en línea con que cuenta esta herramienta es amplio. Desde el sitio web http://www.postgresql.org/ se puede acceder a una amplia colección de documentos y manuales, capaces de resolver la mayoría de los problemas que se suelen confrontar. De ser necesaria más ayuda en el sitio, se encuentran los enlaces a las listas de correo, libros y grupos de usuarios.

Se trata, sin duda alguna, de una opción muy atractiva al momento de implementar un manejador de base de datos que puede compararse en iguales condiciones a los productos comerciales. [15]

2.7 PROPUESTA

Finalmente se propone el desarrollo de un portal digital utilizando una arquitectura de portlets funcional. Se dividirá el sistema en *portlets* pequeños e independientes que encapsulen las funcionalidades básicas del portal, debido a la poca estructuración de los procesos de negocio del portal y a que no necesita de datos provenientes de diversas fuentes para su funcionamiento. Estos portlets estarán diseñados de manera que no utilicen servicios u objetos propios del servidor del portal en este caso Exoplatform, sino solo los servicios y objetos especificados en

el estándar JSR 168 puesto que son de obligatoria implementación por cualquier servidor de portales. Esto responde a la necesidad de hacer reutilizables nuestros portlets en futuras aplicaciones, lo cual generalizado puede convertirse en la reutilización de la solución en futuros proyectos con una arquitectura similar.

Se utilizará el servidor de portales Exoplatform por las características y motivos antes expuestos.

Como ambiente de desarrollo integrado (IDE, siglas en ingles) se propone el Eclipse por ser entre otras características un potente editor de código en lenguaje Java y existir un plugin que extiende sus funcionalidades para desarrollar aplicaciones de tipo portlet para Exoplatform.

Como gestor de Bases de datos se propone el PostgreSQL por sus características y por la completa integración de Java con el mismo.

2.8 CONCLUCIONES

Este capítulo realizó un análisis de las tendencias y tecnologías actuales que se pretenden utilizar en el desarrollo del proyecto. Así mismo se ha valorado y justificado la arquitectura y herramientas a utilizar.

En los próximos capítulos se describirá la arquitectura propuesta para dar una visión global de la solución.

CAPÍTULO III

DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN

3.1 INTRODUCCIÓN

En el presente capítulo se hace la descripción de la propuesta que trae este trabajo, para ello se describen los procesos del negocio que tienen que ver con el objeto de estudio, de acuerdo a esto se llega a la conclusión que debido a la poca estructuración de esos procesos, para poder entender el contexto en que se emplaza el sistema necesitamos definir conceptos que podemos agrupar en un Modelo de Dominio, para capturar correctamente los requisitos y poder construir un sistema correcto.

Además se enumeran los requisitos funcionales y no funcionales que debe tener el sistema que proponemos, lo que permite hacer una concepción general del sistema, e identificar mediante un Diagrama de Caso de Uso, las relaciones de los actores que interactúan con el sistema, y las secuencias de acciones con las que interactúan. En este capítulo se hace un primer acercamiento a la descripción de la arquitectura del portal, mediante su estructura funcional.

3.1.1 Estructura funcional de la arquitectura

Esta estructura representa las funciones que ofrece el sistema a los usuarios finales, a otros sistemas o dispositivos, es decir, lo que representa el sistema para los que interactúan con él. Puede verse esta estructura como la visión conceptual del sistema, permitiendo determinar los aspectos del negocio que se desean implementar en el sistema. Esta es una de las estructuras más importantes en un sistema, ya que ayuda a la captura de los requerimientos y se convierte en un medio de comunicación útil con los usuarios, permitiendo ver gráficamente las relaciones de los usuarios con el sistema y los procesos del negocio identificados por ellos mismos. [18]

3.2 DESCRIPCIÓN DE LOS PROCESOS DEL NEGOCIO

El proceso de negocio del portal digital de PDVSA debe ser definido a partir de su compromiso con los postulados de la nueva PDVSA. La principal tarea de este portal por el momento es difundir las realidades de la nueva PDVSA, mediante artículos de interés, noticias que destaquen los logros alcanzados por la entidad, el éxito de su modelo de negocio y su vinculación con el pueblo. Por lo tanto el flujo de trabajo principal será el relacionado con la elaboración, publicación de noticias en el portal y el acceso por parte de los usuarios del portal a dichos materiales.

Se trata de utilizar las posibilidades de espacio, inmediatez y alcance de la web para convertir el sitio en un medio de análisis e información sobre los principales acontecimientos que ocurren alrededor la principal empresa venezolana y su nueva proyección.

Mantener el contenido del portal es una tarea realizada por un equipo destinado a buscar, elaborar, clasificar y conformar las noticias y materiales que se publiquen en el sitio.

Con el presente trabajo se trataría de brindar la mayor flexibilidad para la publicación de contenido en el portal, además de asegurar que todos los usuarios tengan acceso a los materiales y a todos los servicios comunes del portal.

3.3 MODELO DE DOMINIO

Debido a la sencillez de la estructura y los mecanismos en el proceso de administración y actualización del portal, llegamos a la conclusión de que el negocio que se está estudiando no necesita un modelado completo del negocio. Además como la creación del portal es una nueva alternativa en el proceso de búsqueda de información acerca de la integración cubano-venezolana, no existen reglas de negocio definidas para la actualización del mismo.

Por las características antes expuestas se utilizará un modelo de dominio con el cual se pretende contribuir a la comprensión del contexto del sistema, y por lo tanto también contribuir a la comprensión de los requerimientos del sistema que se

desprenden de este contexto. Para capturar correctamente los requerimientos y poder construir correctamente un sistema se necesita tener un firme conocimiento del funcionamiento del objeto de estudio del mismo.

El objetivo del modelo de domino es comprender y describir las clases más importantes dentro del contexto del sistema, en otras palabras el modelado del dominio deberá contribuir a una comprensión del problema que el sistema resuelve en relación a su contexto.

También nos auxiliaremos de un glosario de términos sobre los nombres para identificar todos los conceptos que se utilizarán en el diagrama. El glosario y el modelo del dominio ayudarán a los desarrolladores, usuarios, clientes y otros interesados a utilizar un vocabulario común.

Se entenderá por *Artículos* a los materiales editoriales creados por la casa editorial de Patria Grande en calidad de artículos.

Se entenderá por *Material Multimedia* a los materiales editoriales creados por la editorial de PDVSA en calidad de Materiales Multimedia.

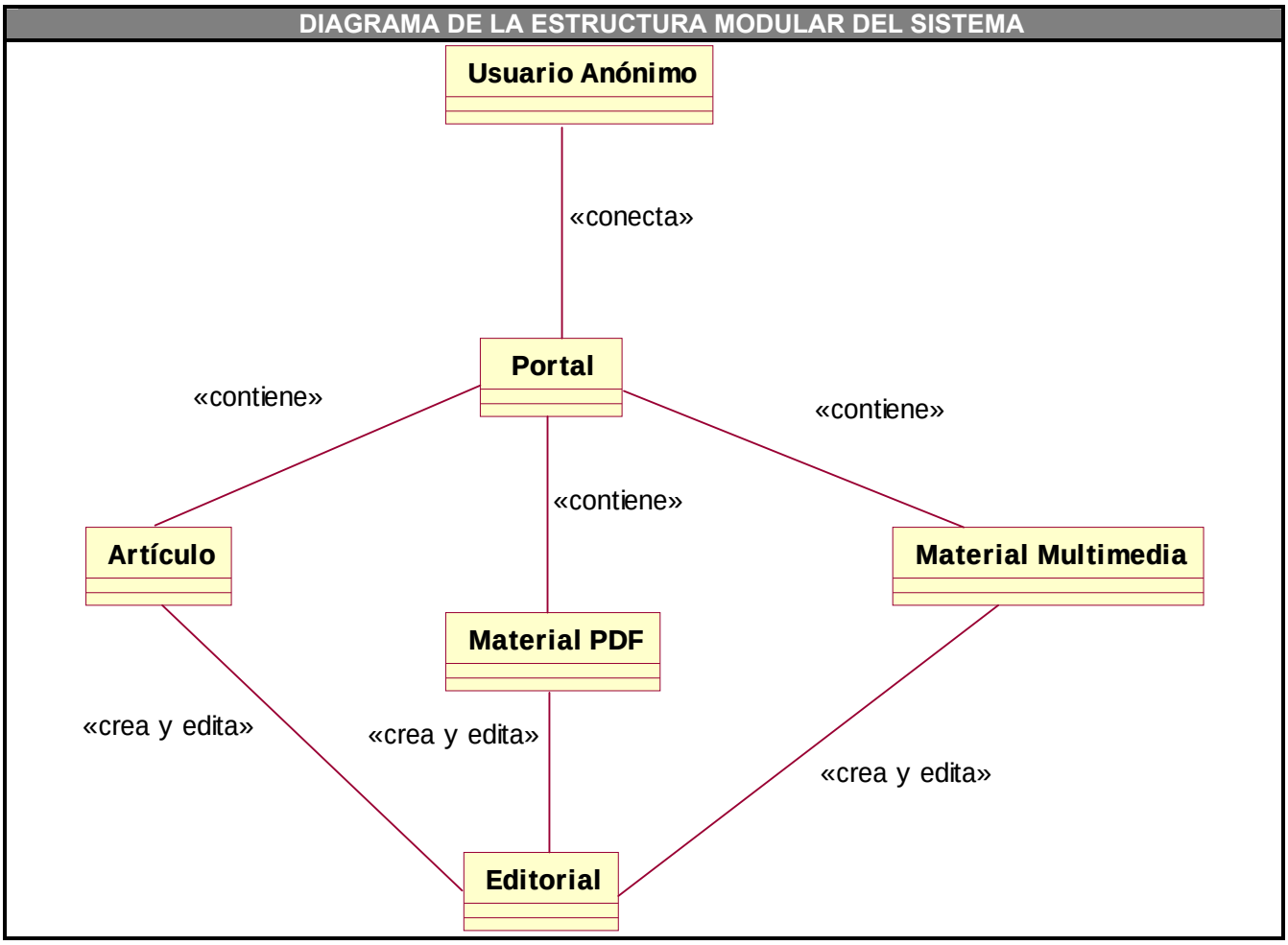
Se entenderá por *Material PDF* a los materiales editoriales creados por la editorial de PDVSA en calidad de Materiales PDF.

Se entenderá por *Usuario Anónimo* a cualquier usuario que desee obtener la información publicada en la revista Patria Grande.

Se entenderá por *Portal* a la aplicación web, en este caso el portal PDVSA, donde se publicarán artículos y materiales multimedia para ser accedidos a través de la web.

Se entenderá por *Editorial* a la casa editorial de PDVSA.

3.3.2 Diagrama del Modelo del Dominio



3.4 CAPTURA DE LOS REQUERIMIENTOS.

3.4.3 Requerimientos funcionales del sistema.

En vista de los objetivos antes expuestos el sistema debe ser capaz de:

1- Recibir información del sistema encargado de las ediciones.

a) El sistema debe ser capaz de recibir y adicionar las noticias enviadas por QuipusNews.

b) El sistema debe ser capaz de recibir y adicionar los artículos enviados por QuipusNews.

c) El sistema debe ser capaz de sobrescribir artículos y noticias según comandos de QuipusNews.

2- Administrar las noticias publicadas en el portal.

a) Eliminar artículo del portal.

3- Administrar usuarios y roles del sistema.

a) El sistema debe ser capaz de permitir el acceso a los administradores del portal así como a los usuarios anónimos.

El administrador puede:

a) Establecer nuevos roles de usuario.

b) Eliminar o adicionar usuarios al sistema.

El usuario anónimo puede:

a) Acceder al contenido publicado en el portal.

b) Descargar contenido del portal.

4- Mostrar noticias.

a) Mostrar noticias que no hallan caducado.

5- Mostrar sección de noticias.

a) El sistema debe ser capaz de mostrar una colección de titulares de noticias agrupadas por temas.

6- Mostrar enlaces con otros sitios de interés.

7- Mostrar materiales multimedia.

a) El sistema debe ser capaz de listar todos los archivos de videos disponibles.

b) El sistema debe ser capaz de listar todos los archivos de audio disponibles.

c) El sistema debe ser capaz de ejecutar y mostrar audio.

d) El sistema debe ser capaz de ejecutar y mostrar video.

8- Mostrar materiales de la biblioteca.

a) El sistema debe ser capaz de mostrar libros, publicaciones, artículos y discursos.

9- Mostrar significado de términos petroleros.

10- Consultar mapas con información de elementos petroleros en zonas determinadas.

11- Permitir que los usuarios interactúen a través de un foro.

a) El portal debe tener en un lugar visible el acceso a un foro de discusión donde los usuarios puedan opinar sobre temas actuales de la nueva PDVSA y sobre la revolución bolivariana.

12- Mostrar artículos de las secciones del portal.

a) El sistema debe ser capaz de mostrar artículos en dependencia de la sección del portal en la que se esté navegando.

13- Mostrar sección de fotorreportajes.

a) El sistema debe ser capaz de mostrar imágenes en forma de fotorreportajes sobre temas petroleros.

3.4.4 Requerimientos no funcionales del sistema.

Los requerimientos no funcionales definen propiedades o cualidades que el producto debe tener. Para que se tenga una idea más exacta son características que hacen al producto atractivo, usable, rápido.

Usabilidad:

El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.

Rendimiento:

Tiempos de respuestas rápidos al igual que la velocidad de procesamiento de la información.

Soporte:

Se requiere un servidor de bases de datos que de soporte para grandes volúmenes de datos y velocidad de procesamiento, además de que muestre un tiempo de respuesta rápido en accesos concurrentes.

Java Virtual Machine 1.4.x o superior.

Servidor de portales Exoplatform 1.0.

Por parte del cliente se requiere un navegador capaz de interpretar JavaScript.

Seguridad:

Identificar al usuario antes de que pueda realizar cualquier acción sobre el contenido del portal.

Garantizar que la información sea publicada únicamente por quien tiene derecho a publicarla.

Garantizar que las funcionalidades del sistema se muestren de acuerdo al nivel de usuario que esté activo.

Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

Verificación sobre acciones irreversibles (eliminaciones).

Político – Culturales:

Debido a que se trata de un producto para la nueva PDVSA, debe recoger todo el espíritu bolivariano, y enmarcarse en el ambiente de cambios revolucionario y nuevos aires que respira nuestra hermana nación venezolana.

Apariencia o interfaz externa:

Diseño orientado a imprimirle una identidad a la empresa basada en su nueva imagen.

Portabilidad:

Necesidad de que el sistema sea multiplataformas.

Confiabilidad:

La herramienta de implementación utilizada debe tener soporte para recuperación ante fallos y errores.

Funcionalidad:

Reducir al mínimo el tiempo en que carga el portal.

Guardar en caché páginas de contenido para agilizar la navegación del portal.

Software:

Navegador superior o compatible con Internet Explorer 4, o Netscape Navigator.

PostgreSql 8.0.1

Exoplatform 1.0.x

Máquina Virtual de Java 1.4.2_05

DESCRIPCIÓN DEL SISTEMA PROPUESTO.

Para cumplir exitosamente con los objetivos propuestos en este trabajo y teniendo en cuenta los requerimientos planteados el sistema propone el desarrollo un módulo, el portal PDVSA, para brindar noticias y artículos relacionados con la empresa, además se incorpora el módulo de administración del sistema el cual no modelaremos ya que es incorporado por el propio sistema servidor de portales que se usará como plataforma para el portal.

Se considera la existencia de dos roles, el rol de administrador del sistema que se encargará de la configuración del portal y el rol de usuario anónimo el cual podrá navegar por el portal obteniendo la información que necesite.

El módulo del portal podrá ser usado por cualquier usuario anónimo que desee obtener información publicada en el portal. Este podrá navegar por el portal como estime conveniente, participar en foros, consultar artículos en fin, beneficiarse de los servicios que ofrece el portal.

3.5 ACTORES DEL SISTEMA

ACTORES	JUSTIFICACION
Usuario Anónimo	Representa a una persona que va a utilizar el sistema para buscar información sobre alguna temática determinada.
QiupusNews	Representa al sistema encargado de editar y enviar al portal los artículos y los materiales multimedia que serán publicados en el portal.
Administrador	Representa a una persona que configura y controla el comportamiento del sistema, controlar a los usuarios, definir roles etc.

3.6 MODELO DE CASOS DE USO DEL SISTEMA.

Utilizando las facilidades que brinda el UML, se representarán los requisitos funcionales del sistema mediante un diagrama de casos de uso. Cada caso de uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia.

A continuación se presentan los casos de uso determinados para satisfacer los requerimientos funcionales del sistema:

CU-1	Recibir contenido.
Actor	QuipusNews
Descripción	El sistema recibe o sobrescribe contenidos publicados según comando de QuipusNews.
Referencia	R1.

CU-2	Mostrar noticias de la Sala de Prensa.
Actor	Usuario anónimo, Administrador.
Descripción	El sistema debe ser capaz de mostrar cualquier noticia completa de la sala de prensa elegida por el usuario.
Referencia	R4, R5

CU-3	Mostrar artículos de las secciones.
Actor	Usuario anónimo, Administrador.
Descripción	El sistema debe ser capaz de mostrar cualquier artículo elegido por el usuario de acuerdo a la sección en la cual esté navegando.
Referencia	R12.

CU-4	Mostrar audio y video.
Actor	Usuario anónimo, Administrador.
Descripción	El sistema debe ser capaz de ejecutar cualquier archivo de audio o video elegido de los disponibles.
Referencia	R4.

CU-5	Mostrar fotorreportajes de la galería.
Actor	Usuario anónimo, Administrador.
Descripción	El sistema debe ser capaz de mostrar un fotorreportaje elegido por el usuario, así como un listado de los que están disponibles.
Referencia	R13.

CU-6	Mostrar materiales de la biblioteca.
Actor	Usuario anónimo, Administrador.
Descripción	El sistema debe ser capaz de mostrar un material completo de la biblioteca elegido de los disponibles.
Referencia	R8.

CU-7	Mostrar significado de términos petroleros.
Actor	Usuario anónimo, Administrador.

Descripción	El sistema debe ser capaz de mostrar el significado de cualquier término petrolero disponible elegido por el usuario.
Referencia	R9.

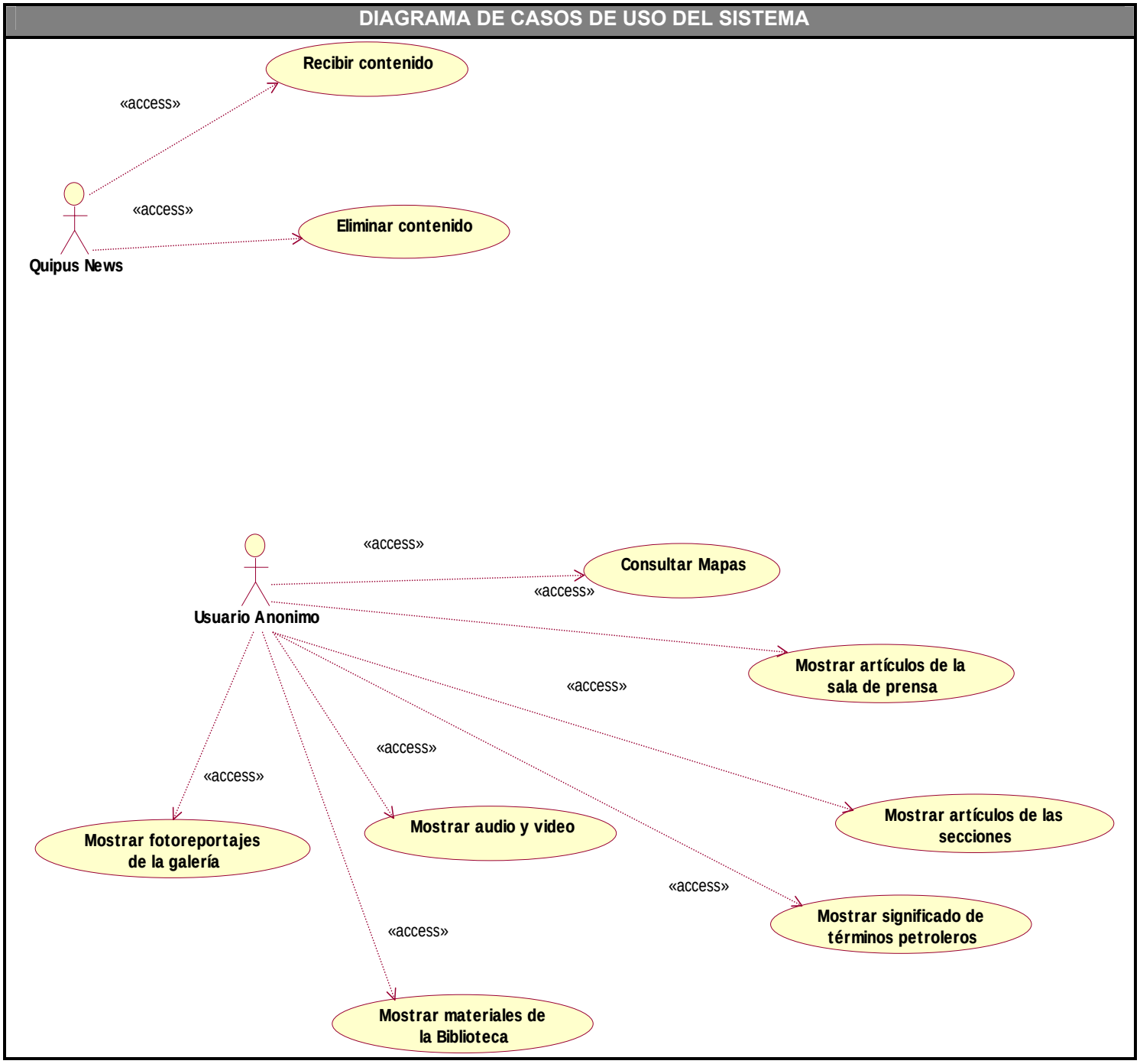
CU-8	Administrar el sistema.
Actor	Administrador.
Descripción	El administrador accede al portal para configurar la seguridad.
Referencia	R3.

CU-9	Consultar mapas
Actor	Usuario anónimo, Administrador.
Descripción	El sistema debe ser capaz de mostrar el mapa elegido por el usuario.
Referencia	R10.

CU-10	Eliminar contenido.
Actor	QuipusNews
Descripción	El sistema debe ser capaz de eliminar de manera lógica contenidos publicados en el portal según comando de QuipusNews.
Referencia	R2

3.6.5 Diagrama de casos de uso.

A continuación se representa la relación entre los actores y los casos de uso del sistema:



3.7 EXPANSIÓN DE LOS CASOS DE USO DEL SISTEMA.

A continuación se presenta la expansión de los casos de uso del sistema de mayor importancia:

CU-1	Recibir contenido del sistema encargado de las ediciones
Actores	QuipusNews.
Propósito	Que el sistema sea capaz de recibir el contenido enviado por el sistema QuipusNews por el protocolo HTTP.
Resumen	El caso se uso inicia cuando el sistema QuipusNews envía algún material al portal. El sistema QuipusNews envía materiales editados (artículos, noticias, videos, audios) listos para publicar a través del protocolo HTTP y el método POST el sistema debe ser capaz de recibir estos materiales, parsear los datos, e insertarlos en las tablas correspondientes de la base de datos que da soporte al sistema y en caso de los ficheros (imágenes y ficheros PDF, audio o video) en los directorios correspondientes.
Referencias	R1
Poscondiciones	El contenido recibido debe quedar correctamente almacenado en la base de datos que da soporte al sistema, así como ficheros asociados en el envió.
Requisitos especiales	Luego de verificar la autentificación e insertar correctamente los datos el sistema debe enviar un mensaje a QuipusNews de reportando si la operación de recibir se realizó satisfactoriamente o su hubo algún problema.
Acción del actor	**Respuesta del sistema**
1. Quipus envía via http un material al portal. Los materiales pueden ser: Artículo Material Pdf Material multimedia	2. El sistema salva los datos del material en la base de datos 3. El sistema salva los ficheros correspondientes en el sistema de archivos**.**

CU-2	Mostrar noticias de la Sala de Prensa.
Actores	Usuario Anónimo, Administrador.
Propósito	Que el usuario pueda seleccionar y ver cualquier noticia publicada en el portal.

Resumen

Este caso de uso inicia cuando un usuario accede al portal como usuario anónimo en la Internet o una intranet, y elige la sección Sala de Prensa entonces el sistema se encarga de mostrar un sumario de las noticias que estén publicadas en ese momento en el portal divididas por temas dando la posibilidad de acceder a su contenido.

Referencias	R4,R5
Precondiciones	Debe existir el cuerpo del artículo en el sistema de archivos.

Escenario Principal	
Acción del actor	**Respuesta del sistema**
1. El usuario elige ver los artículos de la Sala de prensa.	2. El sistema busca los artículos que estén vigentes y muestra un sumario con los titulares de los artículos agrupados por temática.
3. El usuario selecciona el artículo que desea leer.	4. El sistema muestra los detalles del artículo.

CU-3	Mostrar artículos de las secciones.
Actores	Usuario Anónimo, Administrador.
Propósito	Que el usuario pueda acceder a los artículos publicados en cualquiera de las secciones del portal.

Resumen

Este caso de uso inicia cuando un usuario accede al portal como usuario anónimo en la Internet o una intranet, y elige una de las secciones del portal (distinta a la Sala de Prensa, Biblioteca o Glosario), el sistema debe mostrar un artículo completo y un sumario de otros artículos vinculados a esa sección.

Referencias	R12
Precondiciones	Debe existir el cuerpo del artículo en el sistema de archivos.

Acción del actor	**Respuesta del sistema**
Escenario Principal	
1. El usuario elije ver los artículos de una de una sección distinta a Sala de Prensa, Biblioteca o Glosario.	2. El sistema busca los artículos pertenecientes a esa sección que estén vigentes. 3. El sistema muestra los detalles del artículo más actual y un listado de los títulos del resto de artículos como hipervínculos.
4. El usuario selecciona el artículo que desea ver.	5. El sistema muestra los detalles del artículo

CU-4	Mostrar audio y video.
Actores	Usuario Anónimo, Administrador.
Propósito	Que el usuario pueda ejecutar y ver cualquier archivo de audio o video publicados en el portal.

Resumen

Este caso de uso inicia cuando un usuario accede al portal como usuario anónimo en la Internet o una intranet, y elige la sección de la galería de audio o videos, el sistema debe ofrecer un sumario de estos con la posibilidad de elegir ejecutar cualquiera de ellos.

Referencias	R4
Precondiciones	Deben existir los ficheros de audio y video en el sistema de archivo.
Acción del actor	**Respuesta del sistema**
Escenario Principal	
1. El usuario elije ver los archivos de video o audio de la Galería Multimedia	2. El sistema determina el tipo de multimedia que se requiere (audio o video). 3. El sistema busca los datos de los materiales vigentes del tipo solicitado. El sistema muestra un sumario con el título y resumen de los materiales vigentes.
4. El usuario selecciona uno de los materiales listados.	5. El sistema obtiene la URL del material. 6. El sistema ejecuta el material a través del navegador del usuario.

CU-5	Mostrar fotorreportajes de la galería.
Actores	Usuario Anónimo, Administrador.
Propósito	El sistema debe ser capaz de mostrar un fotorreportaje elegido por el usuario, así como un listado de los que están disponibles.

Resumen

Este caso de uso inicia cuando un usuario accede al portal como usuario anónimo en la Internet o una intranet, y elige la sección de fotorreportajes, el sistema debe ser capaz de mostrar un fotorreportaje y mostrar un sumario con los que están disponibles con la opción de ser elegidos.

Referencias	R13
Precondiciones	Deben existir las imágenes del fotorreportaje en el sistema de archivo.
Acción del usuario	**Respuesta del sistema**
Escenario Principal	
1. El usuario elije ver los fotorreportajes de la Galería Multimedia.	2. El busca los datos de los fotorreportajes disponibles. 3. El sistema muestra las imágenes del fotorreportaje más actual. 4. El sistema muestra un sumario del título de los restantes fotorreportajes.
5. El usuario selecciona uno del fotorreportajes listados.	6. El sistema muestra las imágenes del fotorreportaje. 7. El sistema muestra un sumario del título del resto de fotorreportajes.

CU-6	Mostrar materiales de la biblioteca.
Actores	Usuario Anónimo, Administrador.
Propósito	El sistema debe ser capaz de mostrar un material completo de la biblioteca elegido de los disponibles.

Resumen

Este caso de uso inicia cuando un usuario accede al portal como usuario anónimo en la Internet o una intranet, y elige la sección de la biblioteca, el sistema debe ofrecer un listado de categorías bajo las cuales estarán disponibles varios materiales, una vez elegida una categoría, el sistema debe ser capaz de mostrar un sumario de materiales disponibles con la posibilidad de ver el contenido de cualquiera de ellos.

Referencias	R8
Precondiciones	Debe existir el fichero de los materiales en el sistema de archivos.

Acción de actor	**Respuesta del sistema**
Escenario Principal	
1. El usuario elije ver los materiales de la Biblioteca.	2. El busca los datos de los materiales de la temática Biblioteca del tipo Libros y muestra un sumario con los datos de los libros vigentes en la Biblioteca. 4. El sistema muestra un listado con el resto de tipos de materiales que brinda la Biblioteca en forma de hipervínculo. Estos materiales pueden ser : Libros Artículos Documentos Oficiales Discursos Publicaciones
4. El usuario selecciona uno de los tipos de materiales que brinda la Biblioteca.	5. El sistema desarrolla los mismos pasos a partir de 2, pero con el tipo de material seleccionado.
6. El usuario selecciona uno de los materiales listados.	7. El sistema muestra los detalles del material seleccionado.

CU-7	Mostrar significado de términos petroleros.
Actores	Usuario Anónimo, Administrador.
Propósito	El sistema debe ser capaz de mostrar el significado de cualquier término petrolero disponible elegido por el usuario.

Resumen

Este caso de uso inicia cuando un usuario accede al portal como usuario anónimo en la Internet o una intranet, y elige la sección del glosario petrolero, el sistema debe ofrecer un formulario para que el usuario ingrese una palabra o parte de una palabra, y un elija un criterio de búsqueda, a partir de aquí el sistema debe ser capaz mostrar un listado de términos que cumplan con estas condiciones dando la posibilidad de ver el significado de cualquiera de ellos.

Referencias	R9
Precondiciones	Deben existir términos en la base de datos.
Poscondiciones	
Requisitos especiales	
Acción del actor	**Respuesta del sistema**
Escenario Principal	
1. El usuario elije ver el significado de términos del Glosario.	2. El sistema solicita los parámetros para realizar la búsqueda del término. Los parámetros son: Subcadena: parte del término que se desea buscar. Criterio: la búsqueda se hará sobre la base de 3 criterios. - términos que comiencen con la subcadena especificada. - Términos que finalicen con la subcadena especificada. - Términos que contengan la subcadena especificada.
3. El usuario especifica los parámetros.	4. El sistema busca los términos que cumplan con los parámetros solicitados. 5. El sistema muestra un sumario con los términos encontrados.
6. El usuario pincha sobre uno de los términos listados.	7. El sistema muestra el significado del término seleccionado. 8. El sistema continúa mostrando el sumario de términos encontrados anteriormente.
Escenario Alternativo	
	4. No existe ningún término que cumpla con los parámetros solicitados por el usuario 5. El sistema le informa al usuario que no encontró ningún término y le da la posibilidad de

	reingresar los parámetros de búsqueda.

CU-8	Administrar el sistema
Actores	Administrador.
Propósito	Que le permita al administrador controlar la seguridad del portal, controlar los accesos, roles y usuarios.
Resumen	
Este caso de uso inicia cuando un usuario accede al portal como administrador permitiéndole ver los usuarios del sistema crear nuevos usuarios, eliminar usuarios o modificar los datos de un usuario, ver los grupos existentes, crear nuevos grupos de usuarios, modificar los grupos y eliminar los grupos creados, permite además la administración general del portal. Este caso de uso es totalmente realizado por las funcionalidades que brinda Exoplatform.	
Referencias	R3

CU-9	Consultar Mapas.
Actores	Usuario Anónimo, Administrador.
Propósito	El sistema debe ser capaz de mostrar el mapa elegido por el usuario.
Resumen	
Este caso de uso inicia cuando un usuario accede al portal como usuario anónimo en la Internet o una intranet, y elige la sección mapas petroleros, aquí el sistema debe ser capaz de ofrecer un sumario de mapas disponibles con la posibilidad de ver el contenido de cualquiera de ellos.	
Referencias	R8
Precondiciones	Deben existir mapas para ser mostrados en sistema de archivos
Acción del actor	**Respuesta del sistema**
Escenario Principal	
1. El usuario pincha para ver la sección de mapas.	2. El sistema datos de los mapas vigentes. 3. El sistema muestra un sumario con los datos de los mapas.
4. El usuario pincha para ver uno de los mapas.	5. El sistema obtiene la URL del mapa. 6. El sistema muestra el mapa seleccionado.

3.8 CONCLUSIONES

Partiendo del análisis de los procesos del negocio, en este capítulo hemos definido los requerimientos que debe tener el sistema para un exitoso desarrollo, los cuales se representan mediante un Diagrama de Casos de Uso, así mismo se

describieron la acciones que debe realizar cada actor que interactúa con el sistema.

Con los requerimientos y las funciones principales que han sido consideradas como necesarias para la construcción del sistema se puede dar paso al comienzo de la construcción del mismo.

Este ha sido un primer acercamiento a la arquitectura del sistema a través de la estructura funcional del mismo, en este caso el modelo de casos de uso del sistema.

CAPÍTULO IV

CONSTRUCCIÓN DE LA PROPUESTA DE SOLUCIÓN.

4.1 INTRODUCCIÓN

En este capítulo se modelan los artefactos necesarios para la construcción de aplicaciones Web. Los componentes de la aplicación son tratados como portlets y clases, y mediante la utilización de UML se podrán representar a través de diagramas de clases web y diagramas de componentes. Así mismo se presenta la propuesta del modelo de datos la cual nos posibilitará la construcción de la base de datos que servirá como soporte al sistema. Este capítulo además de modelar la lógica del negocio mediante las clases Web, trata algunos principios del diseño del sistema.

4.2 DIAGRAMA DE CLASES WEB.

El diagrama de clases para las Aplicaciones Web, difiere un poco del resto de las aplicaciones que se construyen, puesto que en ellas son más importantes la modelación de la lógica y estado del negocio que los detalles de presentación. Dado que para el desarrollo de este sistema se utilizará el Servidor de Portales ExoPlatform se dará una breve explicación del funcionamiento interno del mismo para una mejor comprensión de los diagramas de clases.

Exoplatform cuenta con 2 componentes fundamentales, la aplicación web del portal y un contenedor de portles.

El contenedor de portlets se encarga de recibir todas las peticiones y dirigirlas al portlet correspondiente. Para esto tiene que controlar el ciclo de vida de los portlets, los crea cuando se agregan por primera ves y a partir de la primera petición los va incorporando a un pool de portlets. Cuando un portlet dentro del pool pasa cierto tiempo sin recibir peticiones es destruido. Una respuesta de Exo a una petición al portal se traduce en mostrar lo que se ha dado en llamar una

"página portal" (portal page), la cual esta compuesta por varios portlets, por lo tanto una petición de render a una página portal, Exo la descompone en varias peticiones render a los distintos portlets que la integran. Un flujo de ejecución genérico podría ser el siguiente: Un usuario hace una petición a un portlet determinado, por ejemplo un submit a un formulario, la petición la recibe el contenedor de portlets, quien la delega al portlet en cuestión, además envía una petición de render para el resto de portlets que conforman la página portal (portal page). Estos devuelven sus respuestas en forma de markups HTML y el contenedor de portlets se encarga de unirlos todos en un HTML único que se muestra como respuesta.

El contenedor de portlets se encarga de gestionar todos los elementos J2EE de los portlets en contextos diferentes. Cada portlet se gestiona en su contexto. No existe un servidor de aplicaciones para cada portlet, sino que existe un único servidor de aplicaciones al cual los portlets delegan la gestión de sus componentes J2EE, como JSPs, Servlets, EjBs. Si bien es cierto que, como dijimos antes, un portlet es una aplicación web, el hecho de trabajar solo con un servidor de aplicaciones tiene sentido puesto que un portal puede tener tantos portlets como considere necesario el desarrollador. Como aplicaciones web que son, los portlets pueden contener páginas JSPs, Servlets, EjBs, y todos estos elementos se gestionan dentro del contexto del portlet.

La aplicación web del portal por su parte implementa los casos de uso del portal (registro y autenticación de usuarios, selección de portlets y layout en las páginas, creación y destrucción de páginas, agregación de las respuestas de los portlets en la página actual, etc).

4.2.1 Estructura modular del sistema.

La estructura modular presenta una división del sistema en módulos más pequeños o subsistemas. [18]

Debido a las características de los portlets estos encajan perfectamente en el perfil de subsistemas de diseño, puesto que exponen una interfaz con operaciones definidas, en este caso el portlet, que se gestiona como un objeto en el contenedor y que expone operaciones como render(), action(), edit(), ect. Aprovechando esto modelamos el sistema dividiéndolo en subsistemas de forma que encapsulamos en portlets las funcionalidades fundamentales del portal. El sistema cuenta con los subsistemas que se listan a continuación:

Capturador de Contenidos. Este encapsula toda la lógica asociada a la recepción de contenido procedente de QuipusNews y su almacenamiento así como la eliminación de contenido a solicitud de QuipusNews.

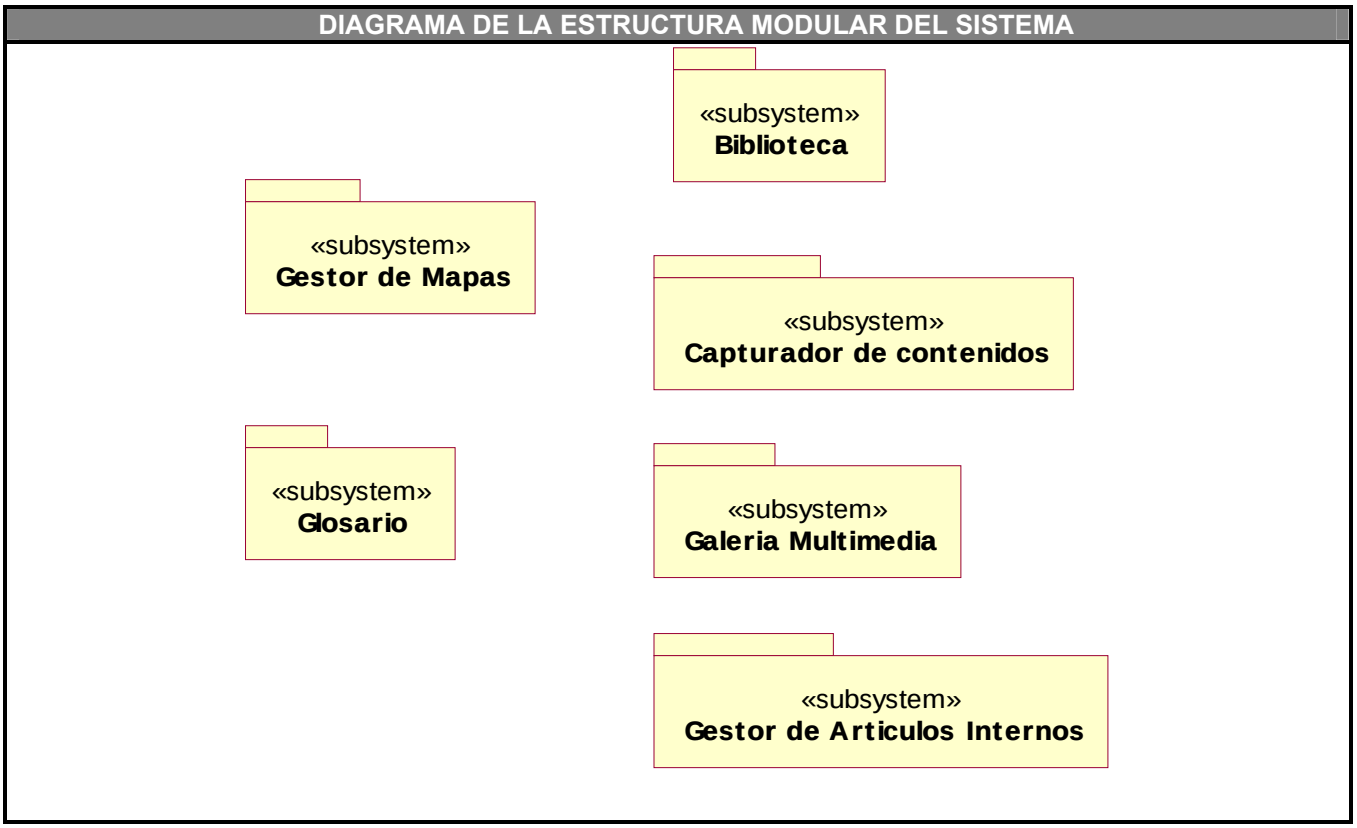
Glosario. Se encarga de manejar los términos petroleros, búsqueda y muestra de su significado.

Gestor de artículos internos. Encapsula la lógica necesaria para mostrar los artículos del portal.

Biblioteca. Se encarga de gestionar los materiales asociados a la temática biblioteca.

Galería Multimedia. Gestiona la ejecución de materiales como video y audio disponibles en el portal.

Gestor de mapas. Gestiona los mapas disponibles en el portal.



4.2.2 Principios de diseño

Como vimos en el capitulo anterior una aplicación de portlets se puede modelar arquitectónicamente como orientada a servicios o como arquitectura de portlet funcional. Para el caso del portal de PDVSA se propone usar la segunda variante. PDVSA busca con su portal la inmediatez en la comunicación con el usuario y difundir la imagen de la nueva PDVSA como sus objetivos estratégicos más importantes; para esto se necesita un portal flexible, atractivo, con atributos que eleven su usabilidad, pero sobre todo un portal con un elevado rendimiento. Este último requisito es el que decanta la decisión. Una arquitectura orientada a servicios tiene como talón de Aquiles precisamente el rendimiento, puesto que los portlets tienen que gestionar el contenido de manera remota. En el capítulo 5 se abordan las ventajas de usar esta variante para la arquitectura del portal.

Internamente cada portlet se diseñó con una arquitectura MVC, teniendo en cuenta que los portlets son una potencial extensión de J2EE, y que el patrón de arquitectura MVC es ampliamente usado en esta plataforma de desarrollo.

Básicamente en el modelo MVC, lo que hace el controller es filtrar peticiones, extraer datos, mapear peticiones a comandos e invocar comandos para gestionar estas peticiones. La parte de Vista determina páginas a visualizar, y formatea la página para su visualización. El cliente interacciona con estas vistas, y en curso de dicha interacción hace peticiones http.

Patrón Service to Worker.

Un patrón de arquitectura puede contener varios patrones de diseño, en este caso utilizamos el patrón de diseño Service to Worker de la plataforma J2EE para modelar cada portlet.

Este patrón es básicamente una implementación de la arquitectura Model-View-Controller. El Service to Worker es la unión de los siguientes patrones como solución:

Front Controller (controlador frontal). Este elemento provee un controlador centralizado para gestionar las peticiones a la aplicación. Un controlador frontal recibe todas las peticiones entrantes de los clientes, remitiendo a su vez cada petición al gestor de peticiones (Dispatcher) adecuado, que se encargará de gestionar la construcción de una respuesta adecuada al cliente. Son los puntos ideales para implementar servicios de seguridad, tratamiento de errores, y la gestión del control para la generación de contenidos.

View Helper (Ayudante o Auxiliar de Vista). Un ‘View Helper’ encapsula los trozos de lógica (código java) correspondientes a la presentación y al acceso a datos y componentes que necesita una vista, haciendo que la vista permanezca de esta forma mucho más simple, reutilizable y mantenible. La lógica de presentación se encarga de formatear datos para que sean visualizados en una página, mientras

que el acceso a datos o componentes implica la obtención de datos. Los View Helpers suelen ser JavaBeans.

Vistas. Páginas JSPs. Se encargan de generar contenido visual específico que responda a las necesidades del usuario. Las páginas JSPs por lo general estarán parametrizadas de tal forma que muestren diferente información según los parámetros que le mandemos. Por lo general una vista (JSP) produce un trozo de la página web que recibe el usuario. [22]

En nuestro caso utilizamos 2 estrategias de este patrón una conocida como “Servlet View”, esta estrategia utiliza un servlet como la vista. Es semánticamente equivalente al uso de JSPs para la vista. [23] La otra conocida como “Business as Helper” es aquella en la que se combina el Helper y la clase de negocio para implementar la clase de negocio como un Helper especializado. [22]

Las vistas serán generadas por el servlet a partir de plantillas. Exoplatform a diferencia de otros servidores de portales como Liferay no cuenta con un sistema de temas gráficos eficaz, en este sentido resulta más sencillo definir plantillas para cada elemento de presentación y aprovechando la tecnología de servlets que permite generar una respuesta html de manera dinámica, se parsean estas plantillas que contienen tags previamente definidos los cuales son sustituidos por valores específicos para lograr una salida html entendible por Exoplatform.

A continuación mostraremos los diagramas de clases web de cada subsistema, más adelante mostraremos como se distribuyen en el MVC los componentes de estos subsistemas.

El subsistema Capturador de Contenidos, encapsula el mecanismo de comunicación de Exoplatform con QuipusNews. Nos basamos en la idea de que el servidor de portlets Exoplatform es una extensión del servidor web tomcat. El componente fundamental de este subsistema es un servlet, el cual se consiguió

desplegar en la carpeta de clases de la aplicación que da inicio a Exoplatform, el descriptor de despliegue de la misma (web.xml) fue alterado para hacer efectivo el despliegue, se le indicó que la prioridad de ejecución del servlet sería -1, así cuando el servidor arranca el servlet comienza a ejecutarse inmediatamente, aún sin que se le hallan hecho peticiones al portal. Este subsistema no fue diseñado siguiendo el patrón MVC, puesto que las clases participantes no tienen una distribución lógica defina sino que solo colaboran con el servlet para persistir el contenido que le llega.

4.2.3 Descripción de las clases de diseño

Nombre: ControlCapturador	
Tipo de clase: control	
Para cada responsabilidad:	
Nombre:	*Instancia* ()
Descripción:	Retorna una única instancia de la clase.
Nombre:	*recibirContenido(pMultiparte)*
Descripción:	Persistir en la base de datos y en el sistema de archivos el contenido enviado por QuipusNews.
Nombre:	*eliminarContenido(id)*
Descripción:	Eliminar lógica o físicamente un determinado contenido, previa orden de QuipusNews.

Nombre: UtilBD	
Tipo de clase: Acceso a datos	
Para cada responsabilidad:	
Nombre:	*Instancia* ()
Descripción:	Retorna una única instancia de la clase.
Nombre:	*getArticulo(idArticulo)*
Descripción:	Recupera de la base de datos los campos de un determinado

	artículo.
Nombre:	*eliminarContenido(quipusId)*
Descripción:	Eliminar lógica o físicamente un determinado material de la base de datos.
Nombre:	*getFotoreportaje(id)*
Descripción:	Devuelve una lista con los datos asociados a cada imagen del fotorreportaje.
Nombre:	*getListadoArticulos(idTematica, cantidad)*
Descripción:	Devuelve la lista de artículos de una temática determinada.
Nombre:	*getListadoFotoreportajes()*
Descripción:	Devuelve una lista de tipo Material con los datos de los fotorreportajes vigentes.
Nombre:	*getListadoMapas()*
Descripción:	Devuelve una lista de tipo MaterialMultimedia con los datos de los mapas vigentes.
Nombre:	*getListadoMultimedia(tipo)*
Descripción:	Devuelve una lista con los datos de los materiales multimedia de un tipo dado. Audio o Video.
Nombre:	*getListadoTerminos(subcadena, criterio)*
Descripción:	Devuelve una lista con los términos que cumplan con los parámetros dados.
Nombre:	*getListaMaterialesBiblioteca(tipo)*
Descripción:	Devuelve una lista de los materiales de la Biblioteca de un tipo determinado. El tipo puede tomar los siguientes valores: Libro, Discurso, DocumentoOficial, Publicaciones, Articulo.
Nombre:	*salvarArticulo(pArticulo)*
Descripción:	Persiste en la base de datos los campos asociados a un artículo tomando como llave el valor del parámetro pArticulo.
Nombre:	*salvarMultimedia(pMultimedia)*

Descripción:	Persiste en la base de datos los campos asociados a un MaterialMultimedia, tomando como llave el valor del parámetro pMultimedia.
Nombre:	*salvarPDF(pPdf)*
Descripción:	Persiste en la base de datos los campos asociados a un MaterialPdf tomando como llave el valor del parámetro P.D..

Nombre: BusinessGlosarioHelper	
Tipo de clase: control	
Para cada responsabilidad:	
Nombre:	*Instancia* ()
Descripción:	Retorna una única instancia de la clase.
Nombre:	*getHtmlSignificadoTermino(idTermino, urlPlantilla)*
Descripción:	Devuelve el significado de un término dado. La salida es formateada a Html usando la plantilla dada.
Nombre:	*getHtmlSumarioTerminos(subcadena, criterio,urlPlantilla)*
Descripción:	Devuelve un listado con los términos que cumplan con los parámetros subcadena y criterio. La salida es formateada a Html usando la plantilla dada.

Nombre: BusinessArticulosInternosHelper	
Tipo de clase: control	
Para cada responsabilidad:	
Nombre:	*Instancia* ()
Descripción:	Retorna una única instancia de la clase.
Nombre:	*getHtmlSumarioArticulo(idTematica, urlPlantilla)*
Descripción:	Devuelve un sumario de artículos de una temática dada. La salida es formateada a Html usando la plantilla dada.
Nombre:	*getHtmlDetallesArticulo(idArticulo, urlPlantilla)*

Descripción:	Devuelve el html de un artículo determinado. La salida es formateada a Html usando la plantilla dada.

Nombre: HelperArticuloSalaPrensa	
Tipo de clase: control	
Atributo	**Tipo**
localArticulo	Articulo
Para cada responsabilidad:	
Nombre:	*Instancia (pArticulo)*
Descripción:	Retorna una única instancia de la clase. Recupera de la base de datos los campos del artículo dado y crea una instancia de localArticulo.
Nombre:	*getHtml (urlPlantilla)*
Descripción:	Encapsula la lógica de cómo parsear la plantilla para obtener una salida para la sala de prensa. Devuelve el html de localArticulo completo. La salida es formateada a Html usando la plantilla dada. El resultado de este método es usado por la clase BusinessArticulosInternosHelper.

Nombre: HelperArticuloSecciones	
Tipo de clase: control	
Atributo	**Tipo**
localArticulo	Articulo
Para cada responsabilidad:	
Nombre:	*Instancia (pArticulo)*
Descripción:	Retorna una única instancia de la clase. Recupera de la base de datos los campos del artículo dado y crea una instancia de localArticulo.

Nombre:	*getHtml (urlPlantilla)*
Descripción:	Encapsula la lógica de cómo parsear la plantilla para obtener una salida para las secciones. Devuelve el html de localArticulo completo. La salida es formateada a html usando la plantilla dada. El resultado de este método es usado por la clase BusinessArticulosInternosHelper.

Nombre: BusinessBibliotecaHelper	
Tipo de clase: control	
Para cada responsabilidad:	
Nombre:	*Instancia()*
Descripción:	Retorna una única instancia de la clase.
Nombre:	*getHtmlListarMateriales (tipo, urlPlantilla)*
Descripción:	Devuelve un html con el sumario de los materiales de la biblioteca del tipo dado. La salida es formateada a html usando la plantilla dada.
Nombre:	*getHtmlMaterial (urlPdf, urlPlantilla)*
Descripción:	Devuelve un html del pdf dado. La salida es formateada a html usando la plantilla dada.
Nombre:	*getHtmlMaterial(idArticulo, urlPlantilla)*
Descripción:	Devuelve un html del artículo dado. La salida es formateada a html usando la plantilla dada.

Nombre: HelperArticuloBiblioteca	
Tipo de clase: control	
Atributo	**Tipo**
localArticulo	Articulo
Para cada responsabilidad:	
Nombre:	*Instancia (pArticulo)*
Descripción:	Retorna una única instancia de la clase. Recupera de la base de datos los campos del artículo dado y crea una instancia de localArticulo.
Nombre:	*getHtml (urlPlantilla)*
Descripción:	Encapsula la lógica de cómo parsear la plantilla para obtener una salida para la sección Biblioteca. Devuelve el html de localArticulo completo. La salida es formateada a html usando la plantilla dada. El resultado de este método es usado por la clase BusinessBibliotecaHelper.

Nombre: BusinessGaleriaHelper	
Tipo de clase: control	
Para cada responsabilidad:	
Nombre:	*Instancia ()*
Descripción:	Retorna una única instancia de la clase.
Nombre:	*ejecutarVideoAudio (urlMultimedia, urlPlantilla)*
Descripción:	Devuelve el html necesario para ejecutar el material multimedia dado en el navegador. La salida es formateada a Html usando la plantilla dada.
Nombre:	*getHtmlFotoreportaje(idFotoreportaje, urlPlantilla)*
Descripción:	Devuelve el html necesario para mostrar un fotorreportaje. La salida es formateada a html usando la plantilla dada.

Nombre:	*getHtmlListadoMultimedia(tipo, urlPlantilla)*
Descripción:	Devuelve el html necesario para mostrar un sumario de los materiales multimedia del tipo dado. La salida es formateada a html usando la plantilla dada.

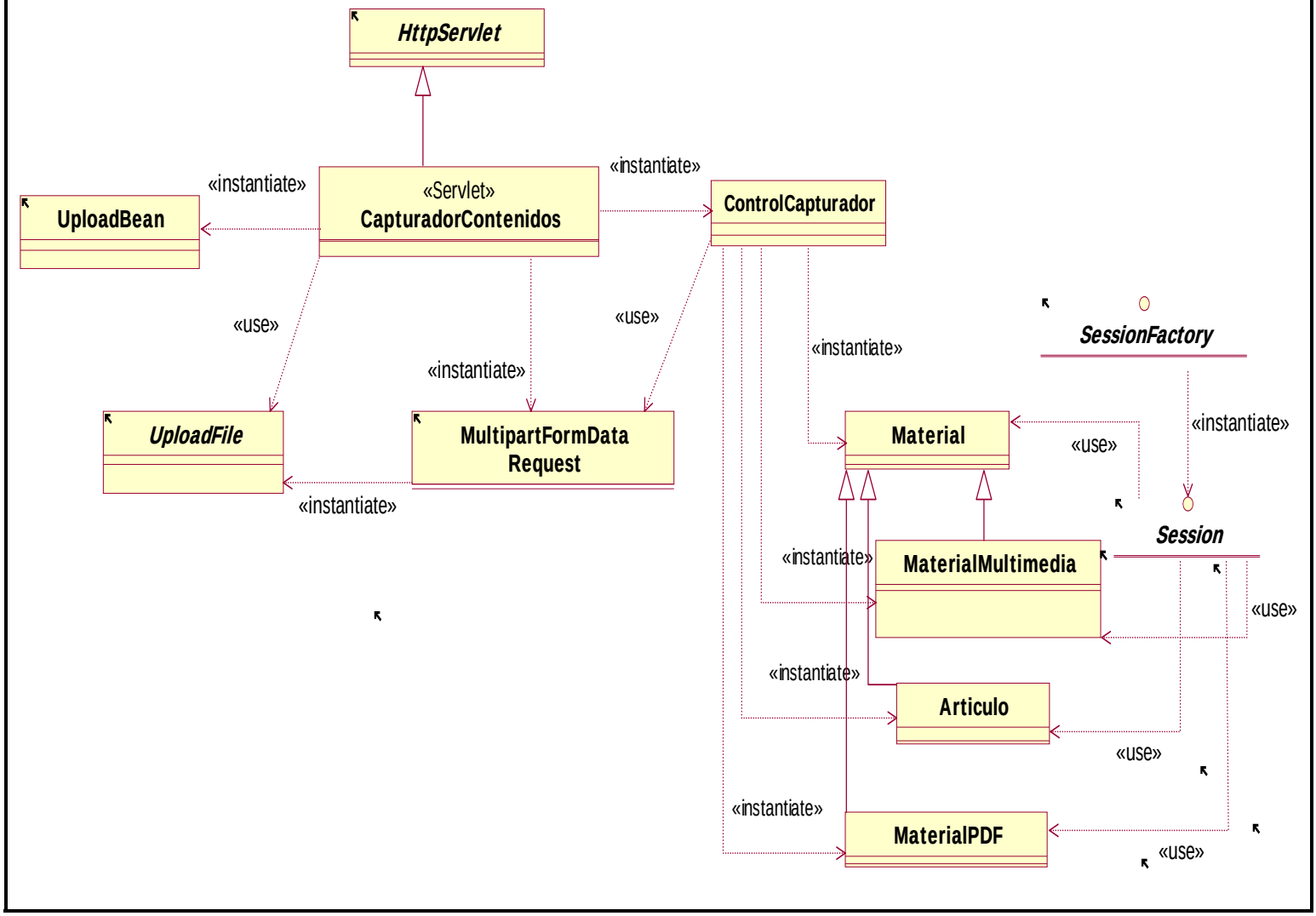
Nombre: BusinessMapasHelper	
Tipo de clase: control	
Para cada responsabilidad:	
Nombre:	*Instancia ()*
Descripción:	Retorna una única instancia de la clase.
Nombre:	*getHtmlDetallesMapas (urlMapa, urlPlantilla)*
Descripción:	Encapsula la lógica de cómo parsear la plantilla para obtener una salida para la sección Mapas. Devuelve el html necesario para mostrar un mapa dado. La salida es formateada a html usando la plantilla dada.
Nombre:	*getHtmlSumarioMapas(urlPlantilla)*
Descripción:	Devuelve el html necesario para mostrar un sumario de los mapas disponibles. La salida es formateada a html usando la plantilla dada.

Nombre: BusinessMapasHelper	
Tipo de clase: control	
Para cada responsabilidad:	
Nombre:	*Instancia (pArticulo)*
Descripción:	Retorna una única instancia de la clase.
Nombre:	*getHtmlDetallesMapas (urlMapa, urlPlantilla)*
Descripción:	Encapsula la lógica de cómo parsear la plantilla para obtener una salida para la sección Mapas. Devuelve el html necesario para mostrar un mapa dado. La salida es formateada a html usando la plantilla dada.

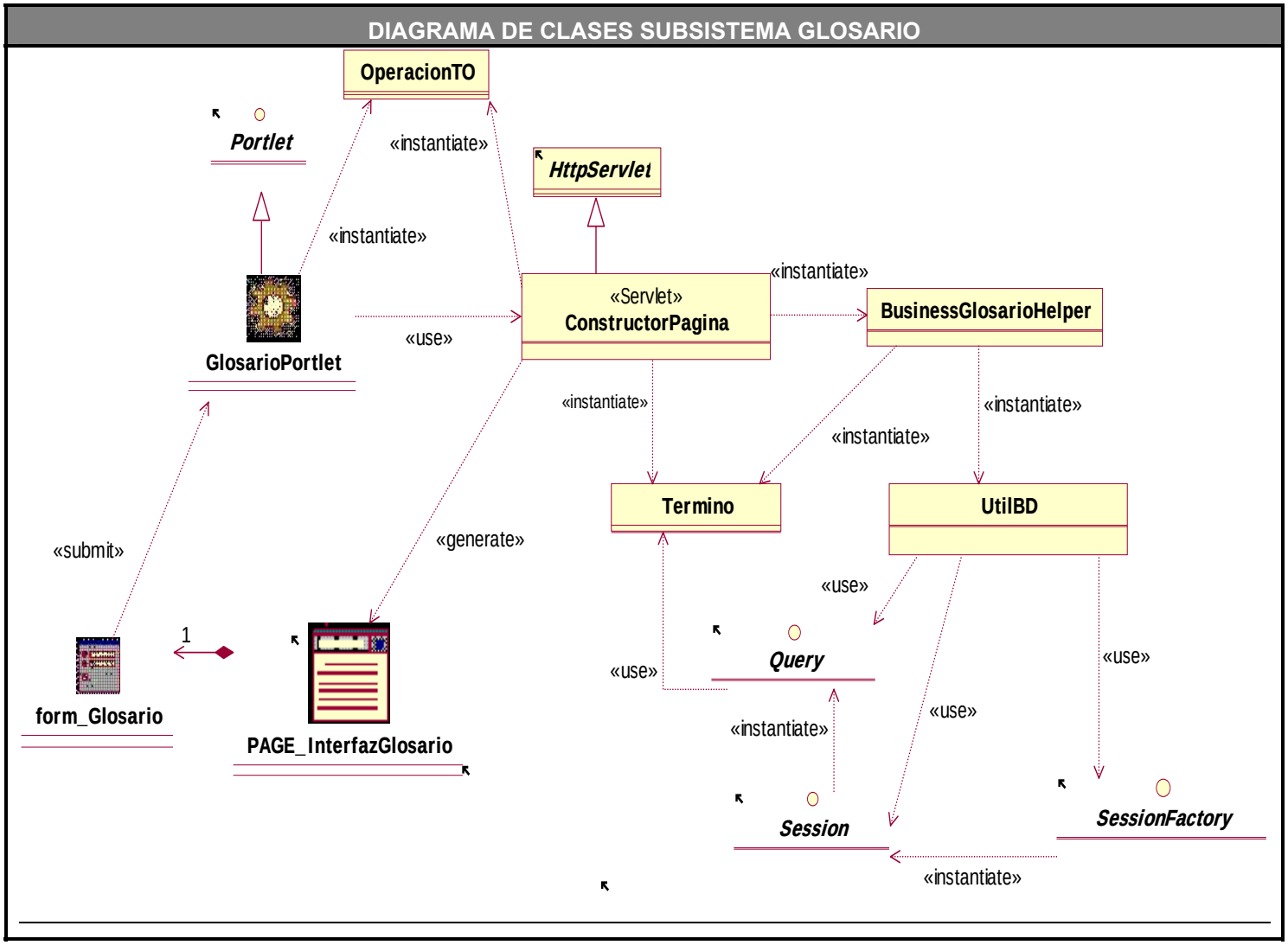
Nombre:	*getHtmlSumarioMapas(urlPlantilla)*
Descripción:	Devuelve el html necesario para mostrar un sumario de los mapas disponibles. La salida es formateada a html usando la plantilla dada.

4.2.4 Subsistema 1: Capturador de Contenidos.

DIAGRAMA DE CLASES DEL SUBSISTEMA CAPTURADOR DE CONTENIDO

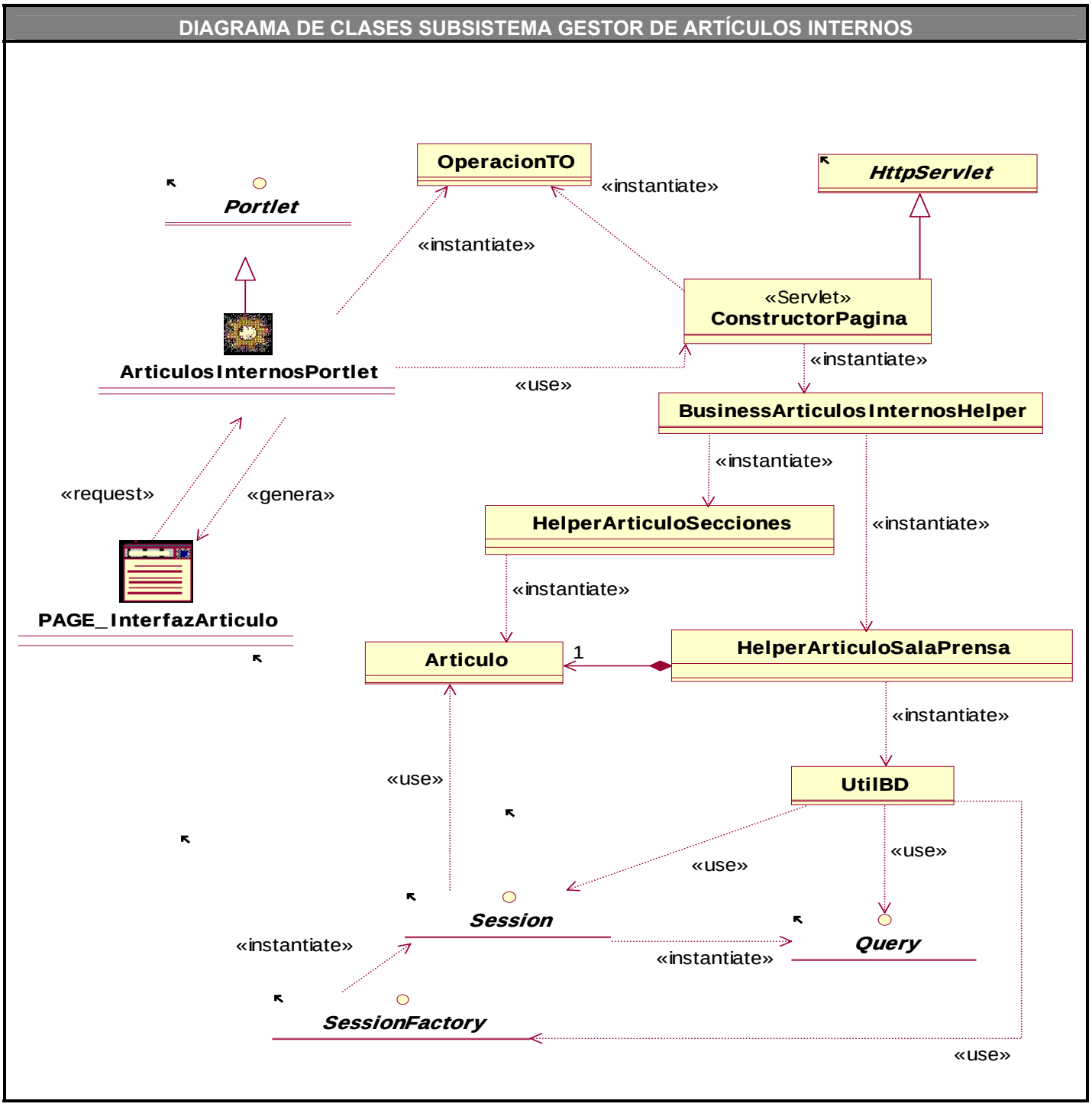


4.2.5 Subsistema 2: Glosario

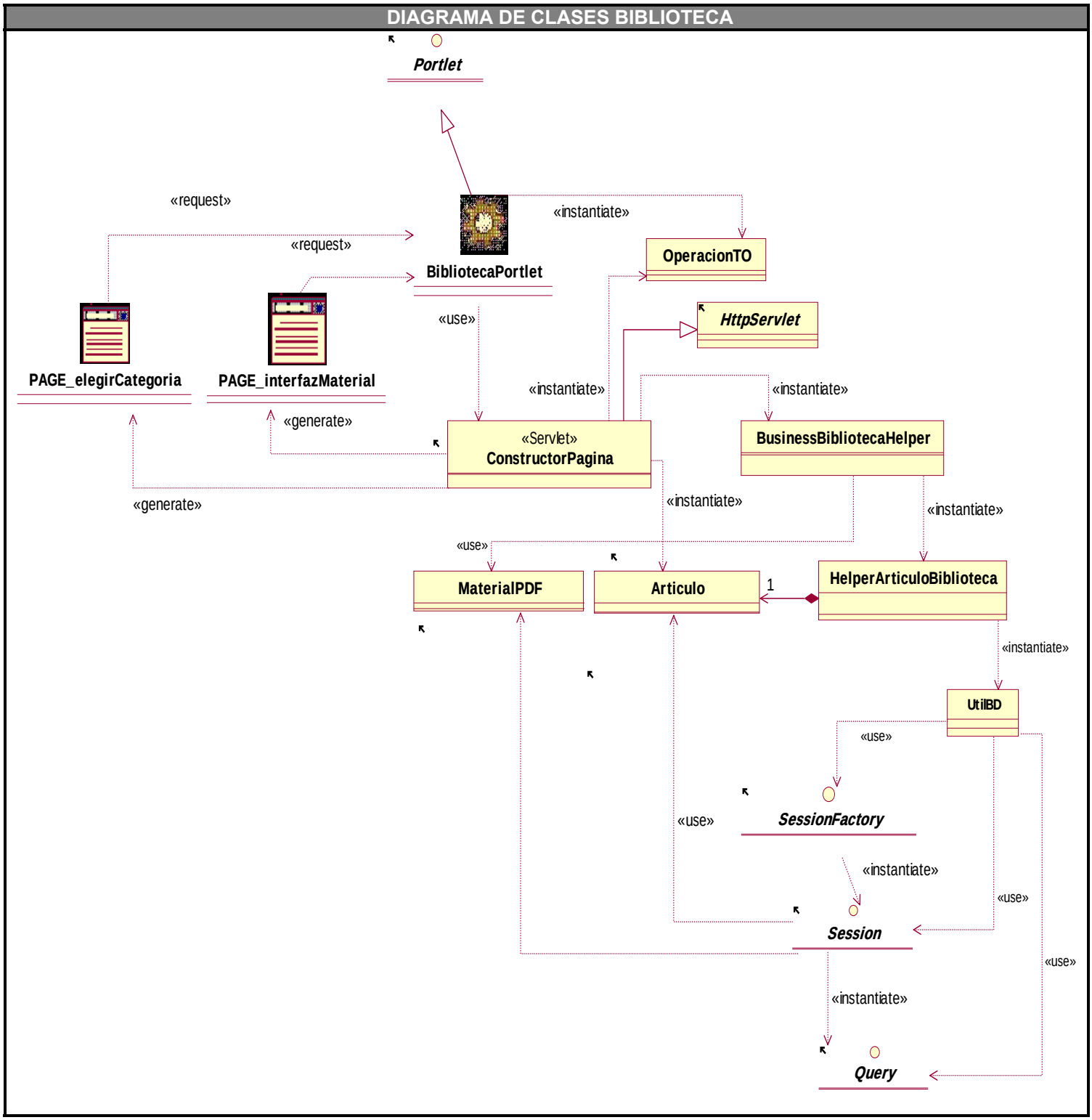


4.2.6 Subsistema 3: Gestor de artículos Internos

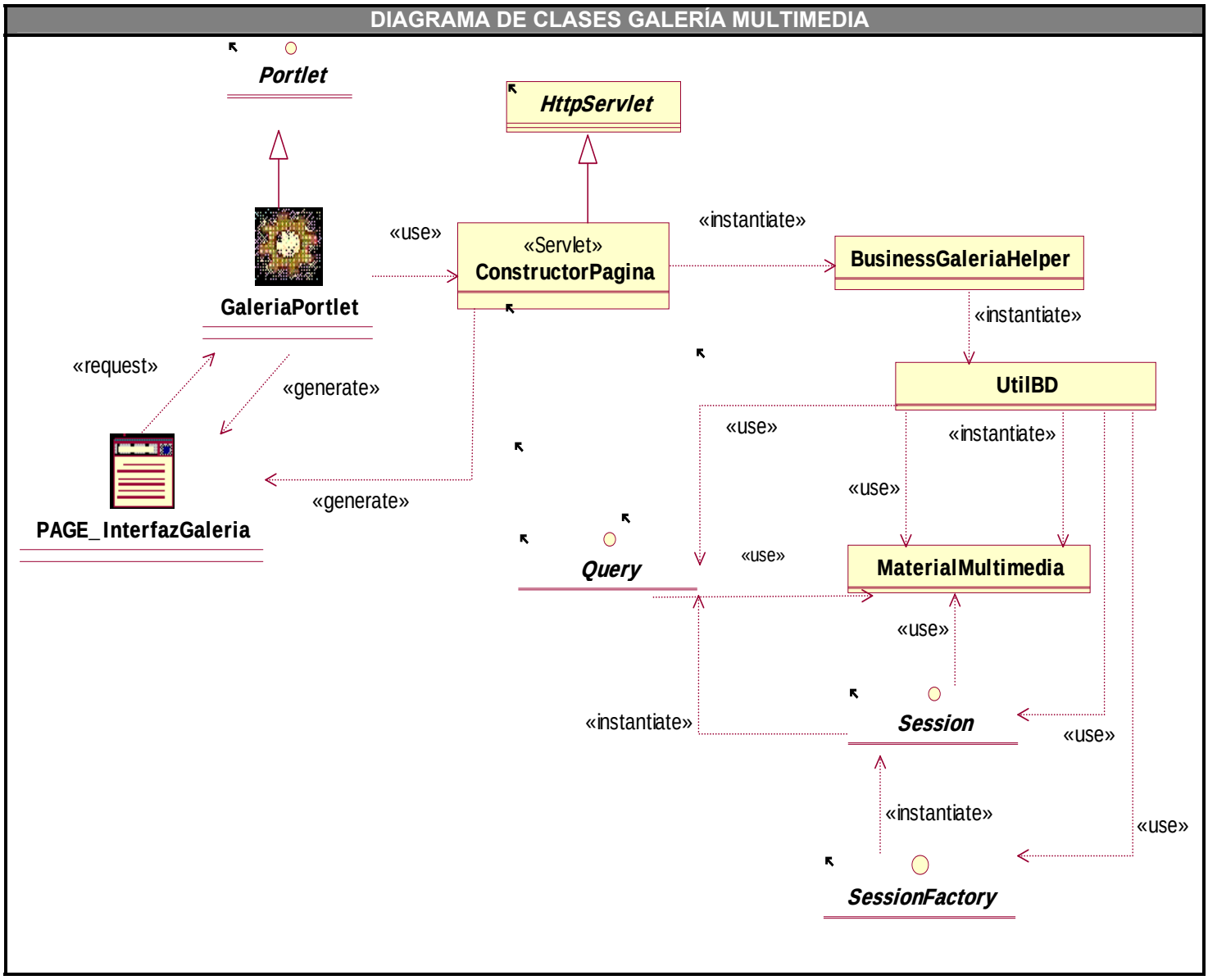
DIAGRAMA DE CLASES SUBSISTEMA GESTOR DE ARTÍCULOS INTERNOS



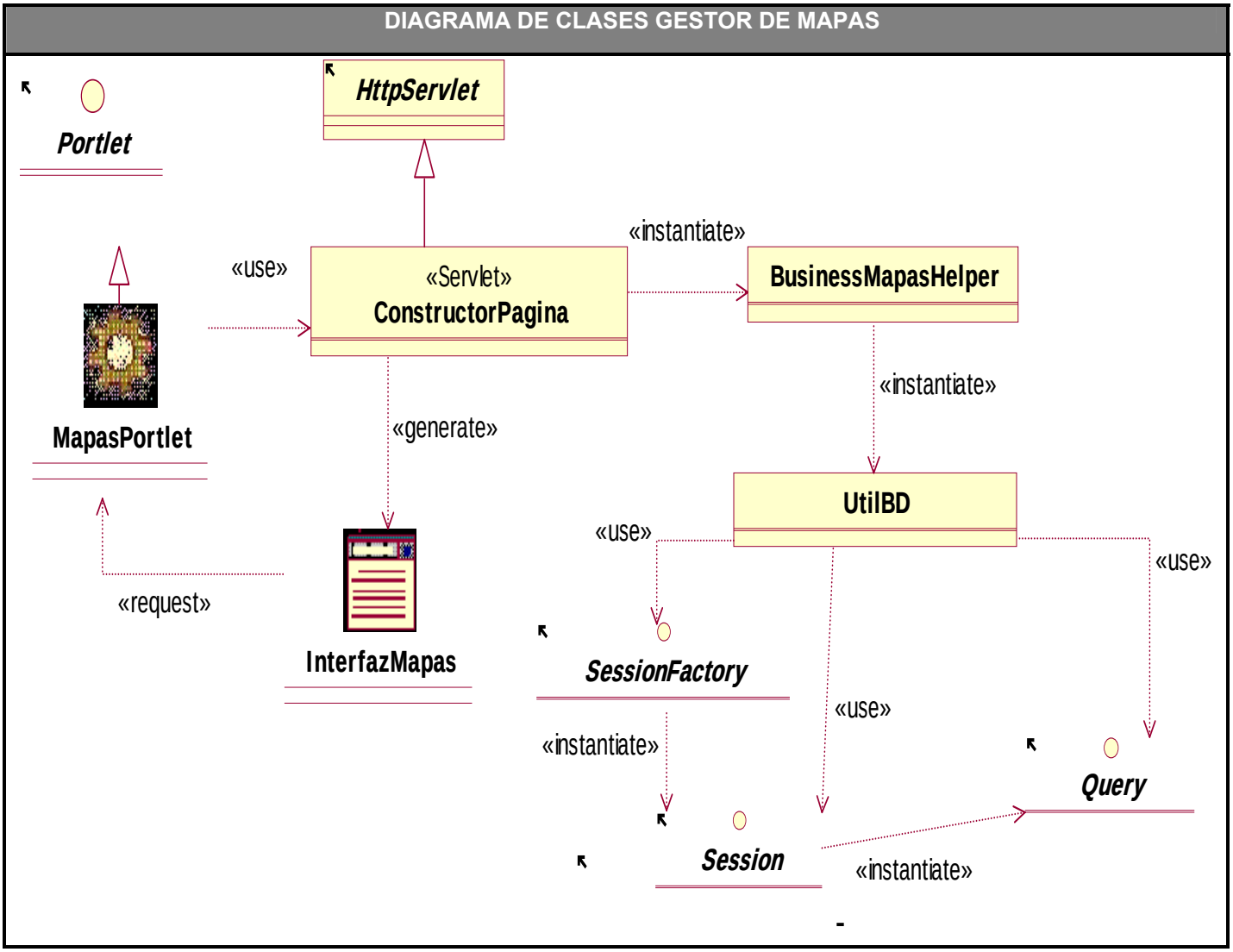
4.2.7 Subsistema 4: Biblioteca



4.2.8 Subsistema 5: Galería Multimedia



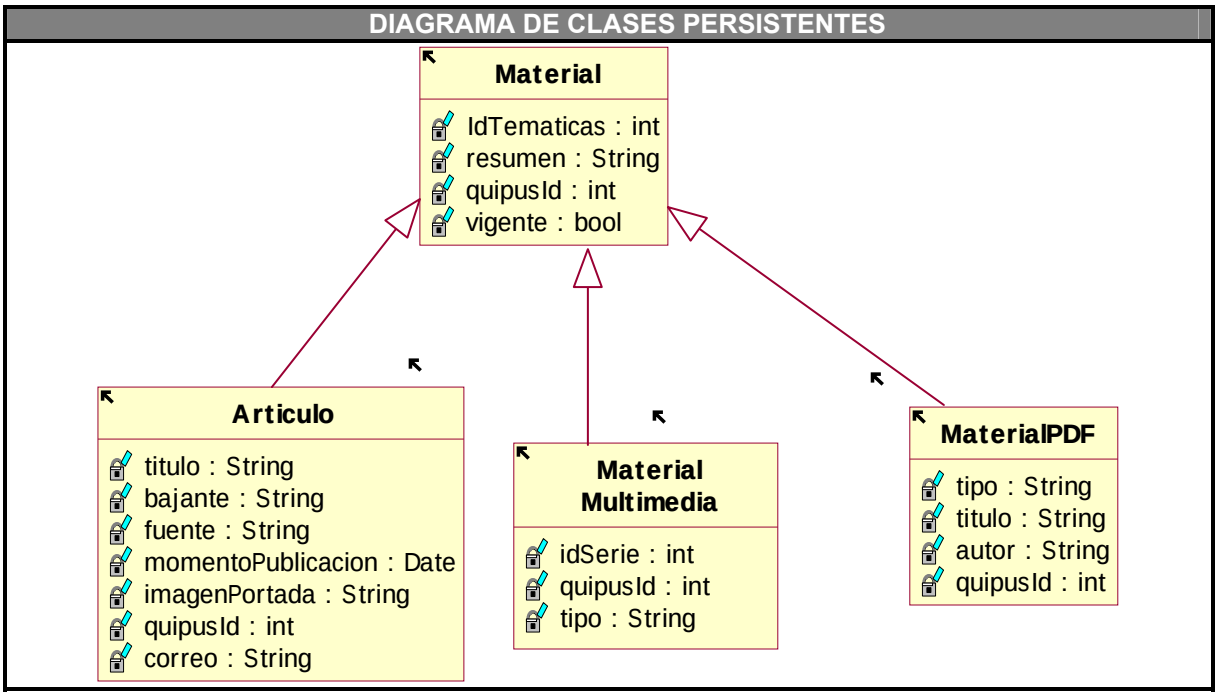
4.2.9 Subsistema 6. Gestor de Mapas



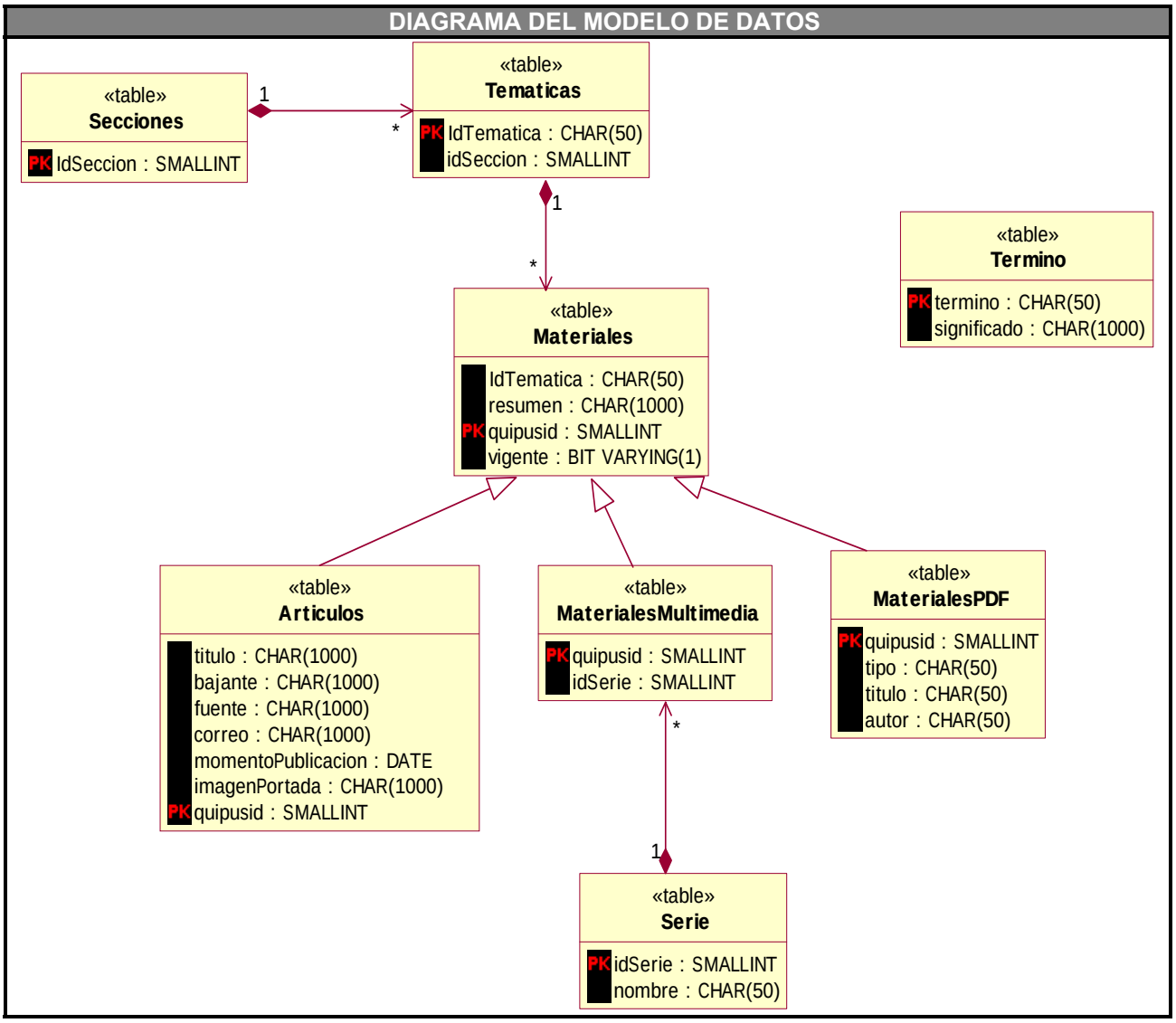
DISEÑO DE LA BASE DE DATOS.

A continuación se brinda el diagrama de clases persistentes y el modelo de datos que dan soporte al contenido manejado por el sistema. Es bueno aclarar que en el modelo de datos no se incluyen las entidades que son manejadas por el sistema servidor de portales Exoplatform, por ejemplo los usuarios.

4.2.10 Clases persistentes.



4.2.11 Diagrama del modelo de datos.



4.3 PRINCIPIOS DE DISEÑO GRÁFICO, MANEJO DE ERRORES Y ESTÁNDARES DE CODIFICACIÓN.

4.3.12 Principios de diseño

El diseño, sea cual sea el objeto del mismo, tiene que basarse en el usuario, y en este caso estamos hablando de cualquier persona en el mundo; donde la gran mayoría no cuentan con una preparación en las cuestiones de la informática.

Para ello, este sistema utiliza ciertos principios generales de diseño que garantizan la usabilidad en los diseños para las aplicaciones Web.

1. Principio de uso equiparable: donde las características de privacidad, garantía y seguridad estén igualmente disponibles para todos los usuarios, y que el diseño sea atractivo para todos los usuarios.
2. Principio de la flexibilidad: donde se ofrezcan posibilidades de elección en los métodos de uso, que facilite al usuario la exactitud y precisión, y se adapte al paso o ritmo del usuario.
3. Principio de la Información perceptible: donde se usen diferentes modos para presentar de manera redundante la información esencial (gráfica y verbal), se proporcione contraste suficiente entre la información esencial y sus alrededores, se amplíe la legibilidad de la información esencial, y que diferencie los elementos en formas que puedan ser descritas (por ejemplo, para las funciones de catalogación).

Principio de tolerancia al error: donde se dispongan los elementos para minimizar los riesgos y errores, por ejemplo utilizando elementos comunes; y los elementos peligrosos eliminados, aislados o tapados, que se proporcionen advertencias sobre peligros y errores. Hay que posibilitar el descubrimiento interactivo y el aprendizaje ensayo-error, y posibilitar la reversibilidad y la recuperabilidad de las acciones.

4.3.13 Manejo de Errores

El manejo de los errores esta a cargo de cada portlet, un portlet como dijimos es una aplicación web por lo tanto aquí se aplicarán las mismas prácticas que se

utilizan para el manejo de errores en cualquier aplicación J2EE en cada portlet. Algunos de los errores más frecuentes que pueden presentarse son aquellos relacionados con el gestor de bases de datos, pero estos son capturados internamente por cada portlet y pueden ser tratados de manera que lleguen al usuario de una forma amigable.

Exoplatform por su parte cuenta con un sistema de manejo de errores, por ejemplo cada vez que ocurre una excepción Exoplatform redirecciona al usuario a una página reportando donde ocurrió el error y detalles del mismo. Estos errores pueden ser causados por referenciar un portlet que halla sido borrado del servidor por citar un ejemplo.

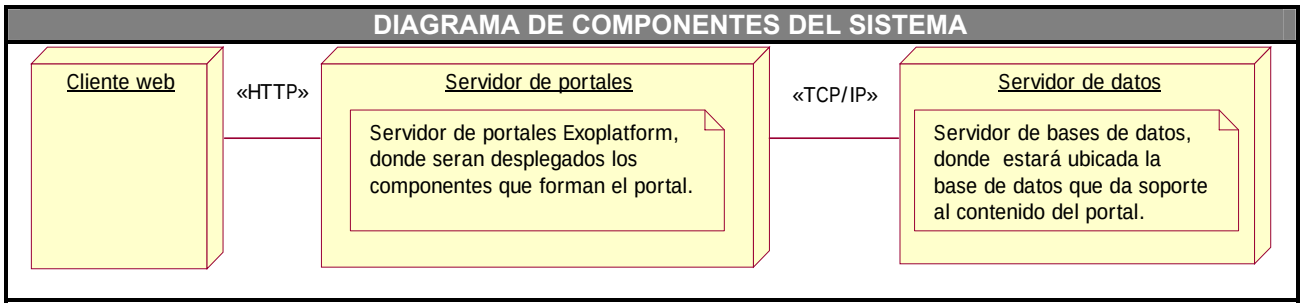
4.3.14 Estándares de codificación

Para obtener un código más claro y comprensible, y facilitar el mantenimiento del software deben seguirse los principios y prácticas expuestas en la convención de código de java (Java Code Convention). [16]

4.4 MODELO DE DESPLIEGUE

Un diagrama de despliegue muestra la configuración de los nodos que participan en la ejecución y los componentes que residen en ellos.

Esta vista de la arquitectura corresponde a la estructura física. La estructura física permite modelar la distribución de los componentes del sistema en los diferentes dispositivos físicos o de hardware de que se dispone.



4.5 MODELO DE IMPLEMENTACIÓN

El modelo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes. Describe también como se organizan y se relacionan unos con otros.

Primero se dará una vista global del sistema, a nivel de sistema los componentes que se modelan son portlets y en el caso del subsistema Capturador de Contenido se modela su servlet; luego se ofrece una vista a nivel de portlets.

La aproximación que se hace aquí de la arquitectura es a través de su estructura de uso. Se entiende por estructura de uso de la arquitectura las relaciones de dependencia que se establecen entre los componentes del sistema. [18]

A nivel de sistema la arquitectura del portal se aproxima a un estilo de capas, identificándose 2 capas, la capa de portlets y la capa de clases comunes.

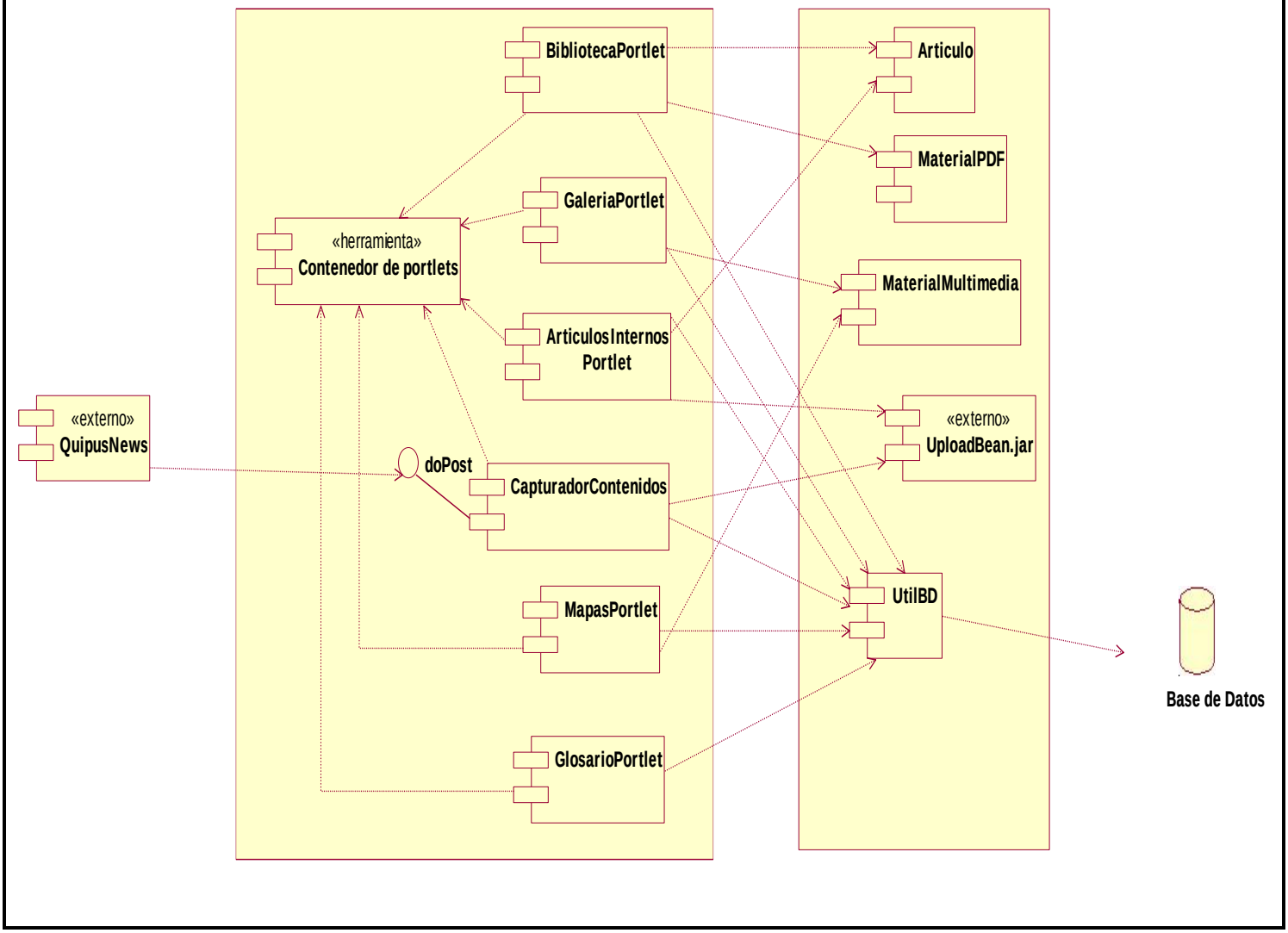
En la capa de portlets se ubican los portlets que conforman el portal y el servlet que sirve de enlace con QuipusNews y en la capa de clases comunes se ubican todas aquellas clases que son usadas por varios portlets, así como ficheros tipo jar desarrollados por terceros que utilizamos en la aplicación, por ejemplo UploadBean.jar, el cual contiene los objetos MultipartFormDataRequest, UploadFile, UploadBean, entre otros que son usados por el subsistema Capturador de Contenido para obtener los datos que vienen en el request y persistirlos en el sistema de archivo.

De ninguna manera esto supone crear dependencia entre los portlets al usar clases comunes, ni afectará su reutilización como componente independiente en otras soluciones puesto que se aprovecharán las ventajas que nos brinda el Eclipse. En el momento de desplegar el portlet en el servidor el Eclipse da la posibilidad de elegir las clases y librerías que se quieren desplegar, se desplegaran solo aquellas que no pertenezcan a las clases comunes, pero el código del portlet con todas sus clases y librerías estará íntegro en el espacio de trabajo de cada programador, listo para ser desplegado en cualquier otro servidor.

Como se ha explicado anteriormente los portlets deberán tener un intercambio mínimo, un portlet además de encapsular funcionalidades del portal que generan contenido al usuario constituye también un componente de presentación del portal puesto que genera su propia interfaz, por lo tanto a la hora de diseñar una arquitectura para un portal usando la tecnología de portlets ésta debe permitir que un portlet sea eliminado sin que ello afecte el funcionamiento del resto. La propuesta que se hace permite mantener el bajo acoplamiento que debe existir entre los portlets.

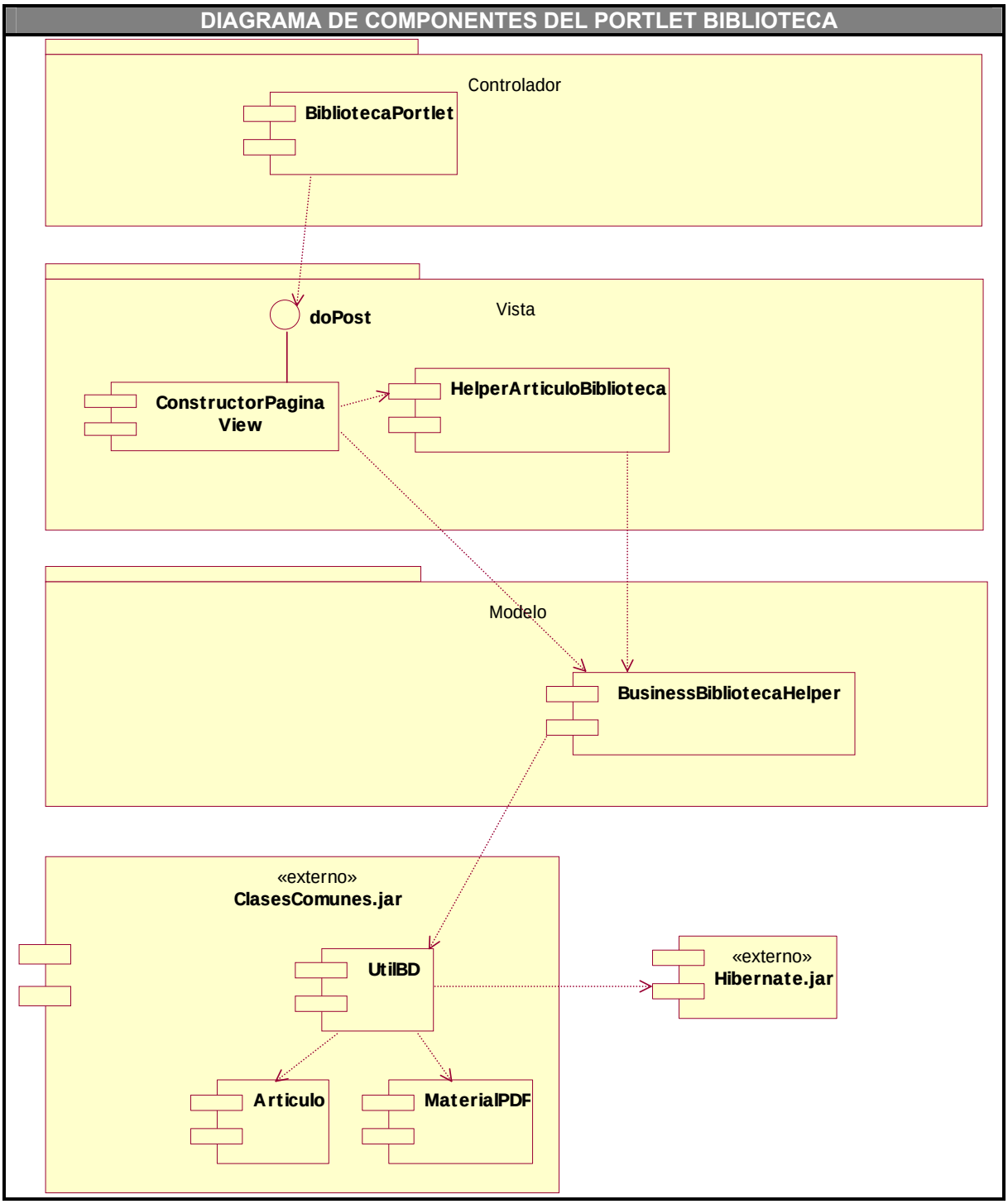
A continuación se muestran el diagrama de componentes del portal y de uno de sus portlets.

DIAGRAMA DE COMPONENTES DEL SISTEMA



4.5.15 Diagramas de componentes a nivel de portlets.

Biblioteca



Para no cargar el documento con infinidad de diagramas, el resto de los diagramas se dejan para los anexos. Por cada anexo se incluirá el diagrama de secuencia y de componente de cada subsistema.

4.6 CONCLUSIONES

En este capítulo se dio una vista de la arquitectura mediante su estructura de clases y su estructura de uso. Se describe como se relacionan los componentes que forman la arquitectura y como es su distribución física y lógica.

CAPÍTULO V

ESTUDIO DE FACTIBILIDAD

5.1 INTRODUCCIÓN

A la hora de enfrentar un proyecto software se han de tener en cuenta diferentes factores como el ámbito del trabajo a realizar, los recursos requeridos, las tareas a ejecutar, las referencias a tener en cuenta, el esfuerzo. Para intentar dar solución a estos problemas se han introducido en la Ingeniería del Software una serie de técnicas, utilizadas dentro de las tareas de planificación, que ayudan a planificar y controlar el esfuerzo y el tiempo necesario de desarrollo. Para realizar estas estimaciones utilizamos el modelo de COCOMO (*COnstructive COst MOdel*).

5.2 PLANIFICACIÓN

Entradas externas

Nombre de la entrada externa	Cantidad Ficheros	Cantidad Elementos Datos	Clasificación
Recibir MaterialPDF	1	8	Simple
Recibir MaterialMultimedia	1	7	Simple
Recibir Articulo	1	12	Simple

Peticiones.

Nombre de la petición	Cantidad Ficheros	Cantidad Elementos Datos	Clasificación
Obtener Articulo	1	12	Simple
Obtener Termino	1	2	Simple
Obtener MaterialPDF	1	8	Simple
Obtener MaterialMultimedia	1	7	Simple
Lista de Terminos	1	2	Simple
Lista de Articulos	1	12	Simple
Lista de MaterialPDF	1	8	Simple
Lista de MaterialMultimedia	1	7	Simple

Archivos lógicos internos.

Nombre del fichero interno	Cantidad Registros	Cantidad Elementos Datos	Clasificación
Termino	1	2	Simple
MaterialPDF	1	8	Simple
MaterialMultimedia	1	7	Simple
Articulo	1	12	Simple

Puntos de función desajustados.

Elementos	Simple		Media		Compleja		Subtotal puntos de función
	No	x Peso	No	x Peso	No	x Peso	
Ficheros lógicos internos	4	7	0	10	0	15	28
Entradas externas	3	3	0	4	0	6	9
Peticiones	8	3	1	4	0	6	24
Total (UFP)							61

Líneas de instrucciones fuentes.

Características	Valor
Puntos de función desajustados	61
Lenguaje(s)	Java
Instrucciones fuentes por puntos de función	63
Instrucciones fuentes por lenguaje (miles de instrucciones)	3843
Instrucciones fuentes (miles de instrucciones)	3,843

Factores de escala.

Factor	Clasificación	Valor
PREC	Nominal	3,72
FLEX	Muy Alto	1,01
TEAM	Muy Alto	1,10
RESL	Nominal	4,24
PMAT	Bajo	6,24
Suma		**16,31**

Multiplicadores de esfuerzo.

Multiplicador	Clasificación	Valor
RELY	Nominal	1,00
DATA	Nominal	1,00
CPLX	Nominal	1,00
RUSE	Nominal	1,00
DOCU	Alta	1,11
PERS	Nominal	1,00
SCED	Nominal	1,00
Producto		**1,11**

La ecuación que plantea COCOMO para calcular el esfuerzo de desarrollo es la siguiente:

$$PM = A \times Size^{E} \times \prod_{i=1}^{n} EM_i$$

donde $E = B + 0.01 \times \sum_{j=1}^{5} SF_j$

E = 0,91 + 0,01 * 16,31 = 1.0731

PM = 2,94 * $(3,843)^{1,0731}$ * 1.11 = 13,83 **≈ 14 Personas – Mes**

5.3 BENEFICIOS TANGIBLES E INTANGIBLES

5.3.1 Ventajas de la arquitectura propuesta.

Con la arquitectura que aquí se describe no solo logramos satisfacer los requerimientos presentados sino que logramos un importante plus en cuanto al rendimiento del portal, puesto que el contenido estará local y los portlets lo gestionarán directamente, sino que se gana en disponibilidad al no depender del contenido expuesto por un tercero.

Por otro lado el desarrollo se agiliza puesto que es mucho más fácil e intuitivo distribuir el trabajo entre los desarrolladores.

5.3.2 Aportes a la dirección del proyecto.

Con el presente trabajo la dirección del proyecto cuenta con una referencia para enfrentar el desarrollo del portal con argumentos sólidos. Se ha definido la

arquitectura del sistema y se ha descrito la misma, además se ha hecho un estimado del esfuerzo necesario para realizarlo.

Estas bases permitirán la construcción del portal de PDVSA de manera más rápida y eficiente, lo que se traducirá en un importante avance para PDVSA permitiéndole multiplicar el mensaje de la empresa y la nueva nación venezolana haciéndolo accesible a internautas en cualquier parte del mundo.

5.4 ANÁLISIS DE COSTOS Y BENEFICIOS

El desarrollo de este sistema no supone grandes gastos de recursos, ni tampoco de tiempo. Las tecnologías y herramientas utilizadas para el desarrollo del mismo son totalmente libres, por lo que no hay que incurrir en gastos por pago de licencias de uso. El sistema es completamente portable, por lo que un cambio de plataforma es viable y factible, es completamente escalable gracias a la estructura basada en portlets de los procesos del negocio que se diseñaron.

5.5 CONCLUCIONES

Luego del estudio realizado, a partir del análisis de los costos y beneficios, se llego a la conclusión que es factible el hecho de desarrollar portal, por todos los aportes esperados.

A partir del análisis de los beneficios, tanto tangibles como intangibles, se concluye que es factible el desarrollo de la aplicación a partir de la arquitectura descrita.

CONCLUSIONES

La información es un elemento fundamental para el desarrollo, con el decursar de los años, la gestión de la información ocupa, cada vez más, un espacio mayor en el desarrollo de los países a escala mundial. Debemos gestionar y hacer el mejor uso posible de esa información, la cual debe ser relevante para alguien en un momento determinado, de lo contrario de poco nos servirá. Cada día son más y más las personas que pueden acceder a la Internet desde cualquier parte del mundo.

En los últimos años, países como Venezuela han combatido la influencia imperialista neoliberal, llevando a cabo los ideales y sueños del Libertador y todos aquellos hombres que lucharon por la libertad de los pueblos de América, por una América libre.

Una de las insignias de Venezuela es PDVSA, es el brazo económico de los nuevos tiempos que corren en esa hermana nación, promover y consolidar la posición de PDVSA en el mundo económico actual es promover y consolidar a Venezuela. El portal de PDVSA es una pieza estratégica fundamental en ese camino.

Para desarrollar un sistema que cumpla con la mayoría de los parámetros de calidad que esperan sus usuarios es necesario que se diseñe una arquitectura que lo permita, esto ha hecho que la arquitectura haya adquirido una importancia sobresaliente en el desarrollo de software en la actualidad.

El presente trabajo se centra en hacer una propuesta de arquitectura para desarrollar el portal de PDVSA, esta propuesta se ha basado en el análisis y la investigación sobre la arquitectura de software, las nuevas tecnologías para el desarrollo de portales corporativos, así como las tendencias que rigen este campo de la informática en la actualidad. La propuesta se ha hecho siguiendo la metodología RUP, y se ha basado en la experiencia recogida en patrones establecidos.

Como resultado se propone una arquitectura que hace posible satisfacer los requerimientos funcionales y no funcionales que aquí se han detallado. Además se describió e implementó un mecanismo para lograr usar QuipusNews como sistema de publicación de contenidos para Exoplatform.

Por tanto se puede concluir que los objetivos propuestos para el presente trabajo han sido cumplidos satisfactoriamente.

RECOMENDACIONES

Como se ha observado los objetivos trazados para este trabajo han sido logrados, sin embargo la propuesta es sólo la primera fase de un proyecto que puede ser mucho más ambicioso. Por lo que hacemos las siguientes recomendaciones:

Profundizar en el estudio de otro servidor de portales como Liferay o Pluto y comunicar al cliente la conveniencia de de su uso si así fuera el caso.

Crear un equipo para desarrollar componentes portlets con vistas a ser usados en futuras o inmediatas aplicaciones.

BIBLIOGRAFÍA

[1]. Maldonado, Roberto C. *"Historia del World Wide Web"*. http://www.x-extrainternet.com/www2.asp.(14/10/2005).

[2].*"¿Qué es el WWW?"*.http://www.x-extrainternet.com/www.asp.(17/10/2005)

[3]. Guerrero, Luis A. "*Modelando aplicaciones Web con UML*". www.dcc.uchile.cl/~luguerre/cc61j/recursos/web-app.ppt.(17/10/2005)

[4]. García, Juan Carlos., Saorín, Tomás."*Los Portales de Internet*". http://www.um.es/gtiweb/curso/seis.htm.(15/10/2005)

[5]. "*Web Services for Remote Portlets*". http://www.oasis-open.org/committees/wsrp

[6]. Bellas, P., Fernando. "*Construcción de portales*". www.w3c.es/gira/paradas/presentaciones/ ConstruccionPortales-GiraW3C-24-11-2004.pdf

[7]. Ramakrishna, Srikanth. Sarathy, Sanjay. "*Strategies for Integrating J2EE-Based Applications into a Portal Server Environment".*http://www.developer.com/java/ent/article.php/10933_3346861_1

[8]. *"Programación en el lenguaje JAVA*".http://www.sc.ehu.es/sbweb/fisica/cursoJava/Intro.htm.

[9]. Ruiz, Cristina, José. "*La nueva generación de portales: de la información a la colaboración*".

[10]. "*Portlets*". http://www.atsistemas.com/portlets.html

[11]. Booch., Rumbaugh., Jacobson. "*El Proceso Unificado de Software*". Edición en español. Madrid, 2000.

[12]. "*Tutorial de UML*". ttp://www.dcc.uchile.cl/~psalinas/uml/introduccion.html

[13]. "*Tutorial de Desarrollo Orientado a Objetos con UML* ". http://www.clikear.com/manuales/uml/

[14]. "*Eclipse (Computación)*". http://es.wikipedia.org/wiki/Eclipse_(computaci%C3%B3n)

[15]. "*¿Qué es Postgre*?". http://uca.guegue.com/postgreSQL/principal.htm

[16]. Java Code Conventions. Sun Microsystems, Inc, 1997.

[17]. “Usabilidad y arquitectura del software”.
www.desarrolloweb.com/articulos/1622.php?manual=5.pdf

[18]. “Representación de la arquitectura de software usando UML”.
www.icesi.edu.co/es/publicaciones/publicaciones/contenidos/sistemas_telematica/1/shurtado_repres-uml.pdf

[19]. Un lenguaje para la especificación y validación de arquitecturas de software.
www.lcc.uma.es/~canal/papers/tesis-//canal_tesis.pdf

[20]. “La importancia de la arquitectura en el desarrollo de software de calidad”.
http://www.eafit.edu.co/NR/rdonlyres/223A8F47-27B5-4EB8-B695-4097F745D701/0/Arquitectura.pdf

[21]. “Patrones y Antipatrones: una Introducción - Parte II
”.http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/mtj_3317.asp#M20

[22]. “Diseño de aplicaciones Internet usuando los patrones de diseño J2EE”.
www.moisesdaniel.com/es/wri/disaplj2ee.pdf

[23]. “Catálogo de Patrones de Diseño J2EE. I.- Capa de Presentación”.
http://www.programacion.com/java/tutorial/patrones/5/

GLOSARIO DE TÉRMINOS

Pagina portal (portal page): Se refiere a una página obtenida como resultado de la agregación de fragmentos HTML generados por portlets, estos fragmentos son procesados por un contenedor de portlets, el cual genera la página final.

CSS: (Hoja de Estilo en Cascada) Dentro del diseño de páginas de Internet se presenta esta como la vanguardia en cuanto a definición de estilos dentro de las plantillas de diseño. A través de instrucciones en código HTML se definen los estándares del conjunto de páginas que conforman el proyecto. La meta es uniformizar el diseño.

CGI: (Interfaz Común de Pasarela) Es un protocolo o interfaz de intercambio de información que se realiza entre el navegador del usuario y un servidor WWW.

Código Abierto: Es una tendencia internacional del desarrollo de software que profesa la distribución del código junto a las aplicaciones, se rigen por licencias tales como GNU/GPL.

HTTP: Es el conjunto de reglas para intercambiar archivos (texto, gráfica, imágenes, sonido, video y otros archivos multimedia) en la World Wide Web.

IDE: Ambiente de desarrollo integrado. Es como se le llama al ambiente que proporciona al usuario una determinada herramienta de desarrollo.

Microsoft: Compañía de software más grande del mundo. Fue fundada en 1975 por Paul Allen y Bill Gates. Aunque también se conoce por sus lenguajes de programación y aplicaciones para computadores personales, el éxito sobresaliente de Microsoft se debe a sus sistemas operativos DOS y Windows.

Unix: Sistema operativo atribuido a Ken Thompson y comercializado por la empresa ATT en la década de los 70s que alcanzó mucho éxito, sobretodo en las universidades y posteriormente en las empresas. Entre sus principales características tenemos que es: portable, robusto, flexible y abierto, actualmente goza de gran popularidad dentro de la tecnología de Internet.

Web Services: aplicación simple que realiza un cometido y que puede formar parte de otros servicios para formar un servicio más completo.

Sun Microsystems: es una empresa informática del Silicon Valley, fabricante de semiconductores y software. Su producto más notable es el lenguaje de programación Java.

J2EE: son las siglas de Java 2 Enterprise Edition que es la edición empresarial del paquete Java creada y distribuida por Sun Microsystems. Comprenden un conjunto de especificaciones y funcionalidades orientadas al desarrollo de aplicaciones empresariales.

ANEXOS

11.1 Anexo 1

11.1.1 Subsistema Glosario.

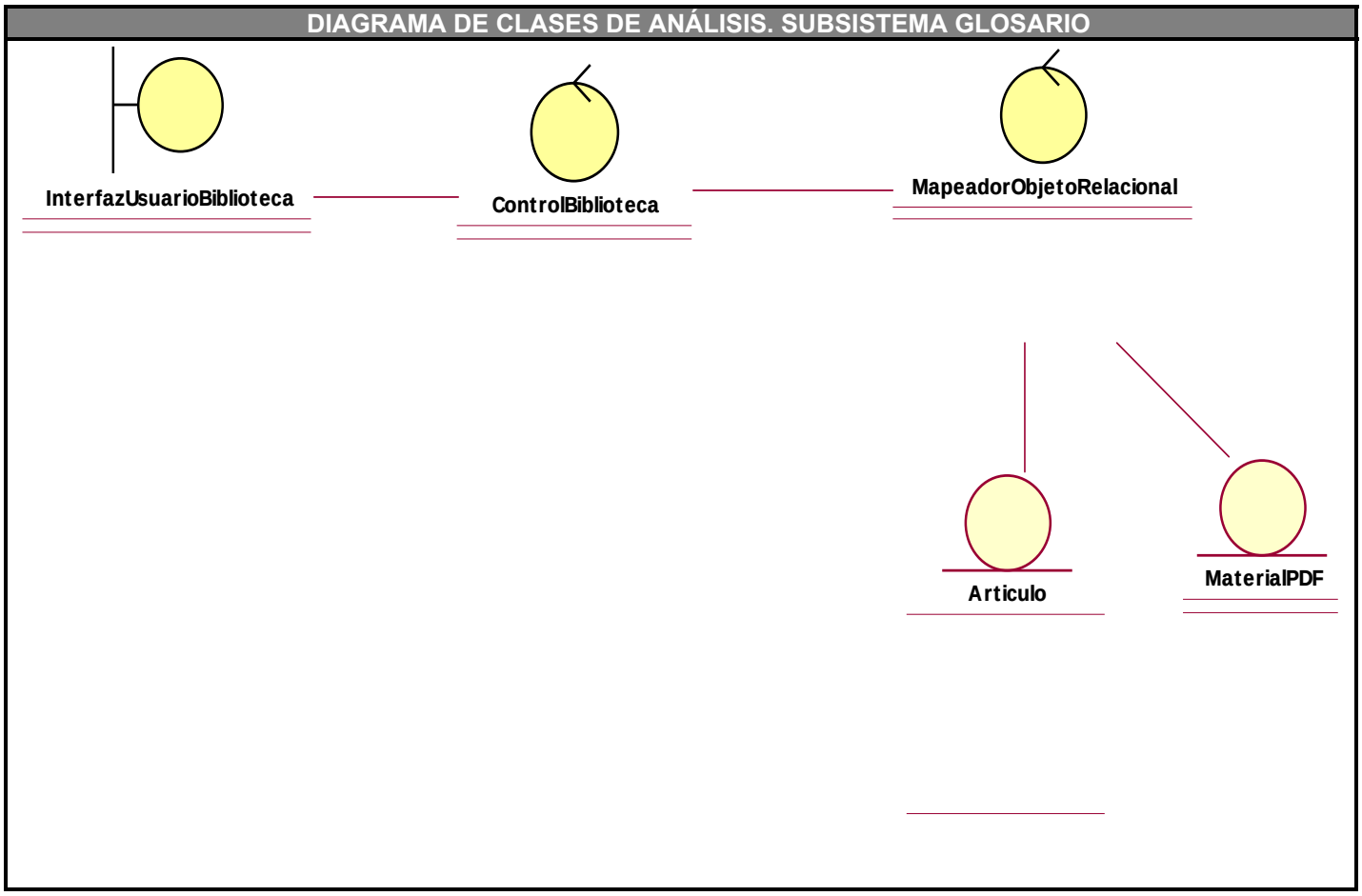
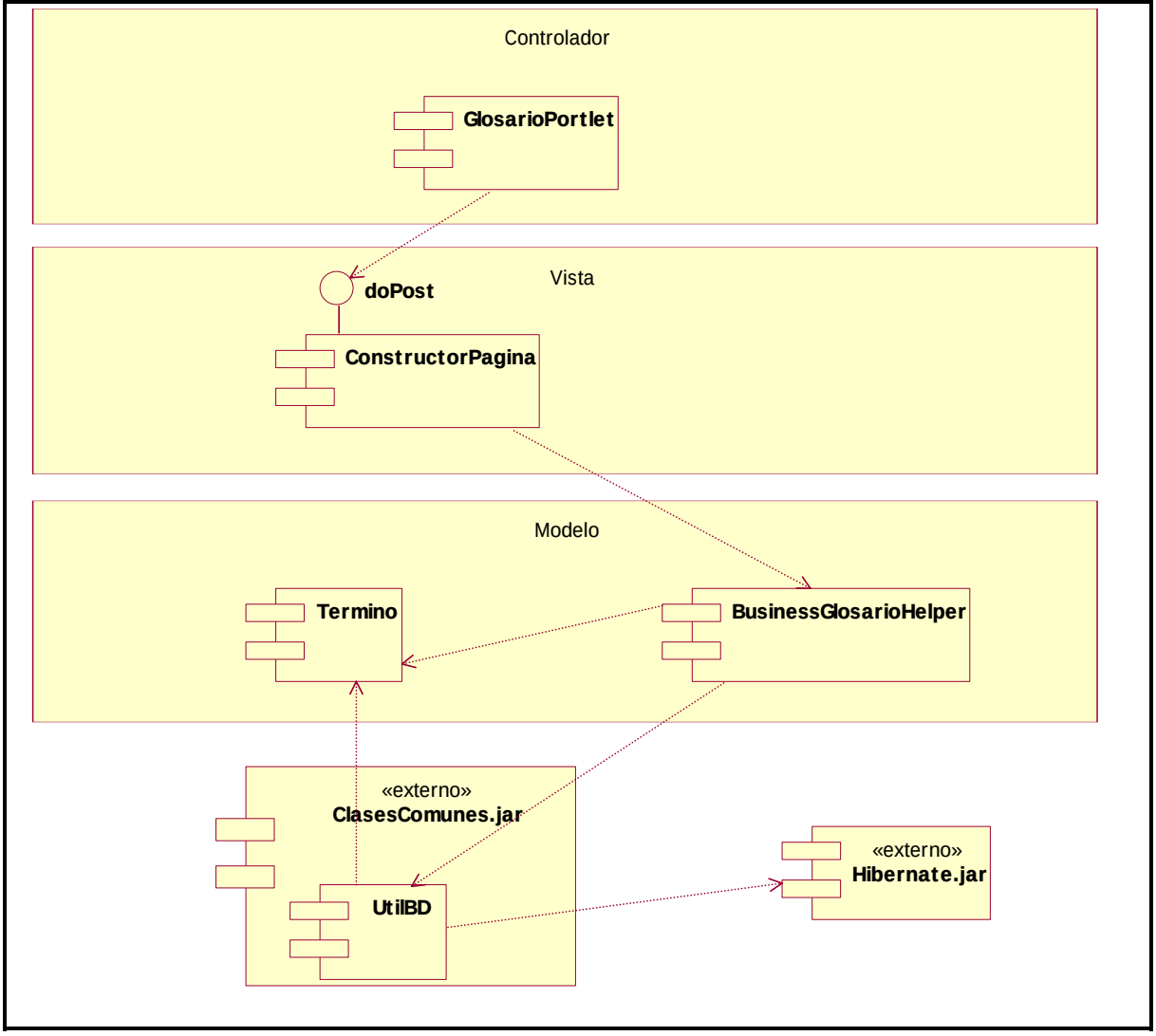


DIAGRAMA DE COMPONENTES. SUBSISTEMA GLOSARIO



11.2 Anexo 2

11.2.2 Subsistema Gestor de Artículos Internos.

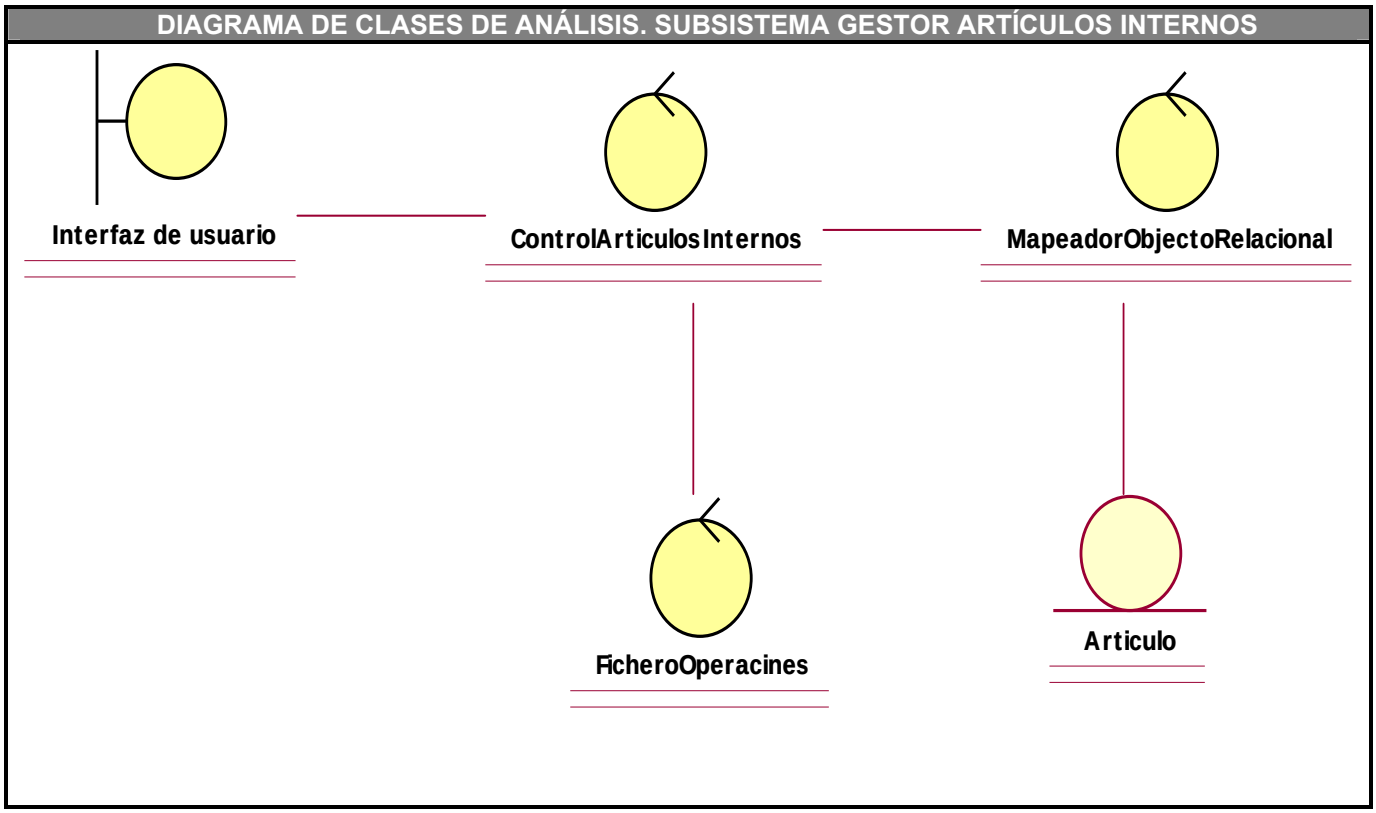
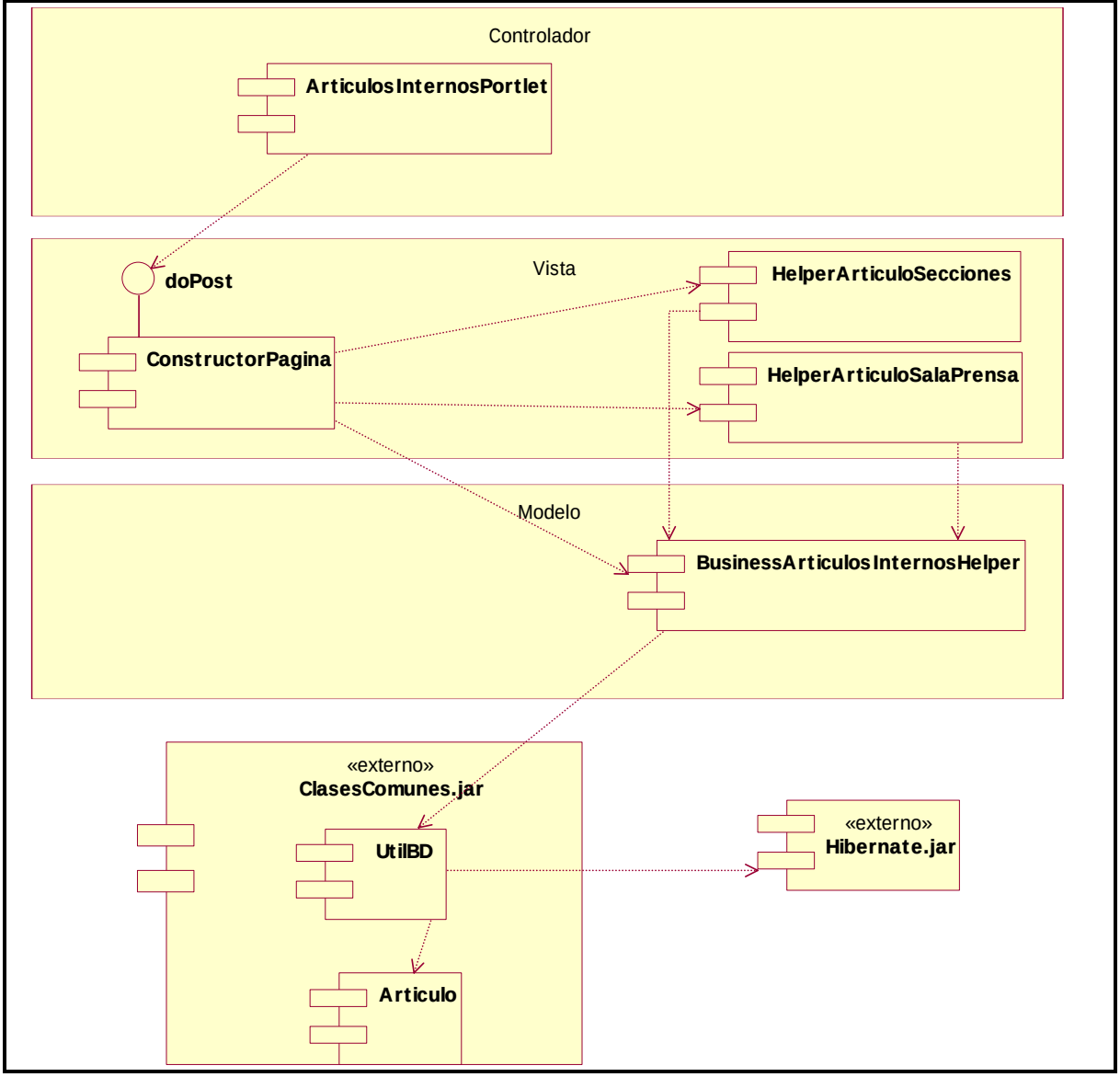


DIAGRAMA DE COMPONENTES. SUBSISTEMA GESTOR ARTÍCULOS INTERNOS



11.3 Anexo 3

11.3.3 Subsistema Galería Multimedia.

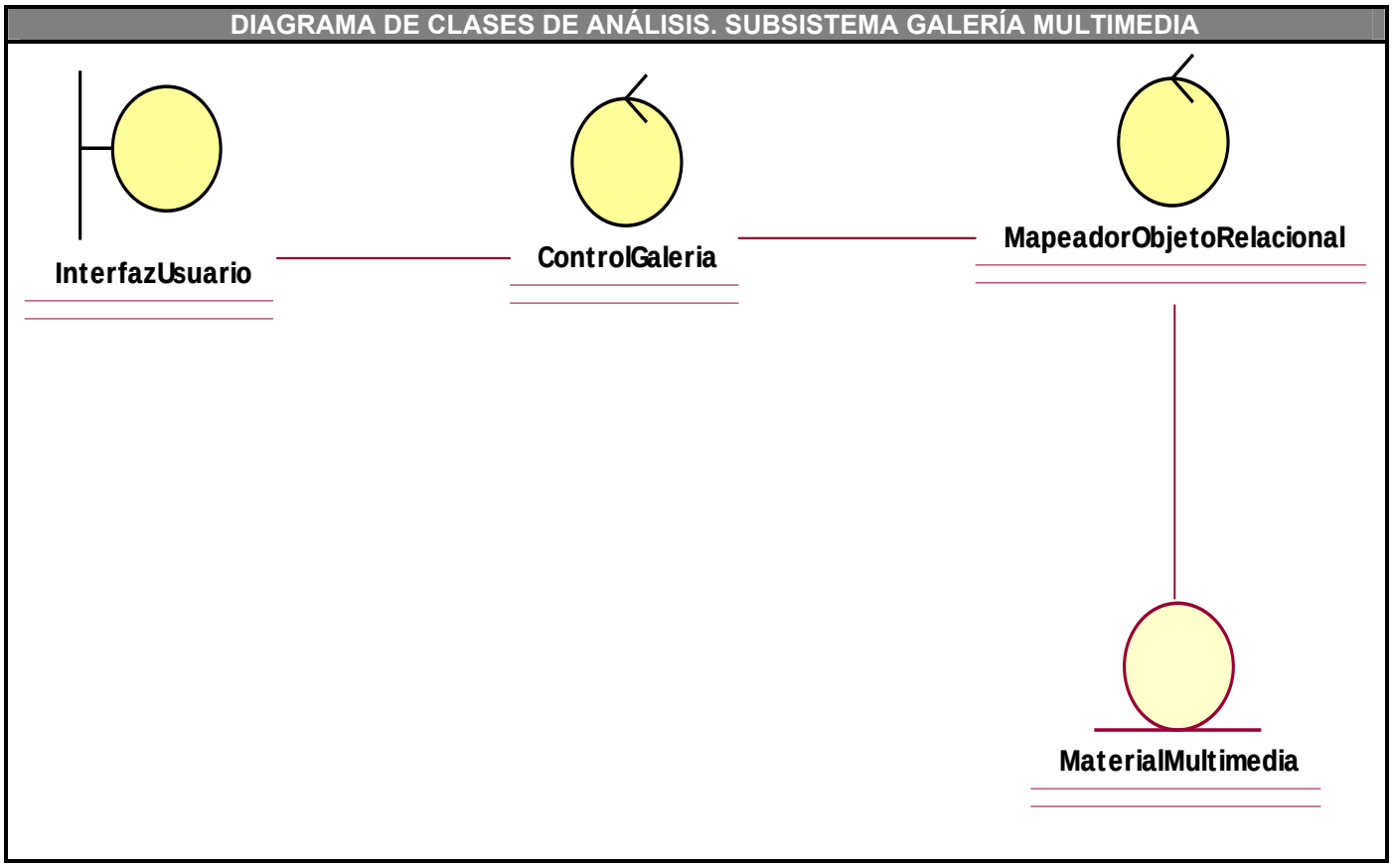
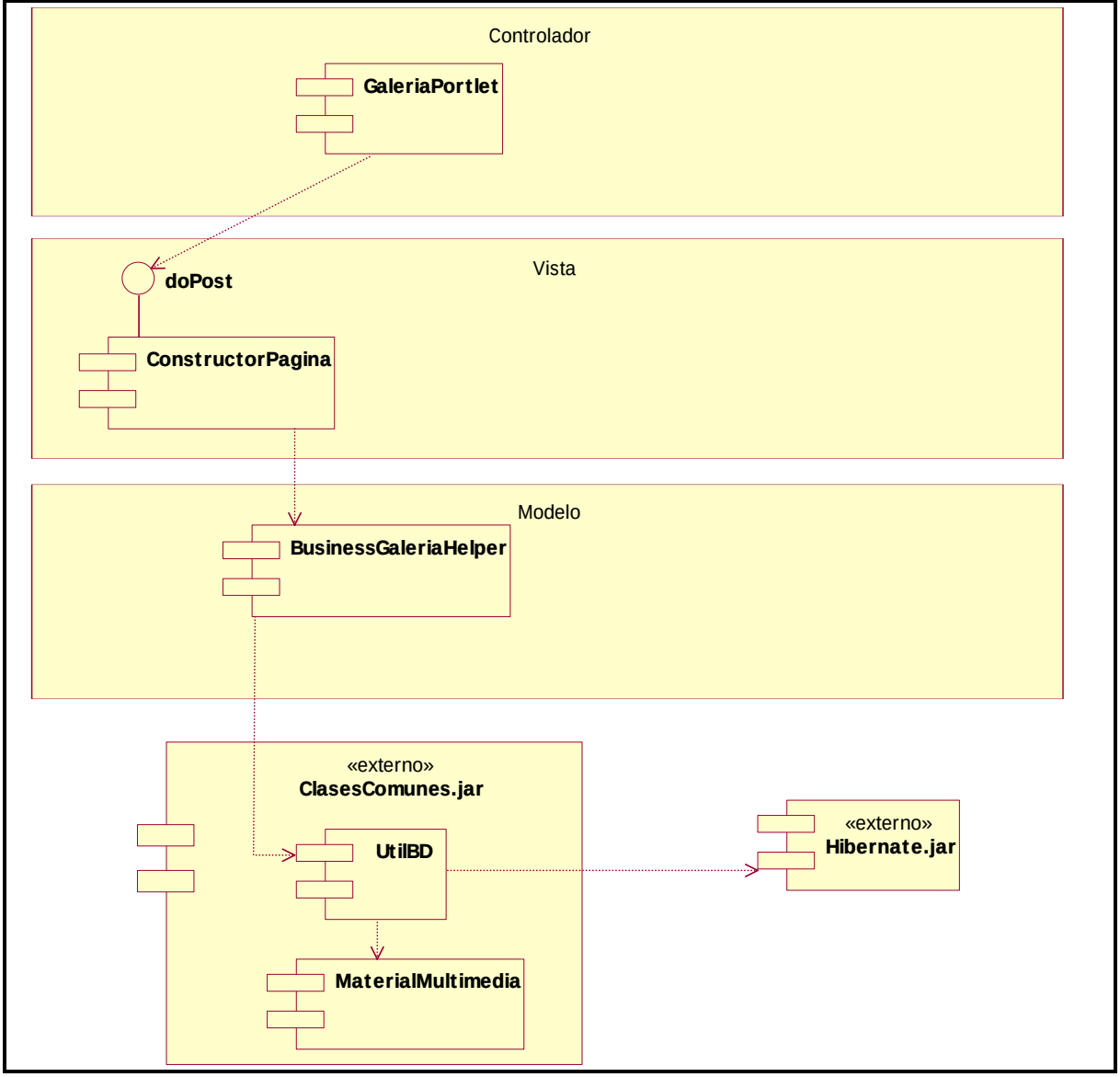
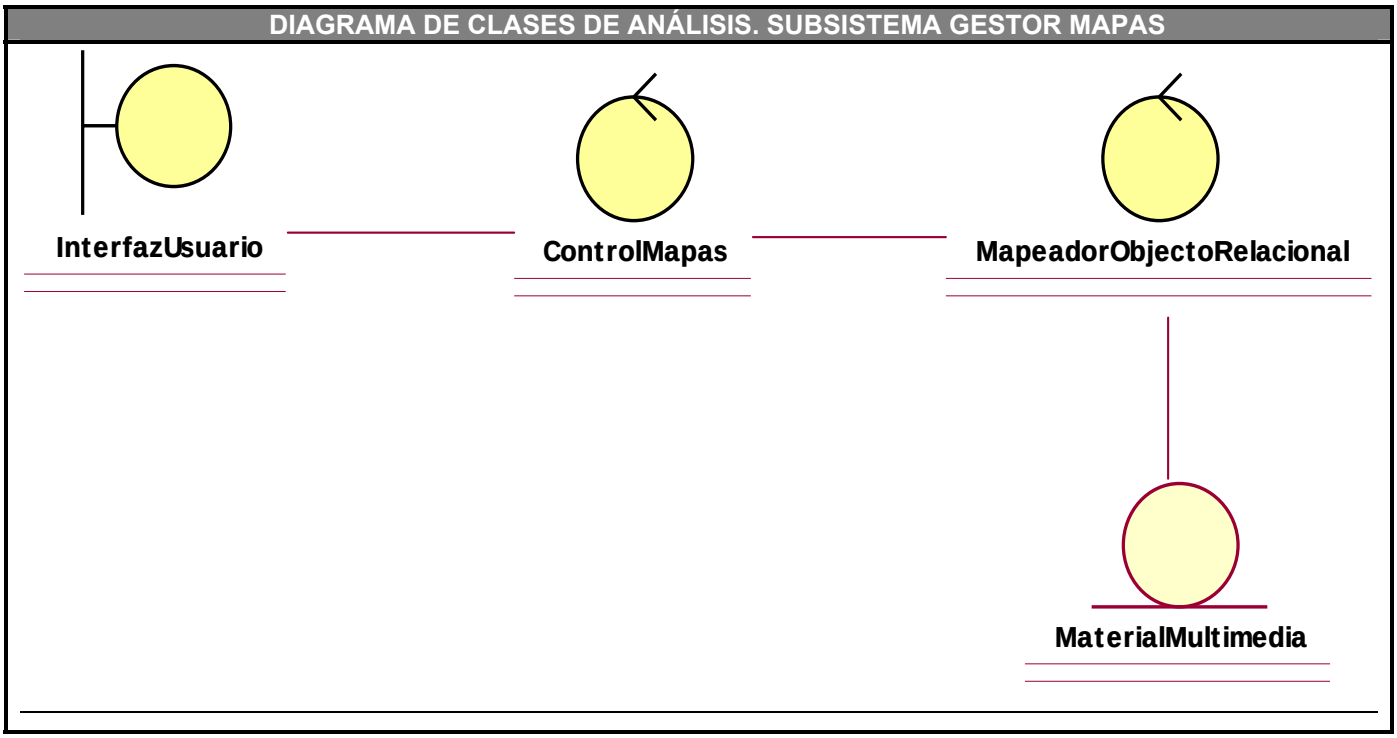


DIAGRAMA DE COMPONENTES. SUBSISTEMA GALERÍA MULTIMEDIA



11.4 Anexo 4

11.4.4 Subsistema Gestor de Mapas.



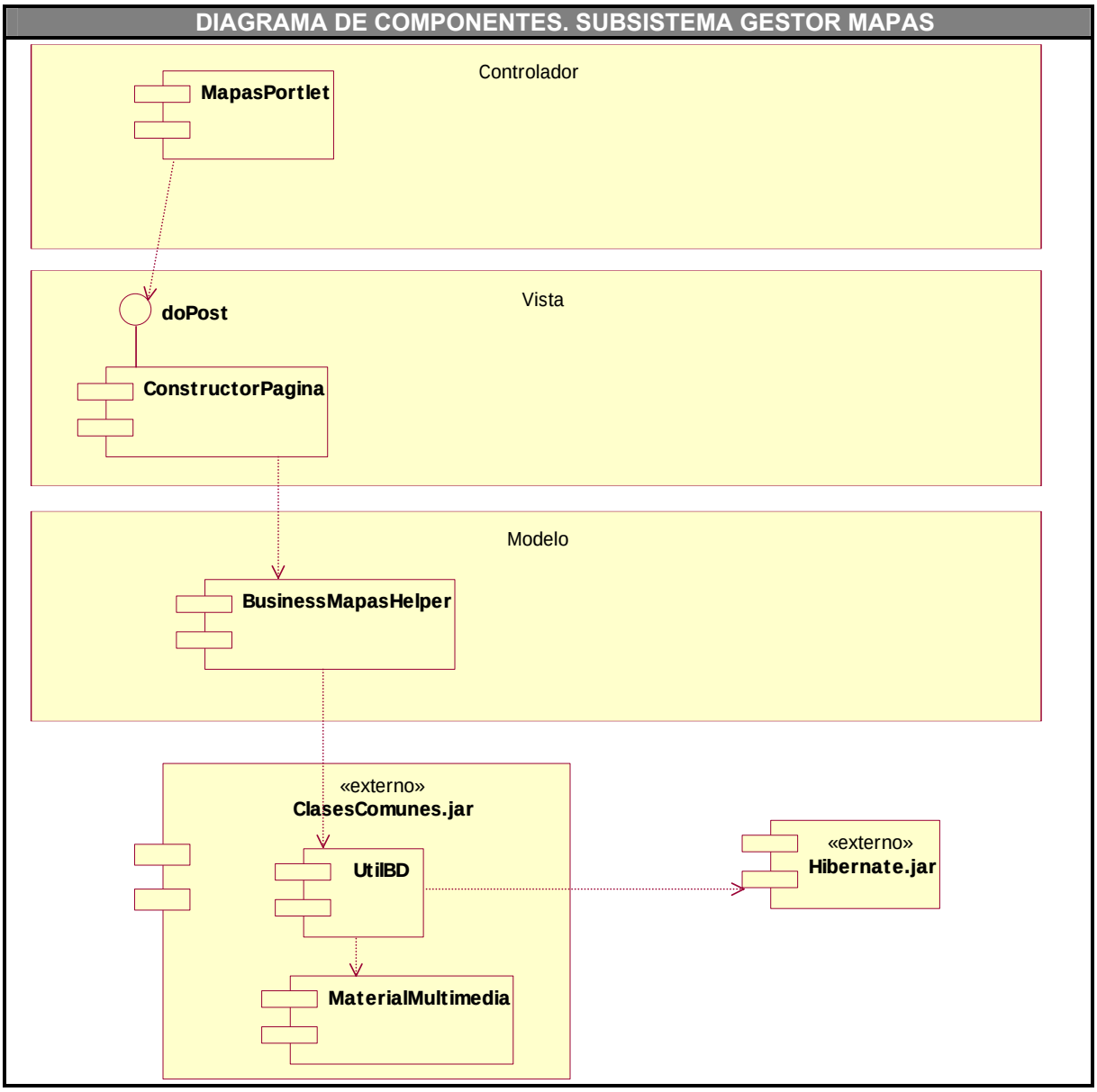


DIAGRAMA DE COMPONENTES. SUBSISTEMA GESTOR MAPAS
Controlador
MapasPortlet
Vista
doPost
ConstructorPagina
Modelo
BusinessMapasHelper
«externo»
ClasesComunes.jar
UtilBD
MaterialMultimedia
«externo»
Hibernate.jar