MODELO DE ARQUITECTURA DE CAPAS PARA SOPORTAR LOS PROCESOS DE ESPECIFICACIÓN Y DISEÑO DE SISTEMAS WEB

LUZ ELENA GUTIÉRREZ LÓPEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS FÍSICO - MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
MAESTRÍA EN INGENIERÍA ÁREA DE INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN
BUCARAMANGA
2009

MODELO DE ARQUITECTURA DE CAPAS PARA SOPORTAR LOS PROCESOS DE ESPECIFICACIÓN Y DISEÑO DE SISTEMAS WEB

LUZ ELENA GUTIÉRREZ LÓPEZ

Trabajo de Investigación para optar al título de
Magíster en Ingeniería

Director
MSc. FERNANDO ROJAS MORALES

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS FÍSICO - MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
MAESTRÍA EN INGENIERÍA ÁREA DE INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN
BUCARAMANGA
2009

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

TITULO:
MODELO DE ARQUITECTURA DE CAPAS PARA SOPORTAR LOS PROCESOS DE ESPECIFICACIÓN Y DISEÑO DE SISTEMAS WEB [*].

Autores:
GUTIÉRREZ LÓPEZ, LUZ ELENA [**].

Palabras Claves:
Arquitectura software, Caracterización, Modelo, UML.

Descripción:

El presente documento describe los resultados de una investigación enfocada al área de la Arquitectura Software. Inicialmente se presenta la oportunidad de mejoramiento encontrada en los procesos de desarrollo software a nivel de aplicaciones Web. Se plantean cuatro (4) objetivos que se pretenden desarrollar a lo largo de la investigación, y todo el referente teórico que sustenta la propuesta. Se hace un breve recorrido por las aplicaciones actuales más significativas que tienen relación con el presente proyecto mostrando sus fortalezas y debilidades. El documento continua con la descripción de la metodología de investigación utilizada, y todas las fases e hitos planteados.

El núcleo de la propuesta se encuentra en la presentación de la Caracterización de aplicaciones Web, pues es en este apartado donde se describen los criterios bases para la selección de los tipos de aplicaciones trabajados. Luego se hace la presentación de la Arquitectura propuesta que consta de cinco (5) vistas, algunas construidas con el Lenguaje de Modelado Unificado (UML). La verificación de la arquitectura se hace por medio de un prototipo software que contempla todas las vistas de la Arquitectura.

El trabajo cierra con el cumplimiento de objetivos, las conclusiones y recomendaciones para la continuidad del presente proyecto.

---

[*] Trabajo de grado

[**] Facultad de Ingenierías Físico-Mecánicas, Escuela de Ingeniería de Sistemas e Informática, Director: Fernando Rojas Morales.

ABSTRACT

TITLE:
LAYER ARCHITECTURE MODEL TO SUPPORT SPECIFICATION PROCESSES AND WEB SYSTEMS DESIGN [*].

Authors:
GUTIERREZ LOPEZ, LUZ ELENA [**].

Key words:
Software Architecture, Characterization, Model, UML.

Description:

This project describes research results focused on the software architecture model. First, improvement opportunity found along the processes development concerning web applications is presented. Four main objectives are posed which pretend to develop, through the investigation, all about theoretical matters to support this proposal. A brief round along most significant actual applications which have a relationship with this project, showing strengths and weaknesses is provided. This document continues with used methodology description and all raised phases and milestones.

Proposal centre in this project is the presentation of application web characterization, because is in this place, where basic criteria is described in order to select working application types. The, proposed architecture is presented. This proposed architecture consists of five (5) views, some of which are built with United Modeling Language (UML). Architecture verification is done trough a prototype software that provides all the architecture views.

This work ends with objective commitment, conclusions and recommendations for the project continuity.

---

[*] Degree Work

[**] Faculty of Physic-Mechanics Engineerings, School of Engineering of Systems and Computer science, Director: Fernando Rojas Morales.

INTRODUCCIÓN

El presente documento tiene como propósito presentar los resultados obtenidos a partir del desarrollo del trabajo de investigación, “MODELO DE ARQUITECTURA DE CAPAS PARA SOPORTAR LOS PROCESOS DE ESPECIFICACIÓN Y DISEÑO DE SISTEMAS WEB”, como requisito para optar al título de Magíster en Ingeniería de la Universidad Industrial de Santander.

Dicho trabajo de investigación tiene por objeto la definición y construcción de un modelo que sirva de apoyo a los procesos de especificación y diseño en proyectos de pregrado, dicho modelo será descrito por medio de UML como lenguaje de representación.

Como línea de base para el desarrollo del modelo, se realizará una caracterización de sistemas informáticos orientados a la Web, para identificar artefactos, funcionalidades, componentes y demás variables que interactúan en los sistemas. Para la verificación del modelo se plantea el desarrollo de un prototipo. Aunque es un trabajo de investigación aplicada, el enfoque principal será la investigación descriptiva, pues ésta ayuda a definir bases, identificar características y propiedades del objeto de estudio de la presente propuesta. Cómo marco metodológico para el desarrollo del prototipo se utilizará el prototipado evolutivo con el fin de presentar avances tempranos del proyecto.

Este trabajo de investigación puede ser de utilidad a: miembros de grupos y centros de investigación, estudiantes de pregrado y postgrado que se encuentren realizando proyectos que refuercen o fortalezcan la labor investigativa de los grupos a nivel universitario, y en general, a toda persona interesada en ingeniería del software aplicada.

1 PLANTEAMIENTO DEL PROBLEMA U OPORTUNIDAD

Desde algunos años, los conceptos de reutilización, patrones y marcos de trabajo empezaron a ser parte fundamental en los procesos de ingeniería que involucraban desarrollo software. Este auge tomó fuerza con la evolución de las nuevas tecnologías y los continuos avances de las TICs[1], en otras palabras el mercado empezó a ser más competitivo, pues a más oferta, el cliente se vuelve más selectivo. Está competitividad obligó a los desarrolladores de software a pensar en un criterio adicional que gobierna el mundo de los negocios a nivel internacional en todas las áreas el cual se denomina "calidad[2]". Pero, ¿Cómo asegurar calidad en un proceso de desarrollo de software?. Para lograr esto, los ingenieros de software han tomado como referencia estándares, modelos y en general buenas prácticas de desarrollo que se han ido depurando durante los últimos años, como es el caso de los estándares ISO 9000, el modelo de Madurez de la Capacidad Integrada más conocido como CMMi y buenas prácticas como revisión por pares, verificación de diseño y código entre otras.

Ahora bien, no basta con utilizar modelos complejos, o seguir lineamientos y reglas de estándares internacionales, cuando al interior de una organización ó un equipo de desarrollo no se tiene conciencia o en su defecto no se conocen a fondo las posibilidades que existen para llevar a cabo procesos de desarrollo, con calidad de manera eficaz y eficiente. A todo esto se le puede sumar la cantidad de información que la ingeniería del software ha generado durante los últimos 30 años como lo son teorías, modelos y en general conjuntos de métodos y técnicas que le brindan al ingeniero de software la oportunidad de hacer las cosas mejor y más rápidamente, "en teoría". Sin embargo, de la teoría a la práctica existe una brecha que puede hacer que esa teoría nunca se aplique, y que por el contrario se dé prioridad a métodos poco confiables para la solución de problemas. Si existe la teoría y ha sido aplicada por otros ¿por qué no simplemente se toma como base para una buena práctica en el desarrollo software? La pregunta aunque obvia, plantea una dicotomía muy

[1] Tecnologías de la información y de la comunicación: son instrumentos y procesos utilizados para recuperar, almacenar, organizar, manejar, producir, presentar e intercambiar información por medios electrónicos y automáticos.

[2] Calidad (Quality, ISO 8402, 1994) Conjunto de propiedades y de características de un producto o servicio, que le confieren su aptitud para satisfacer unas necesidades explícitas e implícitas.

interesante: Por un lado, el costo beneficio que trae consigo el aplicar un método o modelo que implique asegurar calidad en un proceso de desarrollo software, y por otro lado, la velocidad con que un cliente real requiere los productos en un mercado netamente competitivo, en donde gana no el mejor, sino el más rápido.

En la actualidad y en el contexto local, el panorama del desarrollo software se encuentra en un proceso de mejora y transformación, el cual ha sido soportado en la implantación de sistemas de aseguramiento de calidad, los cuales han dejado de ser una estrategia exclusiva de empresas internacionales, y por el contrario, se han convertido en una realidad en Colombia y en nuestro medio (Bucaramanga). Actualmente en Colombia hay más de 50 empresas con certificación ISO 9001:2000 en procesos de desarrollo software, y actualmente Proexport[3] con su programa bandera "Pasión por Colombia", está apoyando a más de dieciocho (18) entidades entre las cuales hay empresas y centros de investigación, en la implantación del modelo de madurez de la capacidad integrada. Aunque esto de por sí ya es una avance, hace falta un engranaje en este ciclo, y es la capacitación de recurso humano competente y cualificado en procesos de desarrollo software. En este momento el lector puede pensar lo siguiente ¿Cómo preparar personal cualificado en las instituciones de educación superior, de tal manera que se conviertan en recursos eficientes y eficaces para las empresas de desarrollo?. Contextualizando el escenario, y tratando de dar respuesta a esta pregunta, se puede hablar de la realidad de las instituciones de educación superior, y en particular de los proyectos y trabajos de grado que están desarrollando los estudiantes. ¿Cuántos trabajos de grado en promedio utilizan criterios de reutilización de componentes, patrones, marcos de trabajo, calidad ó arquitectura software?, la siguiente tabla presenta los posibles resultados a esta cuestión:

Tabla 1. Proyectos de pregrado con orientación a Internet[4]

Criterios avanzados de Ingeniería del software	Número proyectos
Reutilización de Código	2
Reutilización de Diseño	1
Arquitecturas software	1

[3] Agencia Nacional de Promoción a las Exportaciones: Organización encargada de la promoción comercial de las exportaciones no tradicionales, el turismo internacional y la Inversión Extranjera en Colombia.

[4] Fuente, Universidades en Bucaramanga con programas de ingeniería de sistemas con registro calificado, en los cuales se han desarrollado proyectos de grado con enfoque en el área de ingeniería del software.

Patrones	2
Marcos de Trabajo (frameworks)	1
Total de Trabajos revisados	74

Por lo anteriormente expuesto y partiendo de la base de que el trabajo de grado en ingeniería de sistemas, es uno de los mecanismos para que los estudiantes profundicen, pongan en práctica lo aprendido durante la carrera y planteen soluciones reales para el entorno social en el cual interactúan, es posible inferir que realmente existe la posibilidad y la oportunidad de plantear trabajos de maestría, que le permitan a los estudiantes de pregrado realizar trabajos con la ayuda y soporte de estudios actualizados en el área de ingeniería del software, los cuales les permitirían en parte al estudiante:

1. Tener bases conceptuales para aplicar métodos y técnicas avanzadas de ingeniería del software.
2. Ver un caso aplicado donde la reutilización de diseño es una realidad.
3. Profundizar en temas actuales como la arquitectura software.
4. En general fortalecer su proceso de formación, para el trabajo que les espera en el sector productivo.

La presente propuesta de investigación pretende proveer un espacio a los estudiantes de pregrado de Sistemas para profundizar y visualizar aplicaciones reales de la ingeniería del software, que para el caso actual es el análisis y la reutilización de diseño.

2 OBJETIVOS

El objetivo general presentado a continuación enmarca el contexto del trabajo de investigación, así mismo se puede encontrar en el siguiente apartado, la relación de objetivos específicos que permitirán evidenciar el cumplimiento del objetivo general y del trabajo de investigación.

2.1 Objetivo general

Desarrollar un modelo de arquitectura software orientado al tipo de arquitecturas de capas[5], para apoyar las actividades de especificación y diseño software, utilizando el Lenguaje de Modelado Unificado -UML-, y basándose en el uso de patrones de diseño y técnicas de reutilización para la definición de componentes básicos de la arquitectura.

2.2 Objetivos específicos

Para la consecución del objetivo general planteado anteriormente, se enuncian a continuación los siguientes objetivos específicos:

Tabla 2. Objetivos específicos del proyecto

Nro	Descripción del Objetivo
1	Realizar una caracterización de aplicaciones Web que utilizan código abierto para su implementación, para identificar características y artefactos reutilizables en cada tipo de desarrollo.
2	Diseñar un modelo de arquitectura software utilizando casos de uso, diagramas de clases y diagramas de paquetes, que sirva de apoyo a los procesos de especificación y diseño de proyectos software, permitiendo a los desarrolladores la reutilización de diseño y la especificación de requisitos.
3	Verificar la funcionalidad, acoplamiento y la cohesión de los componentes diseñados en la arquitectura software caracterizada, mediante la construcción de un prototipo que evidencie la reutilización de diseño.
4	Publicar y transferir los resultados del trabajo de investigación a grupos de interés y en general a la comunidad científica.

[5] Shawn Mary; Garlan David, Software Architecture- Perspectives on an, Emerging Discipline, Prentice Hall, 1996

3 JUSTIFICACIÓN

La investigación en la Universidad no debe ser entendida como una dimensión aislada de la docencia y la extensión, de hecho, en la investigación y la docencia se conjugan los conceptos epistemológicos que luego son trasmitidos a la sociedad a través de la extensión. Ahora bien, es la universidad una de las responsables del proceso de formación de los futuros profesionales que la sociedad requiere, por ende las estrategias, instrumentos y mecanismos que implante para que dicho proceso de formación sea exitoso, son fundamentales para el desarrollo sostenible de una sociedad. Este trabajo de investigación pretende ser un aporte a ese proceso de formación, de tal manera que los estudiantes del área de sistemas de la Universidad Industrial de Santander, puedan profundizar en el conocimiento adquirido durante su carrera, y cuenten con experiencias y herramientas que les permitan auto formarse para el entorno real en el cual se van a desempeñar.

Ante toda la información que se genera en el mundo día a día, es casi imposible que un profesional egresado tenga a la mano todo el conocimiento que se requiere para ser competitivo en el sector productivo y/o académico. Esto hace que este tipo de trabajos de investigación sean de utilidad no solo para los estudiantes en proceso de formación sino para las mismas empresas que ven como los nuevos profesionales egresan de las instituciones de educación superior no solo con teoría sino con teoría aplicada a la realidad local, nacional y mundial.

Teniendo en cuenta lo anteriormente expuesto, es posible justificar el desarrollo del presente trabajo de investigación desde las siguientes perspectivas:

3.1 Social

La formación y capacitación de investigadores es un aporte significativo para la comunidad universitaria, teniendo en cuenta la retribución de este personal a la sociedad en lo ámbitos tecnológicos, económicos y sociales. La realización de la presente propuesta no solo contribuirá a la formación personal del autor sino también a los demás miembros del grupo investigador al cual pertenece y a la comunidad estudiantil de sistemas en general. Esto se

verá reflejado en la preparación de profesionales con una visión más global y con habilidades para resolver problemas reales de la sociedad.

3.2 Académica

La realización del presente trabajo de investigación proporcionará una alternativa en la presentación y formalización de contenidos curriculares en el área de ingeniería del software, particularmente en lo relacionado a los conceptos de Arquitectura software, además le brindará a los estudiantes de pregrado, un conjunto de conceptos y herramientas aplicadas para identificar y valorar situaciones en donde el conocimiento acerca de reutilización a nivel de diseño y de código son importantes. Como apoyo al proceso de formación académica, existe una oportunidad para continuar en pro del mejoramiento continuo, tanto en la calidad de la educación que se imparte como en la calidad de los futuros profesionales.

3.3 Tecnológica

El uso de estándares y lenguajes de modelado constituye un aporte a la generación y uso de nuevo conocimiento, lenguajes como el Lenguaje de Modelado Unificado -UML-, permiten una mejor comprensión de los problemas a través del modelado de los mismos. Por otro lado, el uso de nuevas tecnologías para aplicaciones soportadas en Internet, permitirán la verificación y validación de los modelos planteados en este trabajo de investigación.

4 ESTADO DEL ARTE

Este apartado contiene información de proyectos y trabajos de investigación realizados en las áreas de: Arquitecturas Software, diseño, modelado, patrones y reutilización; lo cuales son los ejes centrales del presente trabajo de investigación. El objeto de este apartado es presentar una perspectiva global de los trabajos realizados en las temáticas del proyecto, para así analizar las ventajas y recursos que la presente investigación puede ofrecer a la comunidad científica y académica como complemento a la investigación realizadas en las áreas relacionadas anteriormente.

Tabla 3. Proyecto ABLE

ABLE	
Descripción	Es un proyecto de la Universidad Carnegie Mellon, y busca establecer una línea de base para la disciplina de la Arquitectura Software. Los componentes de esta investigación incluyen formas para describir y aprovechar los estilos arquitectónicos, brindando herramientas para la práctica y construcción de arquitecturas, y la definición de métodos formales para la creación nuevos ADLS. Además el proyecto trata temas emergentes como la ubicuidad, de la penetrabilidad, la heterogeneidad y la movilidad de los usuarios.
Análisis	Es una mega proyecto de investigación que pretende formalizar muchos aspectos de la Arquitectura Software, sin embargo, sigue con la premisa inicial de resolver los problemas académicos y luego los del sector productivo, algo que ha permitido que esta disciplina se divida en dos mundos: El mundo de lo real, aplicado y necesario y el mundo de la teoría y la academia.

Tabla 4. Proyecto OLIVANOVA Modeler

OLIVANOVA Modeler	
Descripción	Es un producto software que permite a desarrolladores y analistas definir software desde una perspectiva orientada a objetos, captura los objetos de negocio como clases y los conecta a través de las relaciones. Permite además definir la lógica de negocio, sus reglas y restricciones del sistema sin necesidad de programar. Estas reglas y restricciones serán traducidas por La Máquina de Programar (otro sistema) en el lenguaje y arquitectura seleccionados. Tras esta transformación, el código obtenido estará listo para ser compilado e instalado.
Análisis	Como herramienta informática tiene un valor enorme en el modelado de negocios y definición de reglas, pero como elemento que permita aportar en la formación y apropiación de conceptos de ingeniería a una persona es limitado. Dado que es un proyecto de tipo comercial es entendible que la metodología y el valor agregado en conocimiento para la construcción del mismo sean de carácter propietario, limitando de esta forma la socialización de aspectos claves para la construcción de sistemas similares.

Tabla 5. Team Synergy

Arquitectura para una herramienta modelo del recopilador para el campo del desarrollo de programas embebidos	
Descripción	Este trabajo en investigación presenta los requerimientos técnicos de una arquitectura software, así como los elementos claves que debe tener la lógica de negocios para construir una AS. Propone tres vista de alto nivel: vista combinada, flujo de datos y flujo de trabajo por usuarios y realiza además una descripción de cada paquete que compone cada una de dichas vistas.
Análisis	El proyecto presenta bases sólidas sobre los conceptos que debe tener una arquitectura, sin embargo el nivel de abstracción es muy alto y no tiene en cuenta otras propuestas que existen en la comunidad académica para formalizar una arquitectura. Es un proyecto abierto que permite compartir la experiencia en esta área, pero está lejos de ser una referencia para estudiantes o profesionales en el área de desarrollo software para la escritorio y/o la Web.

Tabla 6. Proyecto ALF

ALF: Arquitectura Open-Source para la Gestión de Servicios de Aprendizaje y Colaboración	
Descripción	Es una solución planteada por la UNED (Universidad Nacional de Educación a distancia. España), para la implantación de un modelo educativo a distancia, para esto brinda herramientas para dar soporte al concepto de comunidad virtual. Está arquitectura está basada en el modelo de tres capas donde se proponen los servicios básicos a nivel de infraestructura los servicios adaptados a las necesidades de un dominio y una interfaz de interacción que en este caso es un portal Web.
Análisis	Este trabajo de investigación está orientado hacia los sistemas colaborativos y de aprendizaje, es otras palabras es una caso particular de aplicación de una arquitectura. Habla y expone las características de los marcos de desarrollo, pero no aplica este concepto para construir rápidamente nuevos sistemas con base en esta propuesta, se puede decir que es más un integrador que un generador de posibles sistemas Web.

Tabla 7. Tesis doctoral en Arquitectura Software

Diseño y Desarrollo de una Arquitectura Software Genérica Orientada a Servicios para la Construcción de un Middleware Grid Orientado a la Gestión y Proceso Seguro de Información en formato DICOM sobre un Marco Ontológico	
Descripción	Este proyecto consiste en la construcción de una Arquitectura Orientada a Servicios (SOA) y la implementación de un Middleware Grid basado en esta arquitectura, cuya función principal será la gestión, integración y proceso de información en formato DICOM almacenada de forma distribuida en diferentes dominios administrativos, de forma segura y estructurada semánticamente mediante la definición de ontologías médicas basadas en el informe estructurado y los estudios DICOM.
Análisis	Como punto de referencia para la construcción de proyectos similares es excelente fuente de referencia, su novel de detalle puede llegar a confundir a una persona con pocos conocimientos en el área de Arquitecturas Software, por lo tanto no es muy factible que un estudiante de pregrado utilice esta fuente para un trabajo de desarrollo software. Por otra parte, dado su alto nivel de detalle solo se aplica al área de trabajo sobre el cual se suscribe su alcance. Para el presente trabajo de investigación es una guía interesante pues permite visualizar como el autor plantea una propuesta de arquitectura en términos de diseño y de código.

Estos cuatro proyectos son solo una muestra de los trabajos desarrollados en el área de la Arquitectura Software, como se puede apreciar, ninguno de ellos se solapa con otro, pues está disciplina tiene todo por formalizar, decir y explorar. La búsqueda de información y de proyectos relacionados con la temática del presente proyecto, dio como resultado una adecuada fundamentación teórica, y a su vez permitió evidenciar algunas oportunidades de mejoramiento en los proyectos analizados. Como conclusión final de este análisis, es conveniente indicar que para este tipo de trabajos de investigación, casi toda la literatura encontrada con respecto al tema se podría llegar a considerar estado del arte, pues cada aporte sigue construyendo día a día esta nueva disciplina.

5 MARCO TEÓRICO

5.1 Importancia y ventajas de una caracterización

En general, las áreas del conocimiento generan día a día información, la cual en muchos casos no alcanza a ser procesada en su totalidad, es por esto que se han utilizado estrategias para clasificar dicho conocimiento y así disponer de información organizada y valiosa en determinado momento. Estudios ontológicos, mapas conceptuales, documentos clasificados y caracterizaciones, son y han sido elementos claves para agrupar información acerca de un tema de estudio. Este trabajo de investigación pretende realizar una caracterización que agrupe el conocimiento existente acerca de las funcionalidades de los sistemas Web, es por esto que es importante socializar el concepto de “caracterización” así como enunciar las posibles ventajas en el proyecto.

Una caracterización es una agrupación de información y conocimiento que tiene por fin identificar características claves en el tema de estudio, para así estructurar y acotar una investigación o trabajo. En algunos casos las caracterizaciones contienen información relevante al marco teórico y/o estado del arte del objeto de estudio, con el objetivo de darle al lector una visión completa del conocimiento expuesto en dicho trabajo. Lo anterior se puede analizar como un soporte y apoyo en el desarrollo de la presente propuesta pues con la clasificación del conocimiento se podrá:

- Disponer de un documento guía que permita orientar el modelo planteado en la presente propuesta.
- Clasificar de acuerdo a ciertos tópicos parte del conocimiento asociado al proyecto, de tal manera que se pueda acotar el problema objeto de la propuesta de una manera objetiva.
- Recopilar parte del conocimiento del proyecto para socializarlo con otros grupos o personas que puedan aportar y complementar el trabajo de investigación.

Lo anteriormente mencionado son las ventajas de utilizar una caracterización en un trabajo de investigación, sin embargo también existe dificultades por superar a la hora de realizar o utilizar este tipo de fuente. Entre las mayores desventajas que puede tener un investigador al utilizar o realizar una caracterización se pueden enunciar:

- No presentación de todas las perspectivas posibles con respecto a un conocimiento, cuando se desconoce parte del objeto de estudio.

- Puntos de vista de diversos autores para una misma problemática, pues dependiendo de un enfoque seleccionado se puede cambiar el trasfondo de la caracterización.
- Nivel alto de acotamiento a la hora de clasificar la información y el conocimiento
- Desconocimiento de la memoria del área de estudio, en algunos casos los autores se centran solo en la actualidad desconocimiento los trabajos realizado por otros autores.

5.2 La Arquitectura y su reflejo en la Arquitectura Software -AS-

Cuando se habla de Arquitectura, se asocia el concepto al arte y la ciencia de diseñar edificios. Históricamente hablando, los arquitectos no solo se dedicaban a diseñar edificios, sino que ampliaban su rango a plazas, pueblos y hasta ciudades enteras. Hoy en día no es tan diferente la cuestión, existen arquitectos especializados en la construcción y distribución del espacio urbano, así como también en el diseño de edificios y obras puntuales. En la actualidad no solo de diseñan edificios, los médicos por ejemplo planifican y diseñan las operaciones, los odontólogos diseñan la sonrisa de las personas y para no ir tan lejos los ing. de la industria automotriz, se basan en los diseños para optimizar la producción en masa de sus vehículos. La importancia de una buena arquitectura, radica en la capacidad para anticipar eventualidades que puedan impedir el correcto funcionamiento de las cosas, esto conlleva al planteamiento de alternativas de solución, a una compresión total de lo planteado, la posible reutilización y la optimización de recursos tanto a nivel económico, físico, tecnológico y humano.

Para un arquitecto es importante definir claramente la forma de representar el diseño de un edificio, es por eso que debe utilizar una forma de representación común, que le permita a otra persona identificar claramente elementos como: la estructura, el sistema eléctrico, el sistema de aguas residuales, los cimientos, etc. Esto le permite al arquitecto tener una visión global de lo que se desea construir.

Los conceptos de arquitectura han heredado a áreas de la ciencia y la tecnología su base conceptual, es por eso que hoy en día, las personas relacionadas con la sociedad del conocimiento y las TICs hablan del término “Arquitectura Software - *AS*”. Retomando el concepto básico de arquitectura, el término *AS* se podría definir como “El arte y la ciencia de diseñar software”, y así, de igual forma que un arquitecto diseña edificios, un arquitecto de software ó ing. de software diseña software. Entonces, los Ings. de software deben tener una

forma de representación, que les permita visualizar de manera general un sistema, y adicionalmente que permita dividir en vistas o perspectivas cada una de las piezas claves de su diseño.

Hasta ese punto es claro que existe una relación semántica en cuanto al término arquitectura y la arquitectura software, en los apartados siguientes se presentan entre otras cosas los inicios, conceptos claves, ventajas, componentes y los principios claves de la AS.

5.3 Breve historia de la Arquitectura Software

Los primeros conceptos acerca de la **AS** surgen en la academia en los años sesenta, entre los autores más representativos se encuentra Edsger Dijkstra, conocido por sus críticas hacia las sentencias GOTO, y precursor de las estructuras de control que se conocen en la actualidad. La siguiente tabla presenta un recuento histórico de los hitos más relevantes que han estado vinculados de alguna u otra forma con la AS.

Tabla 8. Reseña histórica de la disciplina AS

Año	Representante	Descripción
1968	Edsger Dijkstra	Concepto de ciencias de la computación Los niveles de abstracción y la programación estructurada entran en contexto
1968	NATO (North Atlantic Treaty Organisation)	Conferencias sobre Ingeniería del software por F. L. Bauer
1969	NATO (North Atlantic Treaty Organisation)	Conferencias sobre Arquitectura de software por P. I. Sharp
1971	C.R. Spooner	Artículo “Una arquitectura de software para los 70s” [1].
1971	Niklaus Wirth	Propuesta de los niveles de abstracción
1972	David Parnas	Concepto de modulo y ocultamiento de información
1974	David Parnas	Estructuras para software
1975	Fred Brooks	Concepto de arquitectura como interfaz de usuario Se compara al arquitecto de software como la persona que diseña una casa
1976	Frank DeRemer & Hans Kron	La programación en grande versus la programación pequeña
1976	David Parnas	Familias de programas Árboles de decisión Alternativas de los diagramas de flujo Desarrollo estructural
1992	Dewayne Perry, Alexander Wolf	Fundación para el estudio de la Arquitectura Software Nacimiento formal de la AS como una disciplina
1995	Mary Shaw, David Garlan, Paul Clements, Robert	Visión conceptual de la AS, aportes a su crecimiento como una disciplina.

	Allen, Rick Kazman	Aunque el trabajo se realiza en el SEI, no se vincula el concepto a CMM
1996	Gamma Erich, Helm Richard, Johnson Ralph, Vlissides John	Patrones, concepto clave en la AS Publicación del libro "Design Patterns - Elements of Reusable Object-Oriented Software", más conocido como la banda de los cuatro.
Actualidad	Academia y sector productivo	Diversidad en la utilización de conceptos, no hay unificación de criterios. Importancia en los conceptos de reutilización, diseño y marcos de trabajo (frameworks).

5.4 Definición del concepto Arquitectura Software

A pesar de ser un concepto utilizado por los ingenieros de software, no existe una definición taxativa acerca de lo que es la ***AS***, a continuación se relacionan algunas de las definiciones más comunes.

Tabla 9. Conceptos de AS

Autor	Definición
IEEE 1471-2000	La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución [2]. *Nota:* es la definición adoptada por Microsoft.
Paul Clements[6]	"La AS es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones" [3].

5.5 Tipos de Arquitectura Software

Luego de conocer las dos perspectivas más comunes del concepto de Arquitectura software, así como su evolución a lo largo de la historia de los sistemas, podemos vislumbrar la importancia de este concepto dentro del desarrollo software. Es así como existen una gran variedad de problemas que requieren soluciones de software y diversas arquitecturas que se encargan de manejarlos. A continuación se presenta una clasificación de los diferentes tipos de arquitecturas según la visión de Mary Shaw y David Garlan [4]. Aunque no todos los problemas se ajustarán totalmente a estas arquitecturas, si pueden proporcionar una aproximación a la solución.

[6] Autor del libro "Software Architecture in Practice". Profesor de la "Western Michigan University"

Tabla 10. Arquitecturas alternativas

Categoría	Subcategoría	Patrones de diseño
Flujo de datos	Secuencial por lote	Decorador
	Tuberías y filtros	
Componentes independientes	Procesos de comunicación paralela	Observador
	Sistemas cliente-servidor	Apariencia
	Sistemas de eventos	Estado
Maquinas virtuales	Interpretes	Interprete
	Sistemas basados en reglas	
Arquitecturas de almacenamiento	Bases de datos	Observador, Iterador
	Sistemas de hipertexto	
	Pizarrones	
Arquitecturas por capas		Fabrica Abstracta, Sencillo

5.6 Problemas que resuelve la Arquitectura Software

La Arquitectura Software surge como una disciplina para tratar de apoyar los problemas relacionados con la complejidad de los sistemas, busca en parte modularizar un sistema para entender de manera detallada las interacciones entre cada elemento. Adicionalmente ayuda al ingeniero de Software a tomar decisiones importantes en etapas iniciales de un proyecto. A Continuación se enuncian los principales problemas que ataca la AS.

- Visiones limitadas o confusas del sistema: permite demarcar y presentar por medio de diversas vistas del mismo sistema características importantes.
- Ambigüedad en la selección de plataformas: es posible la correcta escogencia de la plataforma a usar, debido al uso de diagramas físicos para la evaluación de las diversas alternativas.
- Reprocesamiento de código: por medio del planteamiento de los posibles módulos es posible definir claramente cuales serán los componentes a implementar, y así evitar posibles retrocesos en la codificación.
- Mala interconexión entre componentes: se puede definir la forma como interactuaran y se comunicaran los componentes principales del sistema.
- Ambigua asignación de recursos humanos, técnicos, tecnológicos y económicos al desarrollo de un sistema: debido a la clara definición de componentes, relaciones y funcionalidades del sistema es posible la correcta asignación de los recursos.
- Mal nivel de desagregación de módulos o componentes: se hace posible la descomposición funcional y lógica de cada uno de los componentes que hacen parte del sistema global.

5.7 Principales problemas en la actualidad de la Arquitectura Software

El aparatado anterior expone los problemas que soluciona la AS, sin embargo es innegable que en esta área del conocimiento aun existe mucho camino por recorrer, a continuación se enuncia la problemática asociada a la AS.

- Falta de criterio unificado, diferencias entre academia y sector productivo: no hay consenso de los conceptos utilizados en esta disciplina, además la academia continúa sus investigaciones desconectada de los procesos que se llevan a cabo en la industria.
- No hay un modelo de proceso desde el inicio hasta el final.
- Desarrollo en paralelo de conceptos antagónicos o no coordinados.
- Métodos ágiles: el desarrollo de las metodologías de este tipo conlleva a no prescindir del uso de métodos de análisis y diseño exhaustivos como lo es la Arquitectura software, porque su objetivo es desarrollar el producto en el menor tiempo posible.
- Metodologías de ciclo de vida (adecuación de métodos formales).
- Patrones, estilos y tácticas.
- Apropiación nominal de la AS por estrategias que no implementan principios arquitectónicos pero se benefician de su prestigio: se ha hecho común el uso del termino de Arquitectura software de una forma no apropiada, es decir, que la aplicación del concepto no corresponde a la semántica original de la misma.
- Poca masa crítica de herramientas y lenguajes de modelado arquitectónico (de alto nivel, con conectores de primera clase).

5.8 Concepto de modelo, metamodelo y meta metamodelo

En toda investigación es importante contextualizar la terminología y los conceptos claves del objeto de estudio, pues dependiendo de la visión de uno u autor se pueden llegar a tener diversas apreciaciones, las cuales pueden llegar a ser totalmente válidas. A continuación se presentan y analizan tres (3) de los conceptos más relevantes en el desarrollo de la presente propuesta de investigación:

5.8.1 Modelo

Los modelos constituyen una representación que puede o no incluir todos los detalles del producto original, además por ser simples representaciones se pueden utilizar para encontrar errores mucho antes de empezar a construir el verdadero producto final. Lo anterior induce tres de los principales propósitos de un modelo:

- Ayudar a comprender soluciones o problemas complejos
- Socializar ideas entre equipos de trabajo
- Servir de guía en un proceso de desarrollo de productos

Eric Braude en [5] expresa lo siguiente con respecto a la definición de modelo "Por lo común es necesario describir las aplicaciones desde varias perspectivas. Esto se puede comparar con la arquitectura de una casa, que requiere múltiples perspectivas como planos, vista vertical, vista frontal, plano de plomería, etc. En el mundo del software, las perspectivas se llaman modelos".

Ahora bien, existe otra terminología que se asocia al concepto de modelo que se ha descrito hasta ahora, en ella se encuentran los metamodelos y los meta metamodelos. Expresar la definición de un metamodelo como un modelo de modelos, es tan solo una descripción semántica que genera ambigüedad, por lo tanto es necesario analizar los componentes necesarios que debe tener un modelo antes de plantear una definición taxativa.

5.8.2 Metamodelo

Para la OMG (Object Management Group) un metamodelo debe definir los elementos esenciales, sintaxis y estructuras que son utilizadas para la construcción de de modelos, particularmente un metamodelo debe proporcionar:

- Un modelo abstracto para la representación de sus modelos, objetos y asociaciones
- Un conjunto de reglas para entender los00 modelos, independientes de un lenguaje de programación
- Reglas que definan el ciclo de vida (pueden ser vistas), la composición y la semántica de los elementos utilizados
- Provea mecanismos de extensión para incorporar nuevos tipos de especificaciones

James Rumbaugh, Ivar Jacobson y Grady Booch definen el concepto de metamodelo en [6] como "Modelo que define el lenguaje empleado para expresar un modelo, también puede ser una instancia de un metamodelo".

5.8.3 Meta metamodelo

Como concepto un meta-metamodelo es lo mismo que un meta modelo, la real diferencia se encuentra en la práctica, pues son las capas inferiores las que definen lo que es y no es específicamente, es decir, son las herramientas de bases de datos u otras similares las que deciden lo que en realidad se es.

La siguiente figura puede aclarar un poco más los conceptos vistos hasta el momento.

6 METODOLOGÍA PROYECTO

6.1 Metodología Investigación

La investigación en general cuenta con niveles que describen el grado de profundización del estudio que se esté realizando, es así como se pueden definir cuatro (4) niveles: Exploratorio, Explicativo, Descriptivo y Predictivo. Ninguno de estos niveles es superior a los otros, el uso de un nivel depende del tema a estudiar y de los objetivos planteados. La siguiente figura, presenta estos niveles:

Figura 1. Estado del arte y marco de referencia

Dado que es importante definir un modelo de investigación que permita establecer las bases para ésta y futuras investigaciones, se utilizará en esta propuesta la *investigación descriptiva*, utilizando el método de análisis para lograr caracterizar el objeto de estudio, así como señalar sus características y propiedades. Este tipo de investigación permitirá ordenar, agrupar y sistematizar los objetos involucrados en la propuesta, siendo por ende una excelente estrategia para desarrollar la presente investigación.

6.2 Metodología de Desarrollo del Proyecto

La metodología de desarrollo de proyectos software contempla dos aspectos básicos: el modelo de ciclo de vida y el lenguaje de representación. El primero, describe las actividades a realizar y el orden en que deben cumplirse; el segundo, conforman el conjunto de convenciones utilizadas para representar y expresar los artefactos del sistema software en la fase de análisis y diseño.

6.2.1 Entrega Evolutiva

El modelo de ciclo de vida elegido para el desarrollo del presente trabajo de investigación es la entrega evolutiva. Este modelo ofrece el control que se obtiene con la entrega por etapas y la flexibilidad que proporciona el prototipado evolutivo. Como puede observarse en la Figura 2 el énfasis principal se establece en el núcleo del sistema, que está constituido por funciones de bajo nivel que probablemente no van a ser modificadas por la realimentación del cliente. En este caso, el núcleo del sistema lo constituye el concepto de arquitectura software; de ahí se desprenden en cada ciclo, la caracterización, el diseño y la verificación del modelo.

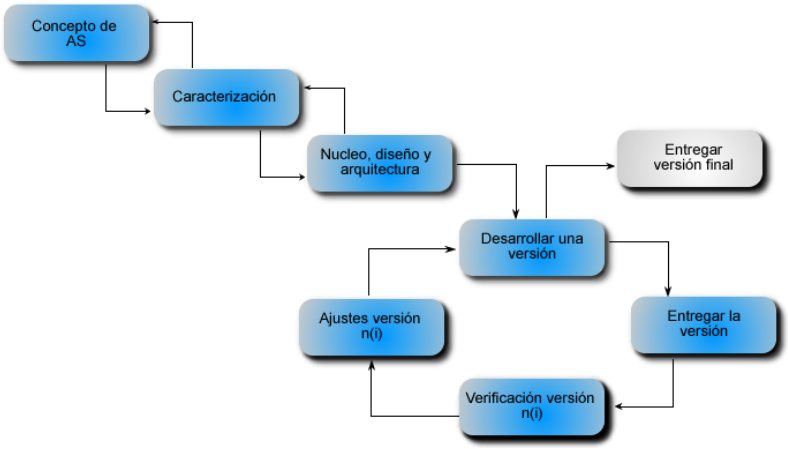


Figura 2. Entrega evolutiva, adecuada al trabajo de investigación

En el apartado “9. PLAN DE TRABAJO” de la propuesta inicial, se describe en detalle cada una de las fases anteriormente mencionadas, exponiendo los instrumentos y recursos a utilizar así como los productos y resultados de cada una.

6.2.2 El lenguaje de representación

El lenguaje de modelado es la notación (principalmente gráfica) de que se valen los métodos para expresar sus diseños con el objetivo de comprender claramente el sistema. UML (Unified Modeling Language - Lenguaje de Modelado Unificado) es el lenguaje de modelado utilizado en el desarrollo del proyecto para tal fin, éste permite visualizar, especificar, construir y documentar la arquitectura de un sistema mientras que evoluciona a través del ciclo de vida de desarrollo de software.

UML se vale de diagramas para representar un sistema en toda su extensión. Un diagrama es una representación gráfica de una colección de elementos del modelo, construida a menudo como un gráfico conexo de arcos (relaciones) y de vértices (otros elementos modelo). Los diagramas básicos de UML son los diagramas de clases, diagramas de objetos, diagramas de casos de uso, diagramas de secuencia, diagramas de colaboración, diagramas de estados, diagramas de actividad, diagramas de componente y diagramas de despliegue.

UML ha logrado posicionarse como un lenguaje de modelado estándar, tras la gran oleada de lenguajes de modelado orientado a objetos. Su evolución integra el trabajo de muchos investigadores, lo cual lo hace una opción bastante sólida y confiable para el desarrollo de proyectos, dentro de los cuales se pueden destacar los proyectos relacionados con sistemas distribuidos basados en la Web.

A lo largo de los últimos años UML se ha establecido como un lenguaje de modelado para estilos arquitectónicos, a pesar de que UML no es un lenguaje formal para la descripción de arquitecturas, su uso generalizado y las mejoras en la última versión como son en el desarrollo de estereotipos, que son tipos de elementos que permiten definir particularidades del modelo dentro del mismo modelo [6], lo han colocado como el lenguaje de facto para la representación visual de sistemas.

7 MARCO DE REFERENCIA DE APLICACIONES WEB

7.1 Evolución del desarrollo Web

Para analizar los antecedentes del desarrollo de aplicaciones Web, es necesario contextualizar los hechos que dieron origen a la Web, por ende es imprescindible observar cuales fueron los hitos fundamentales que llevaron a la creación de una organización mundial que hoy en día orienta de manera indirecta los hilos de este tipo de desarrollos.

La tabla 11 presenta la información histórica recopilada antes de la creación de "consorcio World Wide Web" en adelante y para efectos de este documento ***W3C***.

Tabla 11. Recuento histórico y nacimiento de la Web

Año	Descripción
1965	El término **Hipertexto** fue introducido por primera vez en 1963 por Ted Nelson[7], y su significado inicial fue "escritura no secuencial", en 1965 aparece el término publicado en un artículo titulado "Computadoras, creatividad, y la naturaleza de la palabra escrita", adicionalmente se le atribuyen otros términos como "hypermedia" y "virtualidad".
1968	Douglas C. Engelbart[8], planteó las bases para el primer sistema en línea, denominado **NLS** por sus siglas en ingles (Online System).
1969	La agencia de proyectos de investigación avanzada (ARPA por sus siglas en ingles) inicia la investigación de lo que conocemos hoy en día como Internet. El doctor Engelbart trabajó en la versión de esta red denominada **ARPANET**.
1971	En el verano de este año un programador llamado Raymond Samuel Tomlinson[9] implementó el primer sistema de **correo** para usuarios en diferentes máquinas que estaban conectadas a ARPANET.
1972	Raymond Tomlinson introduce el uso del carácter "@" en el correo para permitir a los usuarios de otras redes utilizar este sistema.
1974	Bajo la tutela del proyecto ARPANET, es mejorado el NCP (Network Control Protocol) el cual venia siendo desarrollado para establecer las políticas de comunicación entre dos computadoras. De este trabajo nació el protocolo de paquetes para la interconexión de redes, **TCP** (Transmission Control Protocol).
1978	Parte del protocolo TCP fue presentado como protocolo **IP** (Internet Protocol). El proyecto para llevar a acabo la transición del protocolo para computadoras de ARPANET desde NCP a TCP/IP se fortaleció con este avance.
1980	Tim Berners[10], investigador del Laboratorio Europeo de Física de Partículas (CERN) de Ginebra, concibió la idea de un proyecto de hipertexto global, el cual, años más tarde se convertiría en la famosa **World Wide Web**.

[7] Licenciado en filosofía de la universidad de Swarthmore en 1959, magíster en sociología de la universidad de Harvard en 1963 y Doctor en medios digitales de la universidad de Keio en 2002.

[8] Ingeniero eléctrico de la Oregon State University en 1948, ingeniería de la Universidad de Berkeley en 1952 y doctor de la universidad de Berkeley en 1955. Creador del ratón ó "mouse" de las computadoras.

[9] Ingeniero eléctrico de Rensselaer en 1963, magíster en ingeniería eléctrica del Massachusetts Institute of Technology MIT en 1965, en 2004 IEEE le entregó el premio Internet por su trabajo y aporte a la red.

1983	Nacimiento de **INTERNET**, finaliza la transición de los protocolos de ARPANET. Todos los sistemas empiezan a utilizar el protocolo TCP/IP.
1984	La arquitectura DNS (Domain Name System) evidencia el problema de la asignación de nombres para máquinas en la Internet, su autor el doctor Paul Mockapetris[11], propone no tener una única tabla para la asignación de direcciones de computadoras, sino por el contrario distribuir en muchas máquinas la asignación de los nombres de manera dinámica.
1990	De la mano de Tim Berners se da inicio a la construcción del primer navegador y editor para Internet denominado **WorldWideWeb**, de igual forma construyó el primer servidor Web denominado "**httpd**". Creó la primera versión del "Lenguaje de Etiquetado de Hipertexto" (**html**), lenguaje de formateo de documentos con enlaces de hipertexto que se convirtió en el formato de publicación principal para la Web. *Nota:* El primer servidor Web del mundo es propiedad del Laboratorio Europeo de Física de Partículas -CERN-.
1991	Marc Andreesen, estudiante del NCSA (National Certer for Supercomputing Applications) de la Universidad de Illinois, desarrolla MOSAIC, un navegador novedoso para Internet, que permitió a varios sistemas operativos tener acceso a Internet.
1992	El primer servidor fuera de Europa es configurado en la Universidad de Stanford, su primera publicación consistió en un documento acerca de partículas físicas del científico Paul Kunz.
1993	Se populariza el uso de navegadores para Internet, Microsoft retoma el proyecto Mosaic. La Web toma fuerza, múltiples aplicaciones y desarrollos en todo el mundo preparan el auge y la globalización de la gran red de redes.

La reseña anterior hace énfasis en las bases fundamentales de lo que se conoce hoy en día como Internet, fuente primaria de los desarrollos Web que se implementan en la actualidad. A partir de 1994 muchas empresas y personas particulares empezaron a trabajar de manera separada en la conformación de un nuevo paradigma de programación, el "Desarrollo Web", la evolución vertiginosa de este paradigma y la competencia por un mercado potencialmente lucrativo, evidenciaron la necesidad de establecer normas y políticas adecuadas para este nuevo modelo de trabajo. Es así como en 1994 fue creado un pequeño código en CGI que permitía ejecutar cierta cantidad de comandos, este sistema fue denominado "PHP - Personal Home Page" [7]. Si bien su uso era bastante complejo abrió las puertas a una nueva generación de desarrollos y el desarrollo Web se enriqueció entonces con la interacción de los formularios que se conocen hoy en día. Tres años más tarde, en 1997, y luego de un arduo trabajo en el desarrollo del "Service Pack 3.0" del servidor Windows NT,

[10] Graduado de la universidad de Oxford Inglaterra, responsable por el éxito mundial de 3com e investigador en informática e inteligencia artificial del Massachusetts Institute of Technology (MIT). Es considerado el inventor y el protector de la Web que se conoce hoy en día.

[11] Licenciado en física e ingeniero eléctrico del Massachusetts Institute of Technology (MIT) en 1971, y doctor en informática de la Universidad de California en 1982.

Microsoft presentó a la comunidad el concepto de ”ASP ó Active Server Pages”[8], el cual entró en competencia inmediata con el naciente lenguaje de secuencias PHP.

Al mismo tiempo que todos estos desarrollos se iban realizando, Tim Berners creó en el Instituto tecnológico Massachussets y con el apoyo del CERN el consorcio “W3C - World Wide Web Consortium”, el cual buscaba estandarizar el uso de las nuevas tecnologías de desarrollo Web, y además lograr la interacción de cualquier sistema a través de la Web. En sus inicios, el trabajo de este consorcio fue definitivo, pues empresas como Microsoft trataron de crear sus propios estándares de desarrollo, lo cual contradecía la filosofía base de la Web de Tim Berners, la cual estaba orientada a la estandarización pero con fines de código abierto. La figura 3 ilustra la evolución y los principales hitos que se presentaron antes de la internacionalización de la W3C.

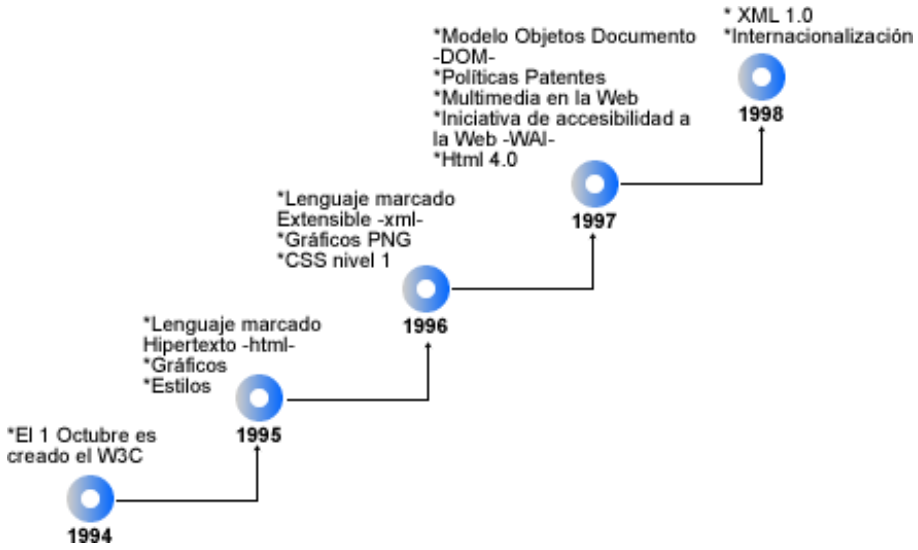


Figura 3. Hitos claves de la W3C antes de su Internacionalización.

Luego del reconocimiento mundial del consorcio W3C, de la necesidad de realizar esfuerzos conjuntos para lograr una verdadera interacción de los sistemas, y de la evolución acelerada en cuanto a aspectos técnicos y tecnológicos, la Web logra un nivel alto de madurez, y las empresas del mundo visualizan la gran red Internet como la oportunidad de negocios del siglo XXI. La figura 4 presenta un resumen del trabajo del W3c durante los últimos años, en el cual se evidencia claramente la evolución de los sistemas Web.

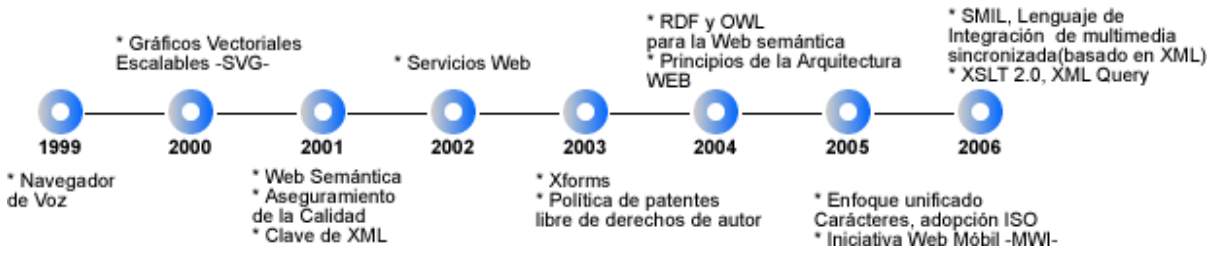


Figura 4. Hitos claves de la W3C en la evolución de sistemas Web.

7.2 Las bases de datos y su contribución al Desarrollo Web

En sus inicios, la Web no disponía de herramientas para acceder a bases de datos, por ende las aplicaciones iniciales fueron estáticas y requerían de un mantenimiento continuo para garantizar una información actualizada. Rápidamente fue creado el CGI (Siglas en inglés de "Common Gateway Interface"), este esquema de desarrollo aunque muy rudimentario permitió acceder a una Web dinámica, en donde elementos como archivos, correos, formularios entro otros podían interactuar. Es así como la Web deja de ser estática y se da inicio a la era dinámica de la Web. Empresas como Netscape y Microsoft colaboraron en la construcción de interfaces para servidor, la cuales optimizaron ostensiblemente los tiempos de respuesta y el acceso a los nuevos objetos en la Web.

Los repositorios iniciales de la Web fueron archivos planos, sin embargo, todo este crecimiento acelerado de la Web provocó que empresas como Oracle, DB2, Microsoft, Sybase y nuevas propuestas de bases de datos en código abierto evolucionaran casi a la misma velocidad de la Web. Este crecimiento no fue exclusivo de las bases de datos relacionales, Computer Associates – CA, creó una base de datos cien por ciento orientada a objetos denominada "JASMINE" la cual ofrecía conectividad con la Web a través de un lenguaje de programación bastante limitado, sin embargo las limitaciones de este lenguaje de programación llevaron al fracaso a esta base de datos. Entendida la problemática anterior por los proveedores de sistemas de gestión de bases de datos, éstos se concentraron en potenciar sus bases para la Web de datos en tres aspectos:

1. Optimización y tiempos óptimos de respuesta
2. Carga y volúmenes grandes de información
3. Herramientas para el mantenimiento y seguridad

Entonces, la Web dinámica toma forma gracias a la contribución de tres grandes frentes, por un lado la evolución de los sistemas de bases de datos, por el otro, los avances significativos en los lenguajes de programación Web que interactúan con dichas bases de datos y por el otro, el esfuerzo de la W3C por unificar y estandarizar de manera adecuada todas las iniciativas para contribuir al crecimiento paulatino de la Web. La figura 5 presenta el esquema general que se utiliza en la Web para el acceso a bases de datos.

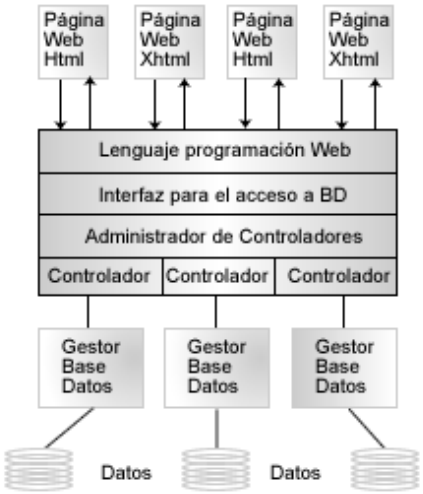


Figura 5. La Web dinámica y las bases de datos

7.3 Los lenguajes de programación Web

Los lenguajes de programación para la Web se pueden dividir en dos grande grupos: Lenguajes para Servidor y Lenguajes para cliente. Los lenguajes para servidor reciben peticiones de un cliente, dichas peticiones son interpretadas línea a línea por el servidor y luego de procesarlas envían una respuesta al cliente. Por su parte los lenguajes para cliente solo gestionan información en el cliente de tal manera que todo se procese antes de ser enviada al servidor. La figura 6 permite identificar los dos tipos de lenguajes.

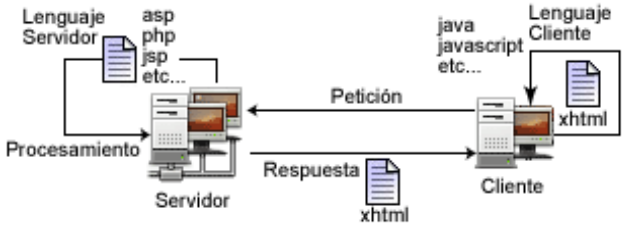


Figura 6. Esquema de lenguajes para programación Web

7.3.1 Lenguajes para cliente

La tecnología de scripting[12] mejoró con la aparición de nuevos lenguajes de programación interpretados basados en los dos entornos dominantes, VBScript y JScript por parte de Microsoft y JavaScript por parte de Netscape. La siguiente tabla resume la información más relevante para estos lenguajes:

Tabla 12. Lenguajes de programación Web para cliente

Empresa	Lenguaje	Descripción
Microsoft	VBScript	Es una implementación de Visual Basic para crear aplicaciones cliente exclusiva para Internet Explorer.
	JScript	Es el nombre de Microsoft para la implementación del lenguaje de scripting basado en Java, ofreciendo mayores prestaciones que VBScript a cambio de una mayor complejidad.
Netscape	LiveScript	Denominado “Mocha” en su versión inicial, es el padre del Javascript que se reconoce hoy en día.
	JavaScript	No es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de Herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.
ECMA[13]	ECMAScript	Es el lenguaje de scripting que soporta el estándar ECMA-262, basado en JavaScript con adicionales retoques para los navegadores de Microsoft y Netscape.

7.3.2 Lenguajes para servidor

Los lenguajes para servidor generalmente compilan las instrucciones cada vez que éstas son requeridas, en este instante son ejecutadas una por una hasta que el script ó la secuencia de comandos finaliza, esta es la gran diferencia con los lenguajes de programación tradicionales como C++ en donde el código es compilado y convertido en código ejecutable para después ser ejecutado.

El proceso de compilación para generar códigos binarios es una de las fortalezas de estos lenguajes, pues al aplicar un cambio este se ve reflejado de manera inmediata en toda la aplicación Web sin necesidad de precompilar todo el código. La siguiente tabla presenta una breve descripción de los lenguajes de programación para servidores Web más representativos.

[12] Es una tecnología que utiliza secuencias o unidades de código, las cuales se procesan e interpretan en un servidor o cliente sin necesidad de estar compilados.

[13] European Computer Manufacturers' Association, es un organismo internacional aunque su nombre indique que es europeo.

Tabla 13. Lenguajes de programación Web para servidor

Lenguaje	Descripción
Cold Fusion	Este lenguaje se hizo popular con la "explosión de las *.COM*"[14] debido a la simplicidad en la interfaz de desarrollo, pues estaba orientada hacia desarrolladores con poco conocimiento de programación. Aunque el rendimiento de una aplicación desarrollada en este lenguaje no es el óptimo, la velocidad con la cual un desarrollador crea una aplicación es muy rápida. Su gran fortaleza es a la vez su gran debilidad, la interfaz amigable para los programadores novatos es agradable y práctica, sin embargo, para los desarrolladores expertos esto es una camisa de fuerza, pues no disponen de la flexibilidad y potencia de otros lenguajes similares en donde los desarrolladores tiene el control de lo que sucede y el porqué sucede. No es recomendado para el desarrollo de aplicaciones de misión crítica, por ende este lenguaje ha sido relegado a los programadores que están en un proceso de desarrollo Web aun incipiente. Su lenguaje CFML (ColdFusion Markup Language) es fácil de aprender, pero limita al desarrollador a realizar las cosas de una única forma.
ASP net	Las páginas activas de servidor fueron la primera respuesta de Microsoft para ingresar al mercado del Desarrollo Web, durante 6 años, Microsoft recopiló la experiencia de los desarrolladores que había utilizado este lenguaje, y lo integró con buenas practicas de desarrollo software, lo cual dio como resultado un lenguaje más robusto y estable, el cual se conoce hoy en día con el nombre de ASP.NET. Por su carácter propietario[15], ASP y ASP.NET han recibido muchas críticas, sin embargo su rapidez, facilidad de aprendizaje y las herramientas que Microsoft suministra para soportar el desarrollo Web, hacen de este lenguaje una excelente estrategia para el desarrollo Web. En la actualidad, ASP.NET se ha convertido en algo más que un lenguaje de programación de servidor, ha evolucionado en un marco de trabajo que le permite a los usuarios desarrollar aplicaciones para la Web utilizando la sintaxis de múltiples lenguajes de programación como por ejemplo: Visual Basic, C#, J# entre otros. Entre sus mayores ventajas están: 1. El acoplamiento con casi cualquier otro producto Microsoft. 2. La velocidad de procesamiento cuando se utiliza adecuadamente en un servidor Windows con el IIS (Internet Information Server). 3. Los múltiples lenguajes de programación que brinda para realizar los desarrollos. 4. Compatibilidad con los modelos de objetos definidos por la W3C. Su mayor debilidad no radica propiamente en alguna característica del lenguaje o marco de trabajo, sino en la competencia, pues otros lenguajes como PHP y JSP siendo de código abierto permiten realizar las mismas funcionalidades con un mejor rendimiento y practicidad.

[14] Esta expresión se acuñó a finales de 1998 y hasta el año 2000, para expresar del número de empresas que ofrecían productos o servicios en la Web.

[15] También llamado software no libre o software privativo. Esta expresión se refiere a cualquier programa informático en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo.

PHP	Es un lenguaje de programación de secuencias usado generalmente para la creación aplicaciones Web comerciales y de misión crítica. Usa una mezcla entre interpretación y compilación que permite a los desarrolladores una combinación excelente entre rendimiento y flexibilidad. Por si solo PHP es tratado como un lenguaje de secuencias para expertos, en donde es necesario tener conocimiento profundo del lenguaje para desarrollar aplicaciones robustas, sin embargo, con la ayuda de un programa escrito en JAVA denominado ZEND STUDIO este lenguaje se encuentra a la altura de otros de su mismo tipo como es el Caso de ASP.NET. Entres sus fortalezas se encuentran : 1. Velocidad de respuesta. 2. Conectividad con casi cualquier base de datos que de soporte a la Web. 3. Millones de servidores instalados en el mundo que brindan alojamiento para las aplicaciones desarrolladas en este lenguaje. 4. La metodología utilizada para el cambio de versiones, la cual no implica la adición de parches sino por el contrario la reescritura de todo el motor de desarrollo. 5. Es multiplataforma. Aunque existen miles de componentes desarrollados en PHP, el desarrollador tiene que generar una competencia adicional a la hora de programar, ésta se orienta hacia la capacidad de reutilización de código, lo que implica que un desarrollador profesional de PHP tenga una disciplina mucho más estricta que otros desarrolladores que usan lenguajes diferentes para la Web.
JSP	JSP es el acrónimo de Java Server Pages, lo que traduce al español “Páginas de Servidor Java”. Es un lenguaje de secuencias orientado a crear páginas Web con programación Java. Al igual que PHP, JSP es multiplataforma lo cual hace de este lenguaje una excelente elección a la hora de emprender desarrollos Web. Este lenguaje utiliza una estrategia de desarrollo denominada “JavaBeans”, lo que consiste es separar totalmente la lógica de presentación y la lógica de acceso a datos. Es un concepto similar al de abstracción de bases de datos que usa PHP con las librerías de acceso a datos. Entre las ventajas de este lenguaje están: 1. Multiplataforma. 2. La estructuración de capas en el desarrollo de aplicaciones. 3. La exigencia de un estilo de programación estandarizado. 4. El mantenimiento de los desarrollos, pues su programación tiene una estructura básica que deben manejar todos los desarrolladores. Una debilidad identificada en este lenguaje de programación de secuencias es su velocidad, la cual está muy por debajo de otros lenguajes como PHP y ASP.net, una de las razones posibles es la necesidad de hacer de este un lenguaje multiplataforma, y la otra puede ser la estrategia de compilación de las paginas JSP, sin embargo son conjeturas no confirmadas que hacen los desarrolladores expertos.
PERL	Es un lenguaje eficiente para la extracción y reporte de datos, fue creado en 1987 lo cual incide en que sea el lenguaje preferido por lo antiguos desarrolladores de sistemas cliente servidor tradicionales. En la actualidad se considera un aliado estratégico de otros leguajes de programación como es el caso de PHP y JSP.

	Su mayor ventaja es la gran cantidad de desarrollos del tipo código abierto (ver CPAN[16]) que existen, los cuales pueden ser realizados en otros desarrollos. Su mayor defecto radica en la costumbre de los desarrolladores que usan este lenguaje por "empaquetar" varias funcionalidades en una sola línea, lo cual contradice muchos de los estándares para desarrollo de código que existen en la actualidad.
Phyton	Es un lenguaje de programación creado por Guido Van Rossum en el año 1990. Es comparado habitualmente con TCL, PERL, Scheme, Java y Ruby. En la actualidad Python se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation. Python es considerado como la "oposición leal" a PERL, lenguaje con el cual mantiene una rivalidad amistosa. Los usuarios de Python consideran a éste mucho más limpio y elegante para programar. Python permite dividir el programa en módulos reutilizables desde otros programas realizados con el mismo lenguaje. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender el lenguaje). También hay módulos incluidos que proporcionan E/S de ficheros, llamadas al sistema, sockets y hasta interfaces a GUI (interfaz gráfica con el usuario) como Tk, GTK, Qt entre otros Aunque es un lenguaje de programación interpretado, no tiene la potencia de otros lenguajes como ASP.net, PHP ó JSP, sin embargo es una referencia obligada cuando se habla de lenguajes de programación para la Web. Su principal fortaleza es la interacción con lenguajes y herramientas para cliente, y su principal debilidad es la falta de un marco de trabajo que le permita al desarrollador implementar aplicaciones Web robustas.

7.4 Tendencia para el desarrollo Web

Es indiscutible que existe una relación directa entre los tipos de lenguajes de desarrollo Web, los lenguajes cliente necesitan de los lenguajes de servidor para que se procese una solicitud ó petición, y así mismo los lenguajes de servidor requieren procesos de validación antes que la información se envíe al servidor.

Durante muchos años, el desarrollo en la Web se orientó al modelo tradicional expuesto en la figura 6 "Esquema de lenguajes para programación Web", casi la totalidad de los desarrollos Web se realizaban bajo esta filosofía, la cual planteaba que las solicitudes de los clientes se hacían por páginas completas y no por segmentos de la misma, la página era procesada en el servidor y luego la información era entregada al cliente en formato html ó xhtml. Este modelo descrito cambió sutilmente con la interacción del lenguaje de

[16] Es un repositorio de librerías bastante amplio, y con una gran cantidad de código que puede ser reutilizado y/o personalizado por otros desarrolladores.

programación cliente -JavaScript- y del lenguaje para intercambio de información que ha revolucionado el desarrollo Web XML, En la figura 7 se puede apreciar la sutil diferencia entre el modelo clásico y el que actualmente está siendo utilizado para el desarrollo de las aplicaciones Web.

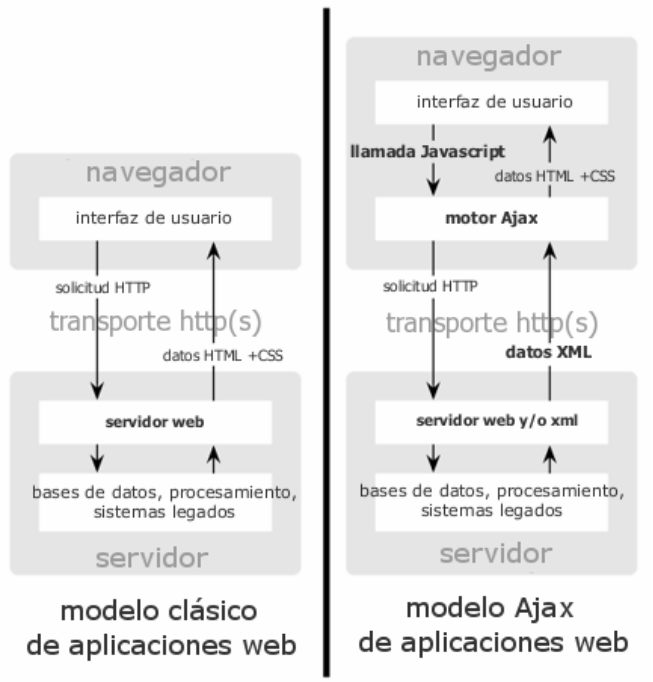


Figura 7. Comparación modelo clásico Web y las nuevas tendencias[17]

La tendencia actual ha sido denominada AJAX (*Acrónimo de Asynchronous JavaScript and XML*) lo que significa javascript y XML asíncronos. Esta tendencia no es una tecnología como tal, por el contrario es la combinación de tres (3) tecnologías ya existentes:

Tabla 14. Tecnologías que componen AJAX

Tecnología	Descripción
Xhtml y/o html	Xhtml es la normalización realizada por la W3C del lenguaje de marcado de hipertexto, su finalidad es asegurar que los desarrolladores construyan páginas semánticamente bien estructuradas, de tal manera que el paso hacia XML sea transparente. Para la creación de efectos visuales, se soporta en las hojas de estilo en cascada -CSS-.
Javascript, ECMAScript,	Este lenguaje de programación cliente permite interactuar dinámicamente con la información presentada en el navegador, así mismo permite acceder al modelo

[17] Figura tomada de www.w3c.es

JScript	objeto de documento -DOM- de cada una de las páginas Web, de esta forma se pueden hacer llamados al objeto XMLHttpRequest[18] para intercambiar datos asincrónicamente con el servidor Web.
XML	El lenguaje de marcas extensible -XML- (sigla en inglés de eXtensible Markup Language), es un metalenguaje de etiquetas desarrollado por el W3C, con el cual se permite definir la gramática de nuevos lenguajes, entre los ejemplos más importantes de aplicaciones de XML están el Xhtml y el SVG. El objetivo fundamental de XML es convertirse en un estándar para el intercambio de información estructurada entre diferentes plataformas, incluso utilizando bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa que sirva como repositorio de datos. Aunque AJAX utiliza XML para la transferencia de datos con el servidor, cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON[19] y hasta EBML[20].

Ajax como tendencia actual, se da a conocer en el año 2005, sin embargo, este estilo de desarrollo nació en 1998 con la iniciativa de Microsoft de crear el denominado "Scripting Remoto" y con la utilización de un elemento ó etiqueta html denominada "IFRAME", el cual solo estaba disponible para las versiones del navegador Internet Explorer. Finalmente fueron los grupos de noticias y en general la comunidad de desarrolladores los que impulsaron la creación del objeto "XMLHttpRequest" el cual es requerido en los navegadores actuales para dar soporte a esta tendencia mundial [9].

7.4.1 El desarrollo por componentes en la Web

Entre las definiciones más relevantes de componentes se encuentran:

> "*Es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio*" [10].

---

[18] Es una interfaz que permite transferir y manipular datos XML desde y hacia un navegador Web, estableciendo un canal de comunicación alterno con el servidor sin necesidad de procesar la totalidad de la pagina Web en la cual es utilizado.

[19] JSON, acrónimo de "JavaScript Object Notation", es un formato ligero para el intercambio de datos el cual no requiere el uso de XML.

[20] EBML, acrónimo de Extensible Binary Meta Language (Meta Lenguaje Binario Extendible), fue diseñado como una extensión binaria simplificada de XML, con el propósito de almacenar y manipular datos de forma jerárquica con campos de longitud variable.

> "*Una parte física reemplazable de un sistema que empaqueta su implementación, y es conforme a un conjunto de interfaces a las que proporciona su realización*" [6].

La construcción de fragmentos de código que pudieran ser utilizados en diferentes sistemas bajo diferentes condiciones, fue la premisa fundamental del desarrollo por componentes. Con el progreso de las interfaces de programación de aplicaciones (API por sus siglas en inglés), la creación de estos pequeños programas a nivel de sistemas tipo cliente se incrementó de manera exponencial, entre los componentes más utilizados estaban los controles VBX y las tecnologías OLE de Microsoft.

El desarrollo de aplicaciones basado en componentes permite la posibilidad de construir un sistema de la misma forma que se ensambla un componente electrónico, es decir a partir de piezas ya prefabricadas. Como base fundamental para fabricar estas piezas, los desarrolladores usaron la programación orientada a objetos como la estrategia para definir y especificar posibles componentes, pues ésta permite construir sistemas de alta complejidad y fácil mantenimiento, sin embargo, no permite por si misma separar cada objeto de manera independiente, de tal forma que pueda ser reutilizado y acoplado a otro sistema. Es entonces cuando nace la programación orientada a componentes conocida como *POC* [11], la cual es una extensión natural de la programación orientada a objetos pero que permite definir y especificar la forma como un componente puede interactuar con otro.

El auge de la programación en la Web y la evolución acelerada de la misma, llevaron a las empresas y desarrolladores a implementar desarrollos bajo el modelo de *COTS* (Siglas en inglés de *commercial off-the-shelf*), este concepto que cambió los clásicos y típicos desarrollos, en donde cada empresa desarrollaba y era propietaria de lo suyo, permitió a los desarrolladores Web realizar especificaciones más precisas y diseños más ajustados a la realidad. A pesar de ser un paso adelante en el desarrollo software trajo múltiples implicaciones, entre ellas la complejidad en la búsqueda y reconocimiento de los componentes realmente necesarios, la adaptación de los mismos y la resolución de solapamientos entre las funciones que un componente podía ofrecer.

En la actualidad los componentes de servidor ActiveX y JavaBeans son dos de las principales alternativas para solventar el problema de la portabilidad en el uso de componentes [12]. No se trata de modelos específicos de servidor, ni siquiera de productos de servidor, sino que representan un medio de empaquetar software en forma de componentes para luego enlazarlos o incrustarlos en páginas Xhtml mediante el uso de scripts. Existe otra iniciativa para el manejo de componentes y es la que se está imponiendo con el uso y desarrollo de código abierto, en donde las implementaciones son empaquetadas en clases y la interfaces de conexión entre el componente y otros sistemas es escrita en código script o en secuencias de comando, esto permite a los usuarios reutilizar un componente aunque este no sea binario, y mejor aún adecuar el componente a una necesidad puntual pues el tipo de licenciamiento del mismo así lo permite. La siguiente tabla presenta los modelos de componentes para la Web más importantes en la actualidad.

Tabla 15. Modelos de desarrollo por componentes para la Web

Empresa	Producto	Descripción	Perspectiva
Microsoft	Activex	Es un entorno de trabajo para el desarrollo de documentos compuestos con controles ActiveX, escritos en los lenguajes de programación que provee el marco de trabajo de Microsoft, por lo tanto la portabilidad es una de sus debilidades.	✓ Empaquetamiento[21] ✓ Integridad[22]
SUN	JavaBeans	Es un entorno de trabajo para el desarrollo de aplicaciones uniendo componentes Java (Beans). Un Bean es similar a un control ActiveX, pero el Bean al estar escrito en Java, tiene la seguridad y la portabilidad asociadas a Java (Multiplataforma).	✓ Empaquetamiento ✓ Servicio[23]
Código Abierto	Clases e interfaces	Este entorno permite la construcción de clase de escritura y lectura al igual que lo hace un Bean, sin embargo, este código no está empaquetado como control, por lo tanto su fuente puede ser modificada a discreción del usuario que está reutilizándolo. Las interfaces son	✓ Integridad ✓ Servicio

[21] Componente como una unidad de empaquetamiento y distribución. Bajo esta perspectiva, el componente está pensado para distribuirse en un formato binario.

[22] Considera al componente como una cápsula de implementación, que mantiene la integridad de los datos que en él se gestionan. Por tanto, es independiente de la implementación de otros componentes. Permite la sustitución de componentes.

[23] Considera al componente como una entidad que ofrece servicios a sus clientes. Los servicios se agrupan en interfaces, las cuales describen lo que un cliente potencial espera de los servicios del componente.

		construidas a nivel de métodos, lo cual permite que los desarrolladores aprendan del código mientras lo ensamblan con nuevos componentes.	

En la actualidad, y gracias en parte a la filosofía de código abierto (en inglés *Open Source*) impulsada por el W3C, los desarrolladores que contribuyen a este movimiento se han involucrado directamente en la generación de código reutilizable, o lo que se puede decir en otras palabras, se han dedicado al desarrollo de código por componentes fundamentado en funcionalidades. Lo que hace unos años era exclusividad de Microsoft con la implementación, venta y uso de controles, es hoy en día una realidad con otros lenguajes de programación para la Web, siendo los más populares los lenguajes para servidor JSP y PHP.

8 CRITERIOS DE CLASIFICACIÓN DE SISTEMAS WEB

Antes de realizar una caracterización en términos generales, se puede considerar importante conocer el pasado inmediato, el presente real y el futuro posible del objeto de estudio. En el capítulo anterior se presentó la evolución de la Web, la tendencia actual de los desarrollos y una perspectiva de lo que al corto plazo serán las implementaciones para la Web, en este capitulo se presenta la tipología de sitios Web, así como la justificación de los criterios de clasificación identificados y utilizados en el presente trabajo de investigación.

8.1 Análisis de criterios para la clasificación de sistemas Web

Para realizar una caracterización es clave identificar los elementos del dominio de acuerdo a un conjunto de criterios claros y que sirvan realmente para agrupar, de esta forma se puede analizar como se comporta un elemento frente a un criterio, e incluso comparar el comportamiento entre los elementos del dominio. El dominio de la presente caracterización son los desarrollos de sistemas Web y el objeto de la misma es encontrar una clasificación de sitios Web que permita identificar características comunes, utilizando para ello los criterios establecidos en dicha caracterización.

El marco de referencia con respecto a los posibles criterios de clasificación que se pueden tener en cuenta para una taxonomía Web, son tomados en parte del Doctor Aníbal de la Torre en "Web educativa 2.0"[24] [13] y del artículo Tipología de sitios Web de la Doctora Gemma Ferreres[25] [14].

8.1.1 Modo de acceso - estado

El modo de acceso hace referencia a la posibilidad que tiene un sitio Web de permitir la lectura y escritura sobre el código o páginas del sitio. En sus inicios la Web solo ofrecía sitios estáticos, sin embargo, los sitios actuales "deben" ser dinámicos, pues el volumen de información que se maneja implicaría una cantidad exagerada de recurso humano para mantener actualizada la información del sitio, por lo tanto, económicamente este tipo de

[24] Fuente "Web Educativa 2.0": Aníbal de la Torre, Coordinador TIC I.E.S. http://www.uib.es/depart/gte/gte/edutec-e/revelec20/anibal20.htm, revisado el 30 de abril de 2007

[25] Periodista en la Universidad Complutense de Madrid, especialista e-learning de la UNED y Master en Sistemas de Información del ISE. Durante cinco años fue autora del blog tintachina. Es co-autora del libro La blogosfera hispana. Pioneros de la cultura digital, publicado por la Fundación France Telecom.

aplicaciones Web se hacen inviables en la actualidad. Este criterio permite clasificar los desarrollos Web en dos grupos:

Tabla 16. Clasificaciones según criterio modo de acceso

Criterio	Clasificación
Modo de acceso	1. Solo lectura
	2. Lectura - escritura

Si bien es cierto, esta clasificación es válida, es poco significativa para el nivel de agrupamiento que requiere esta caracterización, además, los desarrollos actuales permiten un modo de acceso dual, es decir lectura y escritura, por lo tanto este criterio debe ser analizado en forma grupal con otros criterios, púes por si solo no permite emitir una conclusión contundente.

8.1.2 Mínima Unidad de contenido

La mínima unidad de contenido es la expresión más pequeña que se utiliza para presentar información a un usuario en la Web. En sus inicios, la Web solo tenía una unidad y era la página completa. En la actualidad y con el uso de las nuevas tecnologías esta unidad se ha diversificado, en general por la facilidad para estructurar contenidos en una página, y más específicamente en la forma como los desarrolladores construyen las aplicaciones Web actuales.

Tabla 17. Clasificaciones según criterio mínima unidad de contenido

Criterio	Clasificación
Mínima unidad de contenido	1. Mensajes
	2. Artículos
	3. Segmentos de página
	4. Marcos dinámicos

Este criterio brinda una agrupación un poco más amplia que la entregada por el modo de acceso, sin embargo la inclusión de un determinado sitio Web en esta clasificación puede llegar a ser es muy subjetiva y a la vez compleja. Subjetiva por que sino se conoce en detalle la forma como fue desarrollada la aplicación no podremos determinar a ciencia cierta la forma como están siendo usadas las unidades de contenido, por otro lado si fuera posible conocer esta información (la forma cómo se desarrolló el sitio Web), es muy probable que no todos accedan a brindarla. Este criterio de clasificación aunque complejo en su forma de categorizar los sitios Web, puede ser utilizado siempre y cuando la persona que realiza el

análisis del sitio Web tenga conocimiento amplio y la experiencia suficiente en el desarrollo de aplicaciones Web para poder identificar los elementos claves de este criterio.

8.1.3 Modo de visualización

Este criterio permite agrupar las aplicaciones y sitios Web dependiendo de las alternativas que éstos ofrecen para la visualización de la información. Tradicionalmente el acceso a los sitios Web se realiza a través de los navegadores, en donde la información es visualizada en la página Web que generan las aplicaciones, sin embargo existen otros medios o formas de visualizar el contenido de un sitio como son:

Tabla 18. Clasificaciones según criterio modo de visualización

Criterio	Clasificación
Modo de visualización	1. Navegador Web
	2. Lectores RSS[26]
	3. Mensajes de texto

Este criterio planteado por Aníbal de la Torre en "Web educativa 2.0" es un criterio válido para la caracterización. Sin embargo, al diferenciar los sitios Web por medio de este criterio, lo que se analiza realmente es la condición: "¿Qué servicios de visualización presenta el sitio Web? Lo cual conduce a que el verdadero criterio de clasificación no sea la forma de ver la información, sino si posee o no un componente que permita visualizar dicha información.

8.1.4 Temáticas y/o contenidos

En sus inicios, la información que se publicaba en la Web era de carácter especializado (*no queriendo decir con esto que fuera veraz*), y hacía referencia principalmente a un tema de estudio, cada recurso estaba orientado a satisfacer una demanda puntual del ínternauta, los sitios Web de universidades, docentes y/o grupos de investigación llegaron a ser muy populares, pues trataban temas de interés para cada persona. La creciente necesidad de información llevó a las grandes empresas informáticas (*Netscape y Yahoo como líderes*) a crear sitios Web dinámicos que ofrecieran información con temáticas y contenidos orientadas a cautivar casi cualquier tipo de usuario. De esto nace el concepto de "**PORTAL WEB**", el cual es considerado como "una puerta de acceso a información en la Web". La clasificación

[26] Por sus siglas en inglés: Really Simple Syndication, significa "Distribución realmente sencilla" , hace parte de la familia de los formatos XML desarrollado específicamente para todo tipo de sitios que se actualicen con frecuencia y por medio del cual se puede compartir la información y usarla en otros sitios Web.

de sitios Web por su tema o su contenido deja de tener importancia y entonces aparecen las dos clasificaciones más comunes de portales: Horizontales y verticales:

Tabla 19. Clasificaciones según criterio temáticas

Tipo Portal	Descripción
Horizontales	También conocidos como mega portales, son de carácter general y están orientados a todo tipo de público. Ofrecen contenidos de carácter muy amplio, cuya pretensión es cubrir las temáticas más demandadas. Estos sistemas disponen de servicios de valor agregado que agrupan a usuarios en comunidades virtuales, ofreciéndole al ínternauta todo lo que éste necesita, logrando lo que en marketing se conoce como "El cliente Fiel".
Verticales	También conocidos como Vortal (*Vertical Portal*), son sitios Web que proveen información y servicios a un público en particular. El servicio de valor agregado más importante de este tipo de sitios es la estructuración de contenidos particulares sobre un tema. Cuando un portal Horizontal no tiene éxito para captar un ínternauta, estos portales son los llamados a realizar esta labor, pues la posibilidad de profundizar en los contenidos requeridos por el usuario es bastante alta, además de esto, ofrece los servicios tradicionales de un portal en general.

Aunque el análisis del criterio de contenidos y temáticas como tal, no permite hacer una clasificación rigurosa de los sitios Web, si permite identificar claramente un tipo de sitio Web clave: "***Los portales***", ahora bien, los portales se hacen más o menos útiles dependiendo de la cantidad de servicios de valor agregado que ofrezcan a sus usuarios, esta idea resalta la conclusión expresada en el criterio "modo de visualización", en donde se expresa la importancia de clasificar un sitio Web por los servicios y/o componentes que el sitio le pueda brindar a un usuario.

8.1.5 Funcionalidades

Antes de analizar este criterio para clasificar un sistema Web, es apropiado contextualizar su significado. Las siguientes son algunas definiciones del término *funcionalidad*:

Tabla 20. Definiciones del término funcionalidad

Definición	Fuente
Conjunto de características que hacen que algo sea práctico y útil.	Diccionario de la lengua española, 2007.
Coherencia entre las necesidades detectadas y los resultados que se obtienen con el uso del material.	Glosario de tecnología educativa, 26 agosto de 2005. Pere Marqués.
Agrupación de atributos que hacen referencia a la existencia de un	Norma ISO 9126.

conjunto de funciones y sus características. Las funciones son aquellas que satisfacen una necesidad especificada.	

La funcionalidad entonces, se puede definir como el conjunto de opciones que han sido especificadas para un sistema software, siendo estas opciones útiles para el usuario final. Teniendo en cuenta las definiciones de otros autores, y la expresada anteriormente, se puede concluir que este criterio de clasificación busca agrupar los desarrollos para sitios Web bajo las siguientes perspectivas:

1. Objetivo para el cual fueron construidos (*Perspectiva 1*).
2. Servicios que ofrecen (*Perspectiva 2*).

Luego del análisis de los criterios especificados en esta sección para la clasificación de sitios Web, y tomando como base las clasificaciones realizadas por Doctor Aníbal de la Torre y la Doctora Gemma Ferreres, y analizando las dos perspectivas de clasificación a nivel funcional, se puede construir una clasificación como la que se presenta a continuación:

Tabla 21. Clasificaciones según criterio funcionalidad

Clasificación	Perspectiva 1	Perspectiva 2
Blogs	Estos sitios Web fueron creados con el fin de presentar contenidos en orden cronológico y con una baja tasa de tiempo para la actualización de la información. La diferencia con un sitio Web normal radica en que los usuarios que utilizan el sistema no requieren conocimientos previos sobre html o páginas Web. Además, disponen de un conjunto de servicios definidos que encapsulan toda la complejidad del sistema.	✓ Validación de usuarios registrados. ✓ Creación y edición de contenidos. ✓ Publicación de contenidos. ✓ Seguimiento a contenidos presentados. ✓ Interfaz para la creación de páginas Web. ✓ Plantillas para la presentación de contenidos.
Buscadores	El crecimiento acelerado de la publicación de información en la Web, motivó la creación de sistemas Web que permitieran filtrar la información que el usuario desea encontrar. Con una interfaz simple, los buscadores son puentes entre el usuario y la información que requiere.	✓ Búsqueda de información: o Básica o Avanzada ✓ Traducción de páginas encontradas. ✓ Enlace y agrupación de servicios. ✓ Interfaz Web y escritorio para las búsquedas.
Comercio electrónico	Estos sitios son aquellos que permiten al usuario realizar transacciones electrónicas. Su objetivo principal son las ventas, compras, transferencias y en general casi cualquier operación financiera que se pueda realizar por la Web.	✓ Validación de usuarios registrados. ✓ Acceso a servicios de valor agregado: o Saldos o Movimientos o Transacciones electrónicas

Catálogos	Fueron creados con el fin de permitirle a las empresas, el ofrecimiento de sus productos y/o servicios al mercado globalizado que existe hoy en día.	✓ Creación y edición de productos. ✓ Publicación de productos. ✓ FAQ´s: Sistema de preguntas y respuestas comunes de los clientes.
Educación en línea	Su objetivo principal es facilitar la interacción entre docentes y estudiantes, permitiendo la presentación, valoración y seguimiento a las actividades que se pueden llevar a cabo en un proceso de enseñanza aprendizaje.	✓ Validación de usuarios registrados. ✓ Administración de usuarios. ✓ Creación y edición de contenidos. ✓ Publicación de contenidos. ✓ Seguimiento a contenidos presentados. ✓ Evaluaciones en línea. ✓ Foros de discusión. ✓ Clasificados.
Intranets y Extranets	Estos sistemas Web tienen como función facilitar la interacción de las personas que tienen vínculos con una empresa. Los usuarios son registrados en el sistema y realizan las operaciones para lo cual tienen permisos de acceso. Aunque en cuanto a funcionalidad Intranets y Extranets son iguales, la diferencia radica en que las primeras son para empleados y personal interno de una compañía, mientras que la segunda es más abierta y permite el acceso a clientes, proveedores y en general los usuarios que a su juicio considere la empresa.	✓ Validación de usuarios registrados. ✓ Administración de usuarios. ✓ Correo electrónico. ✓ Foros. ✓ Clasificados. ✓ Gestión de documentos. ✓ Gestión de procesos institucionales de la organización.
Medios de comunicación	Son aquellos que permiten a los usuarios obtener información actualizada de la realidad nacional y mundial. Su aspecto es dinámico y visualmente agradable. Utiliza para la publicación de información múltiples herramientas y estrategias de tal manera que el ínternauta se sienta atraído por la simplicidad y actualidad del sitio Web.	✓ Creación y edición de contenidos. ✓ Publicación de contenidos. ✓ Interfaz para la creación de páginas Web. ✓ Plantillas para la presentación de contenidos. ✓ Múltiples servicios para la visualización de información.
Portales	Los portales son la evolución de los sistemas Web comerciales. Su objetivo principal es brindarle al usuario toda la información y utilidades que pueda necesitar, de tal manera que el usuario no necesite de otro sitio Web para satisfacer su necesidad de información y/o recursos.	✓ Validación de usuarios registrados. ✓ Administración de usuarios. ✓ Creación y edición de temáticas. ✓ Publicación de contenidos clasificados por temáticas. ✓ Correo electrónico. ✓ Foros temáticos. ✓ Clasificados. ✓ Encuestas. ✓ Múltiples servicios para la visualización de información.
Webs corporativas	Estos sitios Web permiten presentar la imagen institucional de una empresa en particular. Su objetivo principal es la	✓ Creación y edición de contenidos. ✓ Publicación de contenidos. ✓ Seguimiento a contenidos

	publicidad de los productos y/o servicios de una empresa. Se complementan con los sitios Web del tipo “Catálogos”.	presentados. ✓ Correo electrónico. ✓ FAQ´s: Sistema de preguntas y respuestas comunes de los clientes.

Los cinco (5) criterios analizados hasta el momento permiten agrupar en mayor o menor proporción los sistemas Web, si bien es cierto el criterio denominado “*funcionalidad*” es el que permite realizar una mayor clasificación, no se deben descartar los demás criterios, pues sirven como base para identificar las características que poseen los sistema de información para la Web en la actualidad.

Finalizado el análisis individual por criterios, se hace necesario el diseño y construcción de una prueba que permita verificar las conclusiones encontradas hasta el momento, analizando los desarrollos Web que existen en la actualidad a la luz de los criterios anteriormente expuestos. Como es sabido en la actualidad existe una gran cantidad de desarrollos Web, siendo imposible realizar un análisis de la totalidad de los mismos. Por tal motivo, la selección de una muestra de estudio representativa se convierte en una estrategia ideal para este trabajo de investigación. El siguiente apartado analiza la forma y método que fue utilizado para el cálculo del tamaño de la muestra.

9 ANÁLISIS ESTADÍSTICO PARA LA SELECCIÓN DE LA MUESTRA

La selección de la muestra para el análisis de los desarrollos Web con base en los criterios estudiados en este trabajo de investigación, se soporta en las siguientes razones:

1. El tamaño de la población real de sistemas Web es demasiado grande, y aunque no se puede concluir que tiende a infinito, se puede decir que este valor crece diariamente.
2. El costo de analizar la totalidad de sitios Web para realizar una caracterización con un margen de error mínimo sería muy elevado con respecto al beneficio a obtener, por ende, se puede trabajar con deducciones aproximadas producto del análisis de una pequeña parte de la población total.
3. El tiempo necesario para analizar todos los datos sería muy extenso, y dada la rapidez con la que evolucionan las nuevas tecnologías probablemente los resultados al finalizar el estudio no serían útiles.

Dado que el tamaño exacto de la población objeto de estudio es desconocido, es conveniente utilizar entre las metodologías para muestreo la que se conoce como "***Tamaño de la población infinito ó desconocido***".

Uno de los puntos claves de esta técnica de muestreo radica en el porcentaje de error que se está dispuesto a admitir, o en su defecto el porcentaje de error que se espera cometer, para este trabajo de investigación de define un porcentaje de error "***e***" del diez (10) por ciento. Este porcentaje de error es seleccionado, debido a que es el valor máximo permitido para el cálculo de la muestra. Si selecciona un valor menor del 10% el tamaño de la muestra aumentará de manera considerable, aumentando así el nivel de complejidad del estudio de muestreo. Irónicamente, aunque se desconoce el tamaño total de la población objeto de estudio, este método estadístico requiere que se asigne un valor porcentual a la población "***P***", por ende se toma como referencia el valor estadístico más pesimista el cual equivale al cincuenta (50) por ciento [15].

A continuación, se presenta la fórmula estadística utilizada para calcular el tamaño de la muestra:

$$n = \frac{P * Q * Z^2}{e^2}$$

La expresión, "***Q***" representa el complemento de la proporción muestral conocida, dado que se usa el valor más pesimista de "***P***", el valor de dicho complemento será del cincuenta (50) por ciento. "***Z***" corresponde al nivel de confianza de la muestra, se toma el valor correspondiente a la tabla muestral para Z(90) equivalente a 1.6449. La siguiente tabla resume los valores a utilizar en este proyecto de investigación para el cálculo de la muestra:

Tabla 22. Valores de las variables aplicadas en la formula

Variables	Valores
P	50%
P	50/100 = 0.5
Q	(1-0.5) = 0.95
Z	Z[90] = 1.6449
e	10/100 = 0.1
N	**68**

En la tabla 22, se puede apreciar el valor del tamaño de la muestra calculado, este valor indica que se deben analizar por lo menos sesenta y ocho (68) desarrollos Web, a la luz de los criterios analizados y definidos. Este valor mínimo representa el número de sitios Web que se hace necesario analizar para que la caracterización tenga validez investigativa.

10 CARACTERIZACIÓN DE SISTEMAS WEB

10.1 Construcción del Instrumento

Luego de todo el análisis realizado, y con miras a construir la caracterización formal de sitios Web, se procede a diseñar un instrumento que permita relacionar: los criterios analizados, la muestra seleccionada y una posible unidad de medida que evidencie la existencia o no de un criterio en cada uno de los sitios Web verificados.

Una de las claves para la construcción del instrumento radica en la adecuada relación que se pueda dar entre los criterios de clasificación de sitios Web. Dado que el criterio "***funcionalidad***" permitió realizar una agrupación heterogénea más amplia que los otros, se sugiere entonces realizar un cruce entre la "funcionalidad" y los demás criterios. Dicha relación puede tener la siguiente forma:

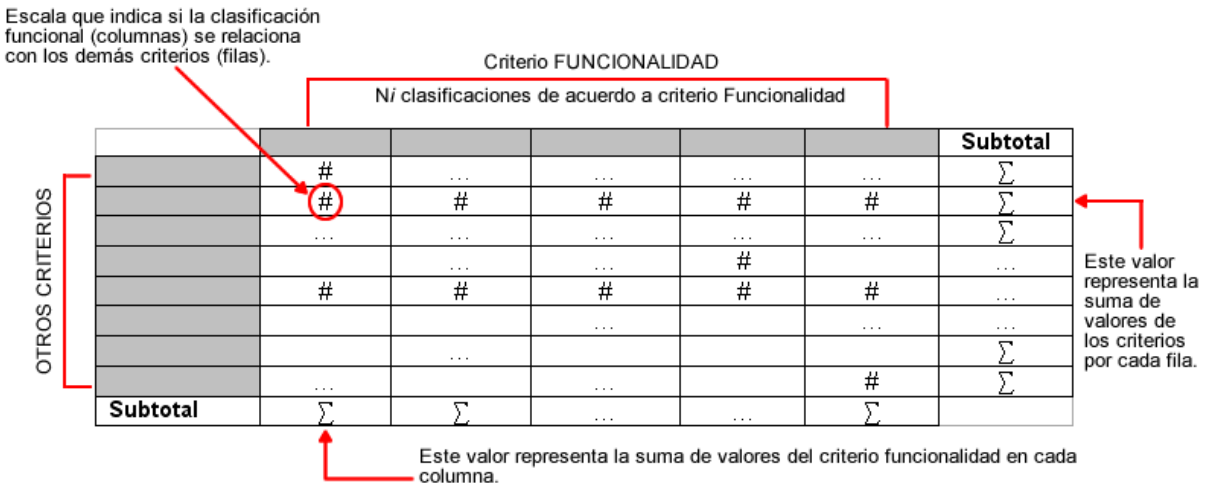


Figura 8. Esquema de convenciones del instrumento

La matriz de relación de criterios permite clasificar los sitios Web con base a los criterios analizados en este trabajo de investigación, sin embargo no todos los sitios Web que se analicen de la muestra encajarán en todas las agrupaciones, por ende es importante construir una escala que permita identificar cuales agrupaciones tienen mayor o menor proporción de sitios Web en su clasificación. La escala propuesta es la siguiente:

Tabla 23. Convenciones de las escalas del instrumento

Valores	Descripción
Tres (3)	Indica que la totalidad de los sitios Web están en esta clasificación.
Dos (2)	Indica que algunos sitios Web se encuentran en esta clasificación.
Uno (1)	Indica que muy pocos sitios Web se encuentran en esta clasificación.
Cero (0)	Indica que no hay sitios Web que se encuentren en esta clasificación.

10.2 Aplicación del Instrumento

10.2.1 Paso 1: Agrupación de Sitios por Funcionalidad

En el capitulo 3 se analizó y calculó el tamaño de la población objetivo, es decir se definió el número mínimo de Sistemas Web que sería necesario verificar para analizar los criterios de clasificación en el presente trabajo de investigación. La selección de los sistemas se realizó teniendo en cuenta sitios conocidos, sitios recomendados y por búsquedas en el sitio http://www.google.com.co. La tabla 24 presenta la cantidad de sitios que se encontraron en cada una de las clasificaciones que agrupa el criterio funcionalidad y establece una convención visual, de tal manera que el lector pueda verificar en el listado completo de sitios visitados (Tabla 25), de que tipo exactamente es el sitio Web relacionado.

Tabla 24. Resumen sitios visitados

Clasificación del Criterio Funcionalidad	Cantidad	Convención
Blogs	5	
Buscadores y Portales	13	
Comercio Electrónico	13	
Catálogos	4	
Educación en Línea	8	
Intranets y Extranets	3	
Medios de Comunicación	11	
Webs Corporativas	11	
TOTAL	**68**	

A continuación se presenta el listado completo de sitios verificados (ver tabla 25):

Tabla 25. Listado de sitios visitados

SITIOS VISITADOS	
1. http://www.solarhoteles.com/website/index.php	
2. http://www.mbas.es/	
3. http://www.elpais-cali.com/	
4. http://www.motorola.com/	
5. http://www.canon.com	
6. http://dyna.unalmed.edu.co/	

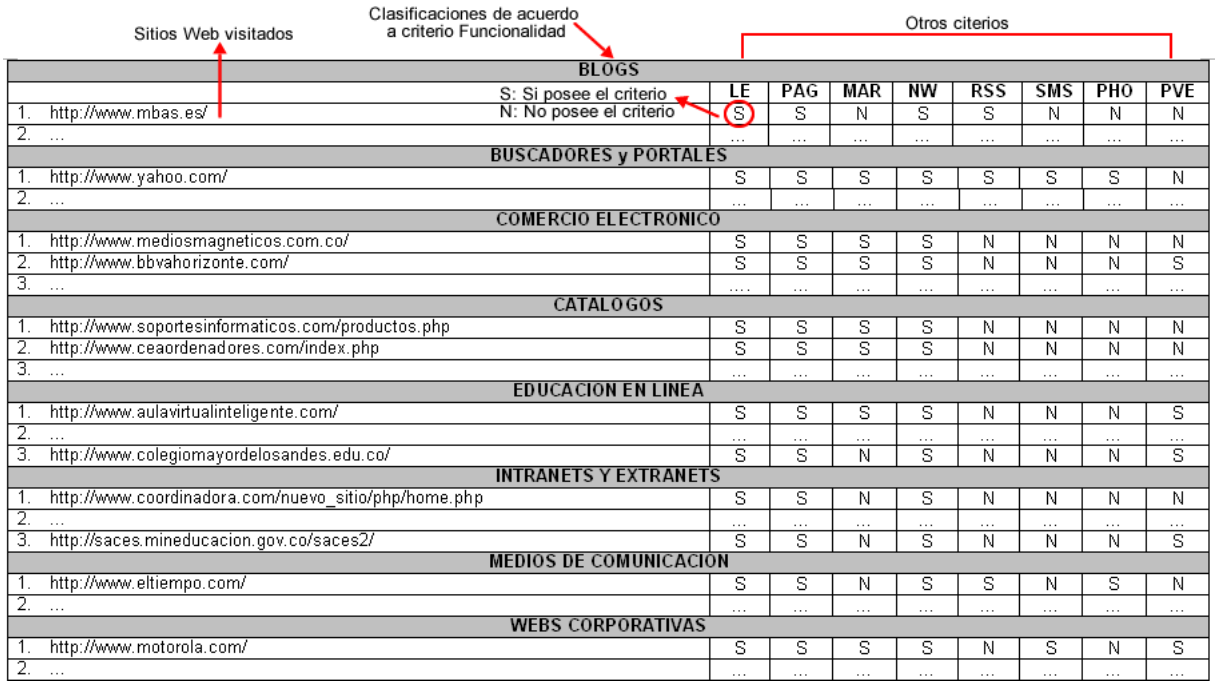
7. http://www.yahoo.com/	
8. http://www.cumpletusmetas.blogspot.com/	
9. http://www.bbvahorizonte.com/	
10. http://www.larepublica.com.co/	
11. http://www.bancosantander.com.co/	
12. http://skrdz.wordpress.com/2007/09/01/ver-particiones-linux-en-windows/	
13. http://www.ceaordenadores.com/index.php	
14. https://aristoteles.gate.upm.es/moodle/login/login.php	
15. http://www.eltiempo.com/	
16. http://www.coordinadora.com/nuevo_sitio/php/home.php	
17. http://www.rcn.com.co/	
18. http://lanota.com.co/noticias/	
19. http://www.aulavirtualinteligente.com/	
20. http://www.eldiario.com.co/	
21. http://diariodelhuila.com/	
22. http://www.colegiomayordelosandes.edu.co/	
23. http://www.elisnet.com/index.php?section=catalogo&idioma=es	
24. http://www.zonaphp.com/	
25. http://www.proteccion.com.co	
26. http://www.elmundo.com	
27. http://www.movicell.net/	
28. http://darwinjimenezgarzon.blogspot.com/2007/08/preguntas-ingeniera-software.html	
29. http://www.bancoagrario.gov.co	
30. http://www.google.com.co	
31. http://www.comsatel.com.ec/	
32. http://www.mmdistrib.com/indexhome.php	
33. http://www.tigo.com.co/	
34. http://www.answers.com/	
35. http://www.udgvirtual.udg.mx/	
36. http://es.altavista.com/	
37. http://www.davivienda.com/	
38. http://www.ecci.edu.co	
39. http://www.avvillas.com.co	
40. http://www.coomeva.com.co/	
41. http://www.ask.com/?o=312	
42. http://www.cefa.es/	
43. http://www.happylegs.es/	
44. http://www.aditec-ingenieros.com/	
45. http://www.kartoo.com/index.php3?langue=es	
46. http://www.bancodeoccidente.com.co/	
47. http://www.bancoldex.com/	
48. http://www.overture.com/	
49. http://www.lycos.es/	
50. http://sek-portal10.ucjc.edu/portal/page/portal/sek/Portada	

51. http://www.soportesinformaticos.com/productos.php	
52. http://www.senavirtual.edu.co/	
53. http://www.bancafe.com.co/	
54. http://www.iaeu.es/caratula/index.php	
55. http://portalninos.igac.gov.co:8080/ninos/contenidos/home.jsp	
56. http://www.gimandes.edu.co/colegio/intranet_login.htm	
57. http://www.gisportal.com/	
58. http://movilesnokia.blogspot.com/	
59. http://www.lookin.com.ar/	
60. http://www.todomercado.com/	
61. http://www.mediosmagneticos.com.co/	
62. http://saces.mineducacion.gov.co/saces2/	
63. http://www.coopcomer.fin.ec/	
64. http://www.infogloss.net/	
65. http://www.valledupar.net/	
66. http://www.bancodecredito.com.co	
67. http://www.cambio.com.co	
68. http://www.encatena.com	

10.2.2 Paso 2: Relación de Criterios adicionales contra Criterio Funcionalidad

Cada uno de los sitios Web relacionados en la tabla anterior fue verificado, y en primera instancia se analizó el tipo de funcionalidad que disponía, para así establecer el posible grupo al que podría pertenecer de acuerdo al criterio “Funcionalidad”. En varios casos el análisis realizado se hizo contemplando la posibilidad de incluir el sitio Web en dos o más grupos. El factor de decisión para la ubicación final en un grupo fue el mayor número de características que identificaban el sitio con determinado grupo.

Luego de analizar el tipo de sitio se procede a verificar los demás criterios de la matriz, para esto es necesario construir un instrumento adicional, de tal manera que permita comparar el detalle de cada uno de los sitios ya agrupados por funcionalidad contra los criterios adicionales. La siguiente figura (figura 9), presenta la matriz adicional requerida para recopilar la información básica de cara a la aplicación del instrumento propuesto en la figura 8.



	LE	PAG	MAR	NW	RSS	SMS	PHO	PVE
BLOGS								
1. http://www.mbas.es/	S	S	N	S	S	N	N	N
2. …	…	…	…	…	…	…	…	…
BUSCADORES y PORTALES								
1. http://www.yahoo.com/	S	S	S	S	S	S	S	N
2. …	…	…	…	…	…	…	…	…
COMERCIO ELECTRONICO								
1. http://www.mediosmagneticos.com.co/	S	S	S	S	N	N	N	N
2. http://www.bbvahorizonte.com/	S	S	S	S	N	N	N	S
3. …	….	…	…	…	…	…	…	…
CATALOGOS								
1. http://www.soportesinformaticos.com/productos.php	S	S	S	S	N	N	N	N
2. http://www.ceaordenadores.com/index.php	S	S	S	S	N	N	N	N
3. …	…	…	…	…	…	…	…	…
EDUCACION EN LINEA								
1. http://www.aulavirtualinteligente.com/	S	S	S	S	N	N	N	S
2. …	…	…	…	…	…	…	…	…
3. http://www.colegiomayordelosandes.edu.co/	S	S	N	S	N	N	N	S
INTRANETS Y EXTRANETS								
1. http://www.coordinadora.com/nuevo_sitio/php/home.php	S	S	N	S	N	N	N	N
2. …	…	…	…	…	…	…	…	…
3. http://saces.mineducacion.gov.co/saces2/	S	S	N	S	N	N	N	S
MEDIOS DE COMUNICACION								
1. http://www.eltiempo.com/	S	S	N	S	S	N	S	N
2. …	…	…	…	…	…	…	…	…
WEBS CORPORATIVAS								
1. http://www.motorola.com/	S	S	S	S	N	S	N	S
2. …	…	…	…	…	…	…	…	…

Figura 9. Matriz adicional para el análisis de sitios

El objetivo de esta matriz es identificar la presencia de los criterios adicionales en cada uno de los sitios verificados, dichos criterios adicionales son:

- LE: Lectura – Escritura
- PAG: Páginas Web
- MAR: Marcos Dinámicos
- NW: Navegador Web
- RSS: Lector RSS
- SMS: Mensajería SMS
- PHO: Portal Horizontal
- PVE: Portal Vertical

En caso de presentarse en el sitio verificado la existencia de un criterio adicional, se procede a marcar la casilla respectiva con la letra “**S**”, en caso contrario se asigna la letra “**N**”. Una vez finalizado este paso, que aunque tedioso, es obligatorio para encontrar una clasificación adecuada, se puede proceder a aplicar el instrumento propuesto en este trabajo de investigación.

10.2.3 Paso 3: Aplicación del instrumento propuesto

Luego de finalizar el *paso 1*, en donde se agrupó la lista de sitios Web por funcionalidades y de realizar la identificación de criterios adicionales en cada uno de los sitios Web verificados en el *paso 2*, se procedió a utilizar el instrumento propuesto en la sección ”10.1 Construcción del Instrumento” con toda la información hasta ahora disponible:

Tabla 26. Instrumento aplicado[27]

	Blogs	Buscadores	Comercio Electrónico	Catálogos	Educación Línea	Intranet Extranets	Medios Comunicación	Portales	Web Corporativas	**Subtotal**	
Lectura - escritura	3	3	3	3	3	3	3	3	3	**27**	**Promedio Filas = 15.63**
Página Web	3	3	3	3	3	3	3	3	3	**27**	
Marcos dinámicos	1	2	2	2	2	0	1	2	3	**15**	
Navegador Web	3	3	3	3	3	3	3	3	3	**27**	
Lector RSS	3	3	1	0	0	0	1	2	0	**10**	
SMS	0	0	0	0	0	0	0	1	3	**4**	
Portal horizontal	0	0	0	0	0	0	2	2	0	**4**	
Portal vertical	0	0	2	0	3	1	0	2	3	**11**	
Subtotal	**13**	**14**	**14**	**11**	**14**	**10**	**13**	**18**	**18**	**125**	
	Promedio Columnas = 13.89										

10.3 Análisis de resultados del instrumento aplicado

10.3.1 Análisis a nivel de columnas

Las columnas del instrumento representan los grupos de sitios Web encontrados de acuerdo al criterio funcionalidad, y la suma de las columnas indican si la clasificación (por ejemplo “Buscadores”) posee en algún grado los criterios de cada una de las filas. La suma de los totales encontrados en cada columna es 125 y el promedio de esa sumatoria es **13.89**. Los valores inferiores a este promedio no son relevantes para el presente trabajo de investigación, pues las características analizadas pueden ser encontradas con mayor facilidad en otras clasificaciones de sitios Web, lo que indica que se pueden solapar en una clasificación mayor perdiendo así toda su importancia.

[27] Para una mayor comprensión de este instrumento referirse a la figura 8 Esquema de convenciones del instrumento (Sección 10.1).

10.3.2 Análisis a nivel de filas

Las filas del instrumento representan las clasificaciones resultantes de los criterios adicionales, y la suma de las filas indican si la clasificación (por ejemplo “Lector RSS”) esta presente en algún grado en las columnas. La suma de los totales encontrados en cada fila es 125 y el promedio de esa sumatoria es **15.63**. A diferencia del análisis realizado para la columnas en donde los valores inferiores al promedio no eran representativos, en este caso los valores superiores al promedio no son relevantes, pues las características analizadas se repiten en una proporción alta, lo cual indica que no sirve como elemento de agrupación dado que la mayoría de sitios Web tienen esta característica.

10.4 Resultados de la caracterización

10.4.1 Tipos de sitios Web seleccionados

La línea de base para seleccionar la primera clasificación de sitios Web la ofrece el criterio "*funcionalidad*", y la asociación de éste con otros criterios permite establecer con más precisión el resultado del presente trabajo. La agrupación inicial de sitios Web realizada con base en éste criterio, esta conformada por nueve (9) subgrupos, sin embargo, luego de realizar el análisis del instrumento aplicado, dicha clasificación se reduce a cinco (5) subgrupos, los cuales hacen parte de la caracterización encontrada en el presente trabajo.

La siguiente tabla presenta todas las agrupaciones que el criterio "*Funcionalidad*" permite establecer, y filtra los subgrupos aceptados y rechazados.

Tabla 27. Resumen de columnas

	Subtotal por columnas[28]	**Seleccionado**
Blogs	13	✗
Buscadores	14	✓
Comercio electrónico	14	✓
Catálogos	11	✗
Educación en línea	14	✓
Intranets – Extranets	10	✗
Medios de comunicación	13	✗
Portales	18	✓
Webs corporativas	18	✓
Promedio	**13.89**	

La fundamentación conceptual para rechazar algunos de los subgrupos relacionados en la tabla anterior obedece al análisis realizado en el apartado 10.3.1. A continuación se presenta la tabla resumen con las agrupaciones finales:

Tabla 28. Agrupaciones seleccionadas

	Subtotal por columnas	**Seleccionado**
Buscadores	14	✓
Comercio electrónico	14	✓
Educación en línea	14	✓
Portales	18	✓
Webs corporativas	18	✓

[28] Relación de valores encontrados para las **columnas** del instrumento luego de aplicarlo a la muestra de sitios Web seleccionados.

10.4.2 Tipos de criterios adicionales seleccionados

Conjuntamente con el criterio "*Funcionalidad*", se utilizaron ocho (8) criterios adicionales, y como resultado de la aplicación del instrumento propuesto en este trabajo, se encontró que solo cinco (5) de esos criterios permitían agrupar adecuadamente los desarrollos Web. La siguiente tabla presenta los criterios adicionales utilizados y de igual manera los filtra en dos categorías: Aceptados y Rechazados.

Tabla 29. Resumen de criterios adicionales

	Subtotal por filas[29]	**Seleccionado**
Lectura – escritura	27	×
Página Web	27	×
Marcos dinámicos	15	✓
Navegador Web	27	×
Lector RSS	10	✓
SMS	4	✓
Portal horizontal	4	✓
Portal vertical	11	✓
Promedio	**15.63**	

La fundamentación conceptual para rechazar algunos de criterios adicionales relacionados en la tabla anterior obedece al análisis realizado en el apartado 10.3.2. La tabla 30 presenta los criterios adicionales seleccionados para la presente caracterización:

Tabla 30. Criterios adicionales seleccionados

	Subtotal por filas	**Seleccionado**
Marcos dinámicos	15	✓
Lector RSS	10	✓
SMS	4	✓
Portal horizontal	4	✓
Portal vertical	11	✓

10.4.3 Clasificación Web propuesta

En el apartado "*2.1.5 Funcionalidades*" se presenta un listado de posibles componentes que pueden tener cada agrupación o clasificación de sitios Web. Este estudio realizado por los autores Aníbal de la Torre y Gemma Ferreres es un aporte significativo, sin embargo deja de

[29] Relación de valores encontrados para las **filas** del instrumento luego de aplicarlo a la muestra de sitios Web seleccionados.

lado algunos componentes que en la práctica se encuentran en los desarrollos Web Actuales y que fueron identificados tras la aplicación del instrumento construido y aplicado en el presente trabajo. La siguiente tabla presenta los resultados encontrados del proceso de caracterización de sitios Web con base en los criterios expuestos en el presente trabajo, adicionalmente incluye los componentes emergentes que se identificaron tras la verificación funcional realizada.

Tabla 31. Clasificación de sitios Web

	Componentes otros autores	**Componentes propuesta**
Buscadores	▪ Búsqueda de información. ▪ Traducción de páginas. ▪ Enlace y agrupación de servicios. ▪ Interfaz Web y escritorio para las búsquedas.	▪ Personalización de idioma e interfaz Web. ▪ Ayuda para las búsquedas.
Comercio electrónico	▪ Validación de usuarios. ▪ Acceso a servicios de valor agregado: saldos, movimientos y transacciones electrónicas.	▪ Simuladores de crédito. ▪ Informes económicos. ▪ Compras por medio de la plataforma. ▪ Mensajes personalizados.
Educación en línea	▪ Validación de usuarios. ▪ Administración de usuarios. ▪ Creación y edición de contenidos. ▪ Publicación de contenidos. ▪ Seguimiento de contenidos presentados. ▪ Evaluaciones en línea. ▪ Foros de discusión. ▪ Clasificados.	▪ Gestión de agenda. ▪ Manejo de variables e indicadores. ▪ Control de actividades.
Portales	▪ Validación de usuarios. ▪ Administración de usuarios. ▪ Creación y edición de temáticas. ▪ Publicación de contenidos clasificados por temáticas. ▪ Correo electrónico. ▪ Foros temáticos. ▪ Clasificados. ▪ Encuestas. ▪ Múltiples servicios para la visualización de la información.	▪ Servicio de mensajería online. ▪ Informes climáticos.
Webs corporativas	▪ Creación y edición de contenidos. ▪ Publicación de contenidos. ▪ Seguimiento de contenidos presentados. ▪ Correo electrónico. ▪ FAQ's	▪ Creación y edición de noticias. ▪ Publicación de noticias. ▪ Creación de sugerencias. ▪ Zona de descargas.

Una vez construida la caracterización de Sitios Web bajo las condiciones expuestas en el presente trabajo, es importante identificar las ventajas que ésta puede brindar en nuevos trabajos y procesos de desarrollo:

1. La identificación de grupos y componentes para cada uno de los sitos Web permite optimizar el proceso de especificación de requisitos, pues si un nuevo desarrollo encaja en alguna de las clasificaciones se facilitaría identificar cuales serían los componentes mínimos y la funcionalidad en general que debería tener.
2. Sirve como línea de base para unificar y establecer los conceptos acerca de una arquitectura de desarrollo software que permita construir tipos de desarrollos como los encontrados en la clasificación.
3. Es un marco de referencia para seleccionar clasificaciones y realizar análisis particulares a los mismos.
4. Constituye una fuente bibliográfica para establecer el estado del arte de los Desarrollos Web en la actualidad.

11 ARQUITECTURA PROPUESTA

11.1 Modelo de referencia de la Arquitectura Software propuesta

Hasta el momento se han planteado conceptos, teorías, ventajas y desventajas de la AS, sin embargo no se ha definido claramente cuáles son los productos tangibles que se pueden obtener de una Arquitectura, de hecho, la problemática alrededor de la delgada línea que separa el Diseño de la AS crea confusiones a la hora de identificar los productos de una AS. La tendencia generalizada de los autores que trabajan está área del conocimiento indica que para describir una AS, se puede utilizar un modelo formado por múltiples vistas o perspectivas [16], en donde cada vista debe presentar de una manera diferente un aspecto del sistema. Una AS debe contemplar por lo menos tres vistas fundamentales, las cuales son:

Tabla 32. Vistas básicas AS

Nro	Vista	Descripción
1	Estática	Esta vista define los componentes claves de la AS que se está desarrollando.
2	Funcional	En esta vista se define qué hace cada uno de los componentes de la Arquitectura Software.
3	Dinámica	En esta vista se define como se comportan los componentes de la Arquitectura.

La tabla anterior presenta los conceptos de las tres vistas requeridas en una AS, sin embargo no define claramente los requerimientos mínimos para construir cada una de ellas, por lo que un Arquitecto de Software podría usar prácticamente cualquier forma de representación para construir su arquitectura, siempre y cuando cumpla con el concepto expuesto en cada vista. Para evitar este problema a la hora de construir una AS, es conveniente tomar como marco de referencia una tendencia arquitectónica que permita desglosar un poco más cada una de las vistas requeridas por una AS.

En el presente trabajo de investigación se tomó como referencia el modelo 4+1 de Kruchten [17], el cual clasifica cada una de las vistas de una AS de la siguiente forma:

Tabla 33. Relación vistas arquitecturas

Vista requeridas en una AS	Vistas según el Modelo Kruchten
Funcional	Vista Lógica Vista de Proceso
Estática	Vista Física Vista de Desarrollo
Dinámica	Vista de Casos de Uso

La anterior clasificación sigue siendo un poco ambigua, sin embargo tiene la particularidad de permitir el uso del estándar de facto mundial UML para la representación de sus vistas[30], lo cual permite que una arquitectura que se construya con esta tendencia, pueda ser entendida y por ende utilizada por personas que tengan conocimientos en modelado orientado a objetos con UML, esto incluye desde estudiantes de informática hasta profesionales en ingeniería.

La siguiente figura presenta el modelo de cinco vistas propuesto por Kruchten, en ella se puede observar el núcleo del modelo, el cual se centra en el modelo de escenarios representados por casos de uso.

Figura 10. Modelo de vistas 4+1.

A continuación se encuentra la descripción de cada una de las vistas propuestas en la tendencia de Kruchten, así como la relación directa con este trabajo de investigación.

11.1.1 Arquitectura Lógica (Logical Architecture)

Elemento	Descripción
Objeto	Soporta el análisis y la especificación de los requisitos funcionales: lo que el sistema debería proporcionar en términos de servicios a sus usuarios. El sistema se descompone en un conjunto de abstracciones claves tomadas mayormente del dominio del problema, en forma de objetos o clases.
Notación sugerida	La notación más usada es UML, y dentro de ésta diagramas de clases, estados y colaboración.
Estilo	El estilo más usado para la vista lógica es el Orientado a Objetos.

[30] El modelo genérico de vistas “4+1” puede usar cualquier notación para su representación, además de cualquier método de diseño, esto se debe a la descomposición lógica y de proceso de la vista funcional, pues en una parte esta división separa atributos funcionales de los no funcionales.

En el presente trabajo, ésta vista corresponde al modelo conceptual de toda la arquitectura, en ella se debe especificar cada una de las partes (de acuerdo al tipo de arquitectura seleccionada serían las capas) que la componen, así como los servicios que puede ofrecer a los posibles usuarios de la AS.

Según el estilo sugerido para esta vista la notación debería tener clases y objetos, y particularmente deberían utilizarse los diagramas de Clases de UML, ahora bien, dado que este proyecto no plantea la creación de una arquitectura particular, sino un modelo de AS para proyectos de Open Source, es clave el uso de clases y objetos, pero relacionados con otros artefactos que permitan visualizar y abstraer toda la arquitectura en una sola vista, esto indica que no se utilizarán los diagramas de clase para esta vista, pero si se dará cumplimiento al objeto de la misma con la representación utilizada.

11.1.2 Arquitectura de Procesos (Process Architecture)

Elemento	Descripción
Objeto	Esta vista trata algunos requisitos no funcionales, como por ejemplo: Ejecución, disponibilidad, conexiones, integridad, etc. También especifica que hilo de control ejecuta cada operación que se define o encapsula en la vista lógica. Es usada en los casos donde los sistemas presentan un alto grado de concurrencia, por tanto, es una vista opcional. Es la encargada de presentar los elementos relacionados con el desempeño del sistema, como son: escalabilidad, concurrencia y tiempo base de desempeño.
Notación sugerida	La notación más usada es UML, y dentro de ésta diagramas estados, actividad y secuencia.
Estilo	Para esta vista pueden encajar muchos estilos, desde el uso de tuberías, filtros hasta la representación por capas y su interacción entre ellas.

Si se utiliza UML para la representación de esta vista, se puede apreciar la diferencia concreta entre las dos vistas descritas (lógica y de proceso), sin embargo a la hora de definir modelos genéricos es complejo utilizar diagramas de actividades para representar posibles escenarios, por lo tanto en el presente trabajo de investigación se utiliza un estilo de representación que permite visualizar las conexiones y requisitos no funcionales en términos de tuberías y relaciones entre capas. Lo anterior plantea una sinergia entre la vista lógica y la de procesos, de tal manera que para la representación de cada vista se utilizará un solo diagrama genérico.

11.1.3 Arquitectura de Desarrollo (Development Architecture)

Elemento	Descripción
Objeto	La vista de desarrollo o despliegue se enfoca en la organización de los módulos software en el entorno de desarrollo. El software es empaquetado en pequeños módulos (librerías de programa, subsistemas, unidades, etc.), los subsistemas se organizan en capas jerárquicas, y cada capa proporciona una interfaz bien definida a sus capas superiores. La vista de desarrollo toma por tanto requisitos internos relacionados con facilidad de desarrollo, gestión del software (reparto de requisitos, trabajo del equipo), evaluación de costes, planificación, monitorización del progreso del proyecto, reutilización, portabilidad, seguridad y restricciones impuestas por las herramientas o por el lenguaje de programación. Esta vista se construye cuando se hayan identificado todos los elementos software del sistema.
Notación sugerida	La notación más usada es UML, y dentro de ésta diagramas de componentes y paquetes.
Estilo	Así como las otras vistas la tendencia es a utilizar orientación a objetos, aunque se pueden detallar los paquetes por medio de estereotipos que permitan ver la relación entre paquetes o entre capas, dependiendo de la estrategia de representación.

Está vista permite ver directamente cuales son los servicios que va a prestar la arquitectura software a construir, por ende la definición clara de componentes y módulos es clave. En el presente trabajo de investigación se utilizarán diagramas de componentes, y para su agrupación diagramas de paquetes, definiendo desde las posibles propiedades hasta los métodos que podrían llegar a tener cada uno de los objetos del sistema. Para hacer uso de esta vista no es necesario hacer una interpretación del modelo de referencia, pues se amolda perfectamente a las necesidades del proyecto.

11.1.4 Arquitectura Física (Physical Architecture)

Elemento	Descripción
Objeto	La vista física se centra en los requisitos no funcionales, tales como la disponibilidad del sistema, la fiabilidad (tolerancia a fallos), ejecución y escalabilidad, también presenta la forma como los procesos, objetos, y demás componentes se relacionan a nivel de nodos.
Notación sugerida	La notación sugerida en esta vista hace referencia a máquinas, nodos, relaciones y en general contenedores físicos que permitan visualizar cómo está organizada la información. Dependiendo del objeto a representar se puede utilizar: Componentes: nodos de proceso. Conectores: LAN, WAN, bus, etc. Contenedores: Clases genéricas (a nivel físico) Varias configuraciones físicas pueden usarse. La correspondencia del software a los nodos debe ser altamente flexible y tener el mínimo impacto

	en el código fuente.
Estilo	El estilo se basa en nodos (entidades genéricas o clases) las cuales permiten ver la separación de cada elemento de la arquitectura. Tal como en otras vistas la orientación a objetos predomina.

Una arquitectura particular define en este tipo de vistas los servidores y las interacciones entre ellos, definiendo los protocolos de conexión, los servicios que pueden ofrecer, los lenguajes de comunicación entre servidores, en muchos casos es necesaria la creación de estereotipos que permitan definir cada uno de los objetos que la vista desee presentar. Dado que el presente trabajo de investigación no define una arquitectura particular, sino un modelo de arquitectura, la representación de servidores es reemplazada por la representación de cada una de las capas definidas, así cada una de las capas que se estructuren pueden convertirse en servidores cuando se instancie el modelo propuesto en este trabajo.

11.1.5 Escenarios y/o casos de uso

Elemento	Descripción
Objeto	La vista de escenarios corresponde a las instancias de casos de uso que unifican todas las vistas. Así, desde los casos de uso se puede hacer una trazabilidad a todos los componentes del sistema software, observando por ejemplo, que máquinas, clases, componentes, .jar, procesos, son los responsables de que el sistema cubra una cierta funcionalidad.
Notación sugerida	La notación sugerida es la representación de casos de uso en UML
Estilo	El estilo es orientado a objetos utilizado en UML

Esta vista puede ser analizada desde dos perspectivas:

1. Alto nivel: Permite visualizar como la arquitectura funciona en realidad.
2. Bajo nivel: Permite que un desarrollador llegué hasta la capa inferior de todo el modelo siguiendo los casos de uso planteados en esta representación.

Este trabajo hace uso de esta representación sin interpretaciones, pues los casos de uso al igual que otros diagramas de UML son estándares de facto en la industria de desarrollo software y particularmente en la academia.

11.1.6 Construcción de las vistas

Si bien es cierto el modelo de Kruchten no es una metodología, si sugiere un método de trabajo y un conjunto de pasos para el desarrollo de cada una de sus vistas. Entonces para construir una AS se deberían seguir estos pasos:

1. Iniciar con el análisis de los casos de uso o escenarios,

2. Luego definir una vista lógica que permita identificar claramente lo que deben llevar las vistas de desarrollo y proceso,
3. Construir las vistas de desarrollo y procesos con base en la vista lógica, así se definen las clases y los objetos claves de la AS,
4. Se finaliza con la vista física.

En la forma como se enuncian y describen los pasos puede parecer una tarea ágil y fácil, sin embargo definir cada una de las vistas tiene un nivel de complejidad que requiere no solo conocimientos sino creatividad a la hora de visualizar y abstraer problemas.

Los pasos "sugeridos" para la construcción de una AS bajo la tendencia de Kruchten son válidos para una AS particular, sin embargo este trabajo de investigación busca plantear un modelo de AS que pueda ser instanciado para desarrollos de código abierto, por lo tanto la construcción de la AS debería iniciar por la vista lógica de manera conjunta con la vista de procesos, esto se debe a que el nivel de abstracción que requiere el presente trabajo es alto, y se debe tener clara la visión global del sistema antes que el detalle de sus relaciones. Las vistas de desarrollo y física son el siguiente paso en la construcción del modelo, lo cual deja el modelo de casos de uso para las fases finales de la AS.

11.1.7 Conclusión marco referencia modelo Kruchten 4+1

Del análisis de cada una de las vistas del marco de referencia seleccionado se pueden identificar y relacionar los productos de ésta investigación. A continuación se presenta una tabla resumen que identifica cuales son los diagramas de UML a utilizar en cada una de las vistas, y su posible reemplazo para aquellas vistas en donde no se utilice la notación sugerida por el modelo de referencia:

Tabla 34. Relación entre vistas y diagramas a utilizar

Vista	Notación sugerida	Notación a utilizar
Lógica	▪ Diagramas de clases UML	Utilización de clases y objetos para la representación, más no diagramas de clases. Definición de capas y relaciones.
Proceso	▪ Diagramas de secuencia ▪ Diagramas de colaboración	Enlaces entre objetos, tuberías y representaciones por capas. Esta vista y la lógica se integran en una sola en el presente proyecto.
Física	▪ Diagramas de despliegue	Diagramas de despliegue
Desarrollo	▪ Diagramas de componentes	Diagramas de componentes
Casos de Uso	▪ Diagramas de casos de uso	Diagramas de casos de uso

11.2 Modelo de capas de la AS propuesta

La presente arquitectura está desarrollada bajo el concepto de niveles o capas, y busca primordialmente descomponer la complejidad del diseño y desarrollo de los sistemas orientados hacia la Web que utilizan código abierto para su implementación.

La jerarquía de capas de la AS propuesta, permite que cada capa provea servicios a una capa superior y que a su vez dicha capa sea servida por una capa inferior. La comunicación entre capas adyacentes puede parecer una limitación, sin embargo puede ser visualizada como una fortaleza, pues permite mantener un bajo acoplamiento del sistema. La siguiente figura permite identificar las cuatro (4) capas que forman la AS propuesta. En un nivel de abstracción superior, cada capa corresponde a un componente básico de la AS, en donde cada uno de ellos encapsula su funcionalidad, comportamiento y forma de manipular los datos.

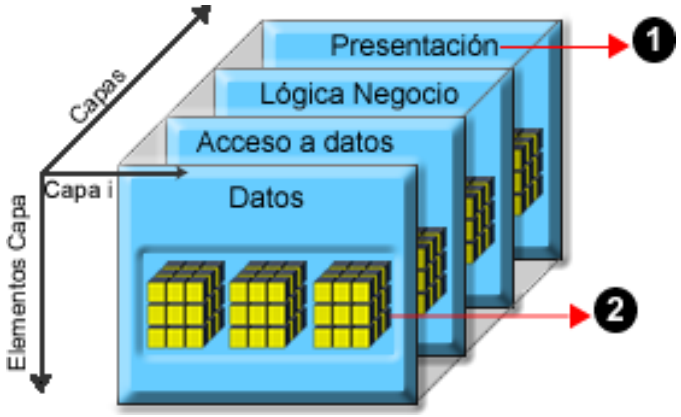


Figura 11. Modelo de capas propuesto.

La representación cúbica de la AS que plantea la figura anterior, es una estrategia para que el lector visualice cómo es posible abstraer algo complejo para luego dividirlo, analizarlo y procesarlo, así como un arquitecto construye una maqueta o ingeniero civil construye un plano, un ingeniero de software puede construir una AS que permita construir nuevas AS para un fin particular. La lectura de la figura anterior es la siguiente:

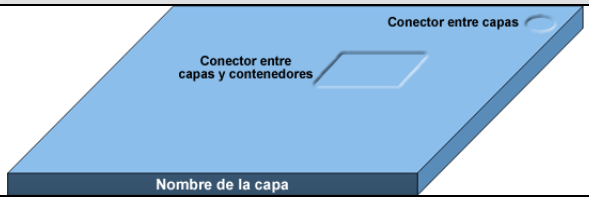
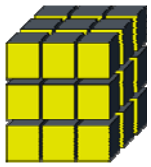
1. Cada componente (capa) tiene una etiqueta ó nombre que indica al usuario su posible funcionalidad (pero no revela cómo lo hace).

2. Los componentes (capas) tiene acceso solo al componente inmediatamente anterior o posterior.
3. Cada componente (capa) tiene a su vez módulos que la conforman, dichos módulos se representan por cubos, lo que indica que esos módulos a su vez también pueden estar compuestos por más sub-módulos.

11.3 Vistas: Lógica y de Procesos

Cuando se construye una Arquitectura Software, es importante no perder de vista el modelo conceptual de lo que se desea presentar, una visión lógica del objeto de estudio ligado a una visión funcional es una alternativa viable, sin embargo el paso previo, consiste en describir cada uno de los elementos que hacen parte de la vista a construir, con el fin de orientar al lector en su compresión y análisis:

Tabla 35. Descripción de los objetos utilizados en la vista lógica y de proceso

Elemento	Descripción
	**Capa:** Este objeto representa cada uno de los componentes en las cuales está dividida la AS propuesta. Una capa permite agrupar contenedores, tuberías internas y conexiones entre objetos. Así mismo establece la forma de comunicarse con una capa contigua.
	**Contenedor:** Es un componente que puede agrupar a otros componentes. Por si solo no posee una lógica funcional, pero si especifica la forma como cada uno de sus componentes puede comportarse cuando interactúa con otros elementos de la AS.
	**Envío de mensajes:** Este objeto permite representar la dirección en la cual viajan los mensajes entre los contenedores y entre las capas.
	**Tubería:** Representa el flujo de información entre uno o más contenedores, el flujo de la información puede tener sentido bidireccional, dependiendo de dirección lógica que se estructure en la capa.

A continuación se describen cada una de las capas y la funcionalidad de de la presente vista:

11.3.1 Capa Almacenamiento Datos

La importancia de la información y los datos en la actualidad, exigen que todo sistema disponga de un repositorio de datos, llámese base de datos relacional, archivos planos o de

tipo particular: XML, imágenes, etc. Estos repositorios se pueden dividir en dos grandes grupos, los cuales se constituyen en los dos contenedores de que dispone esta capa: Contenedor: Bases de Datos Relacionales y Contenedor: Otros repositorios. La división obedece a la forma cómo se gestiona la información en cada uno de ellos, en el primero se utiliza un lenguaje estándar para el acceso y manipulación de datos, y en el segundo se utilizan métodos, técnicas de recuperación y gestión de tipos de archivos.

La siguiente figura presenta los objetos descritos hasta el momento y permite observar la dirección del flujo de información entre los contenedores de la capa.

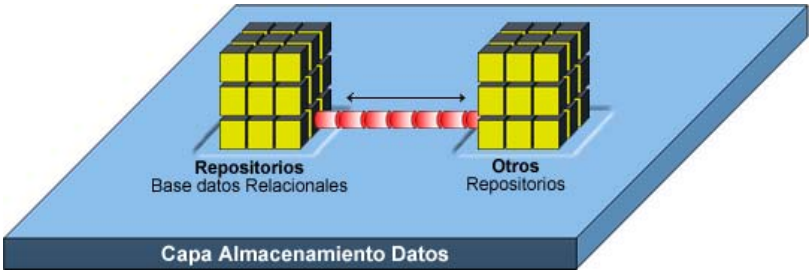


Figura 12. Capa Almacenamiento Datos.

Los dos contenedores de la figura anterior están "***empotrados***" en la capa, esto indica que la capa se comunica directamente con ese contenedor a través de "tubería internas", y es la forma como la capa obtiene y extrae la información de cada contenedor para luego entregarla o recibirla de una capa contigua. Los contenedores tienen a su vez una conexión entre sí "tubería externa", lo que les permite enviar y recibir información de manera directa, obsérvese que la dirección para el envío de mensajes en tres contenedores de esta capa es bidireccional.

Dado que la AS propuesta es para código abierto, se pueden pensar que el uso de bases de datos propietarias no esté contemplado, sin embargo está particularidad es posible siempre y cuando el sistema de gestión relacional utilice el estándar SQL para la manipulación de los datos. En otras palabras no existe inconveniente en que el contenedor de bases de datos

tenga: PostgreSQL[31], MySQL[32], Informix[33], Sybase, Oracle[34], MSSQL[35], ó incluso el mismo MS Access.

Con respecto a los tipos de archivos que gestiona el contenedor "Otros repositorios", la AS propuesta no contempla la manipulación de los datos en formatos propietarios, como es el caso de Microsoft Excel, Microsoft Word, sin embargo, si se contempla la posibilidad de almacenar y descargar cualquier tipo de archivo.

11.3.2 Capa Acceso a Datos

El modelo clásico de tres capas plantea que los datos y el acceso a los datos se encuentren en una sola capa, en donde cada base de datos tiene su propia forma de interactuar con la información y a su vez una forma particular de comunicarse con otras capas para entregar y recibir información. Este modelo deja de tener validez en la medida que las empresas de desarrollo construyen los sistemas Web, pensando más en la funcionalidad que debe tener el aplicativo que desea el cliente, que en algunos aspectos no funcionales como son: el sistema operativo o el repositorio de datos.

La Arquitectura Software propuesta, permite desligar la capa de datos de la capa de acceso a los datos, de tal manera que un desarrollador no deba preocuparse por cual base de datos o que tipo de archivos se van a gestionar en el sistema en la fase de producción. Lo realmente importante para el desarrollador es fabricar las funcionalidades particulares que el cliente requiere para su negocio, los demás componentes del sistema deben ser reutilizables para así optimizar el proceso de desarrollo.

La separación entre los datos como tal y la forma de acceder a eso datos se logra con la utilización de un contenedor denominado "***Mapeador relacional***", dicho contenedor puede interactuar con las bases de datos que se encuentren en el repositorio relacional[36],

---

[31] La mejor base de datos relacional Open Source de la actualidad, compite al nivel de Oracle.

[32] Base de libre distribución pero con licencias que no permiten la manipulación del código fuente.

[33] Fue el segundo sistema de gestión relacional después de Oracle en la década de los 90, es de carácter propietaria

[34] Considerada el mejor sistema de base de datos relacional en la actualidad, es de carácter propietaria

[35] Base de datos propietaria de Microsoft más conocida como SQL Server.

[36] Los requerimientos para permitir el uso de una base de datos en particular no dependen de esta capa, sino de la capa de datos descrita en el apartado anterior.

permitiendo así cambiar en cualquier momento la base de datos sin afectar en ninguna manera el proceso de desarrollo. El contenedor de esta capa puede estar conformado por uno o varios "***mapeadores***", dependiendo del lenguaje a utilizar en el proceso de desarrollo. Para citar algunos ejemplos de este tipo de mapeadores se pueden mencionar:

- ***PDO:*** Es un mapeador nativo para el lenguaje PHP, permite conectarse con un gran número de bases de datos relacional, y hace uso del SQL estándar para acceder a los datos.
- ***ADODB:*** Está conformado por un amplio conjunto de librerías que permiten acceder gran cantidad de bases de datos relacionales, su fortaleza radica en la simplicidad para agregar una nueva librería que permita acceder a una base de datos actualmente no soportada.
- ***DAO:*** Es en mapeador definido para Java, al igual que el anterior permite acceder a múltiples bases de datos relacionales.

La siguiente figura presenta la capa de acceso a datos, en ella se puede visualizar el contenedor propuesto:

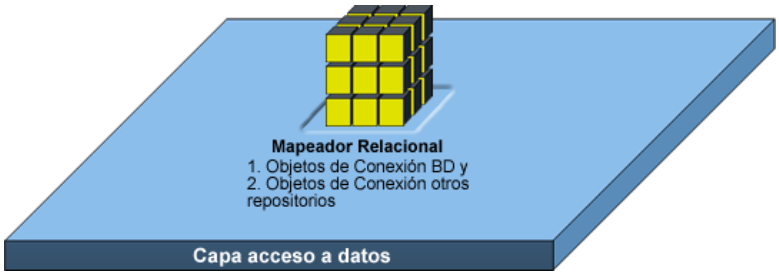


Figura 13. Capa Acceso a Datos.

11.3.3 Capa Negocios

Esta capa es la responsable de implementar las reglas que gobiernan el comportamiento del sistema, de igual forma se encarga de recibir las peticiones de la capa cliente y de dar respuesta a las mismas. La AS propuesta permite dividir la estructura base de esta capa en cuatro (4) contenedores, de tal manera que dada uno de ellos tenga una misión particular en la fase de desarrollo. Esta división está íntimamente ligada a la posibilidad de utilizar un marco de trabajo en el proceso de desarrollo, de tal forma que le permita al desarrollador: ver

en la práctica el uso de algunos patrones, entender y aplicar el concepto de reutilización, y aplicar las herramientas necesarias para construir un sistema, teniendo como referencia un Framework probado y utilizado en proyectos informáticos.

La siguiente figura presenta la composición general de la capa así como la dirección de los mensajes que fluyen entre los contenedores.

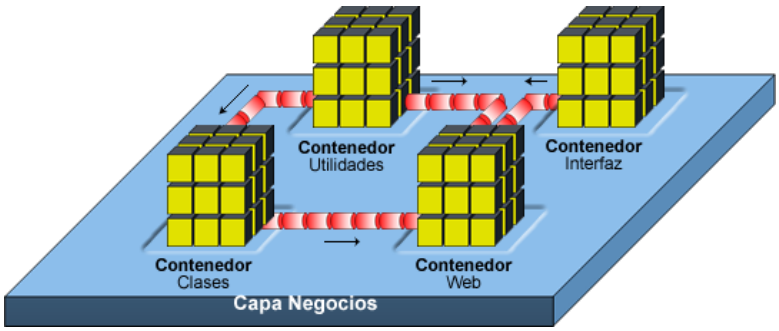


Figura 14. Capa Negocios.

A continuación se describe en detalle cada uno de los contenedores así como las relaciones que existen entre cada uno de ellos.

Tabla 36. Descripción de contenedores capa de negocios

Contenedor	Descripción
Interfaz	Este contenedor almacena las clases que gestionan la parte gráfica del aplicativo como por ejemplo los menús, submenús, barras de navegación, información del usuario en sesión, enlaces para ayuda, etc. Este contenedor no realiza peticiones a ningún otro contenedor (ver dirección del envío de mensajes), y solo acepta peticiones del "contenedor Web".
Utilidades	Este contenedor almacena cuatro (4) tipos de elementos: • Clases para administrar calendarios, mensajes y alertas, • Funciones de validación para formularios, • Funciones para manipulación de fechas, horas y • Editores Web con formato html. No realiza peticiones a ningún otro contenedor, pero si acepta peticiones del "*Contenedor de clases*" y del "*Contenedor Web*".
Clases	Este contenedor es el que se encarga de almacenar las clases que dan cumplimiento a la funcionalidad solicitada por un cliente. Puede hacer uso del "*contenedor Utilidades*" en caso de requerir alguna funcionalidad particular ofrecida por éste. Solo acepta peticiones del "*Contenedor Web*" y no tiene una relación directa con el "*Contenedor Interfaz*".

 *Web*	Este contenedor almacena todas las páginas que pueden llegar a ser procesadas por un servidor Web, generalmente las extensiones son del tipo php, jsp, html, etc. Este contenedor hace peticiones al "*Contenedor Interfaz*" para instanciar los objetos respectivos que permitan manipular la interfaz. En este modelo el desarrollador no manipulará directamente las etiquetas html, utilizará clases y objeto para la construcción de una página. La base de este concepto radica en la utilización de la POO[37] inclusive para crear una página html simple.

11.3.4 Capa Cliente

La capa de presentación o capa cliente permite presentar la información final al usuario, su importancia radica en la interacción directa que tiene ésta con el cliente, pues es donde se identifican problemas de flexibilidad y navegabilidad.

Existen tres (3) contenedores en esta capa: Navegadores Web, Scripts cliente y Ajax. El contenedor "*Navegadores Web*" agrupa todos los posibles navegadores con los cuales un usuario puede acceder a una aplicación Web, además contiene los plugins necesarios para la reproducción de controles y elementos adicionales como lo son: SVGS, ActiveX, secuencias de películas, etc. Este contendedor realiza peticiones al contenedor de Scripts, el cual almacena todas las validaciones de formularios a este nivel. La siguiente figura presenta la estructura base de la capa cliente.

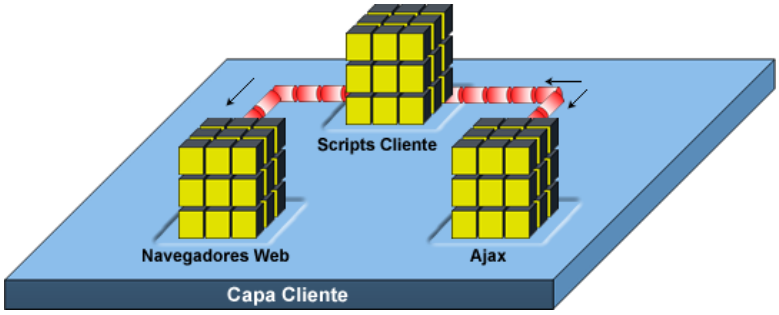


Figura 15. Capa Cliente.

El contenedor "*Scripts clientes*" tiene una doble función, primero entregar al contenedor "*Navegadores Web*" toda la funcionalidad para realizar validaciones y verificaciones en esta capa, y segundo servir de enlace con el contenedor "Ajax", el cual hace peticiones directas a

[37] Programación Orientada a Objetos

la capa cliente para que a su vez realice peticiones a la capa "N*egocios*" recogiendo información sin necesidad de recargar toda la página.

11.4 Vista de Desarrollo

Para realizar una definición más precisa de esta vista se hace necesario describir cada uno de los elementos que se usaran en su representación. A continuación se presenta esta información:

Tabla 37. Descripción elementos de diagramas de componentes

Elemento	Descripción
	Representa la parte física del sistema. Como puede ser un módulo, una base de datos, un programa ejecutable, una biblioteca de programas [18].
	Especifica una biblioteca de objetos estática o Dinámica [18].
	Representa la materialización de una clase, en objeto llamado "instancia".
	Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente se refiere a los servicios ofrecidos por otro componente [18].

Como se describió en el apartado anterior la arquitectura propuesta está formada por cuatro (4) capas que a su vez están realizadas por contenedores específicos que definen el funcionamiento y servicios que aporta la AS. En esta sección se pretende detallar los mecanismos y subsistemas que dan existencia a dichos contenedores. Es el objetivo de la vista de desarrollo representar por medio de diagramas de componentes los métodos básicos que debería tener el sistema. A continuación se presenta la descripción de los contenedores de cada una de esas capas:

11.4.1 Contenedores de acceso a datos

Como se describió en el apartado *11.3.2 Capa Acceso a Datos* para la separación entre la capa de datos y la de acceso a datos se hará uso de un mapeador relacional, que permitirá la portabilidad de la base de datos. Para el caso de la presente arquitectura se recomienda la

utilización de la ADOdb. Este mapeador es producto de un proyecto Open Source, por tanto, ha sido validado y probado por muchos desarrolladores. La ADOdb es un conjunto de librerías de bases de datos. Puede soportar bases como: MySQL, PostgreSQL, Interbase, Firebird, Informix, Oracle, FoxPro, Access y FrontBase, entre otras. A continuación se presenta una visión general de sus librerías:

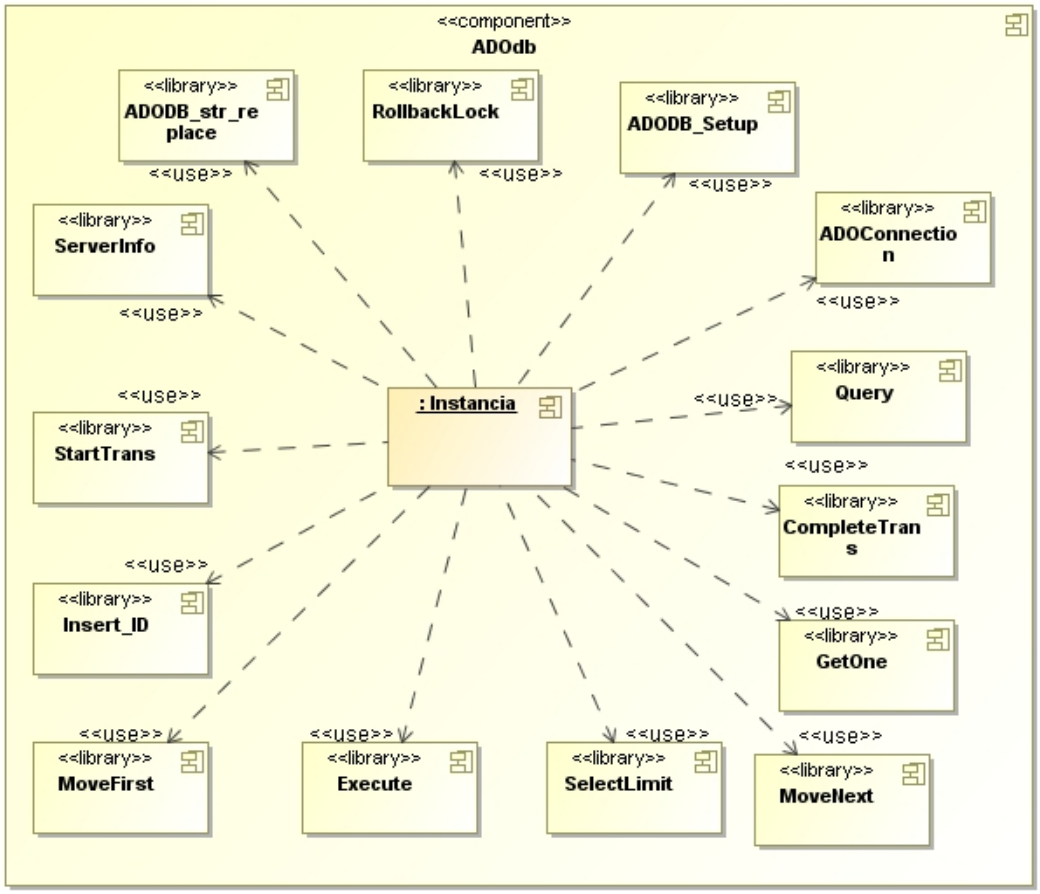


Figura 16. Componente Adodb

La figura anterior presenta el listado de librerías más usadas, pero la clase cuenta con mucho más métodos que no se hace necesario explicar en este apartado.

11.4.2 Contenedores de negocios

La vista de desarrollo describirá la implementación de la capa de negocios de la sección *11.3.3 Capa Negocios*. En la siguiente figura se ubica con un círculo rojo el contenedor a explicar:

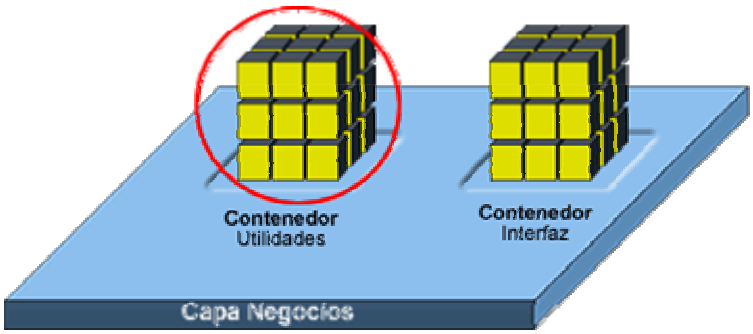


Figura 17. Contenedor utilidades

La distribución del contenedor de utilidades se puede describir así:

Ocho (8) componentes básicos: search, pagination, date, calendar, forms, publisher, files y URL.

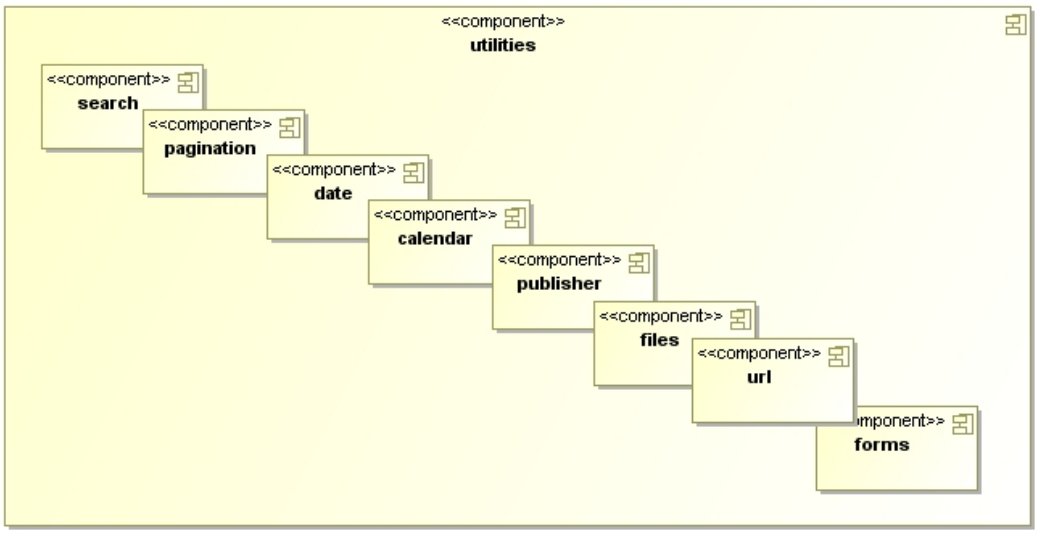


Figura 18. Componente utilidades

11.4.2.1 Componente search

El componente *search* ofrece la funcionalidad de búsquedas de acuerdo a diferentes criterios, que pueden ser por nombres, apellidos, documento de identidad, etc. Dependiendo de las características particulares de la lógica del negocio.

El componte *search* cuenta como eje central con una instancia de su clase, que es la encargada de hacer el llamado de los métodos particulares. Algunos de estos métodos se relacionan directamente entre sí, como es el caso de *print_html* que "hace uso" de *print_hidden*. Esto significa que para que el método print_html cumpla con su objetivo debe llamar desde si mismo a print_hidden. A continuación el diagrama de componentes respectivo:

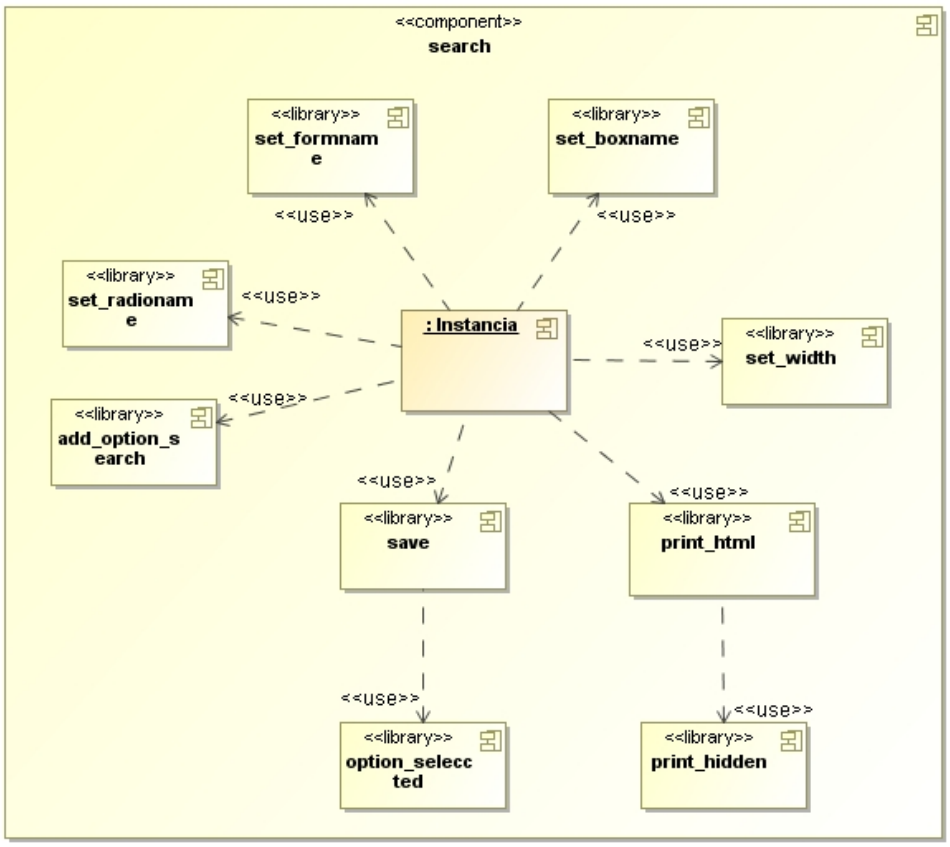


Figura 19. Componente search

La figura presenta las librerías del componente, este listado es descrito a continuación:

- El método *set_formname* permite definir el nombre del formulario a crear.
- *set_boxname* establece el nombre de la caja de texto.
- *set_radioname* establece el nombre del elemento radiobutton de cada una de las opciones de búsquedas.

- *add_option_search* función que adiciona cada criterio de búsqueda, como por ejemplo (por nombre).
- *set_width* establece el ancho de la tabla que contiene los elementos de la búsqueda.
- *save* guarda la opción de búsqueda seleccionada y la palabra a buscar.
- *print_html* presenta en pantalla el objeto de búsquela con todas sus opciones como caja de texto, opciones de radio para selección, botón que active el evento de búsqueda.
- *option_seleccted* almacena la opción seleccionada.
- *print_hidden* almacena los códigos adicionales que se necesiten para la búsqueda.

11.4.2.2 Componente date

Este componente es el encargado de manejar las fechas del sistema, sus formatos, lenguajes y cálculos de las mismas. A continuación se presenta el diagrama de componentes que describe las librerías necesarias así como las interacciones entre ellas.

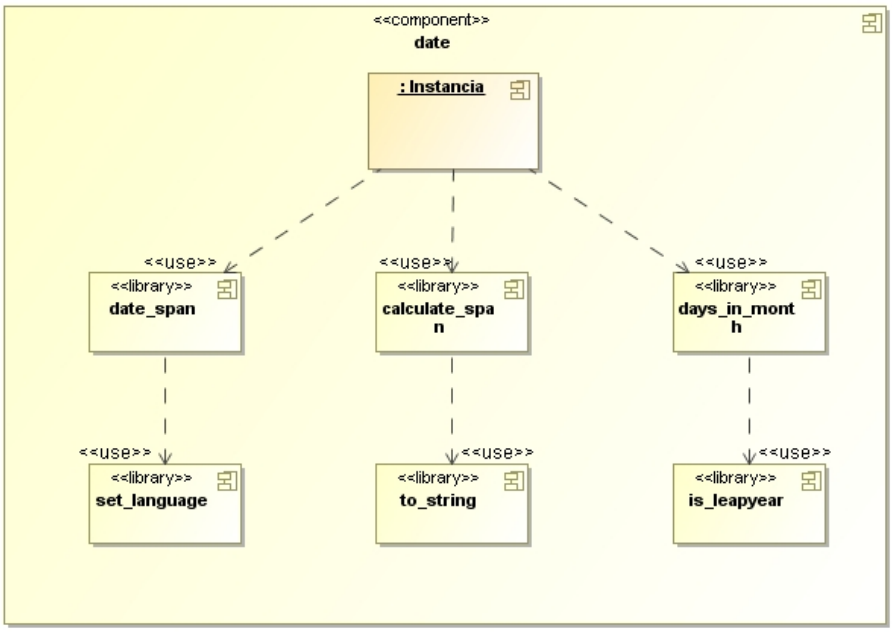


Figura 20. Componente date

La figura presenta las interacciones entre las diferentes librerías del componente date. Se puede apreciar la relación de dependencia entre la librería date_span y set_lenguage, calculate_span y to_string y days_in_month con is_leapyear.

- *date_span* hace el llamado a la función set_language.
- *set_language* establece el lenguaje a usar dentro de la clase.
- *calculate_span* calcula el lapso de tiempo en segundos entre una fecha inicial y final.
- *to_string* convierte un valor entero en una cadena.
- *days_in_month* determina el número de días que tiene un mes.
- *is_leapyear* determina si el año es bisiesto o no.

11.4.2.3 Componente forms

Este es el componente encargado de generar los elementos tipo formulario como listas de selección, radiobutton y cajas de selección.

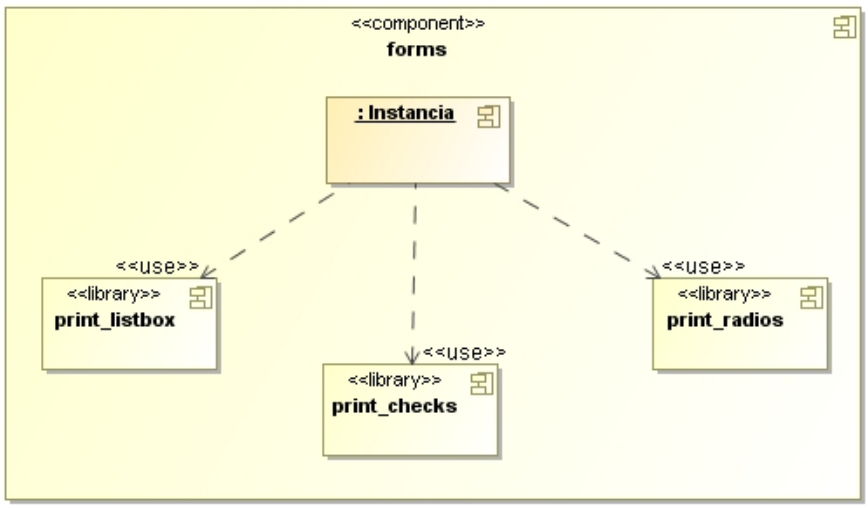


Figura 21. Componente forms

De la figura 21 se puede apreciar que no hay ninguna relación de dependencia directa entre sus librerías, lo que indica que el llamado de las mismas se debe realizar desde el objeto principal de la clase, llamado *instancia*.

- *print_listbox* dibuja en pantalla un combo dinámico de acuerdo con el listado de elementos que se desee incluir, y permite seleccionar la opción almacenada en Base de datos.
- *print_checks* imprime en pantalla un grupo de elementos de selección tipo checkbox.

- *print_radios* presenta en pantalla un grupo de elementos tipo radiobutton.

11.4.2.4 Componente files

Permite realizar el mantenimiento de la información básica que requiere la aplicación para su funcionamiento.

La funcionalidad de este componente radica en encargarse del manejo de archivos del sistema, el registro, lectura y descarga de los mismos.

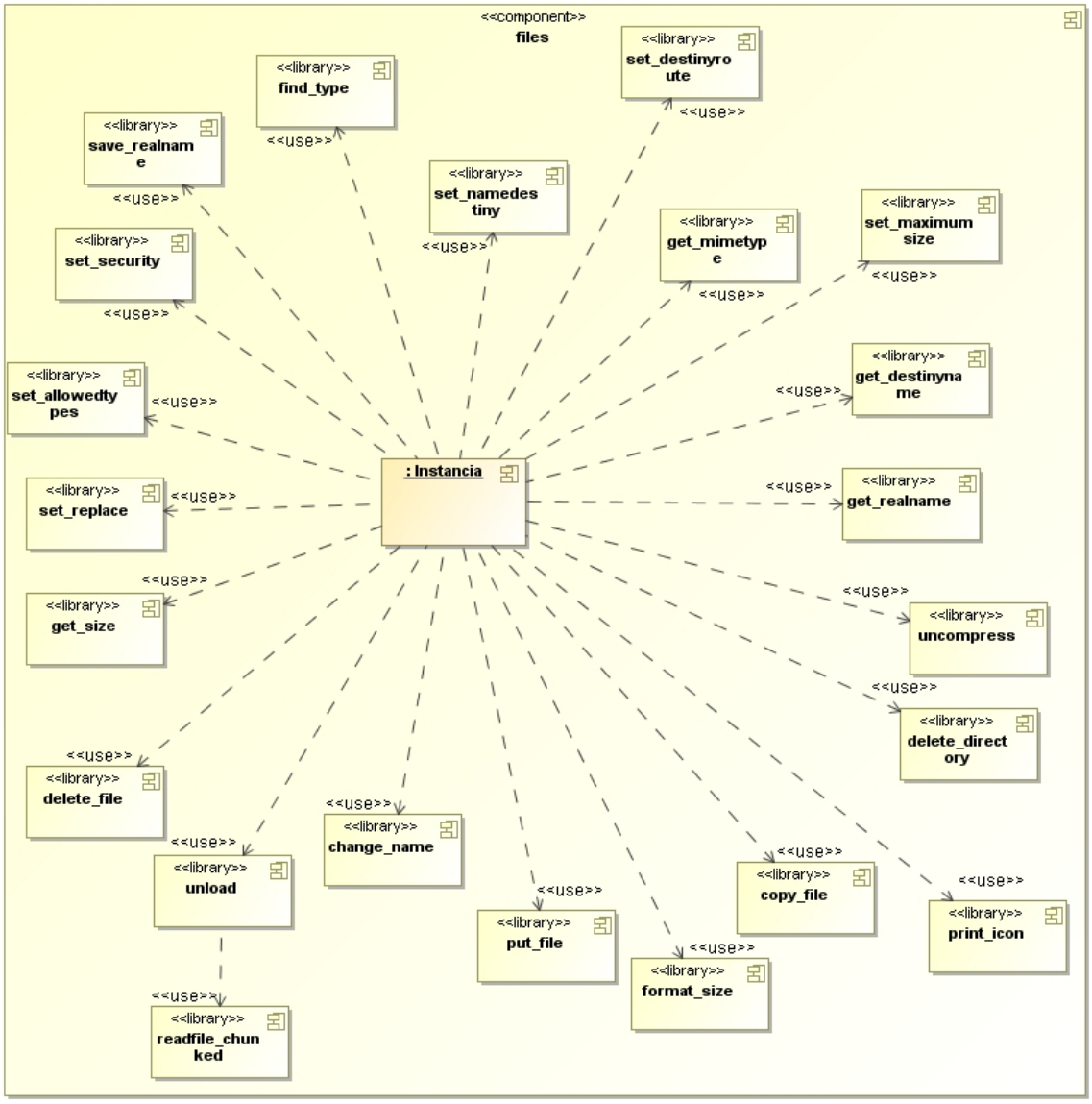


Figura 22. Componente files

La figura anterior presenta todas las librerías necesarias para poder implementar el componente *files*, cuenta con una gran cantidad de librerías, pero se nota una poca dependencia entre ellas. Solo la librería *unload* hace uso de readfile_chunked.

- *find_type* encuentra el tipo de archivo a procesar.
- *set_destinyroute* establece la ruta de destino del archivo.
- *save_realname* método booleano que define si el nombre del archivo es real o no.
- *set_namedestiny* establece el nombre destino del archivo.
- *get_mimetype* obtiene el tipo mime del archivo.
- *set_maximumsize* establece el tamaño máximo del archivo.
- *set_security* establece los permisos de la carpeta destino del archivo.
- *set_allowedtypes* define los tipos de archivos permitidos para cargar en la base de datos.
- *set_replace* define si el archivo se desea reemplazar o no.
- *get_size* obtiene el tamaño del archivo.
- *delete_file* elimina el archivo físicamente del repositorio de documentos.
- *unload* define el tipo de cabecera necesario para descargar el archivo del sistema.
- *readfile_chunked* permite abrir el archivo para lectura.
- *get_destinyname* obtiene el nombre del archivo destino.
- *get_realname* obtiene el nombre real del archivo.
- *uncompress* descomprime el archivo en caso de encontrarse en alguno de los tipos de compresión.
- *delete_directory* elimina directorios.
- *print_icon* presenta el icono correspondiente a cada tipo de archivo.
- *change_name* renombra un archivo.
- *put_file* graba el archivo en el sistema físico.
- *format_size* cambia el formato del archivo.
- *copy_file* copia un archivo de una ruta definida a otra.

11.4.2.5 Componente URL

Este componente es el encargado del paso de parámetros por medio del método *get*, es decir del paso de valores en forma de cadenas encriptadas. Con el fin de que estos valores

no sean manipulados por los usuarios. Este componente se encarga de la encriptación de todos los valores pasados por URL.

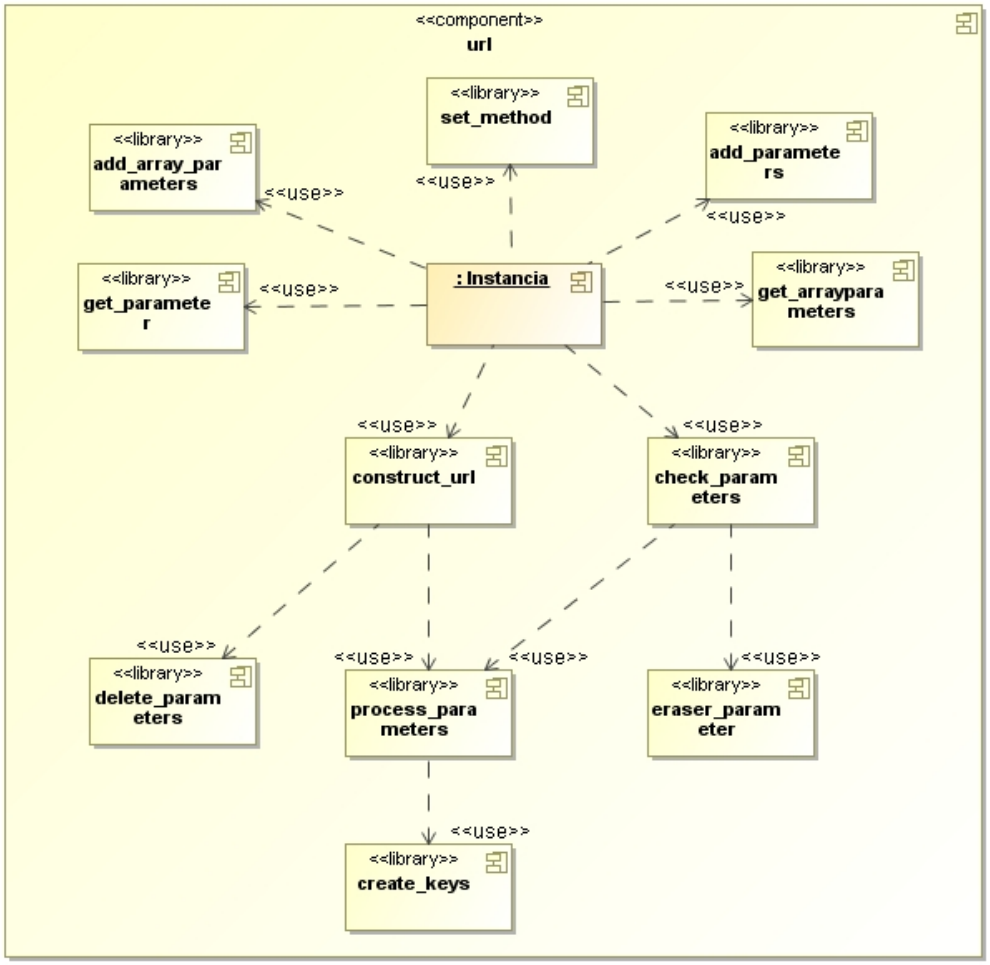


Figura 23. Componente URL

- *set_method* establece el método de encriptación a utilizar.
- *add_array_parameters* adiciona un vector de parámetros.
- *get_parameter* obtiene un parámetro.
- *delete_parameters* elimina parámetros.
- *add_parameters* adiciona un parámetro.
- *get_arrayparameters* obtiene un vector de parámetros.
- *check_parameters* verifica cada uno de los parámetros que vienen en la cadena encriptada.

- *eraser_parameter* elimina un parámetro del listado total.
- *construct_url* concatena todos los parámetros en una cadena.
- *process_parameters* se encarga de encriptar la cadena de parámetros.
- *create_keys* arma las llaves de cada parámetro.

Otro contenedor importante de la capa lógica es el de Interfaz, en la figura se presenta su ubicación en esa capa:

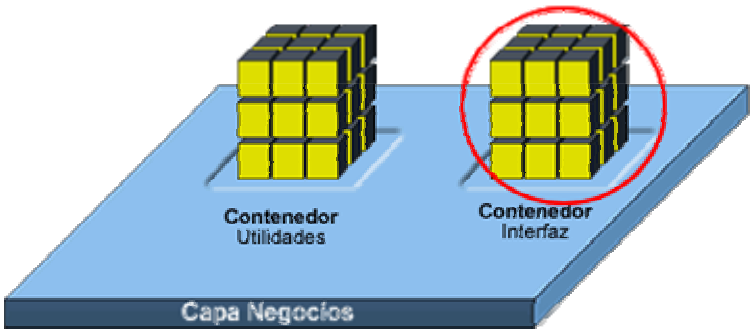


Figura 24. Contenedor Interfaz

La distribución del contenedor de interfaz se puede describir así:

Seis (6) componentes básicos notices, hidden_layer, tabs, panels, menus y warnings.

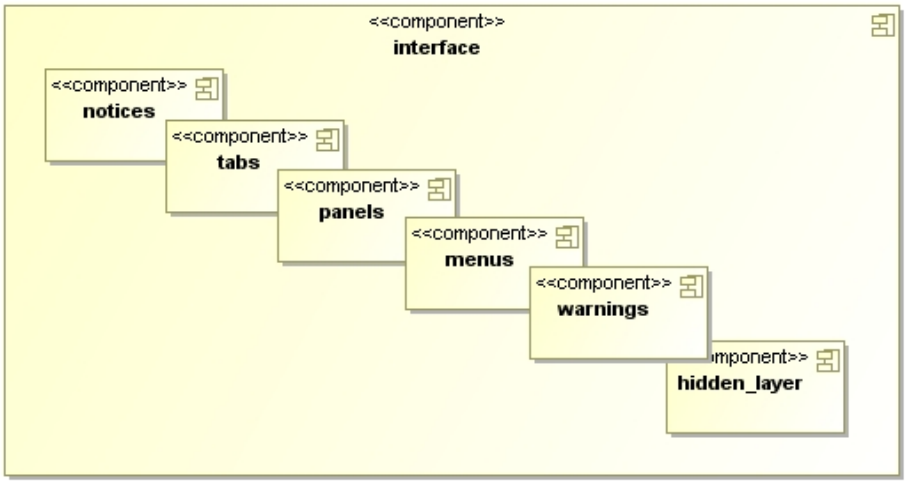


Figura 25. Componente interface

11.4.2.6 Componente notices

Este componente es el responsable de la presentación de mensajes controlados en el sistema. Cuenta con características como tipo de mensajes, mensajes de éxito, de error y de advertencias. Diferentes colores de acuerdo al tipo.

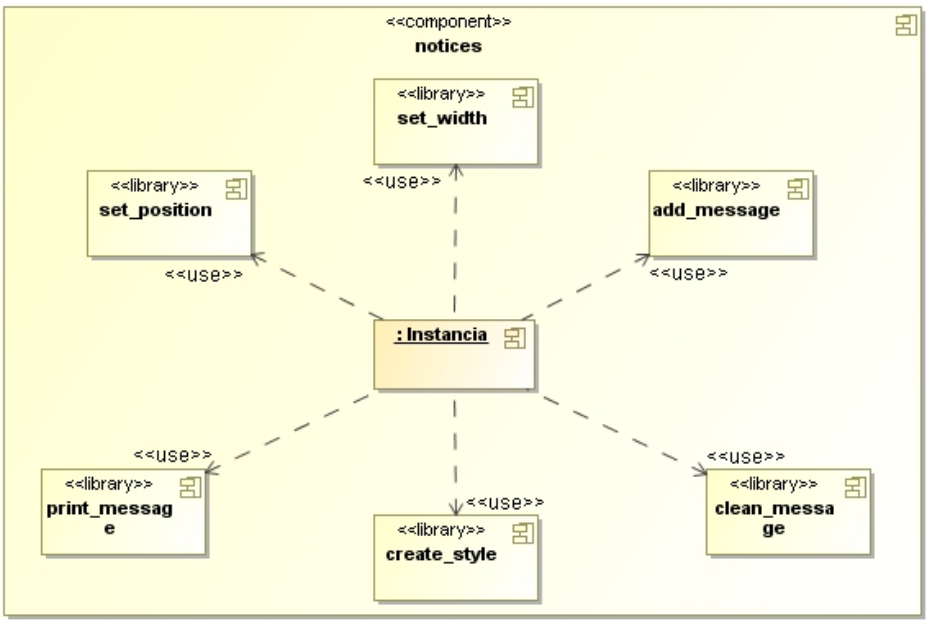


Figura 26. Componente notices

- *set_width* establece el ancho de la ventana del mensaje.
- *set_position* establece la posición vertical y horizontal del mensaje en pantalla.
- *add_message* adiciona un nuevo mensaje.
- *print_message* despliega los mensajes en pantalla.
- *create_style* genera los estilos de los mensajes.
- *clean_message* resetea el objeto de los mensajes.

11.4.2.7 Componente hidden layer

El componente de capa oculta, es el encargado de presentar información en forma de resúmenes, que se visualizan a disposición de un evento onclick.

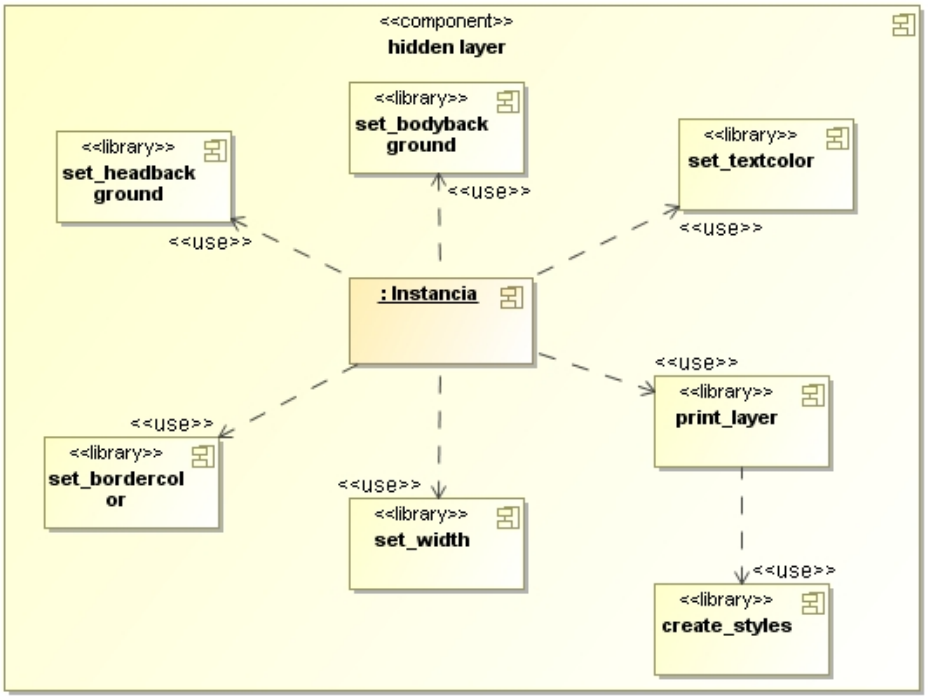


Figura 27. Componente hidden_layer

- *set_bodybackground* establece el color del fondo del cuerpo de la capa oculta.
- *set_headbackground* establece el color del fondo de la cabecera de la capa oculta.
- *set_textcolor* define el color del texto de la capa.
- *set_bordercolor* define el color del borde del objeto capa.
- *set_width* establece el ancho del objeto oculta.
- *print_layer* despliega en pantalla el objeto capa oculta.
- *create_styles* genera los estilos de la capa.

11.4.2.8 Componente panel

Este componente genera un objeto que es capaz de encapsular cualquier otro tipo de elemento dándole la opción de visualizarlo o no.

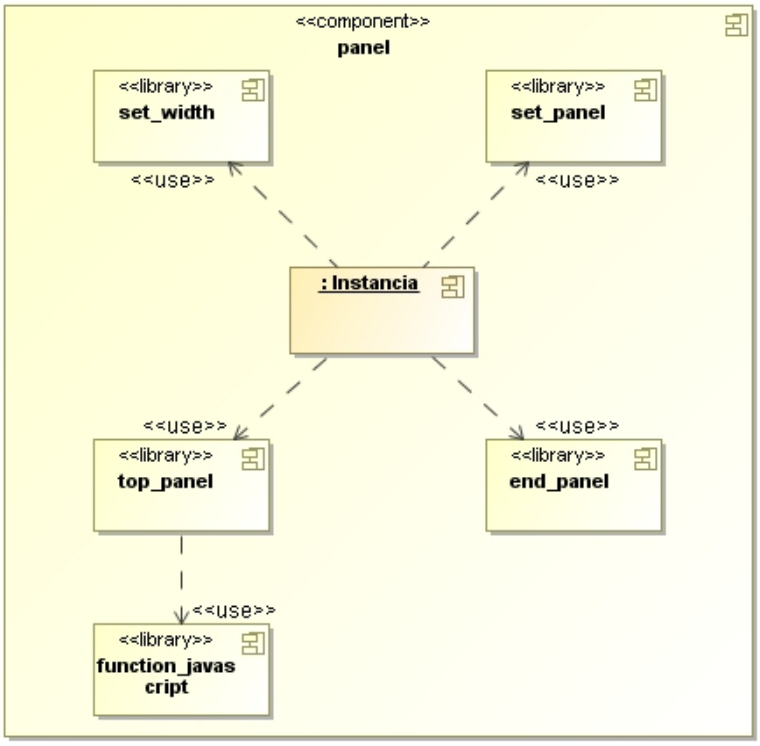


Figura 28. Componente panel

- *set_width* establece el ancho del panel.
- *set_panel* alinea el contenido del panel.
- *top_panel* genera todos los atributos de la cabecera del panel.
- *end_panel* genera todos los atributos de la finalización del panel.
- *function_javascript* genera las cabeceras javascript del panel.

11.4.2.9 Componente menus

Este componente es quizás uno de los más importantes de un sistema porque simplifica el proceso de creación, adición y eliminación de menús y submenús. Con solo instanciar el objeto menús se puede acceder a todas sus propiedades, siendo el sistema configurable dinámicamente.

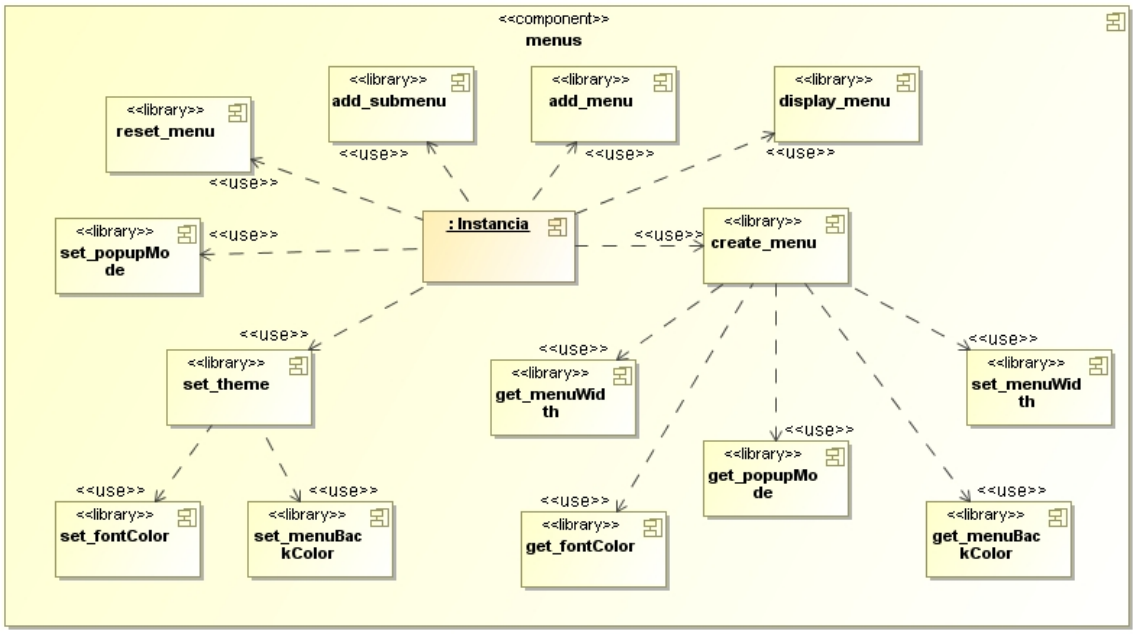


Figura 29. Componente menus

- *set_theme* establece el color de los menús.
- *set_fontColor* establece el color de la letra de los menús.
- *set_menuBackColor* define el color de fondo.
- *set_popupMode* establece el modo de la ventana.
- *reset_menu* inicializa el vector del menú.
- *add_submenu* agrega un submenú a un menú mayor.
- *add_menu* adiciona un menú principal.
- *display_menu* despliega el menú en pantalla.
- *create_menu* genera el menú.
- *get_menuWidth* obtiene el ancho del menú.
- *get_fontColor* obtiene el color de la fuente.
- *get_popupMode* obtiene el modo.
- *set_menuWidth* establece el ancho del menú.
- *get_menuBackColor* obtiene el color de fondo del menú.

11.4.3 Contenedores cliente

Este contenedor corresponde a la capa cliente, en el se describen todas aquellas librerías relacionadas con las validaciones de tipo de datos, integridad de la información, y todas aquellas comprobaciones genéricas, que se deben realizar en cualquier sistema Web.

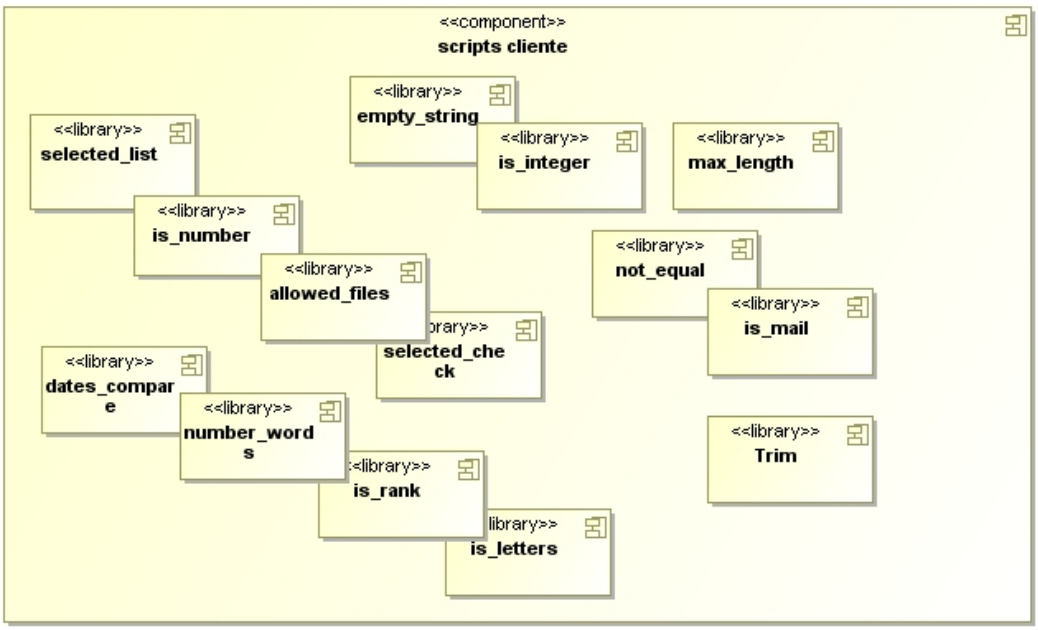


Figura 30. Componente scripts

Los métodos más importantes son:

- *empty_string:* verifica si un campo del formulario se encuentra vacío. En caso tal, presenta un mensaje en pantalla de que el campo específico debe ser diligenciado.
- *is_integer:* verifica si un valor es entero positivo o negativo.
- *max_length:* permite definir la cantidad máxima de caracteres que puede tener una palabra.
- *selected_list:* verifica si en un elemento de selección tipo lista se ha seleccionado un elemento diferente del por defecto.
- *is_number:* verifica si un valor es numérico, puede ser entero o real.
- *allowed_files*: define el listado de tipos de archivos que pueden ser procesados. Por ejemplo sólo .ppt, .txt. Por tanto, si se tratara de procesar un archivo .doc el sistema presentaría un mensaje de advertencia de que ese tipo de archivo no es permitido.

- *selected_check:* método que permite verificar si en un listado de elementos de selección múltiple se ha marcado por lo menos una opción.
- *not_equal:* método que permite comparar el valor digitado en el formulario contra un valor especificado.
- *is_mail:* verifica si el valor digitado corresponde a un correo electrónico valido. Es decir, que este bien armado, por ejemplo: nombreusuario@servidorcorreo.dominio
- *Trim:* elimina los espacios en blanco al principio y al final de una palabra.
- *dates_compare:* método que permite comparar dos fechas para comprobar si la fecha1 es mayor que la fecha2. Donde fecha1 y fecha2 son los parámetros de entrada de la función.
- *number_words:* permite limitar cuantas palabras pueden ser digitadas en un elemento de texto de un formulario.
- *is_rank:* metodo que permite verificar si un valor se encuentra dentro de un rango establecido. Se define un límite inferior y uno superior.
- *is_letters:* permite comprobar si un valor no es carácter, si tiene valores numéricos presenta un mensaje de advertencia, que solo se permiten letras.

11.5 Vista Física

Como se describió en la sección *11.1.7 Conclusión marco referencia modelo Kruchten 4+1*, para la implementación de esta vista se hará uso de los diagramas de despliegue de UML. Por tanto, se hace necesario una breve descripción de sus características. Los diagramas de despliegue presentan la disposición de los elementos físicos que componen un sistema, nodos y la forma como se distribuyen los componentes dentro de dichos nodos. Un nodo es un recurso en tiempo real, tal como un procesador o un servidor. Los nodos se conectan por medio de relaciones bidireccionales. Con esta vista se puede determinar la distribución y la asignación de recursos. Al interior de los nodos se pueden presentar componentes en tiempo de ejecución.

La figura 31, representa los nodos y componentes que interactúan en el modelo propuesto:

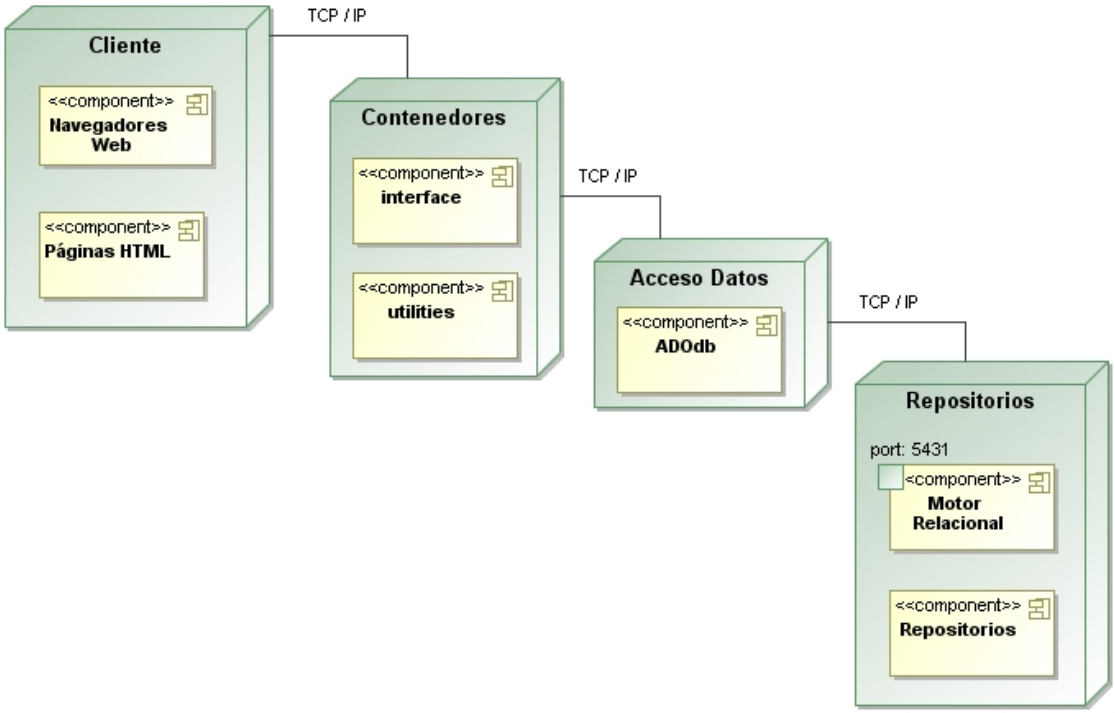


Figura 31. Diagrama de despliegue

El diagrama está formado por tres nodos así:

Cliente este nodo representa la máquina cliente que accede vía TCP/IP a la aplicación mediante páginas HTML, soportadas en un navegador Web, capaz de interpretar páginas CSS y JavaScritp.

Contenedores este nodo representa el servidor Web donde se almacena físicamente los componentes de interface y utilities.

Acceso a Datos los componentes de software del nodo contenedores se encargan de hacerle peticiones al nodo de acceso a datos que está formado por el componente ADODB (Database Abstraction Library for PHP).

Repositorios este nodo está constituido por el motor relacional de base de datos PostgreSQL y los repositorios donde se almacena toda la información necesaria para el funcionamiento de la aplicación.

11.6 Vista de Casos de Uso

En el desarrollo de un sistema software particular los diagramas de casos de uso son la base para la construcción del prototipo, pues dan los lineamientos de las funcionalidades del sistema. La arquitectura planteada en el presente proyecto no corresponde a un modelo particular por tanto, los casos de uso resultan luego del análisis de todas las vistas preestablecidas. A continuación se presentan los diagramas de casos de uso agrupados por contenedores ver sección *11.3 Vistas: Lógica y de Procesos.*

11.6.1 Diagrama de casos de uso contenedores de acceso a datos

Guardando relación con los contenedores descritos, se presentan los casos de uso del contenedor acceso a datos.

En la figura 32, se presenta el actor desarrollador, cuyas funciones están relacionadas con la gestión de transacciones del sistema, el se encarga de administrar el acceso a la base de datos por medio de transacciones basadas en el estándar SQL. Puede iniciar, ejecutar, finalizar transacciones, posicionar un registro de la consulta a la primera, final o siguiente posición.

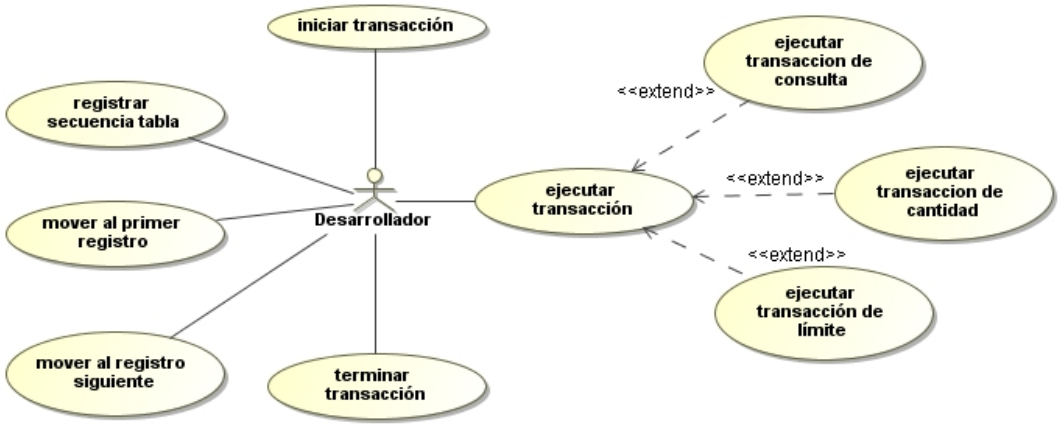


Figura 32. Casos de uso contenedor de acceso a datos

11.6.2 Diagrama de casos de uso contenedores de negocios

El contenedor de negocios está integrado por varios componentes como son el componente de interfaz, utilidades, clases y Web.

La figura 33, ilustra el comportamiento del usuario desarrollador, el cual cuenta con el control total de este componente, puede registrar, editar, eliminar, listar menús, así como también administrar las propiedades de los paneles, mensajes y capas ocultas.

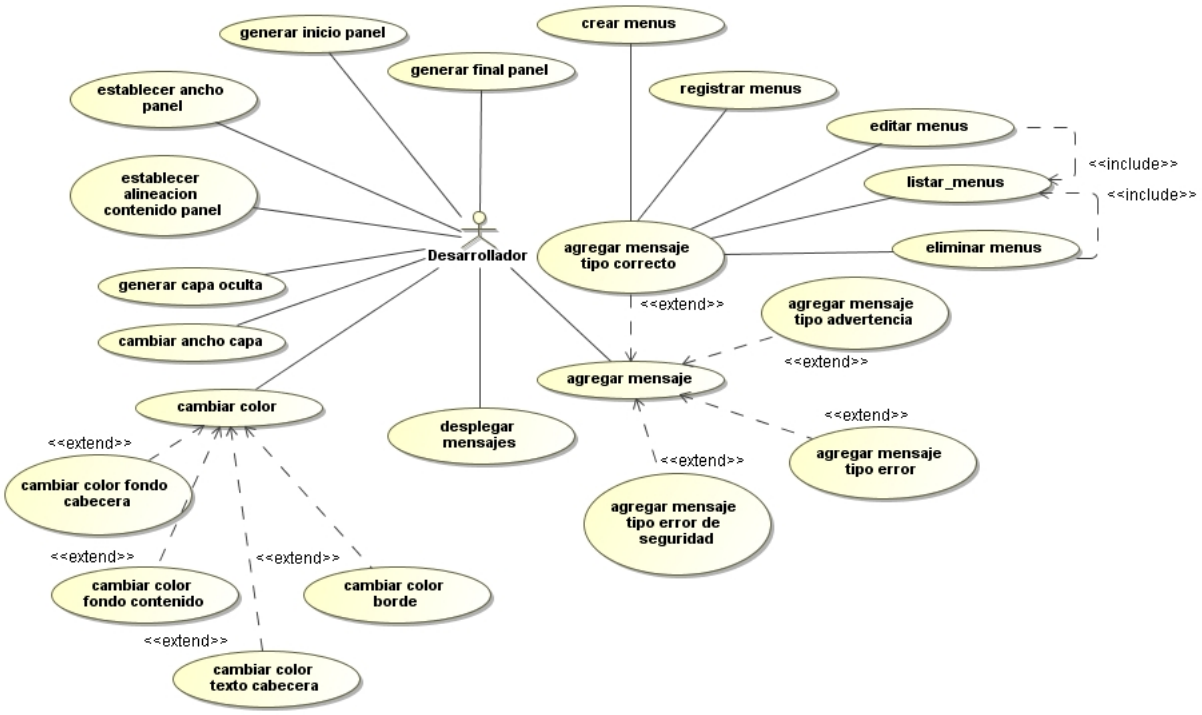


Figura 33. Casos de uso de componente Interfaz

Otro de los contenedores descritos es el de utilidades, a continuación se presenta el diagrama de los casos de uso que se presentan en los diversos componentes que lo conforman.

El diagrama (figura 34), presenta el actor desarrollador. Éste usuario posee todas los privilegios controla la gestión de archivos, búsquedas, fechas, el editor de HTML, horas, paso de parámetros por URL, opciones para armar listas de chequeo, radiobuttons, entre otros.

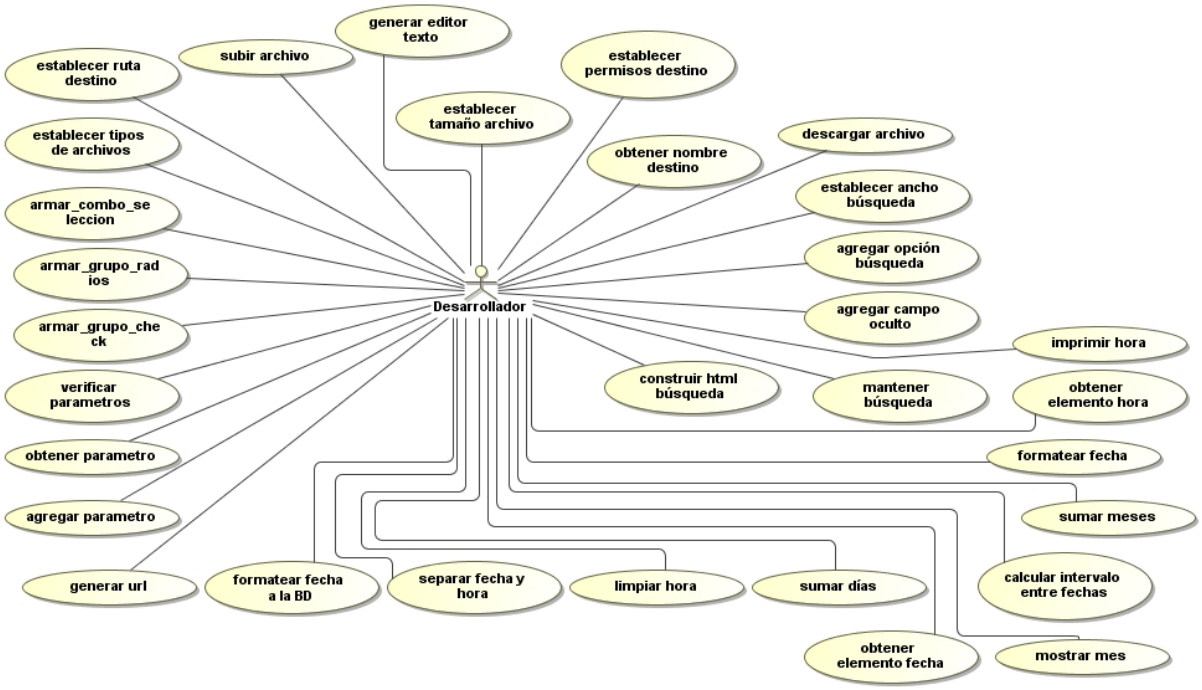


Figura 34. Casos de uso componente Utilidades

En la figura 35, se representan las funcionalidades que puede tener un actor desarrollador con respecto a las áreas de clases del negocio y métodos de las mismas.

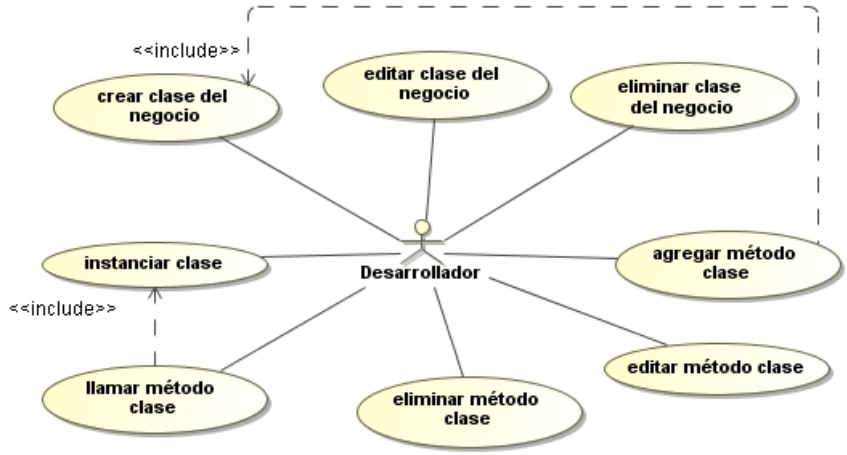


Figura 35. Casos de uso contenedor clases

11.6.3 Diagrama de casos de uso contenedores cliente

El diagrama de casos de uso de este componente (figura 36) ilustra las funcionalidades disponibles para el actor Desarrollador.

El Desarrollador tiene control sobre todas las validaciones del sistema por medio del lenguaje cliente javascript, como son rango de valores, tipos permitidos de archivos, eliminación de espacios en blanco de palabras, correos validos, fechas mayores y menores, tipos de datos numéricos o caracteres, selección de por lo menos un valor en los elementos de formularios, entre otras.

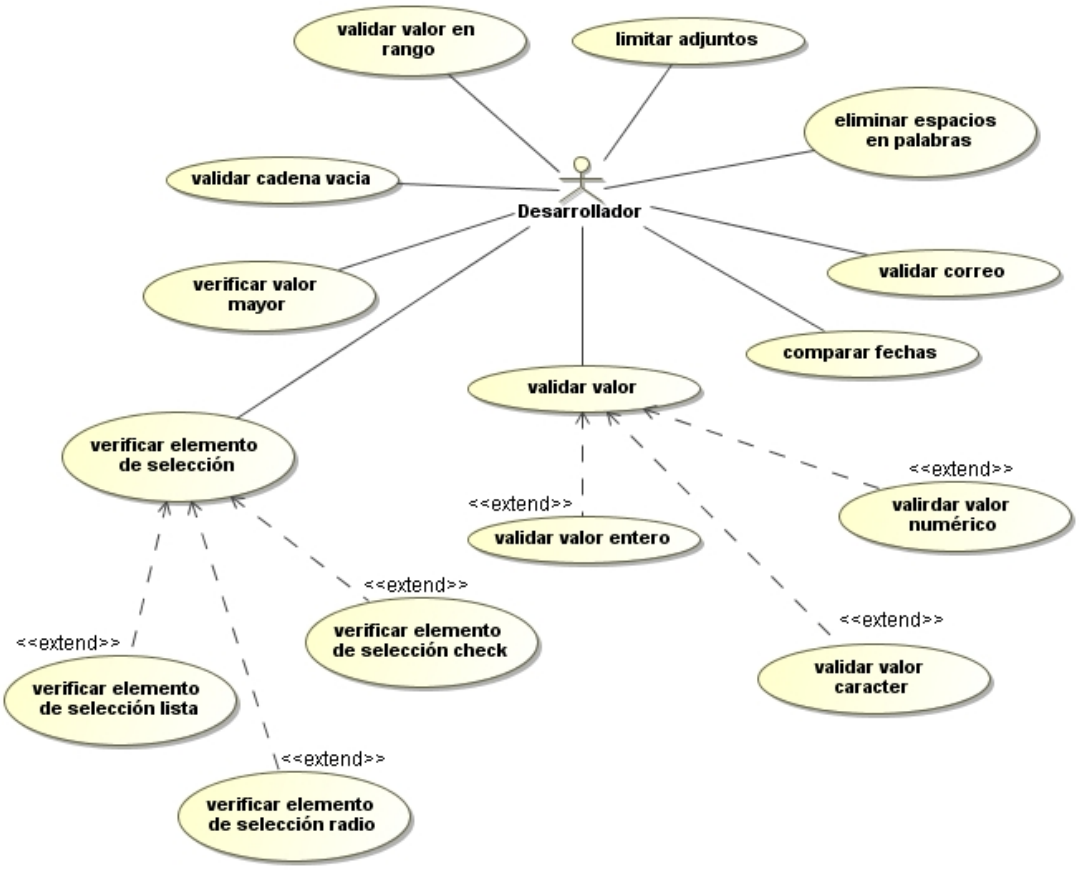


Figura 36. Casos de uso contenedores cliente

12 VERIFICACIÓN DEL MODELO DE ARQUITECTURA

La reutilización de diseño y código fuente es una práctica normal en un proceso de desarrollo, sin embargo en la actualidad y debido a la rapidez con la cual hay que afrontar los compromisos en el mercado, el concepto de reutilización se puede y se debe utilizar en etapas anteriores en el modelo de desarrollo. Si bien es cierto todo sistema informático es diferente, existen requerimientos que se diseñan y construyen una y otra vez cada vez que se realiza un desarrollo, para evitar ésta mala práctica se crearon los patrones, los cuales definen un problema común como un núcleo, de tal manera que pueda ser utilizado las veces que sea necesario en un entorno especifico, por ende, así como se reutiliza un diseño ó el código fuente, así mismo se puede llegar a reutilizar un requerimiento y/o un requisito de un cliente.

Por lo anterior, y para una adecuada verificación del Modelo propuesto, no solo se debe construir un prototipo que permita verificar las ventajas de usar una Arquitectura, sino que se deben contemplar fases de un proceso de desarrollo con calidad. Esto implica analizar desde los requisitos iniciales hasta la implementación del código en la construcción de componentes. A continuación se presenta la verificación del modelo teniendo como etapa de inicio la fase de requisitos.

12.1 Reutilización de requisitos

El Modelo de Arquitectura propuesto no incluye una estrategia metodológica para la construcción de requisitos, sin embargo, es necesario definir un conjunto de pasos que permitan reutilizar un requisito, de tal forma que se puedan aprovechar al máximo los artefactos definidos en el presente modelo. A continuación se presentan tres (3) lineamientos claves para permitirle a un arquitecto de software, diseñador y/o desarrollador potenciar al máximo el uso de la Arquitectura propuesta.

1. Inicialmente, los requisitos deben ser clasificados en dos grupos: Requisitos de Interfaz de usuario[38] y Requisitos Funcionales[39], tal como lo define la norma IEEE 830.

[38] Dado que el presente proyecto propone cambios a la forma en la forma como se construyen los requisitos, en adelante este tipo de requisitos se denominará: requisitos de entorno

2. Los requisitos de entorno deberán usar una plantilla especial que haga referencia a tres elementos:
    a. Titulo del requisito.
    b. Descripción detallada del requisito.
    c. Plantilla en xhtml, o en su defecto una imagen que represente la descripción detallada del requisito, usando para la construcción de esta plantilla los artefactos que se definieron en el apartado *"11.4.2 Contenedores de negocios*".
3. Los requisitos de negocio deberán hacer referencia a:
    a. Titulo del requisito.
    b. Actor o grupo de actores que pueden usar la funcionalidad requerida.
    c. Un resumen del resultado posible que un actor debería observar cuando se inicie este requisito.
    d. Un flujo de eventos que indique los pasos detallados que el actor puede tomar de acuerdo a situaciones propias de la lógica del negocio.
    e. Una descripción de la acción que un actor puede usar en el sistema con su respectiva respuesta, en esta parte, se pueden utilizar los artefactos definidos en cualquiera de los contenedores definidos en el apartado "*11.4 Vista de desarrollo*".
    f. La definición de posibles escenarios, si es que existen.

En el *anexo A,* se presenta la verificación de lo anteriormente expuesto, presentando cómo se deben construir los requisitos para permitir su reutilización con base en el Modelo de Arquitectura propuesto.

12.2 Reutilización de diseño

El paso siguiente luego de la construcción de requisitos, es la realización de las estructuras de diseño, para así conocer el panorama general del sistema a construir. Para el caso de esta propuesta, se cuentan con unos requisitos genéricos que pueden ser reutilizables en cualquier sistema que presente las características definidas en este proyecto. Ahora bien, los

[39] Dado que el presente proyecto propone cambios a la forma en la forma como se construyen los requisitos, en adelante este tipo de requisitos se denominará: requisitos de negocio

diagramas que se construyan como consecuencia del paso anterior, se podrán usar también en forma global. Permitiendo así la reutilización de diseño de un proyecto software.

Para que estos diseños puedan ser aplicados, deben poseer algunas características como:

1. Deben estar construidos con una herramienta de modelado reconocida, como UML, para así manejar elementos comunes (actores, casos de uso, entre otros).
2. El nivel de especificidad de los casos de uso debe ser mínimo, no se deben usar casos de uso generales o que agrupen otras funcionalidades.
3. La parte de interfaz del sistema debe estar completamente especificada, para así mantener una base común.

12.3 Construcción de prototipo Software

El presente trabajo de investigación se vale de la caracterización realizada como parte el objetivo especifico número 1 para establecer un posible repositorio de requisitos, ya que la caracterización hace una clasificación de los sistemas Web por funcionalidades.

De conformidad con la caracterización realizada, se seleccionó "*Educación en Línea*" y dentro de este sistema los componentes:

- Validación de usuarios (acceso y seguridad).
- Administración de usuarios.
- Administración de roles.

Para así evidenciar como se puede reutilizar requisitos, diseño y código. A continuación se presentan los diagramas de casos de uso de los componentes del negocio.

12.3.1 Diagrama de casos de uso

En la sección "*10 Caracterización de Aplicaciones Web"*, se presentan los resultados encontrados del proceso de caracterización de sitios Web con base en los criterios expuestos en el presente trabajo. Encontrando así unos componentes comunes a estas

aplicaciones. A continuación se presenta estos tres componentes básicos: Seguridad de acceso, Gestión de Usuarios y Roles.

En el diagrama (ver figura 37), se describe el actor Desarrollador. Es el usuario con máximos privilegios dentro de este componente puede realizar todas las funciones disponibles como: verificar la integridad del nombre de usuario y la clave que ingresen al sistema, registrar todos los accesos y a la vez permitir el ingreso a cada página de la aplicación. Los permisos que se manejan son por usuario y por funcionalidad.

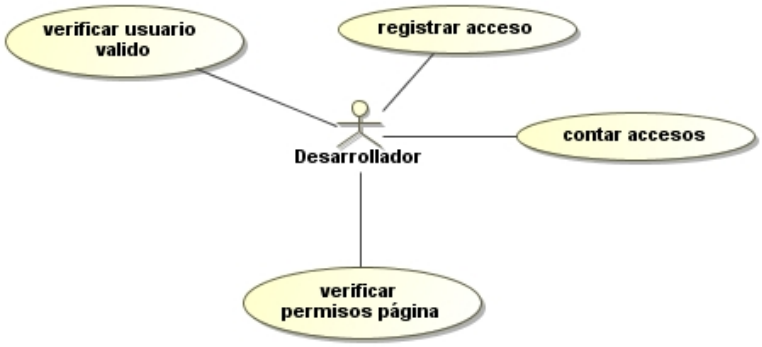


Figura 37. Casos de uso componente de seguridad

En la figura 38, se presenta el actor administrador, cuyas funciones están relacionadas con la gestión de los usuarios del sistema, el se encarga de mantener actualizada la información de los usuarios, puede registrar, editar, eliminar, activar e inactivar los usuarios.

El otro actor Cliente, tiene la opción de actualizar su información personal (nombres, apellidos, dirección, entre otros).

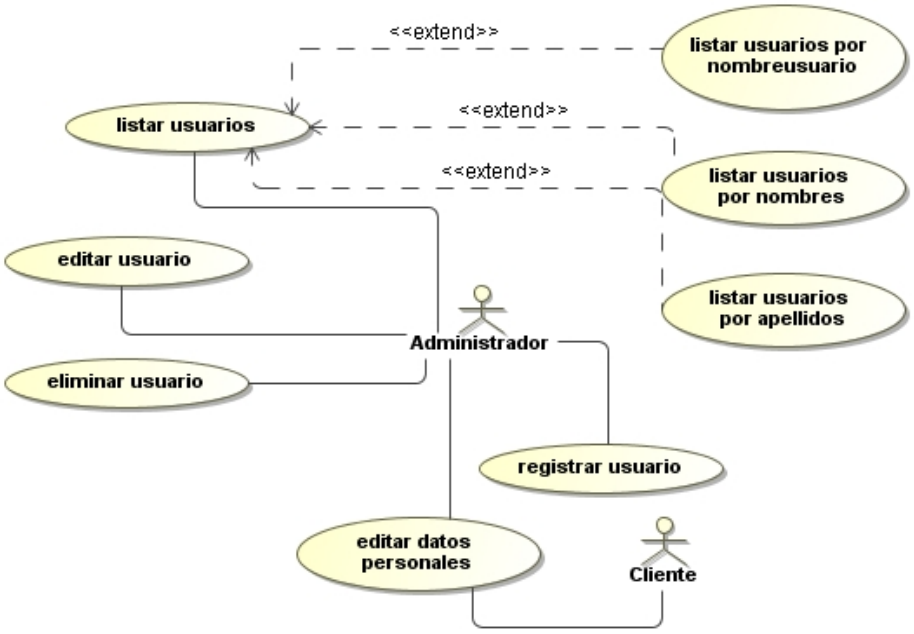


Figura 38. Casos de uso componente usuarios

En el diagrama (ver figura 39), se describe el actor Desarrollador. Éste usuario cuenta con máximos privilegios dentro de este componente puede realizar todas las funciones disponibles como: registrar, editar, listar y eliminar roles, así como la asignación de estos a usuarios. A su vez puede administrar las funcionalidades del sistema. Un usuario puede recibir la asignación de un rol o de una funcionalidad específica. Las funcionalidades también pueden ser asignadas a los roles.

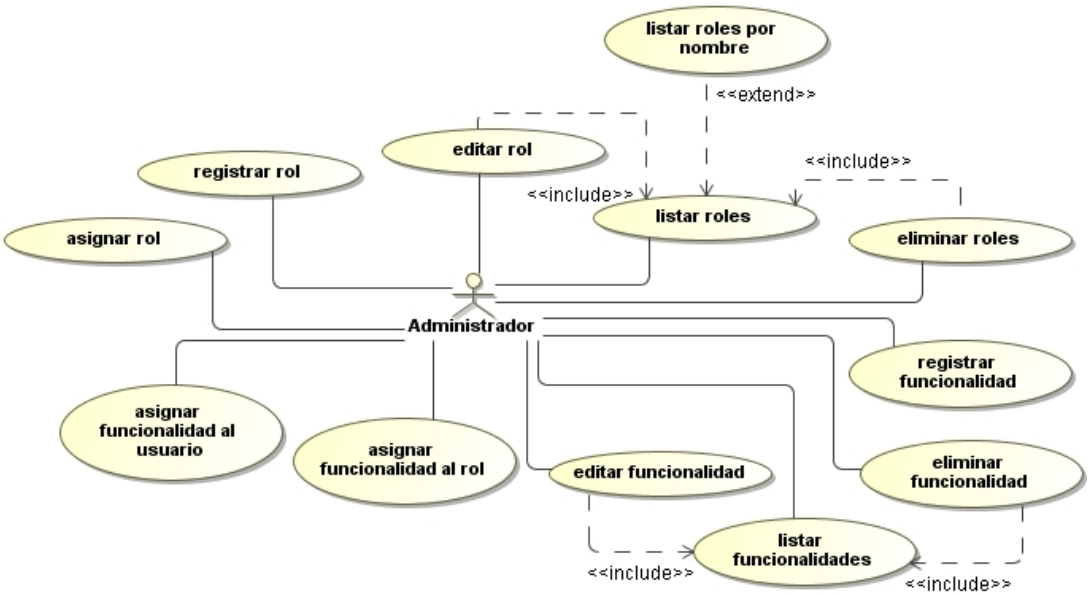


Figura 39. Casos de uso componente roles

13 CUMPLIMIENTO DE OBJETIVOS

Objetivo Relacionado	Resultados Obtenidos	Cumplimiento
1	Un (1) documento con la Encuesta Diagnostico Calidad y Arquitectura Software. Elemento fundamental a la hora de conocer el estado del arte de las empresas desarrolladores de software en la actualidad.	100%
1	Un (1) documento con la Caracterización de Desarrollos Software con Orientación hacia Internet, en Sistemas de Código Abierto.	100%
2	Diagramas de enlaces y tuberías que describen la vista Lógica y de Procesos.	100%
2	Diagramas de Despliegue que representan la vista Física de la AS.	100%
2	Diagramas de Componentes que describen la parte estática de la AS.	100%
2	Diagramas de Casos de Uso que representan la parte dinámica del modelado.	100%
3	Construcción de tres (3) componentes software que permitan visualizar la reutilización de diseño y código.	100%
4	Un artículo titulado Caracterización de Desarrollos Software con Orientación hacia Internet, en Sistemas de Código Abierto.	100%

14 CONCLUSIONES

- La construcción de herramientas para el análisis cuantitativo de las clasificaciones de sitios Web es un aporte significativo del presente proyecto. Se construyeron dos (2) herramientas para la recopilación y análisis de la información.

- Se plantearon criterios para la selección y agrupación de los diferentes tipos de sitio Web, de forma tal que se pudo llegar a un consenso de esta clasificación.

- A pesar de la diversidad de sitios Web que se encuentran en el mercado al final todos cuentan con características comunes y repetibles entre ellos.

- Luego de identificar los elementos comunes entre las clasificaciones se hizo posible la generación de vistas de la arquitectura software que sirvieran como línea de base para la implementación y desarrollo de cualquier sitio Web de código abierto.

- Con la construcción del prototipo se evidenció la reutilización del diseño y código en cualquier aplicación. Esto es de gran utilidad porque cual sería el sentido de “reinventar la rueda”. Evitando así la repetición de código innecesario, proceso que resulta ser doloroso para los ingenieros de desarrollo.

- El uso de componentes reutilizables es propicio para el desarrollo de aplicaciones mas especializadas, puesto que ya no sería necesario gastar tiempo en la construcción de los componentes básicos y se podría invertir ese esfuerzo en la construcción de las características intrínsecas de cada aplicación. Facilitando así el mantenimiento y acelerando el proceso de desarrollo software.

15 RECOMENDACIONES

- Se propone que el modelo de Arquitectura Software planteado, sea usado por los estudiantes de pregrado y postgrados que deban construir aplicaciones Web de código abierto. Este proceso puede ser llevado a cabo a través del centro de investigación al cual está adscrito la propuesta.

- Para el desarrollo de este tipo de trabajos de maestría, es necesario además de conocer y dominar la parte conceptual, tener experiencia en el desarrollo, implementación e implantación de aplicaciones Web con las características analizadas en la propuesta.

- Dar continuidad al desarrollo de este proyecto, mediante la implementación de nuevos componentes que permitan ampliar el funcionamiento del sistema. La caracterización y modelado que se realizó, cubre gran parte de los procesos comunes a los diferentes tipos de aplicaciones; lo que facilitará futuros desarrollos, debido a que la lista de requerimientos a implementar ya está documentada.

REFERENCIAS

[1] C. R. Spooner. “A Software Architecture for the 70's: Part I - The General Approach.” Software - Practice and Experience, 1 (Enero-Marzo), pp. 5-37, 1971.

[2] IEEE-Std-1471-2000 Recommended Practice for Recommended Practice for Architectural Description of Architectural Description of Software-Intensive Systems.

[3] CLEMENTS, Paul. “A Survey of Architecture Description Languages”. Proceedings of the International Workshop on Software Specification and Design, Alemania, 1996.

[4] GARLAN, David y SHAW Mary. An Introduction to Software Architecture. School of Computer Science Carnegie Mellon University Pittsburgh, PA, Enero 1994.

[5] BRAUDE, Eric. Ingeniería del Software Una perspectiva orientada a objetos. Alfaomega, 1ª Edición, 2003.

[6] RUMBAUGH James; JACOBSON Ivar; BOOCH Grady. El lenguaje Unificado de Modelado. Manual de Referencia. Pearson Educación S.A., Madrid 2000.

[7] http://es.wikipedia.org/wiki/PHP

[8] http://www.maestrosdelweb.com/editorial/historiaasp/

[9] GIMENO, José Manuel. Noviembre 2005. http://www.laflecha.net/articulos/blackhats/ajax

[10] Szyperski, C. Component Software. Beyond Object-Oriented Programming. Addison-Wesley. 1998.

[11] http://www.lcc.uma.es/~av/Docencia/Doctorado/tema1.pdf

[12] http://www.astic.es/Estudios%20ASTIC/Monogr%C3%A1ficos/Paginas/desarrolloweb.aspx

[13] http://www.uib.es/depart/gte/gte/edutec-e/revelec20/anibal20.htm

[14] http://mosaic.uoc.edu/articulos/gferreres1106.html

[15] http://webpersonal.uma.es/~AFDEZ/Slab/taman/estimapr.htm

[16] PALLIOTTO Diana y ROMANO Gabriel. La Tecnología de la Ingeniería Inversa: Un Método con UML, guiado por Casos de Uso y basado en el Modelo de Vistas 4+1. Universidad Nacional de Santiago del Estero – Facultad de Ciencias Exactas y Tecnologías Departamento de Informática - Av. Belgrano (S) 1912, (4200) Santiago del Estero, Argentina.

[17] CUESTA, Quintero Carlos E. Ingeniería de Software II Tema 1: Introducción. Universidad Rey Juan Carlos, 2006.

[18] DANIELE, Marcela. Teoría 11: EL ARTE DE MODELAR UML (Unified Modeling Language). UNRC, 2007.

BIBLIOGRAFÍA

- BOOCH, Grady. RUMBAUGH, James y JACOBSON, Ivar. El lenguaje Unificado de Modelado. México. Addison Wesley, 1999.

- GUTIÉRREZ, Abraham y BRAVO, Ginés. PHP 5 a través de ejemplos. Alfaomega, 1ª Edición, 2005.

- ORÓS, Juan Carlos. Diseño de páginas Web interactivas con JavaScript y CSS. Alfaomega, 4ª Edición, 2004.

- BRAUDE, Eric. Ingeniería del Software Una perspectiva orientada a objetos. Alfaomega, 1ª Edición, 2003.

- GARLAN, David y SHAW, Mary. An Introduction to Software Architecture. School of Computer Science Carnegie Mellon University Pittsburgh, PA, Enero 1994.

- REYNOSO, Carlos Billy. Introducción a la Arquitectura de Software. Versión 1.0 – Marzo de 2004. Universidad de Buenos Aires. Disponible en [http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.asp].

- REYNOSO, Carlos Billy y KICILLOF, Nicolás. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Versión 1.0 – Marzo de 2004. Universidad de Buenos Aires. Disponible en [http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp].

- NIETO SANTISTEBAN, María A. Ingeniería Web. Construyendo Web Apps. Universidad de Extremadura. Disponible en [http://www.informandote.com/jornadasIngWEB/articulos/jiw01.pdf]

- ESCALONA CUARESMA, María José y GONZÁLEZ ROMANO, José Mariano. Metodologías para la Ingeniería Web. Universidad de Sevilla. 2007. Disponible en

[http://www.lsi.us.es/docencia/get.php?id=2086].

- SILVA, Darío Andrés. Construyendo aplicaciones Web con una metodología de diseño orientada a objetos. Universidad Nacional de La Plata. Disponible en [http://www.lifia.info.unlp.edu.ar/papers/2001/Silva2001.pdf].

- GUARACHI Y, Franklin F. Reutilización de Requerimientos. Universidad Mayor de San Andrés. Disponible en [http://members.fortunecity.es/gfranklin/archivos/reusoreq.pdf.txt].

- CALVO TUDELA, José Carlos. Reutilización, Parametrización y Patrones de Diseño. Disponible en [http://www.intelligenia.com/documentos/reuparypatdis/ReuParyPatDis.pdf].

- TORRALBA MARTÍNEZ, José. Reutilización del conocimiento del diseño de software. Consideración en la determinación del precio de oferta al cliente. Universidad Politécnica de Valencia. 2004. Disponible en [http://io.us.es/cio2004/comunicaciones/1005-1014.pdf].

- PEREZ SOLTERO, Alonso, BARCELO VALENZUELA, Mario. Diseño de una Ontología para la Reutilización del Conocimiento en los Procesos de Auditoría del Conocimiento. Universidad de Sonora. 2008. Disponible en [http://www.aperez.com.mx/ponenCISCI2008.pdf].

- LÓPEZ QUESADA, Juan Antonio. Fundamentos de Ingeniería del Software. Campus Universitario de Espinardo – Murcia. 2007. Disponible en [http://dis.um.es/~lopezquesada/documentos/FIS_0607/curso/Tema10.pdf].

- MARQUÉS, José Manuel. [http://pisuerga.inf.ubu.es/lsi/Actividades/curso1998_99/reut-burgos.ppt]

ANEXO A

A1. Requisitos de Entorno

Nombre requisito	Tipo de páginas
Descripción	El sistema deberá contar con tres tipos de páginas como son: ***Página principal:*** Este tipo es la página donde se va a desplegar todo el contenido del sitio, donde se colocaran los listados y formularios principales. Ocupa la pantalla completa. Y se puede observar al usar el evento clic en cualquiera de los menús del sistema. Debe contar con las cabeceras del sistema. Esta contiene además los menús principales. ***Ventana flotante:*** Es una página de tamaño mediano, y se despliega al hacer clic en un link de la página principal. Cuenta con las cabeceras del sistema. ***Ventana flotante de otra flotante:*** Este tipo de ventana se despliega al hacer clic en una ventana flotante. No cuenta con cabeceras. Es la mínima expresión de una página.
Interfaz	

Nombre requisito	Cabeceras del sistema
Descripción	En el sistema se debe contar con tres (3) tipos de cabeceras. Estas se distinguirán una de la otra por su tamaño y presencia o no del logo del sistema. Así: ***Cabecera principal:*** La parta superior estará formada por una imagen que ocupe todo el ancho de la pantalla (ancho: 900px, alto: 100px). Debe contener el logo del sistema en la parte izquierda de la imagen principal. En la parte inferior de la imagen principal deben ir las opciones de menú del sistema. En la parte inferior derecha de los menús debe ubicarse el rol del usuario, acompañado del componente en el cual se encuentra y la opción de salir del sistema. Ubicado en la parte inferior izquierda estará el nombre de la opción de menú que se seleccione. Y debajo de este la navegabilidad del sitio. ***Cabecera flotante:*** La parte superior está formada por una imagen que ocupa todo el ancho de la ventana abierta. No cuenta con logo. Puede llevar menús si son necesarios. Luego de los menús se ubica el nombre del rol y el componente donde se encuentre ubicado el usuario en sesión. Luego va un titulo con la opción a realizar. ***Cabecera flotante de flotante:*** La parte superior está formada por una imagen que ocupe todo el ancho de la pantalla y de alto tendrá 20 px. Luego va el titulo de la página que es la acción a realizar.
Interfaz	

Nombre requisito	Opciones de Menús
Descripción	Los nombres de los menús del sistema serán dinámicos, lo que indica que serán guardados en una tabla de la BD, para que puedan ser administrados. La construcción visual de estos se realizara por medio del componente descrito en la sección *11.4.2.9 Componente menus.* Estos menús pueden contener a su vez submenús. Si la opción de menú tiene submenús aparecerá al lado derecho del nombre un icono de "+". Si el menú principal cuenta con submenús, este no poseerá link de acceso, por el contrario los vínculos serán asignados a cada uno de los submenús. Cada uno de ellos contará con una breve descripción que se podrá visualizar al pasar el mouse por la parte superior del nombre. Los menús principales se despliegan de manera horizontal, es decir uno al lado del otro. Y los submenús se desplegarán a de forma vertical en la parte inferior del menú seleccionado.
Interfaz	

Nombre requisito	Opciones de navegabilidad
Descripción	La navegabilidad del sistema estará ubicada en la parte inferior izquierda del nombre del componente. Debe empezar con la opción de Inicio luego seguirá la ruta de opciones por donde el usuario navegue; dentro del mismo componente. Todas las opciones de la navegación contarán con vínculos a excepción de la última opción, porque esta indicará siempre donde se encuentra ubicado el usuario.
Interfaz	

Nombre requisito	Pantalla de Inicio
Descripción	Esta es la página principal del sistema, es la que se llamará "Inicio". Debe contar como primera medida con las características descritas en el requisito *Cabeceras del sistema (Cabecera principal),* a excepción del último ítem donde dice "Ubicado en la parte inferior izquierda estará el nombre de la opción de menú que se seleccione. Y debajo de este la navegabilidad del sitio." Puesto que el nombre del componente será reemplazado por Bienvenido [nombres apellidos]. En la parte izquierda de la pantalla se debe presentar un resumen de datos importantes del sistema, eso dependerá de la lógica del negocio. En la parte derecha de la pantalla deberá ir la foto del usuario en sesión, así como opciones para actualizar su información personal.
Interfaz	

Nombre requisito	Opciones de listados
Descripción	Los listados del sistema deben contar con las características descritas en la sección *Cabeceras del sistema (Cabecera principal).* En la parte inferior de la navegabilidad deben ir en la parte izquierda los nombres de los criterios utilizados como filtros de forma vertical, es decir, uno debajo del otro. En la parte derecha de esa tabla debe ir el listado de filtros disponibles presentados en forma horizontal uno al lado del otro. Debe llevar una opción para deshabilitar todos los filtros. Debajo de las opciones de filtros, deben ir las opciones de búsquedas (se describirán en otro requisito). Luego de esto se debe colocar la paginación la cual se construirá con una utilidad contenida en la AS propuesta. Inmediatamente debajo de la paginación se colocará el listado de registros. Estos deben tener varias columnas entre ellas el nombre del registro y datos básicos y una última columna denominada herramientas donde se colocarán todas las opciones o acciones que pueden realizarse con ese registro, por ejemplo: edición, eliminar, imprimir, entre otras.
Interfaz	

Nombre requisito	Opciones de Búsquedas
Descripción	Esta opción contará con una tabla gris, donde se escribirán los criterios de búsquedas. Antecedidos por un elemento rabiobutton, indicando así que solo se podrá realizar una búsquedas por un criterio a la vez. Debajo de los criterios debe ir una etiqueta denominada “Buscar”, al lado de ésta una caja de texto que servirá para digitar la palabra a buscar. Y al final un botón que active el evento de búsqueda. Esta opción se deberá construir con el componente descrito en la sección 11.4.2.1 Componente search.
Interfaz	

Nombre requisito	Formularios de captura y edición
Descripción	Debe cumplir el requisito *Cabeceras del sistema (Cabecera principal).* Los formularios de captura / edición deben estar construidos con tablas, la primera fila debe contener el titulo del formulario. Las siguientes filas describirán cada una de las opciones a diligenciar. Las filas deben estar a dos columnas; en la primera columna se colocará la etiqueta del campo y en

	la segunda el tipo elemento de formulario. Estos pueden ser cajas de texto, áreas de texto, elementos de selección (radios, checks, listas), caja de fechas, paneles e iconos para desplegar ventanas flotantes. Los elementos de selección, caja de fechas y paneles serán construidos con los componentes que proporciona la AS.
Interfaz	

Nombre requisito	Colores
Descripción	Los colores de las cabeceras, imágenes, listados y formularios deben estar de acuerdo al estándar de la institución para la cual sea objeto el sistema. Todos estos formatos deben estar almacenados en un tipo de archivo .CSS, para permitir así la unificación en el sistema. Se debe manejar una única hoja de estilos.
Interfaz	

Nombre requisito	Formatos y tamaños de letra
Descripción	El tipo de letra debe ser Tahoma de 8.5. Este tipo de letra se encuentra dentro del listado de las clasificaciones estándares para sitios Web.
Interfaz	

A2. Requisitos de Negocio

Nombre del caso de uso	Registrar usuario	
Actor	Administrador	
Propósito: Registrar un usuario en el sistema.		
Resultados		
Presenta en pantalla un formulario para registrar el usuario.		
Flujo de eventos:		
El actor activa el caso de uso al seleccionar en la pantalla principal el vínculo asociado a USUARIOS, luego elige la opción Nuevo Usuario.		
Acción del actor		**Respuesta del Sistema**

Selecciona Nuevo usuario en el menú de Usuarios. Diligencia el formulario de registro para ingresar un usuario al sistema.	Ingreso de los datos correspondientes al usuario: ▪ Identificación ▪ Tipo documento ▪ Nombres ▪ Apellidos ▪ Teléfono ▪ Dirección ▪ Cargo ▪ Correo electrónico ▪ Nombre de usuario ▪ Contraseña ▪ Estado (activo ó inactivo). ▪ Rol: selección desde una lista predeterminada de roles. Registra el usuario en el sistema
Hace clic en el botón "Registrar Usuario"	Verifica que la información esté completa y la almacena en el sistema, de no ser así muestra un mensaje de advertencia y finaliza el caso de uso.
Escenarios	
Finalización del caso de uso a discreción del usuario El caso de uso finalizará en cualquier instante dentro del flujo de eventos, cuando por acción del usuario se realice una operación no contemplada en el flujo normal de eventos del caso de uso.	

Nombre del caso de uso	Listar usuarios
Actor	Administrador
Propósito: Lista los usuarios registrados en el sistema.	
Resultados	
Visualiza en pantalla la lista de usuarios por criterios de búsqueda.	
Flujo de eventos:	
El actor activa el caso de uso al seleccionar de la pantalla principal el vínculo asociado a USUARIOS, luego hace clic en LISTADO DE USUARIOS.	
Acción del actor	**Respuesta del Sistema**

Selecciona Listado de Usuarios en el menú Usuarios.	Verifica si existen registros en la base de datos de los usuarios. Si existen registros, muestra el listado de usuarios con los siguientes campos: ▪ Usuario ▪ Nombres ▪ Documento ▪ Teléfono ▪ Cargo ▪ Estado ▪ Herramientas: Edición, Envío de correos, Eliminar El sistema debe realizar consultas por los siguientes criterios: No. Documento, Usuario, Nombre Completo y finaliza caso de uso.
El usuario introduce los criterios de búsqueda y da clic sobre el botón “Iniciar Búsqueda”.	Verifica la existencia de los registros para el criterio seleccionado. Si existen registros, muestra el listado de usuarios y finaliza caso de uso. Si no existen registros, muestra mensaje de advertencia y finaliza caso de uso.
Escenarios	
Finalización del caso de uso a discreción del usuario. El caso de uso finalizará en cualquier instante dentro del flujo de eventos, cuando por acción del usuario se realice una operación no contemplada en el flujo normal de eventos del caso de uso. ***Inexistencia de registros de usuario*** Si en el sistema, no existen registros de usuario, éste debe presentar un mensaje de advertencia.	

Nombre del caso de uso	Editar usuario
Actor	Administrador
Propósito: Modificar o actualizar los datos existentes de un usuario.	
Resultados	
Muestra en pantalla los datos del usuario, permitiendo introducir modificaciones.	
Flujo de eventos:	
El actor activa el caso de uso al hacer clic en el vínculo USUARIOS, donde se despliega el listado de usuarios del sistema. Luego selecciona el usuario a editar y hace clic en la opción editar.	
Acción del actor	**Respuesta del Sistema**

Selecciona del listado de usuarios la opción Editar.	Muestra la información básica del usuario para modificar o actualizar la siguiente información: ▪ Identificación ▪ Tipo documento ▪ Nombres ▪ Apellidos ▪ Teléfono ▪ Dirección ▪ Cargo ▪ Correo electrónico ▪ Nombre de usuario ▪ Contraseña ▪ Estado (activo ó inactivo). ▪ Rol: selección desde una lista predeterminada de roles. ▪ Funcionalidad: selección desde un listado de opciones del sistema. Edita los datos del usuario en el sistema
Hace clic en el botón "Editar usuario"	Comprueba que la información este completa y la almacena en el sistema, de no ser así muestra un mensaje de advertencia y finaliza caso de uso.
Escenarios	
Finalización del caso de uso a discreción del usuario. El caso de uso finalizará en cualquier instante dentro del flujo de eventos, cuando por acción del usuario se realice una operación no contemplada en el flujo normal de eventos del caso de uso.	

Nombre del caso de uso	Eliminar usuario
Actor	Administrador
Propósito: Excluir o descartar a un usuario del sistema.	
Resultados	
Visualiza en pantalla el listado de usuarios con la opción de Eliminar cada uno de ellos.	
Flujo de eventos:	
El actor activa el caso de uso al hacer clic en el vínculo USUARIOS, donde se despliega el listado de usuarios del sistema. Luego selecciona el usuario a eliminar y hace clic en la opción eliminar.	
Acción del actor	**Respuesta del Sistema**
Selecciona del listado de usuarios la opción Eliminar. Aplicar un criterio de búsqueda por No. Documento, Usuario, Nombre Completo para listar los usuarios a eliminar.	Verifica que el usuario se encuentre registrado en el sistema.
Hace clic en el botón "Eliminar usuario"	Elimina de la base de datos los registros relacionados con el usuario.
Escenarios	

Finalización del caso de uso a discreción del usuario.
El caso de uso finalizará en cualquier instante dentro del flujo de eventos, cuando por acción del usuario se realice una operación no contemplada en el flujo normal de eventos del caso de uso.

Nombre del caso de uso	Editar datos personales
Actor	Administrador, Cliente
Propósito: Renovar o actualizar los datos personales de un usuario registrado en el sistema.	
Resultados	
Se visualiza en pantalla los datos del usuario, permitiendo introducir modificaciones.	
Flujo de eventos:	
El actor activa el caso de uso cuando digita su nombre y contraseña en la página de acceso al sistema, luego selecciona de la pantalla principal el vínculo Datos personales.	
Acción del actor	**Respuesta del Sistema**
Modificar los datos personales, que considere necesarios.	Verifica que el usuario se encuentra registrado en el sistema. Presenta los datos personales del usuario para modificar o actualizar la siguiente información: ▪ Identificación ▪ Tipo documento ▪ Nombres ▪ Apellidos ▪ Teléfono ▪ Dirección ▪ Cargo ▪ Correo electrónico ▪ Contraseña ▪ Confirmar contraseña Edita los datos del usuario en el sistema
Hace clic en el botón "Editar datos"	Comprueba que la información este completa y la almacena en el sistema, de no ser así muestra mensaje de advertencia y finaliza caso de uso.
Escenarios	
Finalización del caso de uso a discreción del usuario. El caso de uso finalizará en cualquier instante dentro del flujo de eventos, cuando por acción del usuario se realice una operación no contemplada en el flujo normal de eventos del caso de uso.	

Nombre del caso de uso	Registrar rol
Actor	Administrador
Propósito: Registrar un rol en el sistema.	
Resultados	
Presenta en pantalla un formulario para registrar el rol.	
Flujo de eventos:	
El actor activa el caso de uso al seleccionar en la pantalla principal el vínculo asociado a ROLES,	

luego elige la opción Nuevo Rol.	
Acción del actor	**Respuesta del Sistema**
Selecciona Nuevo rol en el menú de Roles. Diligencia el formulario de registro para ingresar un rol al sistema.	Ingreso de los datos correspondientes al rol: ▪ Nombre ▪ Descripción ▪ Funcionalidades: Presenta el listado de funcionalidades del sistema. Con un elemento de selección múltiple. Así un rol puede tener una o varias funcionalidades asignadas. Registra el rol en el sistema.
Hace clic en el botón "Registrar Rol"	Verifica que la información esté completa y la almacena en el sistema, de no ser así muestra un mensaje de advertencia y finaliza el caso de uso.
Escenarios	
Finalización del caso de uso a discreción del usuario El caso de uso finalizará en cualquier instante dentro del flujo de eventos, cuando por acción del usuario se realice una operación no contemplada en el flujo normal de eventos del caso de uso.	

Nombre del caso de uso	Listar roles
Actor	Administrador
Propósito: Lista los roles registrados en el sistema.	
Resultados	
Visualiza en pantalla la lista de roles por criterios de búsqueda.	
Flujo de eventos:	
El actor activa el caso de uso al seleccionar de la pantalla principal el vínculo asociado a ROLES, luego hace clic en LISTADO DE ROLES.	
Acción del actor	**Respuesta del Sistema**
Selecciona Listado de Roles en el menú Roles.	Verifica si existen registros en la base de datos de los roles. Si existen registros, muestra el listado de roles con los siguientes campos: ▪ Nombre ▪ Descripción ▪ Herramientas: Edición, Eliminar El sistema debe realizar consultas por los siguientes criterios: Nombre.
El usuario introduce los criterios de búsqueda y da clic sobre el botón "Iniciar Búsqueda".	Verifica la existencia de los registros para el criterio seleccionado. Si existen registros, muestra el listado de roles y finaliza caso de uso.

	Si no existen registros, muestra mensaje de advertencia y finaliza caso de uso.
Escenarios	
Finalización del caso de uso a discreción del usuario. El caso de uso finalizará en cualquier instante dentro del flujo de eventos, cuando por acción del usuario se realice una operación no contemplada en el flujo normal de eventos del caso de uso. ***Inexistencia de registros de roles*** Si en el sistema, no existen registros de roles, éste debe presentar un mensaje de advertencia.	

Nombre del caso de uso	Editar rol
Actor	Administrador
Propósito: Modificar o actualizar los datos existentes de un rol.	
Resultados	
Muestra en pantalla los datos del rol, permitiendo introducir modificaciones.	
Flujo de eventos:	
El actor activa el caso de uso al hacer clic en el vínculo ROLES, donde se despliega el listado de roles del sistema. Luego selecciona el rol a editar y hace clic en la opción editar.	
Acción del actor	**Respuesta del Sistema**
Selecciona del listado de roles la opción Editar.	Muestra la información básica del rol para modificar o actualizar la siguiente información: ▪ Nombre ▪ Descripción ▪ Funcionalidades: Presenta el listado de funcionalidades del sistema. Con un elemento de selección múltiple. Así un rol puede tener una o varias funcionalidades asignadas. Edita los datos del rol en el sistema.
Hace clic en el botón "Editar rol"	Comprueba que la información este completa y la almacena en el sistema, de no ser así muestra un mensaje de advertencia y finaliza caso de uso.
Escenarios	
Finalización del caso de uso a discreción del usuario. El caso de uso finalizará en cualquier instante dentro del flujo de eventos, cuando por acción del usuario se realice una operación no contemplada en el flujo normal de eventos del caso de uso.	

Nombre del caso de uso	Eliminar rol
Actor	Administrador
Propósito: Excluir o descartar a un rol del sistema.	
Resultados	
Visualiza en pantalla el listado de roles con la opción de Eliminar cada uno de ellos.	
Flujo de eventos:	
El actor activa el caso de uso al hacer clic en el vínculo ROLES, donde se despliega el listado de roles	

del sistema. Luego selecciona el rol a eliminar y hace clic en la opción eliminar.	
Acción del actor	**Respuesta del Sistema**
Selecciona del listado de roles la opción Eliminar. Aplicar un criterio de búsqueda por Nombre para listar los roles a eliminar.	Verifica que el rol se encuentre registrado en el sistema.
Hace clic en el botón "Eliminar rol"	Elimina de la base de datos los registros relacionados con el rol.
Escenarios	
Finalización del caso de uso a discreción del usuario. El caso de uso finalizará en cualquier instante dentro del flujo de eventos, cuando por acción del usuario se realice una operación no contemplada en el flujo normal de eventos del caso de uso. ***Existencia de asignación de usuarios al rol*** El caso de uso finalizará sin realizarse la eliminación del rol, si el rol seleccionado tiene asignado por lo menos un usuario del sistema.	