MARCO DE REFERENCIA CENTRADO EN LA ARQUITECTURA PARA LA MEJORA DE CARACTERISTICAS DE USABILIDAD EN EL DESARROLLO DE APLICACIONES WEB CONSTRUIDAS POR MIPYMES

MARIA VERONICA FERNÁNDEZ DE VALDENEBRO
MARIA AMPARO HORMIGA JUSPIAN
ALEYDA TULANDE ARROYO

Director: Dr. JOSÉ LUIS ARCINIEGAS HERRERA
Asesor: Dr. CESAR ALBERTO COLLAZOS ORDOÑEZ

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELEMATICA
POPAYÁN, MAYO DE 2008

MARCO DE REFERENCIA CENTRADO EN LA ARQUITECTURA PARA LA MEJORA DE CARACTERISTICAS DE USABILIDAD EN EL DESARROLLO DE APLICACIONES WEB CONSTRUIDAS POR MIPYMES

MARIA VERONICA FERNÁNDEZ DE VALDENEBRO
MARIA AMPARO HORMIGA JUSPIAN
ALEYDA TULANDE ARROYO

Director: Dr. JOSÉ LUIS ARCINIEGAS HERRERA
Asesor: Dr. CESAR ALBERTO COLLAZOS ORDOÑEZ

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELEMATICA
POPAYÁN, MAYO DE 2008

AGRADECIMIENTOS

Primero y antes que nada, queremos dar gracias a **Dios**, por cumplir su promesa y aparejar los recursos necesarios para llevar a cabo este trabajo de grado, así como también por habernos dado la inteligencia y haber estado con nosotras en todas las circunstancias difíciles, siendo nuestra luz y guía dándonos amor en cada momento de este camino.

Agradecer hoy y siempre a nuestras madres Maria Del Carmen de Valdenebro, Victoria Juspian, Florinda Arroyo por el amor con el que nos han formado, porque siempre estuvieron dispuestas a darnos su apoyo y fortaleza en los momentos que las necesitamos, de igual manera a nuestros padres Rodolfo García, Adolfo Hormiga y Eliodoro Tulande, a nuestro hermanos, a Jaime por ayudar y apoyar constantemente a Aleyda en cada momento de este proceso y a Andrés Felipe.

De igual manera nuestro más sincero agradecimiento al Director de este proyecto, Dr. José Luís Arciniegas Herrera a quien Dios nos puso por guiador y forjador para que este trabajo fuese un éxito, y quien nos ha tenido paciencia desde el comienzo enseñándonos a hacer las cosas con un alto nivel de exigencia y excelencia; así como al codirector César Alberto Collazos Ordóñez.

Agradecemos también a las empresas IKERNELL e INPUT por su colaboración, compromiso, paciencia y amistad, en el proceso de aplicación del marco de referencia.

Un agradecimiento especial a nuestros docentes: Jorge Jair Moreno Chaustre, Carlos Alberto Ardila Albarracín, Carlos Alberto Cobos Lozada, Miguel Angel Niño Zambrano, Pablo Augusto Magé Imbachí, Juan Carlos Vidal, Cesar Alberto Collazos Ordóñez, Elizabeth Granados Pemberty, Luz Marina Sierra Martínez, Roberto Carlos Naranjo Cuervo, Martha Eliana Mendoza Becerra, Erwin Meza Vega, Siler Amador Donado, Ember Ubeimar Mártinez Flor y Wilson Libardo Pantoja Yepez; quienes desde el comienzo de la carrera y hasta el final no solo nos formaron como profesionales excelentes sino también como personas a través de su ejemplo, convirtiéndose en nuestros amigos antes que en docentes.

Así también queremos dar gracias a todo el personal tanto de la Facultad de Ingeniería Electrónica y Telecomunicaciones, como de Secretaría General, Dirección, Recepción, Post-Grado, Administración, TIC's, Biblioteca, Mantenimiento, Limpieza y Fotocopias, ya que dentro de los ámbitos que a cada uno le competen nos han colaborado sin ponernos ningún impedimento, al contrario, nos han brindado siempre una sonrisa.

A nuestros compañeros de carrera: Darío Fernando Gómez, Adelinda Pepicano, Amparo Erazo y a los ingenieros Jimena Timaná, Rene Valencia, Yuly Botina, por su amistad, apoyo y ánimo en cada etapa que pasamos a lo largo de estos años de estudio.

En general queremos agradecer a todas y cada una de las personas que han vivido con nosotras la realización de este proyecto de grado, con sus altos y bajos y que no necesitamos nombrar porque tanto ellas como nosotras sabemos que desde los más profundo de nuestros corazones les agradecemos el habernos brindado todo el apoyo, colaboración, ánimo y sobre todo cariño y amistad.

Autoras: Maria Verónica Fernández de Valdenebro, Maria Amparo Hormiga Juspian, Aleyda Tulande Arroyo.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

Capítulo 1 INTRODUCCIÓN

La industria de software representa una actividad económica de suma importancia para todos los países del mundo, entre ellos Colombia; ofrece múltiples fuentes de negocio y se perfila como una gran oportunidad para los países en vía de desarrollo. Sin embargo, en los países latinoamericanos la industria de software es incipiente e inmadura [1] esto conlleva a una falta de competitividad que a su vez dificulta su crecimiento.

En Colombia la industria del software tiene una serie de problemas y entre ellos se destaca el desconocimiento de la importancia de la calidad tanto en los procesos productivos de software como en los productos finales [2]. Dicho desconocimiento lleva a que los procesos de desarrollo se hagan de forma artesanal y caótica teniendo como resultado tiempos de desarrollo inadecuados, costos por debajo de un valor competitivo, productos software de baja calidad, tiempos y costos excesivos en soporte y mantenimiento de los productos y por tanto baja competitividad frente a los mercados internacionales además de la insatisfacción por parte de usuarios finales y clientes [3].

Aunque la industria del software tiene una desventaja competitiva en Colombia [2]. ésta crece cada día más; por esta razón se hace imperante la necesidad de formular propuestas que encaminen a Colombia hacia la competitividad con los países con gran desarrollo en la industria informática. Una propuesta estratégica es desarrollar productos de calidad.

"Una de las características principales de la industria de software de Colombia, es estar compuesta por micro, pequeñas y medianas empresas – MiPymes. Estas empresas de software tienen serios problemas de calidad en el proceso de desarrollo de sus productos, en muchos casos no existe un proceso real conduciendo a modelos caóticos de operación que afecta a la empresa" [3] y por ende la satisfacción del cliente o usuario final. Al observar las MiPymes del departamento del Cauca es notorio que no son la excepción a este problema, debido a que la mayoría de los emprendimientos nacen como consecuencia de un producto terminado para el cual generalmente no se tuvo en cuenta una metodología para el proceso de desarrollo; además, este tipo de empresas desconocen los procesos de definición de una arquitectura de software y solo tienen en cuenta al usuario final en el momento de levantar requerimientos y de implantar el producto, lo que lleva a productos poco usables.

Pressman [4] dice que los requisitos del software son la base de las medidas de la calidad. La falta de concordancia con los requisitos es una falta de calidad. Estos se ven altamente influenciados por la arquitectura del sistema, la calidad del sistema debe ser considerada en todas las partes del diseño, pero los atributos de calidad se manifiestan de manera distinta a lo largo de estas fases; de esta forma se establece que la arquitectura determina ciertos atributos de calidad del sistema.

En el campo de arquitecturas software un tema que atrae cada vez más interés es la relación directa que existe entre las decisiones arquitectónicas y la satisfacción de ciertos atributos de calidad [5]. Un atributo de calidad que está adquiriendo cada vez más importancia es la usabilidad[1] del software [6]. Reconocidos investigadores ya han adelantado trabajos que involucran la relación entre arquitecturas de software y usabilidad. Por esta razón es interesante e innovador estudiar dicha relación en un dominio específico como es el caso de las aplicaciones Web desarrolladas por MiPymes;

[1] Usabilidad viene del ingles Usability, también conocida como capacidad de uso en el resto del documento se utilizará el termino "Usabilidad" para referirnos a "Usability".

teniendo en cuenta que las aplicaciones Web se están convirtiendo en un elemento clave, tanto en el desarrollo de las empresas como de las instituciones.

Por lo anterior, el proyecto MARCO DE REFERENCIA PARA LA MEJORA DE CARACTERISTICAS DE USABILIDAD, EN EL DESARROLLO DE APLICACIONES Web CONSTRUIDAS POR MiPymes, CENTRADO EN LA ARQUITECTURA busca proporcionar inicialmente a las MiPymes de la industria de software en el departamento del Cauca un marco, aunque no se descarta la posibilidad de aplicarlo en otras MiPymes de características similares, que describa las actividades y los respectivos pasos que se deben seguir en un proceso de desarrollo centrado en la arquitectura, teniendo en cuenta la usabilidad desde las etapas tempranas del proceso de desarrollo, para generar aplicaciones Web que satisfagan las necesidades del usuario, y de esta forma aportar a la competitividad y al posicionamiento en mercados nacionales e internacionales de dicha industria.

Los objetivos perseguidos en este trabajo fueron:

Objetivo General

Proponer un marco de referencia centrado en la arquitectura de software para la mejora de características de usabilidad en el desarrollo de aplicaciones Web construidas por MiPymes.

Objetivos Específicos

- Conformar un conjunto de estrategias y elementos arquitectónicos, los cuales se deben tener en cuanta en etapas iníciales del proceso de ingeniería para construcción de aplicaciones Web por MiPymes, tomando aspectos de usabilidad.
- Conformar un conjunto de guías y pasos que indiquen cómo se deben tener en cuenta aspectos de usabilidad en las primeras etapas, que involucran a la arquitectura, del proceso de ingeniería para desarrollo de aplicaciones en la Web.
- Validar el marco de referencia generado en dos aplicaciones Web construidas por MiPymes del Departamento del Cauca.

Justificación

El trabajar la usabilidad desde etapas tempranas del proceso de desarrollo como lo son las fases que involucran la especificación de la arquitectura software; ya no en un contexto general como se ha realizado en otros estudios sino en un contexto especifico como el de las aplicaciones Web construidas por MiPymes, contribuirá a que estas empresas en general tengan prácticas adecuadas, desarrollando una mejor arquitectura de software y generando aplicaciones Web que satisfagan las necesidades del usuario. Aportando de esta forma a la competitividad y al posicionamiento en mercados nacionales e internacionales de dicha industria.

Además este proyecto pretende aportar a las investigaciones que se adelantan en el campo de la competitividad y productividad de las MiPymes en el sur-occidente colombiano ayudando de esta manera al fortalecimiento de la relación academia-empresa y a contribuir a los trabajos relacionados con la usabilidad desde la arquitectura de software que se trabajan a nivel mundial.

Es preciso aclarar que el modelo obtenido en el marco de ésta iniciativa, se valido en el desarrollo de dos productos realizados por dos MiPymes del departamento del Cauca; sin embargo, su diseño y desarrollo estará proyectado para ser usado en cualquier tipo aplicación Web (sin que se pretenda en este proyecto y por limitaciones de tiempo,

demostrar su uso con otros productos). Igualmente, para su divulgación se realizó un artículo con los aspectos más relevantes del proyecto.

Para la realización y cumplimiento del marco de referencia se han realizado varias actividades las cuales han quedado plasmadas a lo largo de este documento. La monografía se encuentra organizada de la siguiente manera:

Primero se tiene el capítulo 2 donde se recopila la base conceptual necesaria para la elaboración del proyecto, los conceptos que se tuvieron en consideración fueron los siguientes: Aplicaciones Web, Diseño y Desarrollo de Aplicaciones Web, Atributos en las Aplicaciones Web, Metodologías para el Desarrollo de Aplicaciones Web, Arquitecturas de Software, Descripción de Arquitecturas de Software, Diseño de Arquitecturas de Software, Arquitecturas en Aplicaciones Web, Arquitecturas de Software y Aspectos de Calidad, Patrones, Patrones Arquitectónicos, Evaluación de Arquitecturas, Usabilidad, Atributos Generales de la Usabilidad, Beneficios de la Usabilidad, Relación entre la Usabilidad y Arquitecturas de Software.

El capítulo 3 Contiene un análisis de los patrones: Arquitectónicos, Hipermedia y de Iteración que afectan de alguna manera la usabilidad de las aplicaciones desde el momento en el que se define la arquitectura. En dicho análisis se presenta una reclasificación de estos patrones, de acuerdo a los tres elementos que tiene una aplicación Web como lo son: presentación, lógica y datos. por cada patrón se presenta un análisis del impacto que tienen sobre la usabilidad.

En el capítulo 4 se detallan las etapas del proceso de ingeniería para el desarrollo de aplicaciones Web, por cada etapa se presentan las Actividades y tareas necesarias para llevar a cabo el proceso, así como también se detallan las entradas y las salidas en cada actividad y las recomendaciones de cómo realizar las tareas. Además se recomienda en qué momento se pueden utilizar los patrones descritos en el capítulo 3 de acuerdo a la actividad correspondiente.

El capítulo 5 contiene la descripción de los resultados obtenidos al aplicar el marco de referencia en dos aplicaciones Web desarrolladas por: las empresas IKERNELL Aplicaciones Software e INPUT TECNOLOGIES LTDA las cuales son MiPymes de desarrollo de software y pertenecen a la incubadora de empresas de software ParqueSoft Popayán, dichas empresas tienen características particulares de las MiPymes como: Un reducido número de integrantes (entre 2 y 7), procesos de desarrollo no definidos, prácticas de desarrollo poco rentables y desorganizadas, poco presupuesto, talento humano que se caracteriza por la dedicación, conocimientos sólidos en lenguajes de desarrollo, utilización de herramientas libres para el desarrollo de los productos, de uno a dos productos desarrollados o en proceso de desarrollo, entre otras.

El capítulo 6 expone las conclusiones a las que se llegó después de la culminación del proyecto de grado, así como, las ideas para trabajos posteriores relacionados con la continuidad del proyecto.

Por último el capítulo 7, contiene la información adicional que ayuda a precisar el contenido del documento final. Teniéndose los siguiente anexos: Del Anexo 1-5: Contiene la descripción de los patrones que de alguna manera afectan la usabilidad de una aplicación Web desde la arquitectura como por ejemplo los patrones de: usabilidad, arquitectónicos, de presentación Web, sistemas hipermedia, de diseño navegacional, de interfaz, de interacción, del Anexo 6 al 14. Contiene las plantillas para documentar el proceso de desarrollo descrito en el capítulo 4.

Contribuciones

Dentro de las contribuciones más relevantes del proyecto podemos resaltar:

- El presente trabajo de grado aporta un valor significativo a la base del conocimiento de los temas y diferentes trabajos que desarrollan, existen o se relacionan con arquitecturas de software, usabilidad, desarrollo de aplicaciones Web y MiPymes.
- Marco de referencia para la mejora de características de usabilidad, en el desarrollo de aplicaciones Web construidas por MiPymes, centrado en la arquitectura: Descripción y análisis de los patrones que influyen en la usabilidad desde el momento de la arquitectura al construir aplicaciones Web.
- Proceso de ingeniería para desarrollo de aplicaciones en la Web, teniendo en cuenta aspectos de usabilidad: Descripción del proceso que se debe seguir en las etapas de Requerimientos, Análisis y Diseño para construir aplicaciones Web teniendo en cuenta aspectos de usabilidad y patrones.
- Como caso de estudio se aplicó el marco de referencia en dos MiPymes del departamento del Cauca, corroborando de esta manera la validez práctica del proyecto y obteniendo una realimentación sobre la base teórica.

Estas contribuciones quedan registradas en esta monografía, sus anexos, articulo(s), CDs, la experiencia recibida y realimentada de las empresas (anexando cartas donde se certifica su colaboración en los procesos).

Capítulo 2 MARCO TEORICO

En este capítulo se especificarán aspectos básicos que se tienen en cuenta en el desarrollo del Marco de referencia centrado en la arquitectura para la mejora de características de usabilidad en el desarrollo de aplicaciones Web construidas por MiPymes; se da definiciones acerca de aplicaciones Web, arquitectura software, arquitectura en aplicaciones Web, arquitectura software y aspectos de calidad y usabilidad. Así, en la sección 1 se dedica a las aplicaciones Web; en la sección 2 se abarca lo referente a la arquitectura de software; en la sección 3 se describen las arquitecturas de las aplicaciones Web; en la sección 4 la arquitectura de software y aspectos de calidad, se hace una relación entre la arquitectura de software y la calidad de software y en la sección 5 se trata el tema de Usabilidad.

2.1 Aplicaciones Web

En los últimos años Internet ha tenido un crecimiento por el acceso de usuarios, debido a esto han aparecido nuevas tecnologías asociadas a la Web facilitando que los sistemas de información se universalicen. Además empiezan a utilizarse nuevos términos que tiene que ver con aplicaciones informáticas especialmente modeladas y diseñadas para ser ejecutadas en la Web. Uno de los términos es el de aplicaciones Web; debido a esto en esta sección se conceptualizará algunos términos para establecer relaciones entre lo que es una aplicación Web, página Web, sitio Web y portal.

Página Web: Es un documento electrónico que contiene información específica de un tema en particular y es almacenado en un sistema de cómputo que se encuentra conectado a Internet, de tal forma que este documento pueda ser consultado por cualquier persona que se conecte a la red y que cuente con los permisos apropiados para hacerlo. En una página se combina texto con imágenes para que el documento sea dinámico y permita que se puedan ejecutar diferentes acciones, a través del hipertexto o de las imágenes [7].

Sitio Web: Es un conjunto de archivos electrónicos y páginas Web referentes a un tema en particular, que incluye una página inicial de bienvenida, generalmente denominada home page, con un nombre de dominio y dirección en Internet específicos [7]. Empleados por las instituciones públicas y privadas, organizaciones e individuos para comunicarse con el mundo entero.

Portal: El concepto fue divulgado a partir de 1996 y se refiere a lugares dónde era posible encontrar una oferta de servicios de valor añadido, tales como correo electrónico, noticias, información, etc. [8]. Históricamente, el término “portal” se diferenciaba del de “sitio Web” en que se refería a un lugar de partida hacia otros sitios en Internet. No obstante hoy en día estos términos se utilizan indistintamente, haciéndose un uso preferente de la denominación portal, cuando el sitio Web integra diferentes servicios de valor añadido y no sólo páginas estáticas [7].

Aplicación Web: Según Conallen [8] “Una aplicación Web está inmersa en un sitio Web donde la navegación a través del sitio, y la entrada de datos por parte de un usuario, afectan el estado de la lógica del negocio. En esencia, una aplicación Web usa un sitio Web como entrada (front-end) a una aplicación típica. Si no existe lógica del negocio en el servidor, el sistema no puede ser llamado aplicación Web.”.

Las aplicaciones Web son sistemas con interfaces de usuario cada vez más parecidas a las aplicaciones de escritorio, dando servicio a procesos de negocio de considerable

tamaño y estableciéndose sobre ellas requisitos estrictos de accesibilidad y respuesta. Esto ha dado lugar a reflexiones acerca de la mejor arquitectura y de las técnicas de diseño más adecuadas [9].

Por lo tanto una aplicación Web es aquella que los usuarios usan accediendo a un servidor Web a través de Internet o de una intranet mediante un navegador Web que opera como cliente ligero. Se caracteriza por la posibilidad de actualizar y mantener aplicaciones Web sin distribuir e instalar software en miles de potenciales clientes.

En los últimos años, la rápida expansión de Internet y del uso de intranets corporativas ha supuesto una transformación en las necesidades de información de las organizaciones. En particular esto apunta a la necesidad de [9]:

1. La información sea accesible desde cualquier lugar dentro o fuera de la organización.
2. Esta información sea compartida entre todas las partes interesadas, de manera que ellas tengan acceso a la información completa (o a aquella parte que les corresponda según su función) en cada momento.

Esta información puede ser accesible o compartida teniendo en cuenta la seguridad y la información pública y privada.

Estas necesidades han provocado una evolución de las aplicaciones tradicionales de escritorio hacia las aplicaciones Web, que por su idiosincrasia, cumplen a la perfección con ellas. Por tanto, los sitios Web tradicionales que se limitaban a mostrar información se han convertido en aplicaciones capaces de una interacción más o menos sofisticada con el usuario. Inevitablemente, esto ha provocado un aumento progresivo de la complejidad de estos sistemas y, por ende, la necesidad de buscar opciones de diseño nuevas que permitan dar con la arquitectura óptima que facilite la construcción de los mismos.

Según los conceptos de sitios y aplicaciones Web es diferente la implementación de un sitio Web con el desarrollo de una aplicación Web. La aplicación Web es mucho más dinámica, dispone de una lógica de negocio que puede reaccionar y alterar su estado a partir de la interacción con un usuario. Su contrapartida es la complejidad, ya que requiere implementar una arquitectura que se adapte a los cambios constantes, que facilite su ágil integración con otros sistemas y que resuelva picos variables de interacción con un buen rendimiento.

2.1.1 ¿Qué es el diseño y desarrollo de aplicaciones Web?

El diseño y desarrollo de aplicaciones Web consiste en solucionar las necesidades, los objetivos o las ideas de los clientes en Internet utilizando las tecnologías de intranet o extranet según el proyecto.

Las aplicaciones Web pueden ser de acceso público como tiendas virtuales, diarios digitales, portales de Internet, etc. O de acceso restringido como son las intranets para mejorar las gestiones internas de una empresa como el reporte de horas del personal, gestión de proyectos y tareas, control de presencia, gestores documentales, etc., o el uso de extranets para aumentar y mejorar el servicio con los distribuidores, clientes, proveedores, comerciales y colaboradores externos.

Por otro lado el tipo de aplicación Web que más se acerca al concepto es aquella que actualiza el servidor dinámicamente mediante la capa de datos pudiendo modificarla, sus posibilidades son mayores ya que además de incluir las características del tipo anterior está pensada para ofrecer un mayor número de servicios debido a la gran diversidad de funcionalidad que se puede alcanzar (comercio electrónico, generación de contenidos, administración de usuarios, etc.).

2.1.2 Atributos de aplicaciones Web

Los atributos siguientes se pueden encontrar en la gran mayoría de las aplicaciones Web [4]:

- **Intensiva en el uso de la Red:** Por su propia naturaleza, una aplicación Web es intensiva en el uso de la red. Reside en una red y debe dar servicio a las necesidades de una comunidad diversa de clientes. Una aplicación Web puede residir en Internet (haciendo posible así una comunicación abierta para todo el mundo). De forma alternativa, una aplicación se puede ubicar en una Intranet (implementando la comunicación a través de redes de una organización) o una Extranet (comunicación entre redes) la aplicación Web es intensiva de red, controladas por el contenido y en continua evolución. Estos atributos tienen un profundo impacto dentro de la forma en que se lleva a cabo la Ingeniería Web.
- **Controlada por el contenido**: En muchos casos, la función primaria de una aplicación Web es utilizar hipermedia para presentar al usuario el contenido de textos, gráficos, sonido y vídeo.
- **Evolución rápida:** A diferencia del software de aplicaciones convencional, que evoluciona con una serie de versiones planificadas y cronológicamente espaciadas, las aplicaciones Web están en constante evolución. No es inusual que algunas aplicaciones Web (específicamente, su contenido) se actualicen cada hora.

Un cuidado y una alimentación continua permiten que una aplicación Web crezca (en robustez y en importancia).

Las siguientes características son las que conducen a que una aplicación Web crezca en robustez y en importancia, permitiendo adaptarse a las diferentes necesidades del cliente [4]:

- **Inmediatez**: Las aplicaciones basadas en Web tienen una inmediatez que no se encuentra en otros tipos de software. Es decir, el tiempo que se tarda en comercializar una aplicación Web completa puede ser cuestión de días o semanas. Los desarrolladores deberán utilizar los métodos de planificación, análisis, diseño, implementación y comprobación que se hayan adaptado a planificaciones apretadas en tiempo para el desarrollo de aplicaciones Web.
- **Seguridad:** Dado que las aplicaciones Web están disponibles a través del acceso por red, es difícil, si no imposible, limitar la población de usuarios finales que pueden acceder a la aplicación. Con objeto de proteger el contenido confidencial y de proporcionar formas seguras de transmisión de datos, deberán implementarse fuertes medidas de seguridad en toda la infraestructura que apoya una aplicación Web y dentro de la misma aplicación.
- **Estética:** Una parte innegable del atractivo de una aplicación Web es su apariencia e interacción. Cuando se ha diseñado una aplicación con el fin de comercializarse o vender productos o ideas, la estética puede tener mucho que ver con el éxito del diseño técnico.

Estas características tienen diferente grado de influencia en todas las aplicaciones Web, dependiendo del dominio de la aplicación.

2.1.3 Metodologías para el desarrollo de aplicaciones Web

Existen metodologías para el desarrollo de aplicaciones Web referenciadas y aceptadas como: **UP**[2] (El Proceso Unificado) [10] el ciclo de vida de esta metodología abarca las etapas de inicio, elaboración, construcción y transición, que concluyen en un hito principal cada una. Se tomó la primera fase de esta metodología para el marco porque

[2] Unified Process

tiene en cuenta los requisitos y análisis del sistema a desarrollar. **WSDM**[3] [11] (método para el diseño de sitios Web) es una propuesta para el desarrollo de sitios Web, en la que el sistema se define en base a los grupos de usuarios. Su proceso de desarrollo se divide en cuatro fases: Modelo de usuario, diseño conceptual, implementación y su diseño. La fase que se tuvo en cuenta para el trabajo del marco es la primera en la que detecta los perfiles de usuarios para los cuales se construye la aplicación. **HFPM**[4] [12] (modelado del proceso flexible de hipermedia) describe un proceso detallado que cubre todo el ciclo de vida de un proyecto software. HFPM propone un total de trece fases para las cuales se especifican a su vez una serie de tareas, esta propuesta es un complemento para la metodología OOHDM. **OOHDM**[5] [13] (modelo de diseño hipermedia orientado a objetos) de todas las propuestas de desarrollo Web que han surgido en los últimos años, es la más aceptada y la que más ha sido estudiada. Las tres fases de esta propuesta son: la *realización del modelo de clases conceptuales, definición del modelo de navegación y realización del modelo de interfaz abstracta.* La primera fase permite definir un diagrama de clases mediante el cual se representa la estructura estática del sistema. La segunda fase establece una vista del modelo conceptual y expresa cómo navegar a través de la información representada en este modelo. En la última fase se desarrolla una vista de cómo se va a presentar la información al usuario. Para el marco de referencia se tuvo en cuenta las dos primeras fases, ya que se enfoca en el diseño de una aplicación Web. **UWE**[6] [14] (ingeniería Web basada en UML) es una propuesta metodológica basada en el Proceso Unificado [10], [15] y UML para el desarrollo de aplicaciones Web. Cubre todo el ciclo de vida de este tipo de aplicaciones centrando además su atención en aplicaciones personalizadas o adaptativas. Su proceso de desarrollo se basa en tres fases principales: la fase de captura de requisitos, la fase de análisis y diseño y la fase de implementación; se tuvo en cuenta para el marco de referencia las dos primeras fases. **NDT**[7] [16], [17] (técnicas de desarrollo navegacional) es una técnica para especificar, analizar y diseñar el aspecto de la navegación en aplicaciones Web. Para el marco de referencia, se tomo la definición y captura de requisitos. El flujo de especificación de requisitos de NDT comienza con la fase de captura de requisitos y estudio del entorno. Tras esta fase, se propone la definición de los objetivos del sistema. Con base a ellos define los requisitos que el sistema debe cumplir para cubrir los objetivos marcados.

2.2 Arquitectura de Software

2.2.1 ¿Qué es la arquitectura de software?

Hoy en día no se encuentra una definición unánime para el termino Arquitectura de software; sin embargo, la manera más fácil de entenderlo es haciendo una analogía con la arquitectura referente al área de construcción de edificaciones habitables.

La mayoría de personas han visto la construcción de una edificación y saben que para llevarla a cabo son necesarios los planos arquitectónicos, pues sería ilógico empezar a construir sin tener la idea aterrizada en un plano, en esta secuencia lógica de construcción primero se construye el esqueleto y luego se ensamblan las partes. En la construcción de software sucede algo similar aunque a diferencia de la construcción de una edificación, el software no se rige por leyes físicas ni por procedimientos conocidos, sino que es correspondientemente específico y experimental [18]; sin embargo, sí cuenta con un esqueleto estático y dinámico del sistema el cual es la arquitectura de software.

[3] Web Site Design Method
[4] Hipermedia Flexible Process Modelling Strategy
[5] *Object Oriented Hipermedia Design Methodology*
[6] UML Based Web Engineering
[7] Navegational Development Techniques

Entonces ¿existe una definición exacta de Arquitectura de Software? La respuesta es sí, pero cada autor la define de forma diferente. Entre las definiciones más importantes encontramos las siguientes:

La Arquitectura de Software es, "a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del producto; la vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión del detalle inherente a la mayor parte de las abstracciones" [19].

La definición "oficial" de Arquitectura de Software concertadamente la que brinda el documento de IEEE Standards Description 1471-2000, adoptada también por Microsoft, que dice así: "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" [20].

En el presente proyecto se ha tomado como base la anterior definición, además de otra definición que se considera de gran importancia en esta investigación, debido a que esta menciona las propiedades no funcionales, y que es complementaria a la anterior:

"Una arquitectura de Software es la descripción de los subsistemas y componentes de un sistema software y las relaciones entre ellos, típicamente representado mediante vistas que muestran las propiedades funcionales y no funcionales más relevantes" [21].

¿Por qué es importante la arquitectura?
Bass, Clements y Kazman [22] identifican tres razones claves por las cuales es importante la arquitectura de software, para este trabajo se ha considerado una cuarta, de esta forma:

1. **La comunicación:** Las representaciones de la arquitectura de software facilitan la comunicación entre los diferentes participantes[1] ya que ellos dependen de ella y se utiliza como una entrada para diseñar el proyecto ya ayudar a los gerentes a estimar esfuerzos y administrar los recursos.
2. **El diseño:** La arquitectura destaca decisiones tempranas de diseño que tendrán un profundo impacto en todo el ciclo de vida del software además contribuye para el diseño de nuevos sistemas similares.
3. **La evaluación:** La arquitectura constituye un modelo relativamente pequeño e intelectualmente comprensible de cómo está estructurado el sistema y de cómo trabajan juntos sus componentes esto permite la valoración temprana de los atributos de calidad [21], [23].
4. **Calidad**: El desarrollo de formas sistemáticas para relacionar atributos de calidad de un sistema a su arquitectura provee una base para la toma de decisiones objetivas sobre acuerdos de diseño y permite a los ingenieros realizar predicciones razonablemente exactas sobre los atributos del sistema; recientemente ha sido demostrado que usabilidad tiene implicaciones mas allá de la interfaz del usuario afectando por ejemplo a componentes de la arquitectura software [24].

2.2.2 Descripción de arquitectura de software

Existen varias formas para describir arquitecturas como: la Técnica por Código Fuente, la técnica de Caja y Diagramas de Línea y la técnica por Modelos de Vistas; esta última cumple fielmente con las razones claves [25].

[1] Los participantes son las personas (como clientes, usuarios finales, diseñadores, etc.) que se ven afectados o tiene un interés en el resultado de desarrollo.

Modelos de vista

Una buena manera de describir la arquitectura es desde vistas diferentes. Cada vista tiene su propio método de descripción y estilos, direccionan intereses específicos de cada participante.

La mayoría de los frameworks[2] y estrategias reconocen entre tres y seis vistas, que son las que se muestran a continuación:

Tabla 1: Vistas en los marcos de referencia [26]

Zachman (Niveles)	TOGAF (Arquitecturas)	4+1 (Vistas)	[BRJ99] (Vistas)	POSA (Vistas)	Microsoft (Vistas)
Scope	Negocios	Lógica	Diseño	Lógica	Lógica
Empresa	Datos	Proceso	Proceso	Proceso	Conceptual
Sistema lógico	Aplicación	Física	Implementación	Física	Física
Tecnología	Tecnología	Desarrollo	Despliegue	Desarrollo	
Representación		Escenarios	Escenarios		
Funcionamiento					

Es de especial interés para esta investigación los siguientes modelos:

Modelo “4+1” de Kruchten

- En 1995 Philippe Kruchten propuso su célebre modelo “4+1”, vinculado al Rational Unified Process (RUP), define cuatro vistas diferentes de la arquitectura de software: (1) La vista lógica, que comprende las abstracciones fundamentales del sistema a partir del dominio de problemas. (2) La vista de proceso: el conjunto de procesos de ejecución independiente a partir de las abstracciones anteriores. (3) La vista física: un mapeado[3] del software sobre el hardware. (4) La vista de desarrollo: la organización estática de módulos en el entorno de desarrollo. El quinto elemento considera todos los anteriores en el contexto de escenarios [24].

Vistas en UML

- En su introducción a UML (1.3), Grady Booch, James Rumbaugh e Ivar Jacobson han formulado un esquema de cinco vistas interrelacionadas que conforman la arquitectura de software: (1) La vista de escenarios, como la perciben los usuarios, analistas y encargados de las pruebas; (2) la vista de diseño que comprende las clases, interfaces y colaboraciones que forman el vocabulario del problema y su solución; (3) la vista de procesos compuesta por los hilos y procesos que forman los mecanismos de sincronización y concurrencia; (4) la vista de implementación que incluye los componentes y archivos sobre el sistema físico; (5) la vista de despliegue que comprende los nodos que forman la topología de hardware sobre la que se ejecuta el sistema [27].

[2] En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

[3] Mapeado viene de ingles mapping que significa traducción, en el resto del documento se utilizará el termino “Mapeado” para referirnos a “Mapping”.

El problema con los modelos de vista es que a menudo es difícil mantener las vistas estables. Por ejemplo, en la 4+1 vista un cambio en la vista de desarrollo afecta probablemente algo en la vista lógica (aunque no hay un mapeado uno a uno entre esas vistas). Hay algunas superposiciones y diferencias entre las vistas de UML y las 4+1 vistas pero en general estos modelos de vista son bastante similares. Los modelos de vista proporcionan una descripción más completa de una arquitectura que las cajas y diagramas de línea y cumplen de la siguiente manera las razones claves:

1 **La comunicación**: los modelos de vista pueden describir los detalles como las interfaces, la guía (por medio de los casos de uso o escenarios), razón, etc. Un modelo multi-vistas ayuda a la comunicación; ya que al mostrar los intereses particulares de cada participante se minimiza la confusión.
2 **El diseño**: los modelos de vista proporcionan la semántica y capturan la razón de una arquitectura. Las decisiones del diseño capturadas en los modelos de vistas pueden transferirse a futuros sistemas.
3 **La evaluación**: Los modelos de vista proporcionan un cuadro más completo de la arquitectura suficiente para el análisis por un especialista. Ciertamente las vistas son descritas usando lenguajes de notación [28] lo que hace que se pueda realizar una interpretación y un análisis automático.
4 **La calidad**: Las vistas arquitectónicas, a través de distintos niveles de abstracción, resaltan diversos aspectos que conciernen a los participantes lo cual contribuye a la búsqueda de la mejor solución para un sistema específico. Estas perspectivas de una arquitectura ayudan a todos los involucrados en el proceso de desarrollo a razonar sobre los atributos de calidad del sistema.

Pueden usarse varias técnicas para describir arquitecturas del software. La opción para una técnica particular depende de muchos factores como la organización, la aplicación, el dominio, los objetivos del diseño, etc.

2.2.3 Diseño de arquitectura de software

La arquitectura del software es el resultado de un juego de técnicas, negocios y actividades entre participantes (clientes, equipo de desarrollo, etc.) con el objetivo común de desarrollar software que proporcione una funcionalidad específica teniendo un equilibrio en el cumplimiento de requerimientos [25].

La arquitectura del software es el primer producto de las actividades iníciales del diseño que permite el análisis y discusión acerca de diferentes inquietudes. El diseño de software se desarrolla alrededor de cuatro "aspectos de negociación": la evolución de las características, la calidad, el costo y el tiempo de entrega [25].

Estas arquitecturas deben cumplir con los siguientes aspectos [29]:

Escalabilidad
Es la característica principal de una aplicación Web, debido a que hace posible el rápido incremento del número de usuarios. Es importante tenerla en cuenta para: el correcto dimensionamiento de la aplicación, la adaptabilidad del sistema ante el incremento de demanda. Los tipos de escalabilidad que se presentan son: Escalabilidad Horizontal, Escalabilidad Vertical y Clúster de servidores.

- Escalabilidad Horizontal: se distribuye el volumen de la información y la carga de trabajo entre múltiples máquinas. Distribuida por algoritmos de balanceo predeterminados (ej.: Round Robin) las peticiones HTTP entre los distintos clones del sistema, seleccionados aleatoriamente.
- Escalabilidad Vertical: La separación lógica entre capas se implementa de forma que permita la separación física de las mismas. Se necesita de un mediador entre las capas para la comunicación remota.

- Clúster de servidores: estos son habituales en los servidores de aplicaciones comerciales (Weblogic, WebSphere, iPlanet, etc.). Dependiendo de cómo se aplique puede clasificarse como horizontal o vertical. Distribuye y escala el sistema de modo transparente al usuario y al administrador.

Separación de Responsabilidades
Anteriormente se desarrollaban aplicaciones Web con el modelo estructural mainframe, la cual consistía en una única capa física y lógica, luego la arquitectura basada en transacción y con el paso de los años el diseño de la arquitectura fue especializándose para mejorar la escalabilidad y la facilidad de mantenimiento de las aplicaciones llegando a lo que hoy conocemos como arquitectura en capas, cliente- servidor, aplicación 3 capas, aplicación n capas.
La característica de separar responsabilidades se refiere a la separación de las capas y sus respectivas responsabilidades, la tendencia de los estilos arquitectónicos en las aplicaciones Web es el estilo en n-capas; actualmente, el número de capas más utilizado es el de 3 capas:

- Capa de presentación.
- Capa de negocio.
- Capa de persistencia

Tomando como base este modelo, distintos patrones arquitectónicos han ido apareciendo como evolución del mismo (Modelo Brown [30], los patrones de Sun [31], etc.), los cuales han adicionado más capas con el fin de conseguir una mayor independencia entre éstas.

Portabilidad
Una aplicación Web debe poder adaptarse a las distintas arquitecturas físicas posibles en su despliegue. Las tareas de adaptación a un nuevo entorno tienen que limitarse al ámbito de la configuración y no del desarrollo, por eso se dice que debe ser independiente del lenguaje y las plataformas de desarrollo. Para lograr esta característica es necesario tener en cuenta que los componentes deben ser independientes de los requisitos del negocio, además se debe tener en cuenta una capa adicional que es la de infraestructura.

2.3 Arquitecturas en Aplicaciones Web

A continuación se describen las diferentes clases de arquitecturas que se utilizan en el desarrollo de aplicaciones Web Cliente - Servidor. Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma [32]. “Como se deduce del término mismo, clientes y servidores son entidades lógicas independientes que operan en conjunto a través de una red para realizar una tarea” [33].

En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición) y este envía uno o varios mensajes con la respuesta (provee el servicio). En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras.

En otras palabras, la arquitectura Cliente/Servidor es una extensión de programación modular en la que la base fundamental es separar una gran pieza de software en módulos con el fin de hacer más fácil el desarrollo y mejorar su mantenimiento.

2.3.1 Cliente

El cliente es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor, se le conoce con el término front-end [34].

El Cliente normalmente maneja todas las funciones relacionadas con la manipulación y visualización de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces Figuras de usuario (GUI), además de acceder a los servicios distribuidos en cualquier parte de una red.
Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Generar requerimientos de bases de datos.
- Recibir resultados del servidor.
- Formatear resultados.

2.3.2 Servidor

Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se le conoce con el término back-end [34].
El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos.
Las funciones que lleva a cabo el proceso servidor se resumen en los siguientes puntos:

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

2.3.2.1 Características de la arquitectura Cliente/Servidor

Las características básicas de una arquitectura Cliente/Servidor son [34]:

- Combinación de un cliente que interactúa con el usuario y un servidor que interactúa con los recursos compartidos. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de software que maneja recursos compartidos tales como bases de datos, impresoras, módems, etc.
- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, velocidad y capacidades del disco y dispositivos de entrada y salida.
- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.
- Existe una clara distinción de funciones basada en el concepto de "servicio", que se establece entre clientes y servidores.
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.
- Los clientes corresponden a procesos activos en cuanto a que son éstos los que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes.
- No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.
- El ambiente es heterogéneo. La plataforma de hardware y el sistema operativo del cliente y del servidor no son siempre la misma. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.
- El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema Cliente/Servidor. La escalabilidad horizontal permite agregar más

estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores.

2.3.3 Arquitectura Orientada a Servicios (SOA)

(En inglés *Service-Oriented Architecture* o SOA),existen otras definiciones de SOA [35], [36] pero se tuvo en cuenta para este marco de referencia la de Hao He:

Según Hao He [37] es un estilo arquitectónico apto para implementar bajo acoplamiento entre agentes. Los agentes son proveedores y consumidores de servicios, que son la unidad de trabajo.

SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

En un ambiente SOA, los nodos de la red hacen disponibles los recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. La mayoría de las definiciones de SOA identifican la utilización de Servicios Web (empleando SOAP, WSDL, XML, HTTP, UDDI, etc.) en su implementación. Pero SOA no está atado a una tecnología específica. Puede también ser implementado utilizando un amplio rango de tecnologías, incluyendo REST, RPC, DCOM, CORBA, Enterprise JAVA BEANS, etc.

Al contrario de las arquitecturas orientado a objetos, las SOAs están formadas por servicios de aplicación débilmente acoplados y altamente interoperables. Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación. La definición de la interfaz encapsula (oculta) las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo. Con esta arquitectura, se pretende que los componentes software desarrollados sean muy reusables, ya que la interfaz se define siguiendo un estándar [38].

2.3.3.1 Diseño y desarrollo de SOA

La metodología de modelado y diseño para aplicaciones SOA se conoce como análisis y diseño orientado a servicios SOAD. Es una metodología de diseño para desarrollo de sistemas muy ágiles en un modelo cliente/productor donde se abstrae la implementación del proceso actual, de manera que el servicio proveído puede ser modificado o cambiado sin afectar al cliente. SOA es tanto un marco de trabajo para el desarrollo de software como uno de implantación. Para que un proyecto SOA tenga éxito los desarrolladores de software deben mentalizarse para crear servicios comunes que son fomentados por clientes para implementar los procesos de negocio. El desarrollo de sistemas usando SOA requiere un compromiso con este modelo en términos de planificación, herramientas e infraestructura [39].

Un servicio debe ser una aplicación completamente autónoma e independiente. A pesar de esto, no es una isla, porque expone una interfaz de llamado basada en mensajes, capaz de ser accedida a través de la red. Generalmente, los servicios incluyen tanto lógica de negocios como manejo de estado (datos), relevantes a la solución del problema para el cual fueron diseñados. La manipulación del estado es gobernada por las reglas de negocio [40].

SOA se fundamenta en:

- Ejecutarse rápido, adaptarse al mercado, ganar ante la competencia.
- Reutilizar los componentes de los procesos de negocios.
- Medir los resultados y tomar acción sobre ellos.
- Garantizar resultados que sean repetibles y predecibles.
- Empezar donde sea necesario (área de negocios - área de tecnología).

2.3.4 Arquitectura P2P

Se refiere a una red que no tiene clientes ni servidores fijos, sino una serie de nodos que se comportan simultáneamente como clientes y servidores de los demás nodos de la red. Este modelo de red contrasta con el modelo cliente-servidor básico el cual se rige de una arquitectura monolítica donde no hay distribución de tareas entre sí, solo una simple comunicación entre un usuario y una terminal en donde el cliente y el servidor no pueden cambiar de roles [20].

Este aprovechamiento incrementa la cantidad de valores que cada nodo en la red puede añadir, porque no sólo toma información de un solo recurso, sino que también tiene la habilidad de distribuir información con otros recursos.
Una aplicación típica P2P tiene las siguientes características claves que ayudan a definirla:

- Descubrir otro compañero: La aplicación debe ser capaz de encontrar otras aplicaciones que están dispuestas a distribuir información. Históricamente, la aplicación encuentra esos compañeros de acuerdo a los registros de un servidor central que mantiene una lista de todas las aplicaciones que frecuentemente se disponen a distribuir, dando esa lista a cualquier nueva aplicación, a medida que ellas se conectan a la red. Sin embargo hay otras, formas posibles, por ejemplo redes de emisión – difusión, o algoritmos de descubrimiento.
- Consultando compañeros por contenido: Una vez que esos compañeros son descubiertos, la aplicación puede preguntarles por el contenido que es deseado por la aplicación, o sea por ella.
- Distribuyendo contenidos con otros compañeros: Los compañeros pueden consultar a otros por su contenido, y una vez hecho esto pueden también distribuir el contenido descubierto.

2.3.4.1 Propiedades de las redes P2P

Estas son algunas propiedades de las redes P2P [20] básicas:

- Los peers o nodos son autónomos y pueden actuar como clientes y servidores al mismo tiempo.
- Las conexiones y los propios peers no son estables. Las redes P2P están preparadas para no ser vulnerables respecto a la caída de nodos.
- Se organizan en redes virtuales (red overlay).
- No es necesaria una coordinación central ni una base de datos central.

Modelos de redes P2P: p2p puro, parcialmente centralizados o híbridos (donde existen nodos, los super-peers, que actúan de servidor para un conjunto de nodos)

- P2P puro
  Dos dispositivos (clientes o servidores) dialogan e interaccionan entre ellos sin ningún dispositivo central.
- P2P con dependencia parcial de un servidor central.
  Se utiliza un servidor central para obtener la configuración de un usuario (p.ej. Aplicación mensajería instantánea) o para realizar una búsqueda centralizada (la descarga del contenido se haría vía P2P)
- Sistemas híbridos, basados en Super-peers.
  Los super-peers actúan como servidores centrales para un conjunto o clúster de clientes.

Los super-peers se comunican entre ellos al mismo nivel.
En un sistema de archivos compartidos, los super-peers se utilizarían para realizar las búsquedas. La descarga de contenidos ser haría por conexión directa entre los peers.

- Los peers cooperan entre ellos para conseguir servicios (mayor escalabilidad, fiabilidad y rendimiento).

2.3.4.2 Beneficios de la comunicación P2P

El modo de trabajo de las empresas también se ha visto modificado ya que P2P permite reducir costos, brindando los siguientes beneficios:

- Costos más efectivos: Recursos de manejo centralizado reducido, Recursos de almacenamiento reducido, Recursos de red y computación optimizados.
- Eficiencia personal
- Habilidad y adaptabilidad.

2.4 Arquitectura de Software y Aspectos de Calidad

Hay una relación fuerte entre la arquitectura del software y la calidad del software debido a:

1. La calidad se define a partir del diseño de la arquitectura [21]-[23], [27], ya que un buen diseño debe definir hasta qué punto es adecuado permitir cambios para incrementar la calidad del software.
2. La calidad debe lograrse a través del diseño de la arquitectura. las decisiones tempranas de diseño tienen una influencia considerable en la calidad de un sistema. Ciertas decisiones de diseño de arquitectura han demostrado impactar significativamente en algunas características de calidad; por ejemplo, un estilo en capas [21] mejora la modificación pero afecta negativamente la eficacia. Esto muestra que las características de calidad no son independientes; una decisión de diseño que afecta positivamente una característica puede impactar negativamente otra. Algunas características presentan conflictos entre si; por ejemplo ciertas decisiones de diseño que mejoran la modificación afectan negativamente el desempeño lo mismo puede ocurrir con la seguridad y la usabilidad [25]. Es deber de un arquitecto establecer prioridades entre las características de calidad que sean acordes a las necesidades del sistema, ya que cambiar las decisiones con respecto a dichas características resulta muy costoso en fases posteriores del ciclo de vida.

2.4.1 Arquitectura y Características de Calidad de los Sistemas

La arquitectura es un artefacto que determina algunas características de calidad del sistema y por consiguiente cada decisión incorporada en una arquitectura de software puede afectar potencialmente la calidad del software. Sin embargo, esto no es evidente, porque [22]:

- Existen características de calidad como: confiabilidad, disponibilidad y funcionamiento que, luego de ser estudiadas durante años, poseen definiciones generalmente aceptadas. En cambio otras características como modificación, seguridad y usabilidad no las tienen.
- Las características no están aisladas ni son independientes entre sí.
- El análisis de características de calidad no se presta para la estandarización de éstas.
- Las técnicas de análisis son específicas para una característica en particular.

A grandes rasgos, se establece una clasificación de las características de calidad de software en dos categorías [22]:

- Observables en ejecución: las que se determinan del comportamiento del sistema en tiempo de ejecución.

- No observables en ejecución: las que se establecen durante el desarrollo del sistema.

Las Características de calidad también se pueden clasificar de la siguiente manera [40]:

- De ejecución: funcionamiento, seguridad, eficiencia.
- Del ciclo de vida: Facilidad de comprobación, portabilidad, integrabilidad, capacidad de mantenimiento, despliegue, reusabilidad
- De uso: productividad, satisfacción, eficacia, seguridad.

El estándar ISO/IEC 9126 ha sido desarrollado en un intento de identificar las características de calidad para un producto de software [29]. Este estándar es una simplificación del Modelo de Losavio [41], e identifica seis características básicas de calidad que pueden estar presentes en cualquier producto de software. Estas son: Funcionalidad, confiabilidad, usabilidad, eficiencia, facilidad de mantenimiento y portabilidad.

Losavio propone una adaptación del modelo ISO/IEC 9126. El modelo se basa en las características de calidad que se relacionan directamente con la arquitectura: funcionalidad, confiabilidad, eficiencia, facilidad de mantenimiento y portabilidad.

Aunque en las características de calidad involucradas con la arquitectura de software planteadas por Losavio no se tiene en cuenta la usabilidad; Bass, Clements y Kazman, plantean que: La usabilidad involucra aspectos arquitectónicos y no arquitectónicos. Los aspectos no arquitectónicos incluyen una producción de interfaz de usuario clara y fácil de usar, algunas de las preguntas que se pueden hacer respecto a esto son: ¿Debe proporcionarse una caja de texto o una lista de selección? ¿Qué diseño de pantalla es más intuitivo? ¿Qué tipo de letra está más claro?; aunque estos detalles le importan en gran manera al usuario final e influyen en la usabilidad, no son arquitectónicos porque pertenecen a los detalles de diseño. En cambio se consideran arquitectónicos los que involucran que un sistema proporcione la habilidad de cancelar las operaciones, deshacer las operaciones al usuario o re-utilizar datos ingresados previamente [22].

2.4.1.1 Costo & tiempo para comercializar

Entregar un producto a tiempo y dentro del presupuesto son restricciones muy difíciles de cumplir dadas por la dirección del proyecto. Tener una arquitectura definida contribuye a disminuir los tiempos en el proceso de desarrollo y por lo tanto a cumplir satisfactoriamente con los tiempos de entrega.

2.4.1.2 Complejidad

Analógicamente a como el arquitecto de un rascacielos necesita tratar con una fuerza externa que es la gravedad (el edificio no debe caerse en pedazos), así debe tratar un arquitecto de software con una fuerza interior que es la complejidad (la arquitectura del software no debe volverse compleja). Para cumplir con los requisitos de calidad y permitir la evolución del sistema, los arquitectos de software a menudo deben añadir mayor flexibilidad a la arquitectura [25].

Por ejemplo, para un sistema de dirección basado en contentos Web puede ser desconocido que tipos de bases de datos serán usados en dos o tres años (Relacional / orientado a objeto). También puede ser desconocido si un proveedor continuará dando soporte a la base de datos en el futuro. Para aumentar la modificación un arquitecto puede decidir diseñar una arquitectura que usa el mecanismo de abstracción de datos flexible que permite el empleo de los tipos diferentes de medios de almacenamiento persistentes (por ejemplo tipos diferentes o clases de bases de datos) en vez de usar

una arquitectura que usa un tipo fijo de medio persistente (por ejemplo un archivo de texto o una base de datos) [25].

Un arquitecto no debe olvidarse que el objetivo principal de una arquitectura de software consiste en facilitar la comunicación entre los diseñadores y otros participantes. Una arquitectura que sólo puede ser entendida por el arquitecto no cumple con este principio. En cualquier momento el arquitecto de software debe mantener "una integridad conceptual" [42], p. ej: saber cuando detenerse para no añadir más complejidad y así mantener la simplicidad en el diseño.

2.4.1.3 Reutilización

El software debe ser diseñado de tal modo que ciertos artefactos de software (componentes, módulos, etc.) puedan ser reutilizados en otros contextos para desarrollar nuevos sistemas de software dentro del mismo dominio. Si se cumple con la reutilización el software puede ser desarrollado en un tiempo más corto. Además, pueden esperar una calidad más alta de los artefactos de software ya que la posibilidad de descubrir errores en los artefactos es mucho mayor porque ellos son usados en contextos diferentes.

2.4.1.4 Diseñar bajo restricciones

Un sistema de software debería ser fiel a sus requisitos, de alta calidad y ser entregado a tiempo dentro del presupuesto. Además, la arquitectura no debe hacerse demasiado compleja y debe dar apoyo a la evolución y a la reutilización. A causa de estas restricciones informales, las decisiones de diseño para la arquitectura son, por lo tanto, a menudo, tomadas por intuición, confiando en la experiencia de los diseñadores de software senior [43].

Para hacer el diseño menos intuitivo y accesible a inexpertos las comunidades de diseñadores han hecho un esfuerzo para unir sus experiencias y conocimientos dando un compendio de posibilidades para que los arquitectos puedan resolver problemas particulares.

2.4.2 Patrones

Algunas de las mejores prácticas acerca del diseño de arquitectura de software se han recopilado en los patrones, estos describen un problema que ocurre repetidas veces en algún contexto determinado del desarrollo de software y entregan una buena solución ya probada. Esto ayuda a diseñar correctamente en menor tiempo, a construir soluciones reutilizables y extensibles, y facilita la documentación y la comunicación [21].

Casi todas las soluciones de diseño funcional y de calidad del software que están relacionadas con problemas que se dieron en el diseño de la arquitectura se han descrito por medio de los patrones de diseño [44], los patrones arquitectónicos y los estilos [21].

2.4.2.1 Patrones Arquitectónicos

El patrón arquitectónico es una colección de reglas [45] que pueden ser impuestos en el sistema. Un patrón arquitectónico se diferencia de un estilo de arquitectura en que no es predominante y puede ser combinado con estilos arquitectónicos. Un ejemplo de un patrón arquitectónico es un sistema administrador de base de datos (DBMS). Un DBMS generalmente amplía el sistema con un componente adicional o capas, pero ello también impone reglas a los componentes de arquitectura originales. Las entidades que tienen que ser mantenidas persistentes (por ejemplo capacidad de los datos para sobrevivir al proceso en que fueron creados) tienen que ser ampliadas con una funcionalidad adicional para apoyar este aspecto. El empleo de un DBMS generalmente tiene algún efecto sobre características de calidad como funcionalidad, capacidad de mantenimiento

/ modificación y seguridad. Por ejemplo, la funcionalidad es negativamente afectada ya que todas las operaciones de la base de datos tienen que ir a través del DBMS.

2.4.2.2 Patrones de Hipermedia

Los patrones hipermedia pueden ser vistos como patrones de diseño especializados que han sido adaptados a las características específicas del dominio hipermedia [46].

Según Rossi [46]-[48] las aplicaciones en el dominio de hipermedia pueden organizarse en dos categorías: Sistema hipermedia y aplicaciones hipermedia.

Las aplicaciones hipermedia se subdividen en sistemas de patrones para: sistema hipermedia, diseño navegacional y presentación. Estas categorías las definió Rossi basado en la experiencia de diseñar aplicaciones hipermedia aplicando métodos orientados a objeto y patrones.

2.4.2.3 Patrones de Interacción

Los patrones de interacción es la representación de la información conforme a las necesidades del usuario, y se traduce en la satisfacción, eficiencia y aceptabilidad que sienten los usuarios al utilizar los productos informáticos que requieren. Ayudan a diseñar sistemas fáciles de usar para las personas [50].

2.4.3 Cualidades de una arquitectura

Una arquitectura de software debe poseer determinadas cualidades tales como:

- Conformidad Funcional: Una arquitectura de calidad debe implementar todos los requisitos explícitos contenidos en el modelo de análisis y debe acomodar todos los requisitos implícitos que desea el cliente [51].
- Adaptabilidad: Mide cuan fácil es hacer cambios en una arquitectura.
- Modularidad: Permite que el diseño interno de cada componente esté oculto para el resto de los otros componentes [51].
- Niveles de Abstracción: Ayudan a entender el problema y la solución que debe dar el sistema. Se proponen tres (3) niveles de abstracción [41]: Abstracción procedimental, abstracción de datos y abstracción de control.
- Entendible: El entendimiento estará afectado por: La cohesión, el acoplamiento, la nominación, la documentación y la complejidad [52].
- Cohesión: Es una consecuencia del ocultamiento de la información. La meta es hacer que los componentes sean lo más cohesivos posible [53].
- Acoplamiento: Es un indicador de la fuerza de interconexión entre los componentes o elementos de la arquitectura [52]. Los sistemas altamente acoplados tienen una fuerte interconexión, lo que se refleja en una gran dependencia entre los componentes; Los poco acoplados, por otro lado, tienen poca relación entre sus componentes o elementos. La meta es mantener el acoplamiento en el nivel más bajo posible [53].

¿Cómo saber si una arquitectura cumple con estas cualidades?

La evaluación de las arquitecturas de software puede ser realizada mediante el uso de diversas técnicas y métodos.

2.4.4 Técnicas para evaluación de la calidad de una arquitectura

Las técnicas de evaluación de arquitecturas de software permiten al arquitecto realizar mediciones sobre ciertos atributos de calidad [53]. Clements [54], clasifica las técnicas de evaluación de arquitecturas de software en:

- Técnicas inquisitivas: se destacan los cuestionarios, las listas de chequeo y los escenarios.

- Técnicas de medición: son utilizadas para responder interrogantes específicos sobre atributos de calidad determinados. Utilizan instrumentos como los lenguajes de descripción arquitectónica (ADL), y las métricas.

2.4.4.1 Evaluación basada en escenarios

Un escenario es una breve descripción de la interacción de alguno de los involucrados en el desarrollo del sistema con éste. Un escenario consta de tres partes: el estímulo, el contexto y la respuesta. El estímulo es la parte del escenario que explica o describe lo que el involucrado en el desarrollo hace para iniciar la interacción con el sistema [54]; Puede incluir ejecución de tareas, cambios en el sistema, ejecución de pruebas, configuración, etc. El contexto describe qué sucede en el sistema al momento del estímulo. La respuesta describe, a través de la arquitectura, cómo debería responder el sistema ante el estímulo. Este último elemento es el que permite establecer cuál es el atributo de calidad asociado.

Actualmente las técnicas basadas en escenarios cuentan con dos instrumentos de evaluación relevantes, a saber: el Utility Tree propuesto por Kazman [54] y los Perfiles, propuestos por Bosch [5].

2.4.4.1.1 Utility Tree

Es un esquema en forma de árbol que presenta los atributos de calidad de un sistema de software, refinados hasta el establecimiento de escenarios que especifican con suficiente detalle el nivel de prioridad de cada uno [54].

La intención del uso del Utility Tree es la identificación de los atributos de calidad más importantes para un proyecto particular. No existe un conjunto preestablecido de atributos, sino que son definidos por los involucrados en el desarrollo del sistema al momento de la construcción del árbol.

2.4.4.1.2 Perfiles

Un perfil (profile) es un conjunto de escenarios. El uso de perfiles permite hacer especificaciones más precisas del requerimiento para un atributo de calidad [53]. Los perfiles tienen asociados dos formas de especificación: perfiles completos y perfiles seleccionados.

Los perfiles completos definen todos los escenarios relevantes como parte del perfil. Esto permite al ingeniero de software realizar un análisis de la arquitectura para el atributo de calidad estudiado de una manera completa, puesto que incluye todos los posibles casos. Su uso se reduce a sistemas relativamente pequeños [54].

Para los perfiles seleccionados se toma un conjunto de escenarios de forma aleatoria, de acuerdo a algunos requerimientos Si bien es informal, permite hacer proposiciones científicamente válidas [54].

La evaluación basada en escenarios puede ser empleada para comparar dos arquitecturas y para la evaluación absoluta de una arquitectura. La diferencia está en que la evaluación absoluta requiere mayor cantidad de datos estimados y cuantitativos necesarios para la evaluación. Bosch [5] explica que la técnica consiste en dos etapas: análisis de impacto y predicción de atributos de calidad. El análisis de impacto toma como entrada el perfil y la arquitectura de software. Para cada escenario del perfil, se evalúa el impacto de la arquitectura y se obtienen los resultados que serán usados en la etapa de predicción de atributos de calidad, donde se pronostica el valor del atributo de calidad estudiado de acuerdo a las métricas existentes.

2.4.5 Métodos de Evaluación de Arquitecturas de Software

Hasta hace poco no existían métodos de utilidad general para evaluar arquitecturas de software. En virtud de esto, múltiples métodos de evaluación han sido propuestos: SAAM [54], ATAM [55] y QASAR [5] estos métodos no se enfocan a un solo atributo de calidad, pero proporcionan pasos para la evaluación y el razonamiento sobre diferentes atributos de calidad. Existen otros métodos de evaluación derivados de los anteriores que están enfocados a atributos de calidad específicos, como por ejemplo: SAAMER [56] y ALMA [23]; sin embargo, es evidente que algunos atributos de calidad como por ejemplo la usabilidad son difíciles de medir en fases tempranas del ciclo de vida del proceso de desarrollo de software y para dichos atributos aún no existen técnicas formales de evaluación.

2.5 Usabilidad

2.5.1 ¿Qué es la Usabilidad?

Coloquialmente suele definirse usabilidad como la propiedad que tiene un determinado sistema para que sea "fácil de usar o de utilizar y de aprender"; Esta definición, que en esencia es correcta, no deja de ser incompleta ya que el término engloba muchas más connotaciones. A continuación se citan una serie de definiciones propuestas por prestigiosas organizaciones y autores que ayudan a comprender la usabilidad como concepto.

Definiciones ISO: El organismo propone dos definiciones del término usabilidad, dependiendo de los términos considerados en el momento de especificar o evaluar dicha usabilidad:

1) ISO 9241-11 [57]: proporciona la siguiente definición de usabilidad: *"La medida en la que un producto se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso especificado".*

Esta norma explica cómo identificar la información que se necesita considerar en el momento de especificar o evaluar la usabilidad en términos de medidas de funcionamiento y de satisfacción del usuario.

2) ISO/IEC 9126 [58]: proporciona la siguiente definición de usabilidad: *"La capacidad que tiene un producto software para ser atractivo, entendido, aprendido, usado por el usuario cuando es utilizado bajo unas condiciones específicas."* En esta definición las palabras "…*usado bajo unas condiciones específicas*" hacen que se deje en claro que un producto no dispone de usabilidad intrínseca, sino que tiene la capacidad de ser usado en un contexto particular.

Definiciones de autores destacados: Además de las definiciones "oficiales" ofrecidas por los estándares se dispone de otras definiciones con puntos de vista diferentes, de varios autores e investigadores de reconocido prestigio internacional. Veamos, algunas de las más determinantes:

1) *Jakob Nielsen [59]***:** pionero en la difusión de la usabilidad, sugiere que la usabilidad es un término multidimensional. Indica que un sistema usable debe poseer los siguientes atributos: Capacidad de aprendizaje, eficiencia en el uso, facilidad de memorizar, tolerante a errores y ser subjetivamente satisfactorio. Nielsen no da una definición precisa de la usabilidad, pero presenta los criterios operacionales que definen el concepto claramente.

2) *Jenny Preece [60]*: autora de multitud de estudios de usabilidad y de varios reconocidos libros, propone la definición más corta pero quizás la más intuitiva. Se refiere a la usabilidad como el "desarrollo de sistemas fáciles de usar y de aprender".

3) Janice Redish [61]: reconocida profesional de la usabilidad defiende la idea de que el objetivo de las personas que trabajan en la usabilidad no es otro que el de producir "trabajos para sus usuarios", proporcionando a los usuarios las herramientas para poder encontrar lo que necesitan, entender lo que encuentran, actuar apropiadamente sobre ese entendimiento, y hacer todo esto con el tiempo y esfuerzo que ellos crean necesarios. Porque el término usabilidad no se refiere solamente a hacer que los sistemas sean simples, sino que abarca además la comprensión de los objetivos de los usuarios, el contexto de su trabajo y cuál es el conocimiento y la experiencia de que disponen.
4) Whitney Quesenbery [62]: propone extender la definición de la ISO 9241 para hacerla más comprensible. Propone definir la usabilidad en base a las 5 características que los usuarios deben encontrar en el sistema interactivo, las "5 Es": efectividad[4], eficiencia[5], ser atractivo[6] , tolerante a errores[7] y facilidad de aprendizaje[8].

Como se puede ver existen diferentes conceptos acerca del término usabilidad, en este proyecto se tomará la definición dada por el estándar ISO 9241-11 [57], donde el producto que se menciona en esta puede ser un artefacto[9] a desarrollar, como complemento también se tendrá en cuenta la facilidad de uso mencionada por Preece [60], que en términos del artefacto se puede extender como la capacidad de reutilización de los artefactos del componente, y se tomará la definición dada por el estandar ISO/IEC 9126 [58] debido a que el artefacto a desarrollar deberá tener la capacidad de ser comprendido por el usuario, lo cual es una característica deseable de abstracción a alto nivel de cualquier arquitectura software. Con lo expuesto anteriormente se propone la siguiente definición de usabilidad, la cual será la que se use a lo largo de este proyecto:

Es la capacidad, facilidad y medida de un artefacto para que pueda ser usado por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción (atractivo, entendido y aprendido) en un contexto de uso especificado.

2.5.2 Atributos Generales de la Usabilidad

Al igual que existen variedad de definiciones sobre usabilidad, también existen diversas definiciones para el término atributo de usabilidad, pero el significado generalmente aceptado es que un atributo de usabilidad es un componente preciso y medible del concepto abstracto que es la usabilidad [25].

A continuación se da una visión general de los atributos más comúnmente citados entre los autores, Constantine y Lockwood [63], Hix y Hartson [64], ISO 9126-1 [58], Nielsen [59], Preece et al [60], Shackel [65], Shneiderman [66] y Wixon & Wilson [67] en el campo de usabilidad. Se han agrupado los atributos que se refieren al mismo concepto en la misma fila. Algunos autores usan más atributos que los mencionados en la tabla 2..

4 Efectividad viene del ingles Effective, en el resto del documento se utilizará el termino "Efectividad" para referirnos a "Effective".
5 Eficiencia viene del ingles Efficiency, en el resto del documento se utilizará el termino "Eficiencia" para referirnos a "Efficiency" o a "Efficiency in use".
6 Ser atractivo viene del ingles Engaging, en el resto del documento se utilizará el termino "Ser Atractivo" para referirnos a "Engaging" o a "Attractiveness".
7 Tolerante a errores viene del ingles Error-Tolerant, en el resto del documento se utilizará el termino "Tolerante a errores" para referirnos a "Error-Tolerant".
8 Facilidad de aprendizaje viene del ingles Learnability y de Easy-to-Learn, en el resto del documento se utilizará el termino "Fácilidad de aprendizaje" para referirnos a "Learnability".
9 Artefacto: Es una pieza de información que es producida, modificada o usada por un proceso. Son productos tangibles del proyecto: las cosas que el proyecto produce o usa mientras se trabaja hacia el producto final.

Tabla 2: Atributos de Usabilidad.

Constantine	ISO9126	Nielsen	Preece	Shackel	Shneiderman
Facilidad de aprendizaje	Facilidad de aprendizaje	Facilidad de aprendizaje	Facilidad de aprendizaje	Facilidad de aprendizaje	Tiempo de aprendizaje
Eficiencia en uso	Operabilidad	Eficiencia de Uso	Navegación	Efectividad.	Velocidad de ejecución
Facilidad de recordar		Capacidad de recordar		Facilidad de Aprendizaje	Retención sobre el tiempo
Fiabilidad de uso	Operabilidad	Errores	Navegación	Efectividad	Tasa de errores por los usuarios
Satisfacción del usuario	atractivo	Satisfacción	Actitud	Actitud	Satisfacción subjetiva

Cabe especificar para este proyecto se han seleccionado los siguientes cuatro atributos, debido a que se considera que estos son como la base de algunos de los otros atributos y son los que se ajustan directamente con los objetivos de este proyecto.

- Facilidad de aprendizaje: la facilidad con que los usuarios pueden empezar a hacer el trabajo productivo con un sistema que es nuevo para ellos, combinado con la facilidad de recordar la manera en que el sistema debe ser operado.
- Eficiencia de uso: el número de tareas por unidad de tiempo que el usuario puede ejecutar cuando usa el sistema.
- Confiabilidad en uso[10]: la tasa de error de los usuarios, no los errores del sistema, cuando se usa y el tiempo que toma para recuperarse de estos.
- Satisfacción de usuario[11]: las opiniones subjetivas de los usuarios del sistema.

Se ha agrupado los atributos de usabilidad: capacidad de recordar[12], retención sobre el tiempo[13], y facilidad de recordar[14] en el atributo facilidad de aprendizaje. Algunas definiciones como la ISO 9126-1 y Preece et al, no mencionan este atributo y Shackel también une la retención sobre el tiempo con la facilidad de aprendizaje. También se ha usado el término confiabilidad para hacer referencia a la tasa de error al usar el sistema.

2.5.3 Beneficios de la Usabilidad

Existen estudios de varios autores que constatan los beneficios aportados por la usabilidad. Algunos de estos autores incluso afirman que la usabilidad mejora la productividad de los usuarios e incrementa su moral, reduce costos de formación y de documentación, permitiendo, por ejemplo, aumentar cuota de mercado [67], [68].

Mayhew y Mantei [69] fueron los primeros en describir los beneficios de aplicar la usabilidad al diseño software desde un punto de vista interno y de las ventas realizadas. A continuación se presenta una descripción de estos beneficios organizados en tres áreas: Desarrollo, uso interno y ventas.

[10] Confiabilidad en uso viene del ingles Reliability in use , en el resto del documento se utilizará el termino “Confiabilidad en uso” para referirnos a “Reliability in use”.
[11] Satisfacción de usuario viene del ingles user satisfaction, en el resto del documento se utilizará el termino “Satisfacción de usuario” para referirnos a “User Satisfaction”.
[12] Capacidad de recordar viene del ingles Memorability , en el resto del documento se utilizará el termino “Capacidad de recordar” para referirnos a “Memorability”.
[13] Retención sobre el tiempo viene del ingles Retention over time , en el resto del documento se utilizará el termino “Retención sobre el tiempo” para referirnos a “Retention over time”.
[14] Facilidad de recordar viene del ingles Rememberability , en el resto del documento se utilizará el termino “Facilidad de recordar” para referirnos a “Rememberability”.

1) Desarrollo:
   - Reducción de los costos de producción**:** Aunque parezca contradictorio, los costos y tiempos de desarrollo totales se pueden reducir evitando el sobre diseño y reduciendo el número de cambios posteriores requeridos en el producto. Se optimizan los costos de diseño y rediseño de las aplicaciones.
   - Reducción de los costos de mantenimiento y apoyo**:** Los sistemas que son fáciles de usar requieren menos entrenamiento, menos soporte para el usuario y menos mantenimiento.
   - Reducción de los costos corporativos: ya se consigue como consecuencia de los dos puntos anteriores. el valor que el equipo de usabilidad de una compañía que desarrolla software aporta al conjunto global de dicha compañía proporciona mejoras metodológicas en el sistema global de desarrollo, reduciendo, por tanto, los costos generales [69] , [70]**.**
2) Uso interno:
   - Reducción de los costos de uso**:** Los sistemas que mejor se ajustan a las necesidades del usuario mejoran la productividad y la calidad de las acciones y las decisiones. Los sistemas más fáciles de utilizar reducen el esfuerzo y permiten a los usuarios manejar una variedad más amplia de tareas.
     Mientras que los sistemas difíciles de usar disminuyen la salud, bienestar y motivación y pueden incrementar el absentismo.
   - Reducción de los costos de aprendizaje: Un sistema con elevadas dosis de usabilidad está organizado de manera que se adapta mejor al modelo mental de sus usuarios con lo que se minimiza el tiempo necesario para su aprendizaje.
   - Mejora la calidad de vida de los usuarios**:** ya que reduce su estrés, incrementando la satisfacción y la productividad.
3) Ventas:
   - Incremento de las ventas: Un producto más usable permite un mejor marketing debido a la mejor imagen del producto, es más compresible, y por tanto más fácilmente vendible.
   - Mejora en la calidad del producto: El diseño centrado en el usuario da lugar a aplicaciones de mayor calidad de uso, más competitivos en un mercado que demanda productos de fácil uso. En el entorno Web se mejora la imagen y el prestigio del sitio, lo que favorece el aumento de la tasa de conversión de visitantes a clientes.
   - Menor soporte al cliente: Los sistemas usables son más fáciles de aprender y de utilizar, comportando un menor coste de implantación y de mantenimiento.
   - Retorno de la inversión en usabilidad: Algunos ejemplos que demuestran la validez del retorno de la inversión en usabilidad son: S. Dray [71], InfoWorld [72], Intranet de “La Caixa” [73].

2.5.4 ¿En qué momento se debe considerar la Usabilidad?

La usabilidad debería ser considerada en todo momento, desde el mismo comienzo del proceso de desarrollo hasta las últimas acciones antes de liberar el sistema, producto o servicio a sus destinatarios.

En la primera fase de todo proyecto es esencial tener una idea acerca de las características de los usuarios y de los aspectos del producto de mayor interés y necesidad. Teniendo en cuenta estas consideraciones de forma temprana se ahorra tiempo y dinero, dado que la posterior implementación de nuevos aspectos o nuevas interfaces de usuario implican un enorme esfuerzo adicional. Durante todo el desarrollo se deben realizar pruebas para comprobar que se está considerando la usabilidad del producto. Incluso una vez que el producto está en el mercado se debería preguntar a los usuarios acerca de sus necesidades y actitud respecto del mismo [74]. Esta idea de tener constantemente presente la usabilidad del sistema será uno de los puntos principales de este trabajo de grado.

2.5.5 Relación entre Usabilidad y Arquitecturas de Software

En los últimos años, cada vez más la usabilidad ha sido reconocida como una consideración importante durante el desarrollo de software. Los asuntos como si un producto es fácil de aprender, usar, o si es receptivo al usuario y si el usuario puede eficientemente terminar las tareas, podrían significativamente afectar enormemente la aprobación y el éxito de un producto en el mercado. En el futuro, cuando los usuarios son más exigentes, la mala usabilidad puede ser una barrera muy importante en el éxito de nuevas aplicaciones de software comerciales. Por lo tanto las organizaciones desarrolladoras de software están prestando más y más atención en asegurar la usabilidad de su software.

Uno de los problemas con muchos de los sistemas de software de hoy es que no cubren bien los requisitos de calidad. Además, los cambios necesarios para mejorar su calidad a menudo son difíciles de probar y de identificar. Una razón para esto es que muchos de los cambios necesarios requieren cambios en el sistema, lo que no puede ser fácilmente proporcionado por la arquitectura de software [5].
Muchos productos de software reconocidos se ven afectados por la usabilidad y no pueden ser corregidos sin hacer cambios muy importantes a la arquitectura de software de estos productos. Los estudios de proyectos de ingeniería de software revelan que las organizaciones gastan una cantidad relativamente grande en tiempo y dinero arreglando problemas de usabilidad. Varios de los estudios han indicado que el 80% de los gastos totales de mantenimiento están relacionados con los problemas del usuario con el sistema [75]. Entre estos gastos, 64 % están relacionados con los problemas de usabilidad [76]. Estas cifras muestran que una cantidad grande de costos de mantenimiento es gastado en arreglar asuntos de usabilidad.

Una razón importante para estos costos elevados es que la mayoría de los asuntos de usabilidad son solamente detectados durante la prueba y el despliegue y no durante el diseño y la implementación.

Por consiguiente, muchas solicitudes de cambio para mejorar la usabilidad son hechas después de estas fases. Esto hace que sea costoso cubrir todos los requerimientos de usabilidad. Las causas potenciales para este problema son [25]:

- La evaluación de la usabilidad requiere un prototipo del sistema a elaborar. la mayoría de las técnicas de evaluación de usabilidad existentes requieren por lo menos un prototipo interactivo y un conjunto representativo de usuarios presentes para evaluar la usabilidad de un sistema. Algunas técnicas como rápida creación de prototipos [59], permiten la prueba temprana, por ejemplo usando un prototipo o la simulación de una interfaz. La creación de prototipos temprana, incluso sobre el papel, de cómo será la experiencia del cliente, siempre tiene gran valor. Sin embargo, los prototipos tienen una habilidad limitada para presentar la arquitectura de aplicación, ya que solamente hacen un modelo de la interfaz. Los asuntos de interacción tales como si una tarea puede ser deshecha, o el tiempo que toma para llevar a cabo una tarea específica o las propiedades de sistema como la confiabilidad tienen una gran influencia sobre el nivel de la usabilidad, pero son difíciles de simular con un prototipo.

- Limitación de los requerimientos de ingeniería: las técnicas de la ingeniería de la usabilidad como diseño centrado por uso [63] tienen una habilidad limitada para capturar o predecir todos requerimientos de usabilidad. Durante el desarrollo, ni los usos del sistema no son a menudo documentados completamente ni tampoco una definición está hecha de exactamente quiénes serán los usuarios. Usando técnicas tradicionales los expertos de usabilidad notan aproximadamente la mitad de los problemas que los usuarios reales

experimentan [76]. Por lo tanto, algunos requisitos de usabilidad no serán descubiertos hasta que el software ha sido desplegado.

- Cambio de requisitos: durante o después del desarrollo, los requisitos de usabilidad cambian. El contexto en el que el usuario y el software operan está cambiando continuamente y evolucionando, lo cuál hace difícil la capturar de todos los posibles requisitos de usabilidad al principio [77].

Éstas son tres de las razones principales de que algunos problemas de usabilidad no son descubiertos hasta la prueba y el despliegue. El problema con esta detección tardía es que a veces es muy difícil aplicar las soluciones de diseño que arreglan estos problemas de usabilidad, porque algunas de estas soluciones de diseño podrían afectar la arquitectura software. Algunos cambios que pueden mejorar la usabilidad requieren un grado cuantioso de modificación. Por ejemplo, los cambios que se relacionan con las interacciones que tienen lugar entre el sistema y el usuario, como deshacer una función en particular, no puede ser implementada fácilmente después de la puesta en práctica sin incurrir en grandes gastos.

Un primer paso para solucionar estos problemas es investigar qué soluciones pueden afectar la arquitectura. Una de las motivaciones de este proyecto es adquirir un mejor entendimiento de la relación entre la usabilidad y la arquitectura de software, para así contribuir en la solución.

2.5.6 Propiedades de usabilidad

Las propiedades de usabilidad expresan o representan las heurísticas y los principios de diseño que los investigadores en el campo de usabilidad han encontrado, tienen una influencia directa sobre la usabilidad del sistema. Estas propiedades pueden ser usadas como requerimientos en la etapa de diseño [25]

El concepto de propiedad de usabilidad conecta patrones de usabilidad sensibles a la arquitectura con requerimientos y crea una relación directa entre los atributos de usabilidad y los patrones de usabilidad sensibles a la arquitectura [25].

Debido a que en las investigaciones realizadas en el proyecto STATUS [25], se llegó a la conclusión que el puente que une a los atributos de usabilidad con la parte arquitectónica son las propiedades, ya que los atributos de usabilidad solo pueden ser evaluados cuando la aplicación ha sido desarrollada y no desde las etapas iníciales del proceso de desarrollo. Se definió un conjunto de propiedades de usabilidad, con base en los principios heurísticos de Molich y Nielsen [78] y en las propiedades del proyecto STATUS [25]. A continuación se mencionan y explican las propiedades que han sido identificadas:

1. **Realimentación:** El sistema debe siempre mantener a los usuarios informados del estado del sistema, con una realimentación apropiada y en un tiempo razonable.
2. **Gestionando el error:** El sistema debe prevenir la aparición de errores antes que generar buenos mensajes de alerta; hay dos formas de gestionar el error: una es previniendo que el error ocurra y la otra es corrigiendo el error. En caso de que el usuario cometa un error el sistema debe proveer mensajes expresados en un lenguaje claro (sin códigos extraños), indicando exactamente el problema de manera constructiva.
3. **Consistencia:** El sistema debe prevenir que los usuarios tengan que preguntarse si las diversas palabras, situaciones, o acciones significan la misma cosa. La consistencia puede darse de las siguientes maneras:
   a. Visual: Los elementos de la interfaz de usuario deben ser consecuentes o consistentes en el aspecto y estructura.

b. Funcional: La manera de llevar a cabo las diferentes tareas a través del sistema debe ser consecuente, con otros sistemas similares, e incluso entre diferente clases de aplicaciones en el mismo sistema.

c. Evolutiva: Debe ser fácil tener una nueva versión del software, y que esta sea consistente con las anteriores. Como por ejemplo, en el caso de una familia de producto software, la consistencia sobre los productos en la familia es un aspecto importante.

4. Ayuda u orientación: Aunque es mejor si el sistema se puede usar sin documentación, puede ser necesario disponer de ayuda y documentación. Ésta debe ser fácil de buscar, centrada en las tareas del usuario, tener información de las etapas a realizar, ser contextual y no muy extensa

5. Minimizar la carga cognitiva: Los seres humanos tienen limitaciones cognitivas del tratamiento de la información en memoria a corto plazo, por ello se requiere que lo que se muestra por pantalla sea simple, por ejemplo presentar siete ítems por pantalla es una sobrecarga de información porque según estudios realizados siete u ocho es el medio límite de memoria a corto plazo. De igual forma el usuario no debe tener que recordar la información de una parte de diálogo a la otra; es mejor mantener objetos, acciones, y opciones visibles que memorizar.

6. Explícito control de usuario: Los usuarios desean tener el control total del sistema y que responda a sus acciones. El diseño debe responder a las acciones de los usuarios, y que estos sean sus iniciadores, no solo los que respondan a acciones del sistema. El usuario debe tener la impresión de que él tiene "el control" de la aplicación.

7. Mapeo natural: El sistema debe proveer una relación clara entre lo que el usuario quiere hacer y el mecanismo para llevarlo acabo. Esta propiedad puede estar estructurada de la siguiente manera:

a. Predecible: Para el usuario el comportamiento del sistema debe ser predecible.

b. Lenguaje de los usuarios: El sistema debe hablar el lenguaje de los usuarios, con las palabras, las frases y los conceptos familiares, en lugar de que los términos estén orientados al sistema. Utilizar convenciones del mundo real, haciendo que la información aparezca en un orden natural y lógico.

c. Facilidad de la navegación: Debe ser obvio al usuario cómo navegar en el sistema.

8. Accesibilidad: Los sistemas deben ser accesibles en todas las maneras que es requerido. Tal propiedad podría ser descompuesta de la siguiente manera:

a. Discapacidades: los sistemas deben suministrar el soporte para usuarios que son discapacitados (ciego / sordo/ etc.).

b. Multicanal: el sistema debe ser capaz de soportar el acceso por varios medios de comunicación. Este es un concepto muy amplio que varía desde ser capaz de navegar en un sitio Web por medio de un teléfono o ser capaz de hacerlo a través de audio.

c. Internacionalización: Los sistemas deben proveer el soporte para la internacionalización, porque los usuarios están más familiarizados con su propia lengua, moneda, formato de código postal, formato de fecha, etc.

9. Adaptabilidad: El sistema debe poder satisfacer las necesidades del usuario cuando el contexto cambia o adaptarse a los diferentes usuarios ya sean expertos o inexpertos. Tal propiedad podría ser descompuesta como sigue:

a. Experiencia de usuario: La habilidad de adaptarse al nivel de experiencia de usuario.

b. Personalización: La habilidad de proveer ciertos servicios personalizados. Por ejemplo al crear atajos para los usuarios frecuentes: cuando la frecuencia de uso aumenta, los usuarios agradecen reducir el número de interacciones. La abreviaturas, los comandos ocultos, y las macro son muy útiles para un usuario experto.

c. Facilidad de recordar: La capacidad del sistema para recordar los detalles anteriores de la interacción usuario - sistema.

La relación entre las propiedades de usabilidad y los atributos que afectan se puede observar en la siguiente tabla, en el Anexo 1 se presenta en más detalle esta relación [25].

Tabla 3 Relación de las Propiedades y Atributos de Usabilidad

Propiedad de Usabilidad	Atributo de Usabilidad
Realimentación	Eficiencia
	Facilidad de Aprendizaje
Gestionando el Error	Confiabilidad
	Eficiencia
Consistencia	Facilidad de Aprendizaje
	Confiabilidad
Ayuda u orientación	Facilidad de Aprendizaje
	Eficiencia
	Confiabilidad
Minimizar la carga cognitiva	Confiabilidad
	Eficiencia
Explícito Control de Usuario	Satisfacción
Mapeo Natural	Facilidad de Aprendizaje
	Eficiencia
	Confiabilidad
Accesibilidad	Satisfacción
	Facilidad de Aprendizaje
Adaptabilidad	Satisfacción
	Eficiencia

Capítulo 3 MARCO DE REFERENCIA CENTRADO EN LA ARQUITECTURA PARA LA MEJORA DE CARACTERISTICAS DE USABILIDAD EN EL DESARROLLO DE APLICACIONES Web CONSTRUIDAS POR MIPYMES

La pregunta que seguramente surge después de presentar las propiedades expuestas en el capítulo 2 es: ¿Qué decisiones de diseño se deben tomar en una aplicación Web para poder cumplir con dichas propiedades?, y es en este sentido en el que se ha hecho un gran esfuerzo por analizar los diferentes patrones lógicos, de presentación, de datos y sus implicaciones sobre las propiedades de usabilidad. Teniendo en cuenta que estos patrones pueden ser usados por los patrones de alto nivel como son cliente / servidor, Peer to Peer y SOA, descritos en el capítulo 2.

Para el análisis se tomo en cuenta el conjunto de patrones arquitecturales relacionados con el desarrollo de aplicaciones Web presentados en los libros de Buschman [21] y Fowler [79]; este análisis consistió en determinar cuáles de los patrones arquitectónicos citados por Buschman [21] aplicaban al dominio de aplicaciones Web, ya que estos son mencionados para cualquier tipo de software; para los patrones dados por Fowler no se tomo en cuenta este aspecto ya que el autor los enmarca dentro del dominio de aplicaciones Web. Luego se determinó cuales patrones tenían implicaciones sobre las propiedades de usabilidad descritas en la sección 2.5.6.

Debido a que en general una aplicación Web consta de tres elementos fundamentales presentación, lógica de negocio y datos (aunque en algunos casos es difícil saber el límite entre ellos), se realizó una clasificación de los patrones arquitectónicos, hipermedia y de interacción con base en estos elementos, además se subclasificaron según los niveles de abstracción, así: Nivel Alto en el elemento de presentación hace referencia a la forma como se va a estructurar la presentación de la aplicación, Nivel Intermedio hace referencia a los patrones que contribuyen a definir como se desarrollara la capa de presentación y el Nivel Bajo hace referencia a las decisiones de diseño de interfaz y de interacción de la aplicación Web. Para el elemento de lógica de negocio el Nivel Alto se refiere a la manera en como a nivel macro se esctructurará la aplicación Web, en el Nivel Intermedio se encuentran los patrones que ayudan a definir puntualmente la lógica de la aplicación. Para el elemento de Datos se tomarón aquellos patrones que definen la forma en la cual se tendrá acceso a los datos a partir de la lógica del negocio y del diseño de base de datos que se requiera (orientado a objetos, en archivos planos, relacional, etc).

En la Figura 1 se presenta la propuesta de este trabajo para la clasificación de los patrones que se consideran tienen relación directa con las propiedades de usabilidad.

Figura 1 Clasificación de Patrones

Para definir el impacto que los patrones de la Figura 1 tienen sobre la usabilidad se propone la escala de impacto presentada en la Tabla 4. El impacto en la usabilidad podría ser positivo, en el sentido que mejora los atributos de usabilidad o negativo en el sentido que perjudica a los atributos de usabilidad relacionados, así se ha dejado tres niveles de impacto positivo y negativo A, M y B, siendo A el grado más alto de impacto y el cual quiere decir que el patón garantiza el cumplimiento de la propiedad relacionada, M medio el cual se refiere a que al utilizar el patrón se permite que se cumpla con la propiedad y B bajo lo que implica que la utilización del patrón podría ayudar a que se cumpla la propiedad. De igual manera en ciertos casos hay patrones que no afectan a la Usabilidad por lo cual su impacto será NI.

Tabla 4 Escala de impacto de Usabilidad

Grado de Impacto	Símbolo
Alto Positivamente	A+
Medio Positivamente	M+
Bajo Positivamente	B+
No Influye	NI
Bajo Negativamente	B -
Medio Negativamente	M -
Alto Negativamente	A -

Este capítulo esta estructurado de la siguiente manera en la sección 3.1 se analizan los patrones de presentación clasificados por niveles, en la sección 3.2 se analizan los patrones de lógica de dominio y en la sección 3.3 los patrones de datos.

3.1 Patrones de presentación

Los siguientes patrones son significativos para las Interfaces de Usuario (IU) y controladores de vistas. Su utilización es abstracta e independiente del ambiente usado para la implementación. Debido a que diseñar la interfaz Figura es una tarea compleja, se debe encontrar la combinación de elementos correcta tanto en número como en espacio, de tal manera que esos elementos interactúen para una presentación eficaz de la información.

Según la clasificación por niveles en esta capa se tomo el bajo nivel como interfaz de usuario.

3.1.1 Patrones de Alto Nivel

3.1.1.1 Patrón MVC (Modelo Vista Controlador)

En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 5 Patrón MVC

Nombre del patrón	MVC
Descripción	El patrón MVC divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada.
Estructura	El patrón arquitectónico MVC divide una aplicación interactiva en tres componentes: • El modelo, que contiene la funcionalidad principal del sistema y los datos • La vista, que se encarga de mostrar información al usuario • Los controladores, que manejan las entradas del usuario La unión de las vistas y los controladores dan lugar a la IU. Un mecanismo que propaga los cambios asegura que lo que muestra la IU corresponde con el modelo, este mecanismo de propagación de cambios es lo que se conoce comúnmente como observador [21]. La separación del modelo de la vista y el controlador nos permite tener varias vistas para un solo modelo. Si el usuario cambia el modelo a través del controlador de una vista, todas las otras vistas que dependen de estos datos del modelo deberían reflejar los cambios. Las vistas en uso reciben los nuevos datos del modelo y actualizan la información mostrada. Este mecanismo de propagación de los cambios es la base del patrón observador. Existen una serie de patrones que rigen de alguna manera el funcionamiento de MVC, estos según [44], son: la estrategia, para poder cambiar las vistas y los controladores en tiempo real, el compuesto, para tratar de igual manera a las vistas simples que a las compuestas, y el observador, para establecer el mecanismo de propagación de los cambios necesarios para realizar la comunicación entre la IU, es decir, la vista y el controlador con el modelo
Ventajas	• Soporte de vistas múltiples. Dado que esta se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación Web pueden utilizar el mismo modelo de objetos, mostrado de maneras diferentes. • Adaptación al cambio. Los requerimientos de interfaz de usuario tienden a

	cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos como teléfonos celulares o PDAs. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo. Este patrón sentó las bases para especializaciones, tales como controlador de página y controlador frontal.
Desventajas	• Complejidad. El patrón introduce nuevos niveles de indirección y por lo tanto aumenta ligeramente la complejidad de la solución. También profundiza la orientación a eventos del código de la interfaz de usuario, que puede llegar a ser difícil de depurar. • Costo de actualizaciones frecuentes. Desacoplar el modelo de la vista no significa que los desarrolladores del modelo puedan ignorar la naturaleza de las vistas. Si el modelo experimenta cambios frecuentes, por ejemplo, podría desbordar las vistas con una lluvia de requerimientos de actualización
Cuando usarlo	Fowler [79] hace el análisis de cuando separar el modelo de la presentación, es claro que para aplicaciones sencillas que no impliquen una lógica de negocio cambiante o interfaces cambiantes no sería necesario aplicarlo; pero en cambio si se tiene una aplicación de complejidad media o alta será mejor hacer uso de esta herramienta por las ventajas que se mencionaron antes. En la ingeniería de software en general y en el desarrollo de aplicaciones Web en particular siempre se debe tener presente el concepto de cambio. En algún momento del desarrollo habrá que cambiar las funcionalidades o los requerimientos de esta. Cuando se deba extender la funcionalidad de una aplicación Web, se tendrá que modificar los menús y la navegación en general para acceder a esas nuevas funcionalidades. Por ejemplo si se tiene un sistema que deberá ser portado a otra plataforma con un estándar de UI diferente. El plantearse una serie de cuestiones puede ser bueno para vislumbrar si es necesario o no aplicar el patrón: • La misma información se representa de manera diferente, como barras o gráficos. • La representación y el comportamiento de la aplicación deben reflejar los cambios en los datos inmediatamente. • Los cambios en la UI deben ser fáciles de producir, incluso en tiempo real. Se debe poder soportar varios tipos de IUs y que esto sea fácil de realizar. La separación de vista y controlador es menos importante según Fowler [79], quien recomienda hacer esa separación si es realmente útil; sin embargo este principio es fundamental en aplicaciones Web.

Impacto de usabilidad

Este patrón tiene un impacto de usabilidad alto positivamente en la propiedad ***Consistencia: Visual, Funcional y Evolutiva,*** ya que al separar el modelo de los datos de la vista, ayuda a la consistencia a través de vistas múltiples, garantizando la consistencia visual, funcional y a la vez evolutiva, cuando se tenga una nueva versión de la aplicación, por ejemplo. Sobre la propiedad ***Explicito control de usuario*** tiene el mismo impacto, debido a que la estructura que se maneja facilitará realizar cambios que permitan dar mayor control al usuario, como adicionar controles o comandos.

El patrón MVC tiene un impacto de usabilidad medio positivamente en la propiedad ***Realimentación*** ya que la estructura del patrón permite mantener a los usuarios informados acerca de sus solicitudes y del estado del sistema. Sobre la propiedad ***Gestionando el Error*** también tiene un impacto de usabilidad medio positivamente debido a que el poder separar el modelo de las vistas y además soportar múltiples vistas hará mucho más sencillo controlar las excepciones de la aplicación y mostrar la respectiva información a diferentes tipos de usuarios, aunque ello no garantiza la no aparición de estos.

Tiene un impacto de usabilidad medio positivamente en las propiedades ***Mapeo natural, Accesibilidad y Adaptabilidad,*** junto con sus características particulares, debido a que el poder establecer múltiples vistas para los mismos datos ayuda a que el usuario pueda intuir el comportamiento del sistema, así mismo el tener el modelo separado de las vistas y el controlador facilita realizar cambios a las vistas de una manera mas sencilla, permite expresar la interfaz en términos del lenguaje de los usuarios, facilita la navegación, ayuda a adaptar la vista a personas con diversos tipo de discapacidades, a personalizar la aplicación ya sea al lenguaje moneda del usuario y a que este recuerde los detalles de la interfaz a través de las diferentes vistas. También el tener separado el modelo de los controladores permitirá que la aplicación pueda ser accedida por diferentes clases de dispositivos de entrada.

Tiene un impacto de usabilidad bajo positivamente sobre las propiedades ***Ayuda u orientación y Minimizar la carga cognitiva*** debido a que el tener las vistas separadas puede llegar a influir en una mejor orientación y organización de la información.

En la siguiente tabla se muestra el impacto que tiene el patrón por cada propiedad.

Tabla 6 Impacto de Usabilidad- Patrón MVC

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		M+
Gestionando El Error		M+
Consistencia	**Visual**	A+
	Funcional	A+
	Evolutiva	A+
Ayuda u orientación		B+
Minimizar la carga cognitiva		B+
Explícito control de usuario		M+
Mapeo natural	**Predecible**	M+
	Lenguaje de los usuarios	M+
	Facilidad de navegación	M+
Accesibilidad	**Discapacidades**	M+
	Multicanal	M+
	Internacionalización	M+
Adaptabilidad	**Experiencia de usuario**	M+
	Personalización	M+
	Facilidad de Recordar	M+

3.1.1.2 Patrón PAC (presentación – abstracción - control)

En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 7 Patrón PAC

Nombre del patrón	PAC
Descripción	El patrón PAC define una estructura de la arquitectura de una aplicación interactiva con el manejo de agentes[15] en jerarquías que cooperan entre sí. Cada agente es especializado para una tarea específica, y el conjunto de agentes proporcionan la funcionalidad del sistema.

[15] Un agente se considera como un sistema de software capaz de percibir la información sobre el ambiente, comunicarse con otros agentes por medio de la transmisión de mensajes y efectúa las acciones para cambiar este ambiente según sus metas, planes e intenciones.

	Las aplicaciones interactivas son propensas a cambiar con el tiempo, la utilización de este patrón permite adicionar agentes a la arquitectura sin alterar la aplicación en general.
Estructura	Este patrón se divide en tres componentes: • Presentación: este componente es el encargado de mostrar la información al usuario. • Abstracción: es donde se guarda la información. • Control: coordina la comunicación entre los componentes y otros agentes Cada agente consta de los componentes de presentación, abstracción y control. • El componente de presentación proporciona el comportamiento visible del agente PAC. • El componente de abstracción mantiene el modelo de datos que es la base del agente, y proporciona la funcionalidad que opera sobre estos datos. • El componente de control une el componente de presentación y de abstracción, y proporciona funcionalidad que permite al agente comunicarse con otros agentes PAC
Ventajas	• Separación de intereses: Diferentes unidades en el dominio de la aplicación son representados por agentes independientes de los otros agentes. • Soporta cambios y extensiones: Los cambios dentro de los componentes de presentación o de abstracción de un agente PAC no afectan a otros agentes en el sistema; se pueden integrar varios agentes a la arquitectura. • Soporte para multitareas: Permite distribuir los agentes en diferentes hilos, procesos, etc.
Desventajas	• Complejidad: La implementación de cada concepto semántico dentro de una aplicación como su propio agente PAC puede causar una estructura de sistema compleja. • Eficiencia: El aumento en la comunicación entre agentes PAC puede afectar la eficiencia del sistema • Aplicabilidad: El agente PAC puede ocasionar una estructura de grano fino compleja que es difícil de mantener. Por ejemplo, un editor gráfico en el cual cada objeto individual en un documento es representado por su propio agente PAC. • La facilidad de mantenimiento es otro problema que presenta, debido a que los componentes de control son dependientes
Cuando usarlo	Se utiliza cuando se quiere realizar una aplicación con jerarquía de agentes, separar la funcionalidad principal de la interacción hombre-máquina. Cuando los sistemas ofrecen múltiples vistas del mismo concepto semántico, permitiendo crear agentes para coordinar estas vistas. Por ejemplo cuando un usuario quiere ver los resultados de una votación podría seleccionar un gráfico de barras o un gráfico en torta. Muchas aplicaciones interactivas, son multiusuarios, para esto se utiliza los beneficios del multihilos; el agente PAC puede ser implementado como un objeto activo que vive en su propio hilo de control PAC es sobre todo aplicable a los sistemas que consisten en varios subsistemas independientes

Impacto de Usabilidad

Teniendo en cuenta que las aplicaciones interactivas son propensas a cambiar con el tiempo, este patrón tiene un impacto de usabilidad alto positivamente en la propiedad ***Adaptabilidad: Personalización*** debido a que la jerarquía de agentes PAC del sistema es dinámica, lo cual permite adicionar nuevos agentes a la jerarquía en el nivel inferior, a medida que la aplicación así lo requiera. Además el que las modalidades de interacción sean similares, hace que la interacción hombre máquina del producto sea homogénea, teniendo un impacto de usabilidad alto positivamente en las propiedades: ***Consistencia: Visual, Funcional y Evolutiva.***

Este patrón tiene un impacto de usabilidad alto positivamente en la propiedad ***Accesibilidad: Multicanal*** a través de la utilización del componente de control ya que este proporciona funcionalidad que permite al agente comunicarse con otros agentes PAC. En este caso una interfaz a un dispositivo móvil podría implementarse como un agente. También tiene el mismo impacto de usabilidad en la propiedad ***Accesibilidad: Internacionalización*** debido a que es posible adecuar la aplicación a las diferentes monedas y lenguas, ya que los cambios dentro de los componentes de presentación o de abstracción de un agente PAC no afectan a otros agentes en el sistema.

El PAC tiene un impacto de usabilidad medio positivamente en las propiedades ***Mapeo natural: Facilidad de navegación y Adaptabilidad: Experiencia de usuario,*** porque la ventaja de extensión y cambio que presenta el patrón facilita la adaptación de la aplicación a los distintos tipos de usuarios, expertos o inexpertos, además de permitir una intuitiva navegación por el sistema como consecuencia de la buena organización de la presentación que pueda realizarse.

El tener agentes especializados en una tarea específica, cada uno con un componente de presentación, influirá en que se pueda tener una mejor estructura de presentación de la información, con una mejor ayuda al usuario, teniendo un impacto de usabilidad bajo positivamente en las propiedades ***Ayuda u orientación*** *y* ***Minimizar la carga cognitiva.*** Tendrá el mismo impacto en la propiedad **Accesibilidad: Discapacidades**, debido a que la independencia que hay entre los agentes permite que se pueda dar soporte a los usuarios con diversa discapacidades. También esto puede afectar positivamente la propiedad ***Explicito control de usuario***, al poder adicionar controles a la aplicación si así se requiere. Sobre las propiedades ***Mapeo natural: Predecible, Adaptabilidad: Facilidad de recordar*** el sistema**,** tendrá también un impacto de usabilidad bajo positivamente debido a la consistencia que puede presentarse en la diferentes presentaciones de los agentes.
Sobre las propiedades de ***Realimentación y Gestionando el error*** tiene un impacto de usabilidad bajo negativamente debido a que las jerarquías de agentes muy profundas que pueden llegar a presentarse, pueden incrementar el tiempo de respuesta al usuario y hacer más difícil el manejo de errores, requiriéndose buenas estrategias para gestionarlos.
No tiene influencia sobre las propiedades ***Mapeo natural: Lenguaje de los usuarios***.
En la tabla 8 se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de una manera resumida:

Tabla 8 Impacto de Usabilidad- Patrón PAC

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		B-
Gestionando El Error		B-
Consistencia	Visual	A+
	Funcional	A+
	Evolutiva	A+
Ayuda u orientación		B+
Minimizar la carga cognitiva		B+
Explícito control de Usuario		B+
Mapeo natural	Predecible	B+
	Lenguaje de los usuarios	NI
	Facilidad de navegación	M+
Accesibilidad	Discapacidades	B+
	Multicanal	A+
	Internacionalización	A+
Adaptabilidad	Experiencia de usuario	M+
	Personalización	A+
	Facilidad de Recordar	B+

3.1.2 Patrones de Nivel Intermedio

3.1.2.1 Patrón Controlador de Página (Page Controller)

En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 9 Patrón Controlador de Página

Nombre del patrón	Controlador de Página
Descripción	Un controlador por acción o pantalla en una aplicación Web. Este patrón separa la capa de presentación: por un lado se tendrá lo que son estrictamente las *vistas* (páginas Web) y por el otro se tendrá los *controladores* de esas vistas (los que dicen que hay que hacer en respuesta a tal evento). El controlador de Página recibe una solicitud de la página, extrae toda la información pertinente, invoca las acciones solicitadas al modelo, y determinar la correcta vista para desplegar el resultado; La vista, a su vez, depende del modelo para la recuperación de datos que se mostrarán
Estructura	Vista: Pueden ser páginas Web que despliegan cualquier contenido. Modelo: Actúa como el controlador para cada página en el sitio Web
Ventajas	• Simplicidad: Debido a que cada página Web es manejada por un controlador específico, los controladores tienden a permanecer simples. • El aumento de la reutilización: La creación de una clase base controladora reduce la duplicación de código y permite reutilizar código común a través de los controladores de página. • Facilidad de Expansión**:** Se puede ampliar un controlador de página de una manera muy sencilla mediante el uso de las clases ayuda. Si la lógica interna del controlador llega a ser demasiada compleja, se puede delegar algo de la lógica a las clases ayuda (helper). • Responsabilidades de los desarrolladores: El uso de una clase controlador de página ayuda a separar las responsabilidades entre los miembros del equipo de desarrollo. El desarrollador del controlador debe estar familiarizado con el modelo de dominio y la lógica de negocio implementada para la aplicación. El diseñador de la vista, por el contrario, puede centrarse en el estilo de la presentación
Desventajas	• La limitación fundamental del controlador de página es que crea un controlador por cada página Web. Esto funciona bien para aplicaciones con un conjunto estático de páginas y una ruta simple de navegación; pero algunas aplicaciones más complejas requieren configuración dinámica de las páginas y de los mapas de navegación. Difundir esta lógica a través de muchos controladores de página hace la aplicación difícil de mantener. • El uso de la herencia por sí sola para reutilizar funcionalidad común puede dar lugar a una jerarquía de herencia inflexible. • Debido a que el controlador de página depende de los detalles o especificaciones del framework de aplicación Web (por ejemplo, los strings de consulta y cabeceras HTTP), no se puede instanciar y probar las clases del controlador fuera del framework Web. Si quiere ejecutar un conjunto de pruebas sobre la clase controlador, se tendrá que iniciar el servidor de aplicaciones Web para cada caso de prueba. Este tipo de pruebas es a la vez lento y propenso a errores
Cuando usarlo	El controlador de página se utiliza bien en un sitio dónde la mayoría de la lógica del controlador es bastante simple, además ya que cada controlador de página sólo se ocupa de una sola página, este patrón es especialmente adecuado para aplicaciones Web con navegación sencilla

Impacto de usabilidad

El tener un controlador por página o acción puede ayudar a que se tenga un buen manejo de errores, ya que se contará con un controlador específico para cada vista que se encargue de validar, por ejemplo: los campos, la entrada de datos, selección correcta de un evento, etc., en la vista; ayudando entonces a prevenir posibles errores que puedan aparecer, afectando con un impacto de usabilidad alto positivamente la propiedad ***Gestionando el error***

El tener las vistas separadas del controlador facilitará la realización de cambios a las vistas de la aplicación Web, ya que solo se tendrá que modificar el código de una vista determinada, sin afectar el código del controlador, teniendo un impacto de usabilidad alto positivamente en la propiedad ***Adaptabilidad: Experiencia de usuario, y Personalización.***

Este patrón tiene un impacto de usabilidad medio positivamente en las propiedades: ***Consistencia: Visual, Funcional***, debido a que las diferentes vistas permitidas, cada una con un controlador especifico, pueden hacer que la presentación, y la funcionalidad sea uniforme, favoreciendo además la propiedad **Adaptabilidad: Facilidad de recordar el sistema** con el mismo impacto de usabilidad.

Sobre la propiedad ***Consistencia: Evolutiva*** tiene un impacto de usabilidad medio positivamente, debido a que al realizar modificaciones en una vista solo tendrá que cambiarse el controlador específico de esta, sin llegar a afectarse las demás vistas.

En la propiedad ***Explicito control de usuario*** tiene un impacto de usabilidad medio positivamente debido al tiempo de respuesta favorable que pueda darse en respuesta a las solicitudes del usuario, al tener un controlador por página o acción y al estar separado de las vistas***.*** Sobre la propiedad ***Minimizar la carga cognitiva*** tiene un impacto de usabilidad medio positivamente debido a que el tener varias vistas ayuda a que la información sea distribuida de una manera organizada, sin tener que sobrecargar todos los datos en una sola vista. Esto además influye con igual impacto, la propiedad **Mapeo natural: Facilidad de navegación** y afecta con un impacto de usabilidad bajo positivamente la propiedad ***Ayuda u orientación***.

El tener un controlador por página también puede ayudar a que la ***Realimentación*** al usuario sea rápida y efectiva, teniendo entonces un impacto de usabilidad bajo positivamente en esta propiedad. Sobre la propiedad de ***Accesibilidad*** también puede tener un impacto de usabilidad bajo positivamente debido a la facilidad de cambios que este patrón tiene como ventaja, dar soporte a usuarios discapacitados, personalizar la aplicación y permitiendo que ésta sea accedida por diferentes dispositivos.

Este patrón no tiene influencia sobre las propiedades ***Mapeo natural: Predecible y Lenguaje de los usuarios.***

En la tabla 10 se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 10 Impacto de Usabilidad- Patrón Controlador de Página (Page Controller)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		B+
Gestionando El Error		A+
Consistencia	Visual	M+
	Funcional	M+
	Evolutiva	M+
Ayuda u orientación		B+
Minimizar la carga cognitiva		M+
Explícito control de usuario		M+
Mapeo natural	Predecible	NI
	Lenguaje de los usuarios	NI
	Facilidad de navegación	M+
Accesibilidad	Discapacidades	B+
	Multicanal	B+
	Internacionalización	B+
Adaptabilidad	Experiencia de usuario	A+
	Personalización	A+
	Facilidad de Recordar	M+

3.1.2.2 Patrón Controlador Principal (Front Controller)

En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 11 Patrón Controlador principal

Nombre del patrón	Controlador Principal
Descripción	Un controlador principal dirige todas las solicitudes de una aplicación Web a través de un único controlador, el controlador normalmente se estructura en dos partes: un manejador y una jerarquía de comandos
Estructura	Manejador: Es el objeto que realmente recibe las peticiones del servidor Web, obtiene la información de la URL y de la petición para decidir qué tipo de acción comenzar y entonces delega un comando para llevar a cabo la acción. Jerarquía de comandos: Son también parte del controlador, estos representan las acciones específicas. Representar los comandos como objetos individuales permite al controlador interactuar con todos los comandos de una manera genérica, en lugar de invocar métodos específicos de una clase comando común. Después de que el objeto comando completa la acción, el comando elige cual vista usar para devolver a la página
Ventajas	• Control centralizado. Si la lógica común se repite en diferentes puntos de vista en el sistema, es necesario centralizar esta lógica para reducir la cantidad de código duplicado. La eliminación de la duplicación de código es fundamental para mejorar la facilidad de mantenimiento del sistema. Además la recuperación de los datos es mejor manejarla en un solo lugar, en vez de que cada vista recupere los datos evitando tener que duplicar el código de acceso a la base de datos. • Facilidad de configuración: Sólo un controlador principal necesita ser configurado en el servidor Web, el manejador hace el resto de la solicitud. Esto simplifica la configuración del servidor Web. • Debido a que se crea nuevos objetos comando con cada solicitud, no es necesario preocuparse por hacer programación multihilos
Desventajas	• Consideraciones de rendimiento. El rendimiento podría ser muy bajo si el manejador para elegir el tipo de comando que realizara la solicitud debe hacer la consulta en una base de datos o en un documento XML. • El aumento de la complejidad. El controlador principal es más complicado de desarrollar que el controlador de página
	Se debe usar para aplicaciones muy complejas y cuando el uso del patrón:

Cuando usarlo	controlador de página ha llevado a tener una lógica complicada y difícil de manejar

Impacto de usabilidad

Al tener un único controlador con un buen manejo de eventos a través de una excelente delegación de comandos puede contribuir con la propiedad ***Gestionando el error***, con un impacto de usabilidad medio positivamente. Aunque debe tenerse cuidado en que esta gestión se haga de la manera adecuada para no tener implicaciones contrarias.

Debido a que las vistas están separadas del controlador, la realización de cambios a la presentación de la aplicación Web será más sencilla, ya que solo se tendrá que modificar el código de una vista determinada, sin afectar el código del controlador, teniendo entonces un impacto de usabilidad medio positivamente en las propiedades de: ***Adaptabilidad: Experiencia de usuario, Personalización y Facilidad de recordar el sistema.*** Además debido a que simplemente hay un controlador, el cual puede fácilmente reforzar su comportamiento en tiempo de ejecución con decoradores [80]. Se puede llegar a tener decoradores para autenticación, internacionalización, vistas particulares para ciertos usuarios, etc., y dar la posibilidad de acceder a la aplicación por diversos tipos de dispositivos, de esta manera se tiene un impacto bajo positivamente en la propiedad ***Accesibilidad: Discapacidades, multicanal, e Internacionalización***.

Sobre la propiedad ***Consistencia: Visual y Funcional,*** tiene un impacto de usabilidad bajo positivamente debido a que el tener un controlador puede hacer que los eventos se manejen de igual manera en toda la aplicación o familia de productos, llegando a tener una estructura y presentación similares.

Sobre la propiedad ***Minimizar la carga cognitiva*** puede tener un impacto de usabilidad bajo positivamente debido a que el estar separadas la vistas del controlador facilita la adición de controles que disminuyan la carga cognitiva. Además ayuda a la **Realimentación** al usuario, teniendo el mismo impacto de usabilidad.

En la propiedad ***Explicito control de usuario*** puede tener un impacto de usabilidad medio negativamente, que puede ser bajo negativamente si el tiempo de respuesta se incrementa notablemente.

Este patrón tiene un impacto de usabilidad bajo negativamente en la propiedad: ***Consistencia Evolutiva***, debido a que tener un único controlador para todas las vistas hace que al realizar cambios en este, se afecte todas las vistas, no solo una.
Se tiene un impacto de usabilidad bajo negativamente en la propiedad ***Mapeo natural: Facilidad de navegación,*** debido a que el tener un controlador para varias páginas puede tener implicaciones en el rendimiento y en el mantenimiento del sistema dificultando la navegación de la información.

Este patrón no tiene influencia sobre las propiedades ***Ayuda u orientación, Mapeo natural: Predecible y Lenguaje de los usuarios.***

En la tabla 12 se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 12 Impacto de Usabilidad- Patrón Controlador principal (Front Controller)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		B+
Gestionando El Error		M+
Consistencia	Visual	B+
	Funcional	B+
	Evolutiva	B-
Ayuda u orientación		NI
Minimizar la carga cognitiva		B+
Explícito control de Usuario		M-
Mapeo natural	Predecible	NI
	Lenguaje de los usuarios	NI
	Facilidad de navegación	B-
Accesibilidad	Discapacidades	B+
	Multicanal	B+
	Internacionalización	B+
Adaptabilidad	Experiencia de usuario	M+
	Personalización	M+
	Facilidad de Recordar	M+

Fortalezas que pueden persuadir a utilizar Controlador principal en contraposición a Controlador de página:

- Una implementación común del controlador de página implica la creación de una clase base para el comportamiento común entre páginas individuales. Sin embargo, con el tiempo estas clases base pueden crecer con código que no es común a todas las páginas, debido a que tienen funciones especificas.
- Para evitar la adición excesiva de lógica condicional en la clase base, se podría crear una jerarquía de herencia más profunda para eliminar la lógica condicional. Por ejemplo, en una aplicación que tiene tres áreas funcionales, tal vez sería útil disponer de una única clase base común que tiene la funcionalidad de la aplicación. También podría haber otra clase para cada área funcional, que hereda de la clase base general de la aplicación. Este tipo de estructura, a primera vista, es sencilla, pero a menudo lleva a un muy frágil diseño e implementación y a tener demasiado código.
- La solución del controlador de página describe un solo objeto por página. Esta solución no sirve cuando se necesita controlar o coordinar el procesamiento a través de múltiples páginas. Por ejemplo, si se tiene una aplicación Web con navegación compleja, la cual es almacenada en XML. Cuando llega una solicitud, la aplicación deberá buscar a dónde ir, con base en su estado actual.
- Debido a que el controlador de página está implementado con un solo objeto por página, es difícil de aplicar constantemente una acción a través de todas las páginas en una aplicación Web. La seguridad, por ejemplo, es mejor implementarla de una manera coordinada.
- La asociación de la URL para el objeto controlador particular puede ser una limitante para las aplicaciones Web. Por ejemplo, si se tiene un sitio con un "Wizard[16]" como interfaz para la recopilación de información. Cuando se implementa con controlador de página, las páginas opcionales tendrían que ser implementadas con lógica condicional en la clase base para seleccionar la página siguiente.

[16] Este wizard consiste en una serie de páginas obligatorias y opcionales basadas en las entradas de los usuarios

3.1.2.3 Patrón Controlador de la Aplicación (Application Controller)

En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 13 Patrón Controlador de la Aplicación

Nombre del patrón	Controlador de la Aplicación
Descripción	Muchos diseños de interfaz de usuario separan los objetos de la presentación de los objetos del dominio con una capa intermedia llamada Controlador de la aplicación. El propósito de Controlador de la aplicación es ocuparse del flujo de una aplicación, decidiendo qué pantallas deben aparecer y en qué orden. Este patrón puede implementarse en la capa de presentación o como capa intermedia entre la de presentación y dominio, puede ser escrito para ser independiente de cualquier presentación particular lo cual permitirá reutilizar las presentaciones; trabaja bien si se tienen presentaciones diferentes con el mismo flujo básico y navegación, aunque a menudo es mejor dar un flujo diferente a las presentaciones
Estructura	Capa de presentación: Donde se tiene las vistas y sus controladores. Capa de dominio: Se encuentra la parte lógica de la aplicación.
Ventajas	• Fácil substitución de interfaz del usuario. • Fácil substitución de lógica de negocio. • Reutilización de interfaces del usuario. • Reutilización de funcionalidad de la lógica del negocio
Desventajas	• Menor desempeño con respecto al rendimiento de la aplicación. • Requiere gran cantidad de documentación en el diseño para grandes sistemas
Cuando usarlo	No todos los sistemas necesitan un Controlador de la aplicación. Este patrón es útil si el sistema tiene mucha lógica sobre el orden de pantallas y la navegación entre ellas. También cuando no se consigue un mapeo simple entre páginas y las acciones en el dominio. Pero si el flujo y la navegación entre pantallas son simples, no habrá necesidad de aplicar el patrón. Una pregunta que se puede hacer para saber si debe o no aplicar el patrón es ¿Debe hacer cambios similares en muchos lugares diferentes cuando los flujos de la aplicación cambian? Algunas aplicaciones tienen una cantidad significativa de lógica sobre las pantallas, las cuales pueden invocarse en ciertos momentos en una aplicación. Un ejemplo de este patrón es el estilo Wizard de interacción, donde el usuario navega a través de una serie de pantallas en cierto orden. En otros casos podrían ser pantallas que son invocadas en un cierto orden bajo ciertas condiciones, u opciones entre pantallas diferentes que dependen de una entrada más temprana. Hasta cierto punto varios controladores de entrada en MVC pueden tomar algunas de estas decisiones, pero cuando una aplicación es muy compleja puede llevar a duplicar código ya que se tendría varios controladores para diferentes pantallas los cuales necesitan saber qué hacer en determinada situación. Se puede quitar esta duplicación poniendo toda la lógica de flujo en un Controlador de la aplicación. Los controladores de entrada pedirán las ordenes apropiadas al Controlador de la aplicación para ejecutar el modelo y usar la vista correcta dependiendo del contexto de la aplicación

Impacto de usabilidad

Debido a que este patrón se ocupa del flujo de la navegación tendrá un impacto alto positivamente en la propiedades de usabilidad ***Mapeo Natural***: ***Facilidad de navegación, Predecible*** *y* ***Explicito control de usuario***, al dar la sensación al usuario de tener el control de la aplicación. De esta forma también se ***Minimizará*** la ***carga cognitiva***, teniendo un impacto de usabilidad alto positivamente sobre esta propiedad. Todo lo anterior contribuirá a que el usuario pueda tener la ***Facilidad de recordar*** como de navegar en la aplicación Web.

Este patrón tiene un impacto de usabilidad alto en la propiedades ***Realimentación, Gestionado el error y Ayuda u orientación***, debido a que el patrón muestra las diferentes vistas en un orden natural y lógico, lo que da un alto grado de información al usuario de que hacer, sin tener que pensar que vista activar, la aplicación lo hace, previniendo entonces la aparición de errores y mejorando altamente la ayuda al usuario. Además se puede hacer uso de la voz, para informar por ejemplo a una persona con discapacidades auditivas, que vista sigue, ayudando a la propiedad ***Accesibilidad: Discapacidades***, con un impacto de usabilidad medio positivamente.

Además al tener una navegación bien definida la aplicación tendrá una ***Consistencia Funcional y Visual,*** teniendo un impacto de usabilidad medio positivamente.
En las propiedades ***Consistencia Evolutiva y Accesibilidad: Multicanal e Internacionalización,*** tienen un impacto de usabilidad bajo positivamente, debido a que el patrón es un intermediario que puede facilitar los cambios y dar soporte para varios dispositivos de entrada.

Este patrón no tiene influencia sobre las propiedades ***Adaptabilidad: Personalización y Experiencia de usuario.***
En la tabla 14 se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 14 Impacto de Usabilidad Patrón Controlador de la Aplicación (Aplication Controller)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		A+
Gestionando El Error		A+
Consistencia	Visual	M+
	Funcional	M+
	Evolutiva	B+
Ayuda u orientación		A+
Minimizar la carga cognitiva		A+
Explícito control de Usuario		A+
Mapeo natural	Predecible	A+
	Lenguaje de los usuarios	NI
	Facilidad de navegación	A+
Accesibilidad	Discapacidades	M+
	Multicanal	B+
	Internacionalización	B+
Adaptabilidad	Experiencia de usuario	NI
	Personalización	NI
	Facilidad de Recordar	A+

3.1.2.4 Patrón Plantilla de Vista (Template View)

En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 15 Patrón Plantilla de Vista

Nombre del patrón	Plantilla de Vista
Descripción	Este patrón es usado para manejar la lógica de las vistas de la aplicación. Una Plantilla de Vista es un documento HTML que ingresa marcadores especiales en una página HTML. Estos marcadores son sustituidos, manipulados o evaluados para producir un documento de salida. El propósito de Plantilla de Vista es aislar el HTML del código del lenguaje de programación en una aplicación Web. Esto lo

	hace usando marcadores de texto especiales en el cuerpo del texto, donde se pone lógica condicional que llame a los "scripts[17]" que contiene el lenguaje de programación. Actualmente muchas herramientas usan Plantilla de Vista; esto hace que se tengan diferentes alternativas para utilizarlo
Estructura	Marcadores: Son el puente entre las plantillas y el código. Los marcadores más simples son aquellos que reciben alguna información del resto del sistema y ponen en el lugar correcto la página. Etiquetas: Trabajan bien con editores WYSIWYG porque comprenden que algo entre los corchetes angulares "(<>)" es especial y lo ignoran o lo tratan diferente. Objeto ayudante: Este auxiliador tiene toda la lógica de programación real. La página sólo tiene las llamadas, simplificando la página y haciendo más puro la Plantilla de Vista.
Ventajas	• Es un patrón que es soportado por diferentes herramientas. • No hay un lenguaje que aprender porque utiliza editores. • Soporta editores de HTML no orientados a la programación
Desventajas	• Usa gran cantidad de scripts para desarrollar una página y esto hace que baje el mantenimiento. • Mezcla diferentes capas. • La Lógica en la página es difícil de estructurar. • Hace Duplicación de código. • Utiliza la Lógica condicional para realizar la página.
Cuando usarlo	El patrón Plantilla de vista se usa cuando se quiere presentar al usuario un conjunto de plantillas para una vista sencilla y fácil de navegar proporcionándole a cada usuario de la aplicación su propio fichero de trabajo. Cuando la lógica se encuentra en el servidor.

Impacto de usabilidad

Este patrón tiene un impacto de usabilidad medio positivamente en la propiedad ***Gestionando el error*** debido a que el tener separado el código de la vista del de lenguaje de programación puede hacer que se tengan menos errores en la aplicación. Además esto también facilita la evolución de la aplicación, teniendo un impacto de usabilidad bajo positivamente en la propiedad **Consistencia: Evolutiva.**

La duplicación de código que puede presentarse dificulta la realización de cambios en la vistas de la aplicación, afectando la propiedad ***Adaptabilidad: Experiencia del usuario y Personalización***, con un impacto de usabilidad bajo negativamente.

Este patrón no tiene impacto sobre las propiedades ***Realimentación, Consistencia Visual, Funcional, Ayuda u orientación, Minimizar la carga cognitiva, Explicito control de usuario, Mapeo natural: Predecible, Lenguaje de los usuarios, Accesibilidad: Discapacidades, Multicanal, Internacionalización, y Adaptabilidad: Facilidad de recordar el sistema***.

En la tabla 16 se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 16 Impacto de Usabilidad Plantilla de Vista (Template View)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		NI
Gestionando El Error		M+
Consistencia	Visual	NI
	Funcional	NI
	Evolutiva	B+

[17] Script: Conjunto de Instrucciones.

Ayuda u orientación		NI
Minimizar la carga cognitiva		NI
Explícito control de Usuario		NI
Mapeo natural	**Predecible**	NI
	Lenguaje de los usuarios	NI
	Facilidad de navegación	NI
Accesibilidad	**Discapacidades**	NI
	Multicanal	NI
	Internacionalización	NI
Adaptabilidad	**Experiencia de usuario**	B-
	Personalización	B-
	Facilidad de Recordar	NI

3.1.2.5 Patrón Transformador de Vista (Transform View)

En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 17 Patrón Transformador de Vista

Nombre del patrón	Transformador de Vista
Descripción	El patrón Transformador de Vista es creado para transformar documentos XML dinámicamente. Se puede escribir un Transformador de Vista en cualquier lenguaje sin embargo, la opción dominante, por el momento, es XSLT. La noción básica del Transformador de Vista es escribir un programa que obtenga los datos del dominio y los convierta en HTML. Teniendo transformaciones separadas por cada tipo de entrada.
Estructura	Transformador: Es el encargado de la transformación, este posee un controlador que obtiene cada elemento de la entrada, encuentra la transformación adecuada y entonces llama al transformador. Uno típico puede colocar las reglas de la Vista en cualquier orden sin afectar el resultando de la salida.
Cuando usarlo	Este patrón se usa preferiblemente cuando se va a construir una vista en un documento XML, debido a que XSLT se ajusta naturalmente en un mundo de XML. También se puede utilizar en casi cualquier plataforma Web. Se puede usar el mismo XSLT para transformar XML creado de J2EE o .NET, lo que puede ayudar a tener una vista HTML común en los datos de diferentes fuentes. Sin embargo depende del ambiente en el cual el equipo trabaja la vista y el software que prefiera. Cuando se quiere múltiples apariencias para el mismo dato se utiliza el patrón dos fases para la vista que se analizará a continuación.

Impacto de usabilidad

Este patrón tiene un impacto de usabilidad medio positivamente en la propiedad ***Gestionando el error*** debido a que el trabajar con XSLT proporciona ayuda al usuario cuando sucede un error. También tiene el mismo impacto en la propiedad **Consistencia Visual y Evolutiva**, por que usar XML ayuda a tener una presentación uniforme y organizada, facilitando de esta manera la facilidad de cambios para una nueva versión, lo que puede ayudar a si mismo a tener una interfaz simple, sencilla no tan cargada de información, teniendo un impacto de usabilidad bajo positivamente en la propiedad **Minimizar la carga cognitiva.** Los puntos antes expuestos también favorecen la orientación del usuario en la interfaz, definiéndose un impacto de usabilidad medio positivamente en la propiedad **Ayuda u orientación**.

El tener separado el código de la vista del de lenguaje de programación, sin duplicación de código, facilita la realización de cambios en la aplicación, permitiendo personalizar la aplicación a los distintos tipos de usuarios, teniendo un impacto de usabilidad medio

positivamente en la propiedad ***Adaptabilidad: Personalización.*** Esto también influye, aunque con un impacto menor, sobre la propiedad ***Accesibilidad: Discapacidades, Multicanal e Internacionalización*** con un nivel bajo positivamente.

El tener una estructura bien organizada ayuda a que sea fácil navegar en la aplicación afectando la propiedad ***Mapeo natural: Facilidad de navegación***, con un impacto de usabilidad bajo positivamente; lo cual también ayuda al usuario a recordar la forma de interactuar con la aplicación, favoreciendo la propiedad ***Adaptabilidad: Facilidad de recordar***, con un impacto de usabilidad bajo positivamente.

Este patrón no tiene influencia sobre las propiedades ***Realimentación, Consistencia Funcional, Explícito control de usuario, Mapeo natural: Predecible, Lenguaje de los usuarios, Adaptabilidad: Experiencia de usuario.***
En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 18 Impacto de Usabilidad Patrón Transformador de Vista (Transform View)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		NI
Gestionando El Error		M+
Consistencia	Visual	M+
	Funcional	NI
	Evolutiva	M+
Ayuda u orientación		M+
Minimizar la carga cognitiva		B+
Explícito control de Usuario		NI
Mapeo natural	Predecible	NI
	Lenguaje de los usuarios	NI
	Facilidad de navegación	B+
Accesibilidad	Discapacidades	B+
	Multicanal	B+
	Internacionalización	B+
Adaptabilidad	Experiencia de usuario	NI
	Personalización	M+
	Facilidad de Recordar	B+

3.1.2.6 Patrón Dos Fases por Vista (Two Step View)

En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 19 Patrón Dos Fases por Vista

Nombre del patrón	Dos fases por Vista
Descripción	Con Plantilla de Vista o Transformador de Vista, se construye un módulo de vista por página Web. Con este patrón se tiene dos fases: un módulo de primera fase para la página y un módulo de segunda fase para la aplicación entera. En una aplicación en la que se necesita manejar multi apariencia se tiene una vista en un sólo paso para cada combinación de pantalla y apariencia. Así, diez pantallas y tres apariencias distintas tendrán treinta módulos de vista de fase únicos. Usado dos fases para la vista se puede obtener mejores resultados teniendo diez pantallas en la primera fase y tres apariencias en la segunda. Entre más pantallas y apariencias se tenga se aprovechará mejor el patrón.
Ventajas	• Cualquier cambio a la apariencia de la aplicación en la segunda fase es más fácil de hacer, ya que un cambio en la segunda fase afecta el sitio completo. • Pone la decisión de qué HTML usar en un solo lugar. Esto hace fácil realizar

	cambios globales al HTML ya que hay sólo un objeto para alterar cada pantalla de la aplicación. Claro, esa ventaja solo se consigue si la lógica de la presentación se puede manejar en un solo HTML, ya que funciona mejor con aplicaciones dónde pantallas diferentes usan el mismo diseño básico.
Desventajas	• Dos fases por Vista obliga a los programadores a que escriban los objetos del controlador. Teniendo que estar involucrados en cualquier cambio del diseño. • Al necesitar múltiples capas, se hace más difícil para los desarrolladores aprender a implementarlo, aunque una vez se ha utilizado no es difícil, y puede ayudar a reducir la duplicación de código. • Una variación en el tema de apariencias múltiples está al mantener segundas fases diferentes para los dispositivos diferentes, por ejemplo tener una segunda fase para un navegador y otra para un PDA. La limitación usual aquí es que ambas apariencias deben seguir la misma pantalla lógica, y para los dispositivos muy diferentes esto puede ser demasiado complejo
Cuando usarlo	Dos fases por Vista trabaja bien si se tiene una aplicación Web dónde sus servicios están usándose por múltiples clientes, por ejemplo dos aerolíneas que utilizan el mismo sistema de reserva. También se puede usar el patrón cuando se tienen dispositivos con salidas diferentes, en este caso habría que separar las segundas fases por ejemplo para un navegador de Web regular y para una palmtop. Si se tiene una aplicación Web con muchas páginas, se querrá dar una mirada consistente y organización al sitio. Si cada página aparece diferente, los usuarios finales se confundirán con la aplicación. También se querrá hacer cambios globales fácilmente a la apariencia del sitio, pero soluciones comunes que usan Plantilla de Vista o Transformador de Vista hacen esto difícil porque se reproducen a menudo las decisiones de presentación por múltiples páginas. Un cambio global en estos casos puede obligarle a que cambie varios archivos. El valor importante de Dos fases por Vista viene de la separación de primeras y segundas fases, además permite hacer cambios globales más fácilmente. Ayuda a pensar en dos situaciones: la multiapariencia en las aplicaciones Web y única-apariencia a las aplicaciones de Web. Las aplicaciones Multiapariencia son difíciles de encontrar pero su uso está aumentando. En ellos la misma funcionalidad básica se proporciona a través de las organizaciones múltiples y cada organización tiene su propio y distinto “look and feel “

Impacto de usabilidad

Este patrón tiene un impacto de usabilidad alto positivamente en la propiedad ***Consistencia Visual,*** debido a la uniformidad entre las diferentes páginas de la aplicación, lo que ayuda también a la propiedad ***Adaptabilidad: Facilidad de recordar el sistema,*** con impacto de usabilidad medio positivamente. Tiene el mismo impacto en la propiedad ***Consistencia Funcional*** y un impacto bajo positivamente en la ***Consistencia Evolutiva***, debido a que no puede garantizarse la similar funcionalidad en todas las aplicaciones además al realizar cambios en la interfaz principal puede afectar las otras presentaciones particulares.

Sobre las propiedades ***Ayuda u orientación y Mapeo natural: Lenguaje de los usuarios,*** tiene un impacto de usabilidad bajo positivamente, debido a que el patrón permite adaptar la aplicación a las características de los usuarios. Lo que a su vez beneficia a las propiedades ***Accesibilidad: Discapacidades, Internacionalización y Adaptabilidad Experiencia del usuario***, con el mismo impacto de usabilidad.

En la propiedad ***Accesibilidad: Multicanal*** tiene un impacto de usabilidad bajo negativamente debido a que puede ser complejo tener la misma lógica principal para los diferentes tipos de dispositivos, tal como se mencionó en una de las desventajas del patrón.

Este patrón no tiene influencia sobre las propiedades ***Realimentación, Gestionando el error, Minimizar la carga cognitiva, Explicito control de usuario, Mapeo natural: Predecible y Facilidad de navegación, Adaptabilidad: Personalización.***

En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 20 Impacto de Usabilidad Dos Fases para la Vista (Two Step View)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		NI
Gestionando El Error		NI
Consistencia	Visual	A+
	Funcional	M+
	Evolutiva	B+
Ayuda u orientación		B+
Minimizar la carga cognitiva		NI
Explícito control de Usuario		NI
Mapeo natural	Predecible	NI
	Lenguaje de los usuarios	B+
	Facilidad de navegación	NI
Accesibilidad	Discapacidades	B+
	Multicanal	B-
	Internacionalización	B+
Adaptabilidad	Experiencia de usuario	B+
	Personalización	NI
	Facilidad de Recordar	M+

3.1.2.7 Patrones de Navegación

Patrones de Diseño Navegacional

Los patrones en esta categoría ayudan a organizar la estructura de navegación de una aplicación hipermedia de una manera clara y significante. Ellos dirigen los problemas recurrentes cuya solución determina el grado de éxito de las aplicaciones de hipermedia. No se preocupan del aspecto final de los elementos en una interfaz pero si de la organización de navegación a través de los componentes de la aplicación hipermedia.

3.1.2.7.1 Patrón Enlace como Relación entre Vistas (Link as a Relationship View)

En la siguiente tabla se da la descripción del patrón y se define cuando debe ser usado:

Tabla 21 Patrón Enlace como Relación entre Vistas

Nombre del patrón	Enlace como Relación entre Vistas
Descripción	Al igual que los nodos los enlaces son muy importantes en una aplicación Web debido a que permiten la navegación de la aplicación. Los enlaces representan una relación de interés entre dos o más unidades o páginas. El destino de un enlace puede ser calculado buscando los objetos involucrados en la relación y así obtener el nodo destino.
Cuando usarlo	Se utiliza cuando no se sabe de qué manera representar las relaciones y la navegación entre los componentes de una aplicación en una vista hipermedia

Impacto de usabilidad

Este patrón tiene un impacto de usabilidad alto positivamente en las propiedades ***Mapeo natural: Facilidad de navegación y Predecible*** debido a que los enlaces determinan la navegación de la aplicación Web, el hacer un buen diseño de estos tendrá como

consecuencia influencia en estas. Así como afectará positivamente la ***Consistencia Funcional.*** De otra parte al tener una navegación consistente hará más sencilla la interfaz teniendo un impacto de usabilidad alto positivamente en la propiedad ***Minimizar la carga cognitiva.***

Debido a que los enlaces brindan una buena información al usuario, guiándolo a través de la interfaz, las propiedades ***Realimentación y Ayuda u orientación***, tienen un impacto de usabilidad alto positivamente. Esto también ayuda a que el usuario cometa menos errores, teniendo un impacto de usabilidad bajo positivamente en la propiedad ***Gestionando el error***.

Este patrón tiene un impacto de usabilidad medio positivamente en las propiedades ***Consistencia: visual***, ***Explicito control de usuario y Accesibilidad: Discapacidades,*** debido a que el tener enlaces en las vistas ayuda a un mejor aspecto en la estructura de la presentación, además hace que el usuario pueda tener una mayor control de la aplicación, y a su vez favorece a usuarios con limitaciones.

Por otra parte el tener muchos enlaces, puede hacer que el usuario no recuerde fácilmente como navegar a través de la interfaz, teniendo entonces un impacto de usabilidad bajo negativamente en la propiedad ***Adaptabilidad: Facilidad de recordar.***

Este patrón no tiene influencia sobre las propiedades ***Consistencia Evolutiva, Mapeo natural: Lenguaje de los usuarios, Accesibilidad: Multicanal, Internacionalización, y Adaptabilidad: Experiencia de usuario y Personalización*****.**
En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 22 Impacto de Usabilidad Patrón Enlace como Relación entre Vistas (Link as a Relationship View)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		A+
Gestionando El Error		B+
Consistencia	Visual	M+
	Funcional	A+
	Evolutiva	NI
Ayuda u orientación		A+
Minimizar la carga cognitiva		A+
Explícito control de Usuario		M+
Mapeo natural	Predecible	A+
	Lenguaje de los usuarios	NI
	Facilidad de navegación	A+
Accesibilidad	Discapacidades	M+
	Multicanal	NI
	Internacionalización	NI
Adaptabilidad	Experiencia de usuario	NI
	Personalización	NI
	Facilidad de Recordar	B-

3.1.2.7.2 Patrón Observador de Navegación (Navigation Observer)

Nombre del patrón	Observador de Navegación
	Las aplicaciones Web deben registrar el estado de la navegación de una manera perceptible al usuario, este registro debe actualizarse automáticamente conforme la navegación progresa. Por lo tanto se debe definir la historia de los objetos es

Descripción	decir la información acerca de la navegación que interactúa con los nodos y los enlaces, esta debe hacerse para cada sesión desde que se inicia la aplicación. Diferentes historias deben crearse para las diferentes sesiones en un ambiente concurrente. Además la GUI con la que una historia particular será percibida debe ser definida separadamente y ser capaz de cambiar independientemente. Debe ser posible definir la historia de navegación de diferentes maneras, por ejemplo, una lista ordenada de nodos visitados, una lista ordenada de enlaces visitados, un gráfico coloreado, etc.
Cuando usarlo	Se aplica cuando no se tiene conocimiento de cómo crear un registro comprensible o distinguible del proceso de navegación.

Tabla 23 Patrón Observador de Navegación

Impacto de usabilidad

Este patrón tiene un impacto de usabilidad alto positivamente con influencia directa específicamente en las propiedades ***Adaptabilidad: Facilidad de recordar*** **el sistema** y ***Minimizar la carga cognitiva***, ya que con él se permitirá al usuario reconocer donde se encuentra y por donde ha navegado en la aplicación, el usuario podrá por ejemplo devolverse sin necesidad de memorizarse una url, ya que el sistema lo ha recordado por él. Esto también afecta positivamente la propiedad ***Mapeo natural: Facilidad de navegación.***

Tiene un impacto de usabilidad alto positivamente en la propiedad ***Ayuda u orientación*** debido a que el registrar el estado de la navegación orienta al usuario en el recorrido que ha realizado, además hace que el usuario sienta que la aplicación esta acorde con su manera de pensar, teniendo un impacto de usabilidad bajo positivamente en la propiedad ***Mapeo natural: Lenguaje de los usuarios***.

El poder informar al usuario donde está ubicado o por donde ha navegado, brinda una buena ***Realimentación*** al usuario, teniendo un impacto de usabilidad medio positivamente en esta propiedad, así como también previene que el usuario cometa errores, teniendo el mismo impacto en la propiedad ***Gestionando el error.***

El hecho que el usuario pueda reconocer con facilidad qué lugares ha visitado, mejorará su experiencia con la aplicación, teniendo un impacto de usabilidad bajo positivamente en la propiedad ***Adaptabilidad: Experiencia de usuario*** y ***Personalización.***

Este patrón no tiene influencia sobre las propiedades ***consistencia: visual, funcional, evolutiva, explicito control de usuario, mapeo natural: previsibilidad, y accesibilidad: discapacidades, multicanal e internacionalización.***

En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 24 Impacto de Usabilidad Patrón Observador de Navegación (Navigation Observer)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		M+
Gestionando El Error		M+
Consistencia	**Visual**	NI
	Funcional	NI
	Evolutiva	NI
Ayuda u orientación		A+
Minimizar la carga cognitiva		A+
Explícito control de Usuario		NI
Mapeo natural	**Predecible**	NI

	Lenguaje de los usuarios	B+
	Facilidad de navegación	A+
Accesibilidad	**Discapacidades**	NI
	Multicanal	NI
	Internacionalización	NI
Adaptabilidad	**Experiencia de usuario**	B+
	Personalización	B+
	Facilidad de Recordar	A+

3.1.2.7.3 Patrón Nodo como una Unidad única (Node as a single Unit)

En la siguiente tabla se da la descripción del patrón y se define cuando debe ser usado:

Tabla 25 Patrón Nodo como una Unidad única

Nombre del patrón	Nodo como una Unidad única
Descripción	Un nodo debe abarcar una "unidad" autónoma o autosuficiente de información que debe tener sentido para un conjunto de usuarios que realizan un conjunto de tareas en un dominio dado. El nodo debe enfocarse en un tema específico, Todos los datos que son pertinentes para esa unidad deben ser incluidos en el mismo nodo. No es bueno tener páginas Web muy largas, e incluir diferentes combinaciones de temas, que no sean pertinentes. Ni tampoco fragmentar un tema en varias páginas ya que esto hace la lectura y la impresión más difícil, porque el lector debe navegar de un fragmento al próximo para ver la "unidad" entera.
Cuando usarlo	Se utiliza cuando se debe decidir la magnitud o extensión de un nodo, el número de diferentes "temas" es grande, la cantidad de datos en un tema es largo, la lectura debe ser presentada con una unidad que le de sentido al usuario, que le ayude a lograr sus tareas.

Impacto de Usabilidad

Este patrón tiene un impacto de usabilidad alto positivamente en las propiedades: ***Minimizar la Carga Cognitiva*** y ***Mapeo natural: Facilidad de navegación***, debido a que ayuda a no tener sobrecarga de información y estructurar el contenido de la aplicación de una manera razonable al usuario, afectando de este modo también la propiedad de ***Mapeo Natural: Lenguaje de los usuarios*** ya que la información debe aparecer en un orden natural y lógico, con un impacto medio positivamente.

Tiene un impacto de usabilidad medio positivamente en las propiedades ***Consistencia: Visual y Funcional, Adaptabilidad: Experiencia del usuario y Facilidad de recordar el sistema,*** debido a que el tener la información en una sola unidad permite que la interfaz de usuario sean consecuente o consistente en aspecto y estructura, también se puede facilitar la navegación con otros sistemas similares, teniendo en cuenta la experiencia del usuario y la ***Facilidad de recordar*** el sistema.

La buena organización de la información que se presenta al utilizar el patrón, puede hacer que el usuario intuya de una manera más sencilla como interactuar con la aplicación, teniendo un impacto de usabilidad bajo positivamente, en la propiedad ***Mapeo natural: Predecible***.

Este patrón no tiene influencia sobre las propiedades ***Realimentación, Gestionando el error***, ***Consistencia: Evolutiva, Ayuda u orientación***, ***Explicito control de usuario, Adaptabilidad: Personalización y Accesibilidad: Discapacidades, Multicanal, Internacionalización.***

En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 26 Impacto de Usabilidad Patrón Nodo como una Sola Unidad (Node as a single Unit)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		NI
Gestionando El Error		NI
Consistencia	Visual	M+
	Funcional	M+
	Evolutiva	NI
Ayuda u orientación		NI
Minimizar la carga cognitiva		A+
Explícito control de Usuario		NI
Mapeo natural	Predecible	B+
	Lenguaje de los usuarios	M+
	Facilidad de navegación	A+
Accesibilidad	Discapacidades	NI
	Multicanal	NI
	Internacionalización	NI
Adaptabilidad	Experiencia de usuario	M+
	Personalización	NI
	Facilidad de Recordar	M+

3.1.2.7.4 Patrón Método Crear Nodo y Crear Enlace (Node Creation Method and Link Creation Method)

En la siguiente tabla se da la descripción del patrón y se define cuando debe ser usado:

Tabla 27 Patrón Método Crear Nodo y Crear Enlace

Nombre del patrón	Método Crear Nodo y Crear Enlace
Descripción	En muchas aplicaciones Web, donde los contenidos de un nodo no cambian, o cambian muy poco, deben definirse nodos estáticos. Por otro lado, en aplicaciones funcionalmente ricas dónde se captura la información en bases de datos, deben definirse nodos dinámicos. Se usan los enlaces estáticos en las aplicaciones cerradas, estáticas, y también se crean estáticamente los nodos cuando el mantenimiento en caso de cambios futuros no es un problema. Pueden usarse los enlaces estáticos en las aplicaciones dinámicas cuando la definición de un enlace-cómputo es demasiado compleja, el nodo del punto extremo es muy difícil de ser cambiado, o el enlace sólo es temporal. Se deben crear los nodos dinámicamente cuando la aplicación y la funcionalidad se actualizarán constantemente, cuando los lectores pueden modificar, crear, o formar nuevos nodos, y cuando la actualización instantánea es necesaria. Por otra parte, cuando no hay ninguna aplicación subyacente, y el conjunto de nodos es limitado y fijo, los nodos estáticos son la solución más aceptable. En algunos casos el tener nodos dinámicamente puede ser un proceso pesado, y es bueno guardar los resultados para una posible consulta en el caso en que el dato en la base de datos cambie en una proporción baja. Deben preferirse los enlaces dinámicos en aplicaciones dinámicas dónde los nuevos nodos también son creados dinámicamente, el dato es volátil, y deben lograrse la facilidad de mantenimiento eficazmente. También se usan enlaces dinámicos, incluso con los nodos estáticamente creados, cuando es posible definir los tipos del enlace, y cuando el número de enlaces de cada tipo para crear es considerable (es decir, el esfuerzo exige escribir un cómputo para crear los

	enlaces automáticamente es menos que el esfuerzo para instanciar manualmente todos los enlaces en una relación entre los conjuntos grandes de nodos)
Cuando usarlo	Se utiliza cuando no se sabe si crear los nodos estáticamente o dinámicamente, al igual que cuando no se sabe si crear los enlaces estáticos o dinámicos.

Impacto de usabilidad

El crear los nodos y enlaces de una manera correcta, ya sea dinámica o estáticamente, mejora la ***Experiencia del usuario*** con la aplicación, ya que esta dará una respuesta efectiva y eficiente a sus acciones, así también contribuirá en la ***Consistencia: Funcional*** y en la ***Facilidad de navegación***, con un impacto de usabilidad medio positivamente

El patrón tiene un impacto de usabilidad medio positivamente en las propiedades ***Consistencia: Evolutiva, Explicito control de usuario, Adaptabilidad: Personalización,*** al crear nodos o enlaces dinámicamente se puede tener consistencia evolutiva al poder crear o adicionar sistemas similares, de esta manera el usuario puede personalizar la aplicación haciendo que sienta que tienen el control sobre la aplicación.

El crear nodos o enlaces dinámicamente hace que tenga un impacto de usabilidad bajo negativamente en la propiedad ***Minimizar la carga cognitiva*** debido a que se puede sobrecargar la interfaz de información haciendo que lo que se muestre en pantalla no sea simple.

Tiene un impacto de usabilidad bajo negativamente en las propiedades ***Consistencia: Visual, Adaptabilidad: Facilidad de recordar el sistema,*** debido a que el crear nodos o enlaces dinámicamente, puede dar la posibilidad de que la información no sea consecuente o consistente en aspecto y estructura, y al tener demasiada información es más difícil de recordar el sistema.

Este patrón no tiene influencia sobre las propiedades ***Realimentación, Gestionando el error, Ayuda u orientación, Mapeo natural: Predecible, Lenguaje de los usuarios y Accesibilidad: Discapacidades, Multicanal e Internacionalización.***
En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 28 Impacto de Usabilidad Patrón Método Crear Nodo y Crear Enlace (Node Creation Method y Link Creation Method

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		NI
Gestionando El Error		NI
Consistencia	Visual	B-
	Funcional	M+
	Evolutiva	M+
Ayuda u orientación		NI
Minimizar la carga cognitiva		B-
Explícito control de Usuario		M+
Mapeo natural	Predecible	NI
	Lenguaje de los usuarios	NI
	Facilidad de navegación	M+
Accesibilidad	Discapacidades	NI
	Multicanal	NI
	Internacionalización	NI
Adaptabilidad	Experiencia de usuario	M+
	Personalización	M+
	Facilidad de Recordar	B-

3.1.2.7.5 Patrón Contexto Navegacional (Navigational Context)

En la siguiente tabla se da la descripción del patrón y se define cuando debe ser usado:

Tabla 29 Patrón Contexto Navegacional

Nombre del patrón	Contexto Navegacional
Descripción	Las aplicaciones Web normalmente involucran distribución de colecciones de nodos (ej., Pinturas, Ciudades, Personas, etc.). Estas colecciones pueden ser exploradas de diferentes maneras, según la tarea que el usuario este realizando. Esto significa que no sólo se necesita presentar la información de una manera diferente en todos los casos sino también proporcionar diferentes enlaces o índices. Los contextos navegacionales están compuestos por un conjunto de nodos (como Libros) y Enlaces de Contexto (enlaces que conectan los objetos en un contexto). Los nodos son decorados con información adicional de un contexto particular y enlaces adicionales para los Enlaces de Contexto. El contexto navegacional también puede contener información acerca del mismo contexto. Eso se mostrará en un nodo de Contexto particular. Ese nodo puede proporcionar un índice a todos los nodos en el contexto o un enlace al primero. Encontrar los contextos de navegación es importante porque son la estructura arquitectónica de alto nivel que ayudan a organizar la navegación de manera que se puede describir la estructura de navegación de una aplicación Web no sólo como un conjunto de nodos y enlaces sino también como un conjunto de contextos en que esos nodos serán accedidos.
Cuando usarlo	Se utiliza para organizar la estructura de navegación de la aplicación, proporcionando pautas, información y relaciones que dependen del estado actual de navegación, de tal manera que la información pueda ser mejor presentada y comprendida.

Impacto de usabilidad

El tener en cuenta los contextos navegacionales ayudará a estructurar la información de una manera comprensible y adecuada y proporcionará diferentes caminos para encontrar la información. Lo que contribuirá a que el usuario no se pierda y que encuentre la información más fácilmente. Teniendo un impacto de usabilidad alto positivamente en las propiedades: ***Minimizar la carga cognitiva, Mapeo Natural: Facilidad de Navegación*** y ***Adaptabilidad: Facilidad de r*****ecordar** ***el sistema***.

Se tiene un impacto de usabilidad medio positivamente en las propiedades ***Gestionando el error, Consistencia: Visual, Funcional, Ayuda u orientación, Mapeo natural: Lenguaje de los usuarios,*** debido a que los contextos de navegación ayudan y orientan al usuario en su navegación previniendo errores que se puedan presentar y el tener una estructura bien organizada facilita el acceso a la información.

Este patrón no tiene influencia sobre las propiedades ***Realimentación, Consistencia Evolutiva, Explicito control de usuario, Mapeo natural: Predecible, Accesibilidad: Discapacidades, Multicanal e Internacionalización, y Adaptabilidad: Experiencia de usuario, Personalización.***

En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 30 Impacto de Usabilidad Patrón Contexto Navegacional (Navigational Context)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		NI
Gestionando El Error		M+
Consistencia	Visual	M+
	Funcional	M+
	Evolutiva	NI
Ayuda u orientación		M+
Minimizar la carga cognitiva		A+
Explícito control de Usuario		NI
Mapeo natural	*Predecible*	NI
	Lenguaje de los Usuarios	M+
	Facilidad de navegación	A+
Accesibilidad	Discapacidades	NI
	Multicanal	NI
	Internacionalización	NI
Adaptabilidad	Experiencia de usuario	NI
	Personalización	NI
	Facilidad de Recordar	A+

3.1.2.7.6 Patrón Referencia Activa (Active Reference[18])

En la siguiente tabla se da la descripción del patrón y se define cuando debe ser usado:

Tabla 31 Patrón Referencia Activa

Nombre del patrón	Referencia Activa
Descripción	En muchas aplicaciones Web se necesita proporcionarle al lector una manera de entender donde está y ayudarlo a decidir dónde ir luego. La solución usual puede incluir un índice a los elementos para que el usuario navegue. Esta solución requerirá que el usuario vuelva del nodo actual al índice para ver este o para moverse a otro nodo. El índice proporciona un acortamiento para llegar a ese conjunto de nodos, pero una vez el lector está en uno de esos nodos, la referencia se pierde y además estas operaciones de navegación: mover hacia atrás al índice y adelante pueden desorientar al usuario. Una buena solución es mantener un objeto activo y distinguible de navegación actuando como un índice para otros objetos navegacionales (nodos o sub-índices). Este objeto sigue siendo distinguible junto con los objetos objetivos (lo que el usuario quiere ver), permitiendo al usuario explorar esos objetos o seleccionar otro objetivo relacionado. De esta manera se puede interactuar con ambos el índice y los nodos objetivos. Por ejemplo si se quiere explorar las ciudades del mundo la aplicación debería mostrar una referencia permanente de donde está el usuario localizado mientras esta navegando así: 'localización de: América del Sur - >Argentina - >Buenos Aires'. En particular tiene que poder escoger ir a la región en que la ciudad se localiza. Cuando se usa un objeto activo el lector tiene un registro distinguible y permanente sobre el estado actual de navegación y, de esta manera, se proporciona no sólo una herramienta de orientación sino también que está disponible mientras se navega por los nodos objetivo
Cuando usarlo	Este patrón se debe usar para proporcionar una referencia del estado actual de navegación, de la aplicación al usuario. Combinando una herramienta de orientación con una fácil manera de navegar para un conjunto de nodos relacionados, al mismo o más alto nivel de abstracción.

[18] **Nota:** El patrón historia de navegación puede ser de ayuda, pero normalmente considera todos los nodos al mismo nivel de abstracción, sin ninguna guía de composición o nivel del tema.

Impacto de Usabilidad

Este patrón ayuda a guiar al usuario, haciendo que este no se pierda en la aplicación, teniendo un impacto de usabilidad alto positivamente en las propiedades: ***Ayuda u Orientación, Facilidad de recordar el sistema y Facilidad en la navegación***

Se tiene un impacto de usabilidad medio positivamente en las propiedades de ***Realimentación, Gestionando el error, Consistencia: Visual, Explicito control de usuario,*** al tener una referencia activa le da al usuario realimentación en su navegación previniendo los errores que pueda cometer, teniendo una visión de donde se encuentra para mejor navegación y dando la posibilidad que escoja a donde quiere ir.

Este patrón no tiene influencia sobre las propiedades***, Consistencia: Funcional y Evolutiva, Minimizar la carga cognitiva, Mapeo natural: Lenguaje de los usuarios, Predecible, Accesibilidad: Discapacidades, Multicanal, Internacionalización, Adaptabilidad: Personalización, Experiencia de usuario.***

En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 32 Impacto de Usabilidad Patrón Referencia Activa (Active Reference)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		M+
Gestionando El Error		M+
Consistencia	Visual	M+
	Funcional	NI
	Evolutiva	NI
Ayuda u orientación		A+
Minimizar la carga cognitiva		NI
Explícito control de Usuario		M+
Mapeo natural	*Predecible*	NI
	Lenguaje de los usuarios	NI
	Facilidad de navegación	A+
Accesibilidad	Discapacidades	NI
	Multicanal	NI
	Internacionalización	NI
Adaptabilidad	Experiencia de usuario	NI
	Personalización	NI
	Facilidad de Recordar	A+

3.1.3 Patrones de Bajo Nivel o de Interfaz de usuario

Se han considerado como patrones de bajo nivel los patrones de interfaz de usuario, los cuales se relacionan detalles específicos de la interfaz Figura presentada en las aplicaciones Web. Debido a que los siguientes patrones no tienen una estructura y que su uso depende de la información a mostrar no se detallan las ventajas y desventajas de cada uno de ellos.

3.1.3.1 Patrón Información Solicitada (Information on Demand)

En la siguiente tabla se da la descripción del patrón y se define cuando debe ser usado:

Tabla 33 Patrón Información Solicitada

Nombre del patrón	Patrón Información Solicitada
Descripción	Este patrón presenta la información en la interfaz de usuario como un índice (fijo) y enlaces a la información detallada (dinámico), esto separa los canales de comunicación de entrada y los de salida, el canal de entrada es el contenido de la

	información que ofrece la aplicación y el canal de salida es la información en detalle de cada contenido
Cuando usarlo	Información solicitada se usa cuando en un nodo se tiene gran cantidad de información para mostrar al usuario y esta información no se ajusta en un pantallazo, no es posible tener una vista global de lo que se encontrará en esa página; cuando se quiere proporcionar diferentes atributos en un mismo nodo como ejemplo sonido, imágenes en movimiento, etc., puede distraer la atención del usuario por lo tanto se le da la oportunidad al usuario de activar/desactivar dichos atributos.

Impacto de usabilidad

Este patrón tiene un impacto de usabilidad alto positivamente al presentarse una información ordenada y completa al usuario en una sola página de manera estética, se ***Minimiza la carga cognitiva,*** *impactando de igual forma la propiedad* ***facilidad de recordar,*** además al permitirle al usuario activar y desactivar atributos del nodo se dará un ***Explicito control*** sobre la aplicación. En la medida que este patrón se utilice uniformemente tendrá un impacto medio positivamente en la ***Consistencia Visual***.

Tiene un impacto de usabilidad medio positivamente en las propiedades: ***Gestionando el error y Mapeo natural: Lenguaje de los usuarios***, ***Facilidad de navegación*** esta forma de presentar la información hace que el usuario cometa menos errores cuando esta navegando debido a que es natural, lógico y familiar a usuario. Por lo anterior también tendrá un impacto medio positivamente en la propiedad: **Adaptabilidad: Experiencia del usuario.**

El impacto de usabilidad bajo positivamente en la propiedad de ***Adaptabilidad: Personalización*** debido a que es afectada cuando se quiere proporcionar diferentes atributos en un mismo nodo, como medios auditivos, imágenes, permitiendo personaliza la aplicación.

Este patrón no tiene influencia sobre la propiedad de: ***Realimentación, Consistencia: Funcional y evolutiva, Ayuda u orientación, Mapeo natural: Prdecible, Accesibilidad, Adaptabilidad: Experiencia de usuario.***

En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 34 Impacto de Usabilidad Patrón Información Solicitada (Information on Demand)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		NI
Gestionando El Error		M+
Consistencia	Visual	M+
	Funcional	NI
	Evolutiva	NI
Ayuda u orientación		NI
Minimizar la carga cognitiva		A+
Explícito control de Usuario		A+
Mapeo natural	Predecible	NI
	Lenguaje de los usuarios	M+
	Facilidad de navegación	M+
Accesibilidad	Discapacidades	NI
	Multicanal	NI
	Internacionalización	NI
Adaptabilidad	Experiencia de usuario	NI
	Personalización	B+

	Facilidad de Recordar	A+

3.1.3.2 Patrón Desacoplar la Interacción de la Información (Information-Interaction Decoupling)

En la siguiente tabla se da la descripción del patrón y se define cuando debe ser usado:

Tabla 35 Patrón Desacoplar la Interacción de la Información

Nombre del patrón	Patrón Información Solicitada
Descripción	Este patrón organiza la información en la interfaz de usuario separando los canales de comunicación de entrada y los de salida, los agrupa de manera que deja el "grupo de interacción de entrada" para que permanezca fijo y "el grupo de salida" reacciona dinámicamente al control de activación. También diferencia la información substantiva (es decir, el contenido) de los controles de activación.
Cuando usarlo	Desacoplar la Interacción de la Información se usa cuando hay gran cantidad de información para mostrar al usuario y se quiere mostrar solo lo que el usuario necesita presentándole dos grupos de interacción de entrada (contenido) y salida (muestra la información).

Impacto de usabilidad

Este patrón tiene un impacto de usabilidad alto positivamente en la propiedad: ***Minimizar la carga cognitiva***, debido a que al mostrar la información en grupos de entrada y salida no se saturan las páginas de información.

Las propiedades sobre las que tiene un impacto medio positivamente son: ***Realimentación, Consistencia: visual y funcional, Mapeo natural: Lenguajes de los usuarios y Facilidad de navegación,*** debido a que este patrón permite que se le presente información al usuario del estado del sistema cuando se carga una nueva vista del grupo de entrada, facilita que la información sea consistente con un lenguaje entendible por el usuario, además que sea fácil de navegar.

Se tiene un impacto de usabilidad bajo positivamente en las propiedades: ***Accesibilidad: Internacionalización y Facilidad de Recordar*** el sistema, ***Adaptabilidad: Personalización, Experiencia del usuario,*** la forma en la que este patrón permite agrupar la información aporta para que el usuario la recuerde con facilidad***;*** dependiendo del tipo de información que se maneje puede facilitar la internacionalización adaptándose a personalizar la información que se presenta mejorando de esta forma la experiencia del usuario.

Este patrón no tiene influencia sobre las propiedades de ***Gestionando el error***, ***Consistencia: Evolutiva, Ayuda u orientación,*** **Explicito control de usuario,** ***Mapeo Natural: Predecible, A*****ccesibilidad: Discapacidad, Multicanal y** ***Adaptabilidad:***

En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 36 Impacto de Usabilidad Patrón Desacoplar la Interacción de la Información (Information-Interaction Decoupling)

PROPIEDADES DE USABILIDAD		**Impacto de Usabilidad**
Realimentación		M+
Gestionando el Error		NI
Consistencia	**C. Visual**	M+
	C. Funcional	NI
	C. Evolutiva	NI
Ayuda u orientación		NI

Minimizar la carga cognitiva		A+
Explícito control de Usuario		NI
Mapeo natural	**Predecible**	NI
	Lenguaje de los usuarios	M+
	Facilidad de navegación	M+
Accesibilidad	**Discapacidades**	NI
	Multicanal	NI
	Internacionalización	B+
Adaptabilidad	**Experiencia de usuario**	B+
	Personalización	B+
	Facilidad de Recordar	B+

3.1.3.3 Patrón Agrupación Conductual (Behavioral Grouping)

En la siguiente tabla se da la descripción del patrón y se define cuando debe ser usado:

Tabla 37 Patrón Agrupación Conductual

Nombre del patrón	Agrupación Conductual
Descripción	Este patrón agrupa la información dependiendo de la conducta funcional de los objetos de control (como los links, los botones, etc.) de la interfaz, los selecciona y los coloca en cada área de la pantalla. La interfaz de cada grupo tiene apariencia similar para reforzar la comprensión. En una aplicación de hipermedia típica hay diferentes tipos de objetos de interfaz activos: aquéllos que proporcionan la navegación "general", como el botón “atrás”, o links para devolver a los índices; objetos que proporcionan la navegación dentro de un contexto; objetos que controlan la interfaz (como en "Información solicitada"), etc.
Cuando usarlo	Este patrón es utilizado cuando hay diferentes tipos de objetos en la aplicación a desarrollar y se quiere estructurar la interfaz por áreas según estos objetos.

Impacto de usabilidad

Este patrón tiene un impacto de usabilidad alto positivamente al presentarse la información por grupos de objetos en las áreas de la interfaz permitiendo de esta manera que la presentación sea ***Consistente*** de manera ***Visual*** como ***Funcional***, ***Ayuda y orienta al usuario*** para navegar y mejora la percepción de la interfaz ***Minimizando la carga cognitiva***, lo cual contribuye a la ***Experiencia del usuario*** y a la ***Facilidad de recordar el sistema*** ya que esto lo puede volver predecible.

Las propiedades sobre las que tiene un impacto medio positivamente son: ***Explicito control de usuario, Mapeo natural: Predecible y Adaptabilidad: Personalización***, al presentar la información de manera consistente que le permita al usuario personalizar la presentación, seleccionando y colocando los objetos de controles en cada área de la pantalla, así la interfaz de cada grupo tiene apariencia similar para reforzar la comprensión.

Tiene un impacto de usabilidad bajo positivamente en la propiedad de ***Mapeo natural: Lenguaje de los usuarios,*** debido a que este patrón permite que se presente la información de manera que los conceptos sean familiares al usuario en cuanto a que hace que aparezca en un orden natural y lógico.

No se tiene un impacto de usabilidad en las propiedades ***Realimentación, Gestionando el error, Consistencia: Evolutiva, Accesibilidad.***

En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 38 Impacto de Usabilidad Patrón Agrupación Conductual (Behavioral Grouping)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		NI
Gestionando El Error		NI
Consistencia	Visual	A+
	Funcional	A+
	Evolutiva	NI
Ayuda u orientación		A+
Minimizar la carga cognitiva		A+
Explícito control de Usuario		M+
Mapeo natural	Predecible	M+
	Lenguaje de los usuarios	B+
	Facilidad de navegación	A+
Accesibilidad	Discapacidades	NI
	Multicanal	NI
	Internacionalización	NI
Adaptabilidad	Experiencia de usuario	A+
	Personalización	M+
	Facilidad de Recordar	A+

3.1.3.4 Patrón Anticipación Conducta (Behavior Anticipation)

En la siguiente tabla se da la descripción del patrón y se define cuando debe ser usado:

Tabla 39 Patrón Anticipación Conducta

Nombre del patrón	Anticipación Conducta
Descripción	Cuando el usuario selecciona un objeto, de la interfaz, la retroalimentación se puede proveer de diferentes maneras, por ejemplo se puede proporcionar retroalimentación auditiva o visual. Una manera de hacerlo es resaltando o cambiando de color el objeto, o usar un área de la pantalla para poner una explicación textual corta.
Cuando usarlo	Este patrón es utilizado cuando hay diferentes tipos de objetos activos en la aplicación a desarrollar y para darle un sentido de que sucede y guiar al usuario para que haga una buena selección de la opción.

Impacto de usabilidad

Cuando se selecciona un objeto activo se proporciona una explicación automática corta de lo que hace el objeto, teniendo un impacto de usabilidad alto positivamente en las propiedades de usabilidad: ***Realimentación y Ayuda u orientación al usuario***, este patrón también contribuye a ***Gestionar el error*** ya que se pueden implementar mensajes de alerta para las excepciones que se presenten por acciones del usuario lo cual a su vez da realimentación al usuario acerca de las acciones que le son permitidas o no ***Minimizando la carga cognitiva***.

La propiedad que tienen un impacto medio positivamente ***Consistencia: Visual,*** al resaltar o cambiar de color el objeto, o usar un área de la pantalla para poner una explicación textual corta del objeto visitado o seleccionado con un lenguaje natural se tiene una consistencia visual.

Se tiene un impacto de usabilidad bajo positivamente en la propiedad de ***Accesibilidad: Discapacidad,*** cuando la orientación al usuario se da de manera auditiva.

Este patrón no tiene influencia sobre las propiedades de ***Consistencia: Funcional y Evolutiva, Explicito control de usuario, Mapeo natural, Accesibilidad y Adaptabilidad.***
En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 40 Impacto de Usabilidad Patrón Anticipación Conducta (Behavior Anticipation)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		A+
Gestionando El Error		A+
Consistencia	Visual	M+
	Funcional	NI
	Evolutiva	NI
Ayuda u orientación		A+
Minimizar la carga cognitiva		A+
Explícito control de Usuario		NI
Mapeo natural	Predecible	NI
	Lenguaje de los usuarios	NI
	Facilidad de navegación	NI
Accesibilidad	Discapacidades	B+
	Multicanal	NI
	Internacionalización	NI
Adaptabilidad	Experiencia de usuario	NI
	Personalización	NI
	Facilidad de Recordar	NI

3.1.3.5 Patrón Proceso de Retroalimentación (Process Feed-Back)

En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 41 Patrón Proceso de Retroalimentación

Nombre del patrón	Proceso de Retroalimentación
Descripción	Este patrón proporciona retroalimentación al usuario de las operaciones que está realizando informándole la situación en que esta, indicando el progreso si son procesos pesados por ejemplo una consulta de una gran cantidad de datos.
Cuando usarlo	Este patrón se usa cuando se le quiere dar una constante retroalimentación puntual al usuario sobre las acciones que está realizando.

Impacto de usabilidad
Este patrón implica un impacto de usabilidad alto positivamente en las propiedades de usabilidad: **R*ealimentación*** porque se le está informado al usuario el estado de las acciones que realiza la aplicación, lo cual ***Ayuda y orienta al usuario*** para que no se sienta frustrado y por el contrario sienta que tiene un control sobre la aplicación. Un claro ejemplo de esto es la barra de progreso en las aplicaciones Web para transacciones pesadas.

En las propiedades de ***Gestionando el error y Consistencia Visual*** tiene un impacto medio positivamente, debido a que cuando se le informa al usuario de las acciones que realiza a través de la navegación esta previniendo que cometa errores y hace que las páginas presentadas sean consistentes visualmente y ***Previsibles*** teniendo un impacto bajo positivamente en esta última.

Se tiene un impacto de usabilidad bajo positivamente en la propiedad de ***Accesibilidad: Discapacidad***, si la información del progreso se da a través de medios de audio como el sonido.

Este patrón no tiene influencia sobre las propiedades ***Consistencia: funcional y evolutiva, Minimizar la carga cognitiva, Explicito control de usuario, Mapeo natural, Accesibilidad y Adaptabilidad.***
En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 42 Impacto de Usabilidad Patrón Proceso de Retroalimentación (Process Feed-Back)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		A+
Gestionando El Error		M+
Consistencia	**Visual**	M+
	Funcional	NI
	Evolutiva	NI
Ayuda u orientación		A+
Minimizar la carga cognitiva		NI
Explícito control de Usuario		NI
Mapeo natural	**Predecible**	B+
	Lenguaje de los usuarios	NI
	Facilidad de navegación	NI
Accesibilidad	**Discapacidades**	B+
	Multicanal	NI
	Internacionalización	NI
Adaptabilidad	**Experiencia de usuario**	NI
	Personalización	NI
	Facilidad de Recordar	NI

3.1.3.6 Patrones de Interacción

Nombre en Español	Nombre en Ingles
Narrativa	Narrative
Despliegue de Información de Alta Densidad	High-density Information Display
Formulario	Form
Panel de Control	Control Panel
Editor WYSIWYG	Editor WYSIWYG
Comando Compuesto	Composed Command
Espacios Navegacionales	Navigable Spaces
Visión Global en Detalle	Overview Beside Detail
Instrucciones Paso a Paso	Step-by-Step Instructions
Grupos Pequeños de Cosas Relacionadas	Small Groups of Related Things
Conjunto de Jerarquías	Hierarchical Set
Conjunto Tabulador	Tabular Set
Mapa o Grafica	Chart or Graph
Detalle Opcional en Demanda	Optional Detail On Demand
Desactivar Cosas No Pertinente	Disabled Irrelevant Things
Mostrar el Indicador de Acceso	Pointer Shows Affordance
Descripción corta	Short Description
Apariencia de Fondo	Background Posture
Superficie Activa Central	Central Working Surface
Superficies Activas en Mosaico	Tiled Working Surfaces
Pila de Superficies Activas	stack of Working Surfaces
Montón de Superficies Activas	Pile of Working Surfaces
Mapa de Espacios Navegacionales	Map of Navigable Spaces
Puntos Claros de Entrada	Clear Entry Points

Secciones de Color Codificado	Color-Coded Sections
Ir un Paso Atrás	Go Back One Step
Regresar a un Lugar Seguro o Punto de Control	Go Back to a Safe Place
Acciones Convenientes de Ambiente	Convenient Environment Actions
Acciones de Localización del Objeto	Localized Object Actions
Acciones para Objetos Múltiples	Actions for Multiple Objects
Opción de un Conjunto Pequeño	Choice from a Small Set
Colección Editable	Editable Collection
Perdonar Entrada de Texto	Forgiving Text Entry
Entrada de Texto Estructurado	Structured Text Entry
Barra de Herramientas	Toolbox
Preferencias de Usuario	User preference
Secuencia de Acción Escrita	Scripted Action Sequence
Anotaciones de Usuario.	User's Annotations
Marcadores de Libros	Bookmarks
Framewok Repetido	Repeated Framework
Buenas por Defecto	Good Defaults
Estado Recordado	Remembered State
Historia de la Interación	Interaction History
Indicador de Progreso	Progress Indicator
Mensaje Importante	Important Message
Chequeo Real	Reality Check
Demostración	Demonstration

Tabla 43 Patrones de Interacción

Estos patrones se describen en detalle en el anexo 4

3.2 Patrones de Lógica de Dominio

3.2.1 Patrones de Alto Nivel

3.2.1.1 Patrón Capas (Layers)

En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 44 Patrón Capas

Nombre del patrón	Capas
Descripción	Para organizar una aplicación empresarial, la industria del software ha convergido en una Arquitectura basada en Capas, dividiendo el sistema en tres capas básicas: la capa de presentación, la capa de lógica de dominio y la capa de acceso a datos. El principio detrás de esta arquitectura es que cada capa dependa sólo de los elementos contenidos en ella o en las capas situadas por debajo, teniendo responsabilidades bien definidas y evitando cualquier tipo de acoplamiento con las capas superiores [81].
Estructura	***Capa de presentación*** Maneja la interacción entre el usuario y la aplicación. En algunas ocasiones el usuario puede ser otro sistema comunicándose por ejemplo a través de un servicio remoto. Esta comunicación se hace especialmente notoria en ambientes Web, donde la capa de presentación no sólo tiene que crear documentos entendibles por los usuarios sino manejar los mensajes enviados por el navegador Web como datos

	de formularios. Para esto se suele utilizar un esquema de Controladores y Vistas, donde los controladores son el eje entre las capas inferiores y las vistas, el patrón de diseño Modelo Vista Controlador es un ejemplo popular de esta forma de estructurar la aplicación. ***Capa de lógica de dominio*** Representa las reglas de negocio. Lo que distingue a esta lógica del resto de la aplicación es que mantiene los conceptos centrales del proceso. ***Capa de acceso a datos*** Esta capa brinda servicios para sincronizar la capa de lógica con un medio de almacenamiento. Para esto se deben identificar los objetos en memoria que deben sobrevivir a la ejecución del programa teniendo así una persistencia de largo plazo.
Ventajas	• Soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los desarrolladores la partición de un problema complejo en una secuencia de pasos incrementales. • Admite naturalmente optimizaciones y refinamientos. • Proporciona amplia reutilización ya que se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas.
Desventajas	• Muchos problemas no admiten un buen mapeo en una estructura jerárquica. Incluso cuando un sistema se puede establecer lógicamente en capas, consideraciones de rendimiento pueden requerir acoplamientos específicos entre capas de alto y bajo nivel. • A veces es extremadamente difícil encontrar el nivel de abstracción correcto • Los cambios en las capas de bajo nivel tienden a filtrarse hacia las de alto nivel. • La arquitectura en capas ayuda a controlar y encapsular aplicaciones complejas, pero complica no siempre razonablemente las aplicaciones simples [82].
Cuando usarlo	Este patrón de arquitectura sirve para estructurar aplicaciones que se pueden descomponer en partes a distintos niveles de abstracción, la forma de identificar si se debe o no usar este patrón puede ser haciéndose las siguientes preguntas: ¿los cambios no deben afectar a todo el sistema?, ¿las interfaces deben ser estables (estándares)?,¿debe ser posible intercambiar partes?, ¿las capas de bajo nivel pueden servir en otros sistemas futuros?, ¿no es fácil determinar la granularidad?, ¿componentes complejos deben subdividirse?, ¿Cruzar fronteras puede afectar el rendimiento?, ¿Hay que definir fronteras para dividir el trabajo?.

Impacto de usabilidad

El separar la aplicación en capas hace que sea más sencillo realizar los cambios de una capa a otra, por el bajo acoplamiento que brinda, haciendo posible entonces adaptar la aplicación a diferentes tipos de usuarios, idiomas, y medios de comunicación teniendo un impacto de usabilidad medio positivamente en las propiedades que corresponden a A***ccesibilidad:*** Discapacidades, ***Internacionalización y Multicanal.*** Esta misma característica hace fácil adaptar la presentación a los distintos usuarios, favoreciendo la propiedad de ***adaptabilidad***: ***personalización*** con un impacto de usabilidad bajo positivamente.

Se tiene un impacto de usabilidad medio positivamente en las propiedades de: ***Consistencia Visual, Funcional y Evolutiva*** debido a que la posibilidad de definir interfaces con una capa estándar, facilitando la construcción de extensiones o añadiendo diferentes productos de la misma familia.

Este patrón permite la realimentación entre capas, debido a que una capa informa a la otra, las solicitudes, los cambios etc., así como también los errores ocurridos, teniendo entonces un impacto de usabilidad bajo positivamente en las propiedades ***Realimentación y Gestionando el error***.

Este patrón no tiene influencia sobre las propiedades ***Ayuda u orientación, Minimizar la carga cognitiva, Explicito control de usuario, Mapeo natural: Lenguaje de los usuarios, Facilidad de navegación, Adaptabilidad: Experiencia del usuario y Facilidad de recordar el sistema.***
En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 45 Impacto de Usabilidad Patrón Capas (Layers)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		B+
Gestionando El Error		B+
Consistencia	Visual	M+
	Funcional	M+
	Evolutiva	M+
Ayuda u orientación		NI
Minimizar la carga cognitiva		NI
Explícito control de Usuario		NI
Mapeo natural	Predecible	B+
	Lenguaje de los usuarios	NI
	Facilidad de navegación	NI
Accesibilidad	Discapacidades	M+
	Multicanal	M+
	Internacionalización	M+
Adaptabilidad	Experiencia del usuario	NI
	Personalización	B+
	Facilidad de Recordar	NI

3.2.2 Patrones de Nivel Intermedio

3.2.2.1 Patrón Pizarra

En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 46 Patrón Pizarra

Nombre del patrón	Pizarra
Descripción	Colección de programas independientes que trabajan cooperativamente en una estructura de datos común. Cada programa es especializado en solucionar una parte especial de la tarea y todos trabajan en conjunto sobre la solución. A estos programas especializados se les llama fuentes de conocimiento y estos hacen parte de la estructura del patrón [21]
Estructura	• **Pizarra:** La pizarra es el almacén de datos central. Los elementos del espacio de solución y los datos de control son guardados aquí, así como el vocabulario que es el conjunto de todos los datos que aparecen sobre la pizarra. • **Fuentes de Conocimiento:** Son subsistemas independientes, especializados en solucionar una parte especifica de la tarea, los cuales escriben y leen de la pizarra. Estos programas especializados son independientes el uno del otro, no se llaman, ni hay una secuencia predeterminada para su activación. • **Control**: Monitorea y evalúa el estado actual del sistema y coordina las fuentes de conocimiento.

Ventajas	• Ayuda a la experimentación en dominios de aplicación en los cuales no existen soluciones algorítmicas claras y bien definidas. • Soporte para la facilidad de cambios y de mantenimiento, debido a que las diferentes partes de las cuales consta el patrón pizarra son independientes, permitiendo modificar, eliminar o agregar estos componentes por separado. • Las fuentes de conocimientos son reutilizables, ya que estos son programas independientes, especializados en ciertas tareas. • El sistema siempre provee una solución aunque sea parcial, aproximada y débilmente corroborada.
Desventajas	• No hay garantía de obtener una solución final correcta. Generalmente los sistemas pizarra pueden solucionar sólo un cierto porcentaje de sus tareas dadas correctamente. • Dificultad de establecer una buena estrategia de control. La estrategia de control no puede ser diseñada de una manera simple y requiere un enfoque experimental. • Eficiencia Baja. Los sistemas de pizarra sufren sobrecargas computacionales en el rechazo de hipótesis incorrectas. • Esfuerzo de desarrollo alto. La mayoría de los sistemas pizarra toman años para desarrollarse debido a los varios experimentos que hay que realizar hasta dar con un conjunto de fuentes de conocimientos y una estrategia de razonamiento adecuada que garantice buenas soluciones en la mayoría de los casos. • Ningún soporte para el paralelismo. El patrón pizarra no soporta de forma directa y evidente una implementación concurrente o paralela
Cuando usarlo	El patrón arquitectónico pizarra es útil para dominios inmaduros, en los cuales la experimentación es provechosa y aún no existe una solución conocida o factible. Aplicaciones Web para el reconocimiento de voz y de imágenes pueden ser ejemplos de dominios en los que tales problemas existen. Después de la investigación y de la experiencia que se adquiera, mejores algoritmos pueden desarrollarse, lo que permite usar una arquitectura más eficiente.

Impacto de Usabilidad

Al analizar este patrón, teniendo en cuenta su estructura, las ventajas y desventajas que presenta se puede concluir que sobre la propiedad ***Accesibilidad: Discapacidades, Multicanal e Internacionalización,*** tiene un impacto de usabilidad medio positivamente, ya que este patrón permite experimentar en dominios inmaduros, como por ejemplo el de reconocimiento de voz, experimentos que contribuirían a dar soluciones a problemas que tienen que ver con la accesibilidad de una aplicación Web.

Sobre la propiedad ***Adaptabilidad: Experiencia de usuario y Personalización,*** tiene un impacto de usabilidad bajo positivamente debido al soporte para la facilidad de cambios y de mantenimiento, que brinda este patrón, al ser las partes de las cuales consta independientes.

Al tener diferentes subsistemas independientes, especializados en solucionar una parte específica de la tarea, con paradigmas y lógica diferentes, se pueden llegar a tener estructuras de presentación, y funcionalidad distintas, en toda la aplicación. Afectando por lo tanto la propiedad ***Consistencia Visual,*** con un impacto de usabilidad medio negativamente, la ***Consistencia: Funcional y Evolutiva*** con un impacto alto negativamente. Esto mismo también hace que las propiedades ***Ayuda u orientación***, ***Mapeo natural: Facilidad de navegación***, tenga un impacto medio negativamente.La propiedad ***Gestionando el error*** tiene un impacto de usabilidad alto negativamente debido a que con este patrón se puede llegar a tener soluciones incorrectas, lo que no ayudaría a prevenir los errores sino más bien que estos podrían llegar a incrementarse.

Las propiedades ***Adaptabilidad: Facilidad de recordar y Mapeo natural: Predecible***, tienen un impacto de usabilidad bajo negativamente debido a la estructura desorganizada que puede presentarse con la utilización de este patrón.

Al usuario no obtener respuestas en un tiempo razonable a sus acciones por causa de las sobrecargas computacionales que pueden llegar a presentarse en la aplicación, la propiedad ***Realimentación*** puede tener un impacto de usabilidad bajo negativamente.

Este patrón no tiene influencia en las propiedades ***Minimizar la carga cognitiva, Explicito control de usuario, Mapeo natural: Lenguaje de los usuarios.***
En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 47 Impacto de Usabilidad Patrón Pizarra

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		B-
Gestionando El Error		A-
Consistencia	Visual	M-
	Funcional	A-
	Evolutiva	A-
Ayuda u orientación		M-
Minimizar la carga cognitiva		NI
Explícito control de Usuario		NI
Mapeo natural	Predecible	B-
	Lenguaje de los usuarios	NI
	Facilidad de navegación	M-
Accesibilidad	Discapacidades	M+
	Multicanal	M+
	Internacionalización	M+
Adaptabilidad	Experiencia de usuario	B+
	Personalización	B+
	Facilidad de Recordar	B-

3.2.2.2 Patrón Mediadores (Middleware)

En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 48 Patrón Mediadores

Nombre del patrón	Mediadores
Descripción	Es un patrón arquitectónico aplicado a la estructuración de sistemas distribuidos, en los cuales es necesaria la interacción remota de componentes altamente desacoplados. Se introduce un componente Mediador que logre el desacoplamiento de los clientes y de los servidores. También registra a los servidores, logrando de esta forma que los servicios que éstos ofrecen estén disponibles a los posibles clientes [21].
Estructura	Los componentes que conforman este patrón son: Mediador, servidores, clientes o colegas. • Un mediador: Que define la interfaz de comunicación entre los clientes, permitiendo la cooperación entre ellos para la realización de tareas. Este componente conoce y mantiene a todos los componentes o servicios. • Un servidor implementa objetos que exponen su funcionalidad a través de interfaces que consisten en operaciones y atributos. • Los clientes son aplicaciones que acceden a los servicios de al menos un

	servidor. Para invocar servicios remotos, los clientes envían solicitudes al mediador. Después que la operación se ha ejecutado, los clientes reciben respuestas o excepciones del mediador.
Ventajas	• Transparencia en la ubicación de los servidores. El Mediador recibe las peticiones del cliente y localiza el servidor donde se halla el servicio solicitado, por lo tanto el cliente no necesita conocer la ubicación de los servidores. Y viceversa, los servidores no necesita saber la ubicación del cliente solicitante, ya que reciben todas las solicitudes del componente Mediador local. • Facilidad de evolución ya que se puede modificar solamente un componente, sin tener que modificar el resto. Utilizando los Proxy y los puentes para la implementación de los cambios. • Reutilización: Cuando se construyen nuevas aplicaciones cliente, frecuentemente el desarrollador se basa en la funcionalidad de su aplicación en los servicios existentes. • Interoperabilidad entre diferentes sistemas mediadores. Los diferentes sistemas Mediador pueden interoperar si tienen un protocolo común para el intercambio de mensajes. Este protocolo es implementado y manejado por puentes, los cuales son responsables de trasladar el protocolo específico del Mediador en un protocolo común, y viceversa. • Proporciona una alta flexibilidad que permite agregar o eliminar componentes con facilidad
Desventajas	• Eficiencia no muy alta. Las aplicaciones que utilizan la implementación del Mediador usualmente son más lentas que las aplicaciones cuya distribución de componentes es estática y conocida. Los sistemas que dependen directamente de un mecanismo concreto para la comunicación entre procesos también dan un mejor desempeño que una arquitectura Mediador, porque el Mediador introduce capas indirectas para lograr portabilidad, flexibilidad y facilidad de cambiar. • Dependencia, acoplamiento, máximo al Mediador. Si este cae el sistema entero cae. • Pruebas difíciles y costosas cuando se utiliza muchos componentes.
Cuando usarlo	Este patrón se utilizará cuando se necesita la interacción remota de componentes altamente desacoplados. Esto se consigue introduciendo un nuevo componente Mediador cuya función principal es conseguir el desacoplamiento de los clientes y de los servidores. El Mediador es responsable de coordinar la comunicación entre el cliente y el servidor.

Impacto de Usabilidad

Al utilizar este patrón se tiene un impacto medio positivamente con la propiedad de ***Adaptabilidad***: ***Personalización*** porque se espera que el sistema cambie y progrese continuamente, permite adicionar más clientes y servidores a la red y el Mediador se encarga de realizar la comunicación entre las aplicaciones dentro de la red distribuida por medio de los componentes como Proxy, puentes y Mediador. También se podría implementar con este patrón la propiedad ***Realimentación*** donde se le informa al usuario el progreso de la carga de los procesos que realiza la aplicación. Todo esto también influye en la propiedad ***Accesibilidad****: **Discapacidades, Multicanal e Internacionalización***, por la estructura que este brinda a la aplicación, facilitando la adaptación de estas características, teniendo un impacto de usabilidad bajo positivamente en estas.

Este patrón tiene un impacto bajo positivamente con las propiedades de ***Gestionando el error*** ya que el sistema depende del mediador y si este falla todo el sistema se cae; influyendo con la propiedad ***Consistencia: Funcional, Evolutiva*****,** cuando las aplicaciones van evolucionando con el tiempo permite integrar nuevos sistemas, ya sean dispositivos multimedia, afectando la accesibilidad.

Este patrón no tiene influencia sobre las propiedades ***Consistencia: Visual, Ayuda u orientación, Minimizar la carga cognitiva, Explicito control de usuario, Mapeo natural, Adaptabilidad: Experiencia del usuario y Facilidad de recordar el sistema.*** En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 49 Impacto de Usabilidad Patrón Mediadores (Bróker)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		M+
Gestionando El Error		B+
Consistencia	Visual	NI
	Funcional	B+
	Evolutiva	B+
Ayuda u orientación		NI
Minimizar la carga cognitiva		NI
Explícito control de Usuario		NI
Mapeo natural	Predecible	NI
	Lenguaje de los usuarios	NI
	Facilidad de navegación	NI
Accesibilidad	Discapacidades	B+
	Multicanal	B+
	Internacionalización	B+
Adaptabilidad	Experiencia de usuario	NI
	Personalización	M+
	Facilidad de Recordar	NI

3.2.2.3 Patrón Micro-Kernel

En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 50 Patrón Micro-Kernel

Nombre del patrón	Micro-Kernel
Descripción	Este patrón encapsula la funcionalidad básica de una aplicación, en un componente llamado microkernel, otros componentes se basan en todos o algunos de estos servicios básicos para poder ejecutarse.
Estructura	El patrón Microkernel define cinco tipos de componentes: • Microkernel: Es el principal componente del patrón. Este implementa los servicios centrales, que sirven como base fundamental para las funcionalidades más complejas. • Servidor interno: También conocido como subsistema. Representa un componente separado que ofrece funciones adicionales, extendiendo la funcionalidad provista por el Microkernel. • Servidores externos: Es un componente que usa al Microkernel para implementar su propia vista. (una vista indica una capa de abstracción desarrollada encima de los servicios atómicos proveídos por el Microkernel.) • Adaptadores: Capa intermedia entre el cliente y el servidor externo, la tarea del adaptador es redireccionar las peticiones del cliente, hacia el servidor externo o bien hacia el Microkernel. • Cliente: Es una aplicación que está asociada con exactamente un servidor externo. Este solamente accede a las interfaces provistas por el servidor externo.
	• Flexibilidad. Una de las fuerzas más grandes de un sistema Microkernel es su flexibilidad y extensibilidad. Una vez definido el Microkernel sólo basta agregar vistas a la aplicación, agregando nuevos servidores externos. Si se

Ventajas	necesita implementar una vista adicional, todo lo que se necesita hacer es agregar un nuevo servidor externo. • Extensibilidad: Extender el sistema con las capacidades adicionales sólo requiere la adición o extensión de servidores internos. • El componente Microkernel proporciona todos los mecanismos necesarios para permitirle a los servidores externos implementar sus propias funcionalidades, lo que incrementa la facilidad de mantenimiento y adaptabilidad del sistema entero.
Desventajas	• Complejidad del diseño y la implementación. Desarrollar un sistema basado en Microkernel es una tarea no trivial. Por ejemplo, a veces puede ser muy difícil analizar o predecir los mecanismos básicos que un componente Microkernel debe proporcionar.
Cuando usarlo	La estructura Microkernel es muy atractiva y se presta por si misma a aplicaciones que requieren ser portables, extensibles y adaptables. Además, este patrón es útil en el desarrollo de varias aplicaciones que usan interfaces de programación similares, que contienen la misma funcionalidad principal. El Microkernel lo podemos ver en una aplicación Web como un "framework", un conjunto de API's[19], que contenga la funcionalidad básica de la aplicación Web, sobre la cual se implemente funciones específicas.

Impacto de Usabilidad

El patrón tiene sobre la propiedad ***Consistencia: Funcional*** un impacto de usabilidad alto positivamente, debido a que al tener un programa central, para las mismas funciones se tendrá una homogeneidad en la funcionalidad de la aplicación y aún en la familia de productos software, favoreciendo además la ***Consistencia Visual*** y evolutiva con un impacto de usabilidad bajo positivamente.

Las aplicaciones Web a menudo tienen una larga vida, a veces diez años o más. Sobre estos períodos de tiempo, nuevos requerimientos de usuario pueden aparecer así como nuevas tecnologías, por estos motivos se sugiere considerar el patrón Microkernel en el diseño de la arquitectura de una aplicación Web, ya que este contribuye a que una aplicación sea extensible y adaptable. Estos beneficios conducen a que el patrón tenga sobre la propiedad de usabilidad ***Accesibilidad: Discapacidades, Multicanal e Internacionalización,*** un impacto de usabilidad medio positivamente, debido a que facilita la adaptación de la aplicación a usuarios con diversos tipos de discapacidades, lengua, moneda, etc., ayudando también a dar soporte para varios dispositivos. Estas ventajas a la vez favorecen la propiedad ***Adaptabilidad: Experiencia de usuario y Facilidad de recordar el sistema*** con un impacto medio positivamente***,*** debido a que permitiría fácilmente añadir nuevas funcionalidades y proveer ciertos servicios personalizados a los usuarios. Esto se haría por ejemplo creando nuevas vistas que serian los servidores externos y plugins[20], servidores internos. La propiedad adaptabilidad: personalización también se ve afectada en menor grado, con un impacto bajo positivamente.

Este patrón tiene un impacto de usabilidad bajo positivamente en la propiedad ***Realimentación y Gestionando el error,*** debido a que la estructura que tiene el patrón hace posible la interacción con el usuario y la gestión de errores.

[19] Una API es una Interfaz de Programación de Aplicaciones (Application Programming Interfaz), un conjunto de funciones o métodos usados para acceder a ciertas funcionalidades.

[20] (Plug-in) Programa que puede anexarse a otro para aumentar sus funcionalidades (generalmente sin afectar otras funciones ni afectar la aplicación principal).

Este patrón no tiene influencia directa sobre las propiedades ***Ayuda u orientación, Minimizar la carga cognitiva, Explicito control de usuario, Mapeo natural: Predecible, Lenguaje de los usuarios y Facilidad de navegación.***
En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 51 Impacto de Usabilidad Patrón Micro-Kernel

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		B+
Gestionando El Error		B+
Consistencia	Visual	B+
	Funcional	A+
	Evolutiva	B+
Ayuda u orientación		NI
Minimizar la carga cognitiva		NI
Explícito control de Usuario		NI
Mapeo natural	Predecible	NI
	Lenguaje de los usuarios	NI
	Facilidad de navegación	NI
Accesibilidad	Discapacidades	M+
	Multicanal	M+
	Internacionalización	M+
Adaptabilidad	Experiencia de usuario	M+
	Personalización	B+
	Facilidad de Recordar	M+

3.2.2.4 Patrón Reflexión:

En la siguiente tabla se da la descripción del patrón así como se detallan las ventajas y desventajas y se define cuando debe ser usado:

Tabla 52 Patrón Reflexión

Nombre del patrón	Reflexión
Descripción	El patrón arquitectónico Reflexión provee un mecanismo para cambiar la estructura y el comportamiento de los sistemas software dinámicamente.
Estructura	En este patrón, una aplicación es dividida en dos partes: • Meta Nivel: Consiste en un conjunto de meta objetos, los cuales encapsulan sólo los detalles del sistema que probablemente cambian o varían de cliente a cliente. Aspectos del sistema que se espera permanezcan estables durante el tiempo de vida de una aplicación no deberían estar contenidos en este nivel. • Nivel Base: Define la lógica de la aplicación, su implementación usa la información y los servicios proporcionados por los meta objetos. Esto hace que el nivel base permanezca flexible, ya que su código es independiente de los aspectos que pueden ser sujetos a cambios y a adaptación. También se debe definir una interfaz llamada protocolo MOP[21] (Protocolo Meta Objeto) que soporta la implementación de funciones que operan en varios meta objetos, permitiendo al nivel base tener acceso a la información que mantiene o al servicio que ofrece el meta objeto. Un meta objeto no permite al nivel base modificar su estado interno, la manipulación es posible sólo a través del protocolo meta objeto. El protocolo del meta objeto es usualmente diseñado como un

[21] MOP (Protocolo Meta Objeto): Asumiendo que el nivel meta describe la arquitectura del modelo de objetos, lo que describe MOP es qué meta-objetos son necesarios para implementar dicho modelo, de modo que se proporciona soporte para la reflexión. por lo tanto el protocolo describe una interfaz, que viene dada por la combinación de las interfazs de otros meta-objetos instanciados.

	componente separado. La estructura general de una arquitectura reflexiva es mucho más como un sistema en capas. El meta nivel y el nivel base son dos capas, cada uno de los cuales proporciona su propia interfaz. La capa del nivel base específica la interfaz de usuario para explotar la funcionalidad de la aplicación. La capa del meta nivel define el protocolo del meta objeto para modificar los meta objetos.
Ventajas	• Ninguna modificación explícita del código fuente. No se necesita tocar el código existente cuando se modifica un sistema reflexivo. El propio protocolo del meta objeto es responsable de integrar sus solicitudes de cambio: realiza las modificaciones y extensiones al código del meta-nivel, y recompila las partes cambiadas y las conecta con la aplicación mientras se está ejecutando si es necesario. • Cambiar un sistema software es fácil. El protocolo del meta objeto proporciona un mecanismo seguro y uniforme para el software cambiante. • Soporte para muchos tipos de cambios. Los Meta objetos pueden encapsular cada aspecto del comportamiento del sistema, estado y estructura. Una arquitectura basada en el patrón Reflexión potencialmente soporta los cambios de cualquier tipo o escala. Incluso pueden ser cambiados los aspectos del sistema fundamentales, como los mecanismos de llamado de función o el tipo de estructuras. Con la ayuda de técnicas reflexivas es también posible adaptar el software para encontrar las necesidades específicas del entorno o para integrar los requerimientos específicos del cliente.
Desventajas	• Las modificaciones al meta nivel pueden causar serios daños al software o a su entorno. La robustez de un protocolo meta objeto es por lo tanto de gran importancia [83]. Los errores potenciales dentro de las especificaciones de cambio deben ser detectadas antes que el cambio es realizado. Cada cambio debe tener sólo un efecto limitado en otras partes del software. • Menos eficiencia. Los sistemas software reflexivos son normalmente más lentos que los sistemas no reflexivos. Esto es causado por la relación compleja entre el nivel base y el meta nivel. Siempre que el nivel base es incapaz de decidir cómo continuar con el cálculo, consulta a los meta niveles por ayuda. Esta capacidad reflexiva requiere procesamiento extra: la recuperación de información, los meta objetos cambiantes, la verificación de la consistencia, y la comunicación entre los dos niveles disminuye la ejecución global del sistema
Cuando usarlo	El patrón reflexión se recomienda para las aplicaciones Web debido a que contribuye a especificar una arquitectura abierta a la modificación y a la extensión, facilitando a la aplicación adaptase a los cambios de los requerimientos en demanda. En otras palabras, se sugiere utilizar cuando se quiere diseñar para cambios y evolución

Impacto de Usabilidad

La estructura que ofrece el patrón reflexión, facilita los cambios en una aplicación Web, haciendo que la aplicación Web pueda adaptarse a los diferentes usuarios, a las necesidades de éstos, contribuyendo considerablemente con la propiedad ***Accesibilidad: Discapacidades, Multicanal e Internacionalización,*** teniendo un impacto de usabilidad alto positivamente, esto favorece igualmente la propiedad ***Adaptabilidad: Experiencia de usuario, Personalización, Facilidad de recordar.***, ya que diferentes tipos de dispositivos pueden ser usados, y características especiales puede ser añadidas, con un impacto de usabilidad alto positivamente.

Al tener un nivel base donde se define lógica de la aplicación, puede ayudar a la propiedad ***Consistencia Funcional y Evolutiva,*** debido a que se manejará la misma funcionalidad para toda la aplicación y para otras similares, teniendo un impacto de

usabilidad medio positivamente. También tendrá implicaciones en la propiedad ***Mapeo natural: Predecible, Lenguaje de los usuarios y Facilidad de navegación***, con un impacto de usabilidad medio positivamente, ya que al manejar otro productos de la misma familia, el comportamiento del sistema puede ser intuido por el usuario, facilitando también la navegación del usuario por la aplicación.

También favorece la propiedad ***Ayuda u orientación*** ya que la ayuda se puede añadir o cambiar de una manera sencilla, teniendo un impacto de usabilidad bajo positivamente. Al poderse manejar aun en tiempo de ejecución los eventos ocurridos se puede lograr una buena ***Realimentación*** con el usuario, ya que se lo mantiene informado del estado de la aplicación, aunque el tiempo de notificación puede no ser el apropiado, teniendo entonces un impacto de usabilidad bajo positivamente.

Debido a que un sistema reflexivo es menos eficiente, el tiempo de respuesta de usuario aumentaría, lo que haría pensar al usuario que él no tiene el control de la aplicación ya que no puede hacer nada para disminuir este tiempo y obtener su respuesta rápidamente, afectando entonces la propiedad ***Explicito control de usuario***, con un impacto de usabilidad bajo negativamente. Además, puede afectar la propiedad ***Gestionando el error*** con un impacto de usabilidad medio negativamente, debido a que pueden presentarse errores al realizar los cambios.

Este patrón no tiene influencia directa sobre las propiedades ***Consistencia Visual, Minimizar la carga cognitiva***.
En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 53 Impacto de Usabilidad Patrón Reflexión

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		B+
Gestionando El Error		M-
Consistencia	Visual	NI
	Funcional	M+
	Evolutiva	M+
Ayuda u orientación		B+
Minimizar la carga cognitiva		NI
Explícito control de Usuario		B-
Mapeo natural	Predecible	M+
	Lenguaje de los usuarios	M+
	Facilidad de navegación	M+
Accesibilidad	Discapacidades	A+
	Multicanal	A+
	Internacionalización	A+
Adaptabilidad	Experiencia de usuario	A+
	Personalización	A+
	Facilidad de Recordar	A+

3.3 Patrones de Datos

Existen patrones para las bases de datos relacionales propuestos en el libro de Fowler [79] como mapeador de datos, entrada de datos a las filas, entradas de datos a la tabla, registro activo, etc. Se considera que para efectos de las propiedades de usabilidad aparentemente no tiene influencia directa. Se puede decir que según la complejidad del problema a resolver y de acuerdo al diseño que se haga de las bases de datos ya sea relacional, orientada a objetos y mixtas se puede afectar la propiedad de realimentación

positiva o negativamente debido al rendimiento del software. En general se deja abierta la posibilidad de seguir analizando estos patrones para trabajos futuros.
A continuación se presenta los patrones de acceso a datos que se consideran que tienen influencia en la usabilidad.

3.3.1 Patrón de Script de Transacción (Transaction Script)

En la siguiente tabla se da la descripción del patrón así como se detallan las ventajas y desventajas y se define cuando debe ser usado:

Tabla 54 Patrón de Transacción Script

Nombre del patrón	Transacción
Descripción	Este patrón organiza la lógica de negocio por procedimientos donde cada procedimiento maneja una sola solicitud desde la presentación, teniéndose un proceso por cada acción que vaya a realizar el usuario
Ventajas	• Es un modelo simple basado en procedimientos que la mayoría de desarrolladores pueden entender. • Uno de los beneficios de este enfoque es que no necesita preocuparse por qué están haciendo las otras transacciones. La tarea es conseguir la entrada, enviar las solicitudes a la base de datos y guardar sus resultados en la base de datos. • Es sencillo organizar la lógica de esta manera en aplicaciones simples y esto involucra muy poca sobrecarga en el rendimiento. • Se puede ver fácilmente los límites de una transacción, se observa bastante bien donde empieza y acaba una transacción, esto puede resultar muy útil para determinadas herramientas. • La relación de los scripts con las bases de datos es sencilla, ya que directamente se utiliza el lenguaje de consulta SQL, no intentando ocultar el hecho de que existe una base de datos relacional que persiste la aplicación. Incluso es posible trasladar gran cantidad de la lógica dentro del mismo motor de Bases de Datos. Así conceptos como validaciones, chequeos de integridad, manejo de concurrencia y mantenimiento de logs de auditorías se delegan al motor en lugar de programarse.
Desventajas	• Este patrón tiene desventajas y limitaciones, las cuales tienden a aparecer cuando la complejidad de la lógica del dominio aumenta. Con frecuencia se duplicará código para implementar diferentes scripts que comparten la mayoría de su funcionalidad. Esta repetición se puede arreglar de alguna manera utilizando subrutinas comunes para varios scripts de transacción, pero aún así, muchas veces será imposible evitar dicha duplicación, por lo que la aplicación será más difícil de entender y por lo tanto de mantener, dando lugar a estructuras poco claras y poco robustas. • No sirve para la reutilización. • Al permitir tener parte de la lógica en el motor de la base de datos se rompe el principio de encapsulación por lo que es difícil extender el modelo y por otro lado es propenso a errores de difícil detección a medida que se escala la solución
Cuando usarlo	Se justifica su utilización e implementación para programadores novatos o para aplicaciones Web en las que el tiempo sea el factor más importante. Además el uso de este patrón dependerá de la complejidad de la lógica de dominio de la aplicación Web, si no es tan compleja se puede usar este patrón.

Impacto de usabilidad

Este patrón permite que se cumpla la propiedad de ***Realimentación*** ya que el tiempo de notificación al usuario, será corto relativamente, y la información del estado de la aplicación será por acción o procedimiento, teniendo entonces un impacto de usabilidad medio positivamente.

En la propiedad ***Gestionando el error*** puede tener un impacto de usabilidad bajo negativamente, debido a que se puede llegar a presentar errores no fáciles de detectar por la mezcla de código y repetición que se presenta.

Debido a que implementar cambios o adicionar funcionalidad en este tipo de estructura puede ser complicado, teniendo duplicación de código, la propiedad ***Adaptabilidad: Experiencia de usuario y Personalización*** el patrón puede tener un impacto de usabilidad bajo negativamente.

Este patrón no tiene influencia sobre las propiedades ***Consistencia: Visual***, ***Funcional, Evolutiva, Ayuda u orientación, Minimizar la carga cognitiva, Explicito control de usuario, Mapeo natural: Predecible, Lenguaje de los usuarios, Facilidad de navegación, Accesibilidad: Discapacidades, Multicanal e Internacionalización, Adaptabilidad: Facilidad de recordar***
En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 55 Impacto de Usabilidad Patrón de Transacción (Transaction Script)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		M+
Gestionando El Error		B-
Consistencia	Visual	NI
	Funcional	NI
	Evolutiva	NI
Ayuda u orientación		NI
Minimizar la carga cognitiva		NI
Explícito control de Usuario		NI
Mapeo natural	Predecible	NI
	Lenguaje de los usuarios	NI
	Facilidad de navegación	NI
Accesibilidad	Discapacidades	NI
	Multicanal	NI
	Internacionalización	NI
Adaptabilidad	Experiencia de usuario	B-
	Personalización	B-
	Facilidad de Recordar	NI

3.3.2 Patrón Modelo del Dominio (Domain model)

En la siguiente tabla se da la descripción del patrón así como se detallan las ventajas y desventajas y se define cuando debe ser usado:

Tabla 56 Patrón Modelo del Dominio

Nombre del patrón	Modelo del Dominio
Descripción	Un objeto del modelo del dominio que incorpora tanto comportamiento como datos. El patrón modelo de dominio crea un conjunto interconectado de objetos donde cada uno representa algún significado particular. Este es similar a un modelo de bases de datos, aunque tiene algunas diferencias. Un modelo de dominio maneja datos y procesos, tiene atributos multivaluados y una compleja red de asociaciones y usa herencia. Un aspecto clave del *Modelo del Dominio* es el aislamiento tanto con sus capas superiores como sus inferiores. Se dice que provee un modelado transparente de la presentación y la persistencia.

Ventajas	• Utilizando las técnicas de diseño orientado a objetos, la lógica de la aplicación permite modelarse de forma tal que cambios al comportamiento sean de relativo bajo costo al proceso de desarrollo. Así pueden incorporarse al modelado conceptos como herencia y patrones de diseño comunes a este tipo de aplicaciones, esto contribuye a la extensibilidad de la aplicación. • Se mantiene una estructura bien organizada aunque se aumente la complejidad de los algoritmos o incluso cuando se proporcione otros nuevos. • Software reutilizable.
Desventajas	• Implica mayores costos sobre todo al principio del desarrollo (en especial si el equipo de desarrollo no ha trabajado con este patrón).
Cuando usarlo	En las aplicaciones Web donde la lógica de dominio del sistema sea compleja, se recomienda usar este patrón, por otro lado, si se tiene una lógica simple, el patrón Transacción es un buena opción.

Impacto de Usabilidad

Sobre la propiedad ***Realimentación*** tiene un impacto de usabilidad medio positivamente debido a que la estructura que brinda el patrón permite mantener al usuario informado del estado del sistema.

Este patrón tiene un impacto de usabilidad medio positivamente en la propiedad ***Mapeo natural****:* ***Facilidad de navegación,*** debido a que una estructura bien constituida, facilita la navegación del usuario a través de la aplicación.

La gran ventaja del *Modelo del Dominio* es la extensibilidad, que según Meyer [80] es la facilidad de como los productos de software se adaptan a los cambios en la especificación. La extensibilidad es fundamental en un sistema de software debido a que factores humanos pueden hacer que sea necesario cambiar el sistema. Por lo anterior este patrón tiene un impacto de usabilidad medio positivamente en las propiedades de ***Adaptabilidad: Experiencia de usuario, Personalización,*** *y*a que será más fácil implementar los cambios, y adaptar la aplicación a los distintos perfiles de usuario. Esto también hace que sobre la propiedad ***Accesibilidad: Discapacidades, Multicanal e Internacionalización,*** tenga un impacto de usabilidad bajo positivamente, ya que los cambios para que una aplicación pueda cumplir con estas características pueden llegar a ser sencillos.

Al implementar este patrón tiene un impacto de usabilidad medio positivamente sobre la propiedad ***Gestionando el error***, debido a que el tener una estructura bien organizada ayudará a prevenir los errores y a corregirlos; esto, además, beneficia las propiedades ***Ayuda u orientación*** y ***Adaptabilidad: Facilidad de recordar el sistema***, con un impacto de usabilidad bajo positivamente.

En la propiedad ***Consistencia: Funcional y Evolutiva,*** tienen un impacto de usabilidad medio positivamente debido a que el tener código reutilizable puede incidir en un tipo de estructura similar tanto en la aplicación como en la familia de productos; influyendo también aunque en menor grado en la propiedad ***Consistencia Visual,*** con un impacto bajo positivamente

Sobre la propiedad ***Minimizar la carga cognitiva*** puede tener un impacto de usabilidad bajo positivamente, debido a que los cambios necesarios para que la aplicación sea simple al usuario, serán mas sencillos de hacer con este tipo de estructura, sin tener duplicación de código.

Este patrón no tiene influencia sobre las propiedades ***Explicito control de usuario***, ***Mapeo natural: Lenguaje de los usuarios y Predecible.***
En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 57 Impacto de Usabilidad Patrón Modelo del Dominio (Domain model)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		M+
Gestionando El Error		M+
Consistencia	Visual	B+
	Funcional	M+
	Evolutiva	M+
Ayuda u orientación		B+
Minimizar la carga cognitiva		B+
Explícito control de Usuario		NI
Mapeo natural	Predecible	NI
	Lenguaje de los usuarios	NI
	Facilidad de navegación	M+
Accesibilidad	Discapacidades	B+
	Multicanal	B+
	Internacionalización	B+
Adaptabilidad	Experiencia de usuario	M+
	Personalización	M+
	Facilidad de Recordar	B+

3.3.3 Patrón Modulo de Tabla (Table Module)

Un punto intermedio entre la vista procedimental que brindan los Scripts y la vista orientada a objetos y transparente a la persistencia del modelo del dominio es la del módulo de tabla.
En la siguiente tabla se da la descripción del patrón así como se detalla su estructura, ventajas y desventajas y se define cuando debe ser usado:

Tabla 58 Patrón Modulo de Tabla

Nombre del patrón	Modulo de Tabla
Descripción	En este patrón se tiene una sola instancia para manejar la lógica de negocio de todas las filas de una tabla en la base de datos. Por cada tabla, vista o procedimiento existe una clase en la aplicación Web que la manipula. A diferencia del modelo del dominio las instancias de estas clases representan a las tablas y no las filas de la base de datos.
Ventajas	• Organizar la lógica de dominio en tablas, más que en procedimientos, da lugar a arquitecturas más estables y evita problemas como la duplicación de código. • Encaja muy bien con el resto de la arquitectura. Muchos entornos GUI están construidos para trabajar con los resultados producidos por consultas SQL organizadas en conjunto de registros. • Permite empaquetar los datos y comportamiento juntos y al mismo tiempo toma las fortalezas de una base de datos relacional. • Uno de los beneficios de este estilo es que se puede probar el Modulo de la tabla creando un conjunto de registros en la memoria sin ir a la base de datos.
Desventajas	• No soporta conceptos como la herencia, las estrategias y otros patrones OO. • Depende del soporte que se tenga en la plataforma de desarrollo para el conjunto de registros
Cuando usarlo	El Modulo de Tabla es muy útil cuando es necesaria la persistencia de datos y se cuenta con tecnología que soporte conjunto de registros.

Impacto de Usabilidad

En las propiedades ***Consistencia Funcional y Evolutiva,*** tiene un impacto de usabilidad medio positivamente debido a que este patrón brinda una estructura más estable en comparación con el Patrón Transacción, lo que permitirá utilizar la lógica en diferentes partes la aplicación. Por lo anterior, sobre las propiedades ***Minimizar la carga cognitiva*** y ***Consistencia Visual,*** tiene un impacto de usabilidad bajo positivamente, debido a la facilidad con la que se pueda llegar a realizarse cambios o adiciones para mejorar la interfaz.

Este patrón contribuye con la propiedad ***Adaptabilidad: Experiencia de usuario y Personalización,*** ya que cuando se requiera adaptar la aplicación Web, no se tendrá problemas con la duplicación de código, al querer adicionar nuevas funcionalidades por ejemplo; sin embargo, realizar los cambios o adiciones puede ser aun más complejo que con el patrón modelo de dominio, ya que no se tendrán características de la programación orientada a objetos, por lo que el impacto de usabilidad será bajo positivamente.

En la propiedad ***Gestionando el error*** puede tener un impacto de usabilidad bajo positivamente debido a que este patrón facilitará las pruebas de la aplicación Web, que son necesarias para prevenir los errores que puedan ocurrir cuando el usuario utilicé el sistema.

En las propiedades ***Realimentación*** y ***Ayuda u orientación*** tiene un impacto de usabilidad bajo positivamente, debido a que la estructura que brinda el patrón permite mantener al usuario informado del estado del sistema; además, contribuye con la propiedad ***Mapeo natural: Facilidad de navegación***, al tener una arquitectura mas estable.

Este patrón no tiene influencia sobre las propiedades ***Explicito control de usuario, Mapeo natural: Predecible, Lenguaje de los usuarios, Accesibilidad: Discapacidades, Multicanal e Internacionalización y Adaptabilidad: Facilidad de recordar el sistema.***

En la siguiente tabla se puede ver el impacto que tiene este patrón sobre las propiedades de usabilidad de manera resumida:

Tabla 59 Impacto de Usabilidad Patrón Módulo de la Tabla (Table Module)

PROPIEDADES DE USABILIDAD		Impacto de Usabilidad
Realimentación		B+
Gestionando El Error		B+
Consistencia	Visual	B+
	Funcional	M+
	Evolutiva	M+
Ayuda u orientación		B+
Minimizar la carga cognitiva		B+
Explícito control de Usuario		NI
Mapeo natural	*Predecible*	NI
	Lenguaje de los Usuarios	NI
	Facilidad de navegación	M+
Accesibilidad	Discapacidades	NI
	Multicanal	NI
	Internacionalización	NI
Adaptabilidad	Experiencia de usuario	B+
	Personalización	B+
	Facilidad de Recordar	NI

¿Qué patrón utilizar para acceso a datos?

En cada arquitecto de software reside la decisión de qué patrón utilizar ante una lógica de dominio determinada, sin embargo debido a la influencia positiva que tiene sobre la usabilidad y las ventajas que ofrece manejar el paradigma orientado a objetos el patrón Modelo de Dominio seria el mas recomendado, aunque cada equipo de desarrollo debe analizar las ventajas y desventajas de cada patrón y elegir el que mejor se ajuste a sus necesidades y entorno.

3.4 Conclusiones del capitulo

Al terminar este capítulo se puede concluir que:

- Los patrones de arquitectura tienen una influencia directa sobre las propiedades de usabilidad, así como los patrones de interacción y de hipermedia, en la medida que estos sean correctamente utilizados la usabilidad de la aplicación Web mejorará considerablemente.
- Los patrones a utilizar deben seleccionarse de acuerdo a los requerimientos que conforman la arquitectura de la aplicación Web, una manera de establecer que patrones utilizar es referenciándose al cuando usarlos asi como también es importante guiarse por la clasificación que en este proyecto se presenta ya que será mucho más sencillo determinar que patrón utilizar y en donde
- El análisis de usabilidad presentado sirve para tener una claridad de que decisiones se deben tomar a nivel arquitectonico en el desarrollo de la aplicación Web, para cumplir con las propiedades de usabilidad así como también con los atributos de usabilidad relacionados con dichas propiedades.
- La forma en la que se utilicen los patrones impactará positiva o negativamente la usabilidad de la aplicación.

Capítulo 4 PROCESO DE INGENIERIA PARA DESARROLLO DE APLICACIONES EN LA Web, TENIENDO EN CUENTA ASPECTOS DE USABLIDAD

De acuerdo a los estudios realizados por [84], [85] se puede concluir que las razones principales que obligan a la inclusión de nuevos procesos para el desarrollo de aplicaciones Web y adaptación de los existentes son:

1. La complejidad de las aplicaciones Web: Debido a la existencia de hiperenlaces e hipervínculos, lo que puede causar que el usuario pueda llegar a perderse en el hiperespacio si el sistema no se construye basado en alguna técnica que garantice la calidad del mismo [86].
2. El número de usuarios: Mientras que en los sistemas clásicos el número de usuarios finales estaba normalmente definido y cerrado o al menos controlado a un grupo de usuarios específicos, en los sistemas Web la visión cambia. El número de usuarios no es sólo ilimitado, si no que, además, es totalmente impredecible en la mayoría de los casos, por lo que la usabilidad de las aplicaciones Web tiende a ser un factor crítico [87].
3. Falta de conocimiento: El estudio realizado por [88] ha descubierto que la mayoría de las metodologías existentes no están siendo usadas por los profesionales debido a que no las conocen.
4. Las aplicaciones Web necesitan ser rápidamente desarrolladas, robustas y fáciles de usar.
5. La mayoría de metodologías para aplicaciones Web son muy complejas y extensas como por ejemplo: WSDM [11], HFPM [12], OOHDM [13], NDT [16].

Debido a las razones antes expuestas, se propuso en este trabajo de grado desarrollar un proceso inicial de ingeniería para desarrollo de aplicaciones en la Web, el cual abarca las etapas de requerimientos, análisis y diseño en las cuales la definición de la arquitectura tiene un papel relevante, dicho proceso ha sido definido con base en la investigación realizada sobre las metodologías existentes que tienen en cuenta por lo menos uno de los temas pilares de este proyecto de grado, es decir, Usabilidad, Aplicaciones Web, Arquitecturas de Software o MiPymes; Así como también es complementado con los patrones expuestos en el capítulo 3.

En la Figura 2 se pueden ver las etapas que componen este proceso:

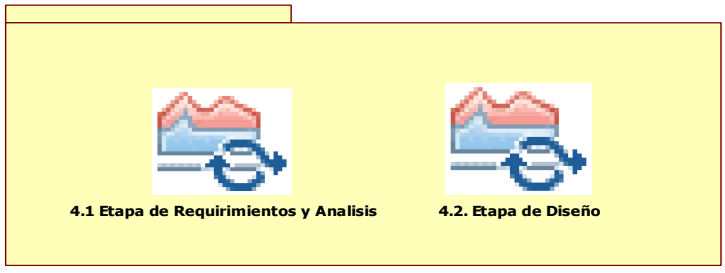


Figura 2 Proceso inicial de ingeniería para desarrollo de aplicaciones en la Web, teniendo en cuenta aspectos de usabilidad

4.1 Etapa de Requerimientos y Análisis

A continuación en la Figura 3 se detallan las etapas con sus respectivas actividades y tareas propuestas en el proceso de desarrollo. Para la descripción gráfica presentada por actividad se utilizó SPEM [89].

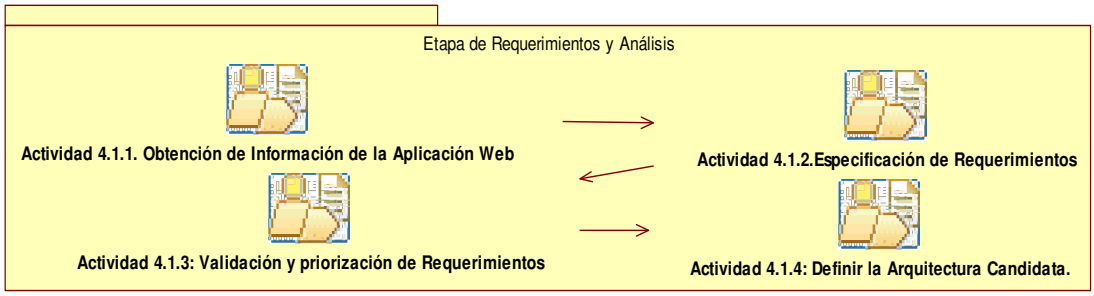


Figura 3 Etapa de requerimientos y análisis

El concepto que en última instancia establece la calidad de un sistema software viene determinado a partir de la concordancia entre los requerimientos fijados y la consecución de los mismos. De allí la importancia que esta etapa tiene en el proceso de desarrollo de aplicaciones Web.

Durante la etapa de requerimientos y análisis se define el modelo del negocio y el alcance del proyecto. Se identifican todos los actores y Casos de Uso (aproximadamente el 20% del modelo completo). Para lograr estos objetivos, es importante entender la definición y alcance del problema que se trata de resolver con la aplicación Web; así como el entorno organizacional en el que se desenvuelven lo usuarios finales. Esta información se obtiene mediante reuniones con los clientes y potenciales usuarios, preferiblemente en el contexto donde éstos trabajan habitualmente, en las cuales se desarrollan documentos (Acta de Reunión de Requerimientos) con las solicitudes de los clientes y/o usuarios.

A partir de las actas de reuniones de requerimientos se describe completamente el sistema (lo que debe hacer el sistema) mediante los documentos: Especificación de Requerimientos, Modelo de Casos de Uso y Alcance del Sistema. Dichos documentos incluyen las características que serán incluidas y cuales consideradas pero no incluidas en el sistema a construir y proporcionan una base contractual para los requerimientos que serán visibles para los clientes y/o usuarios.

4.1.1 Actividad Obtención de Información de la Aplicación Web

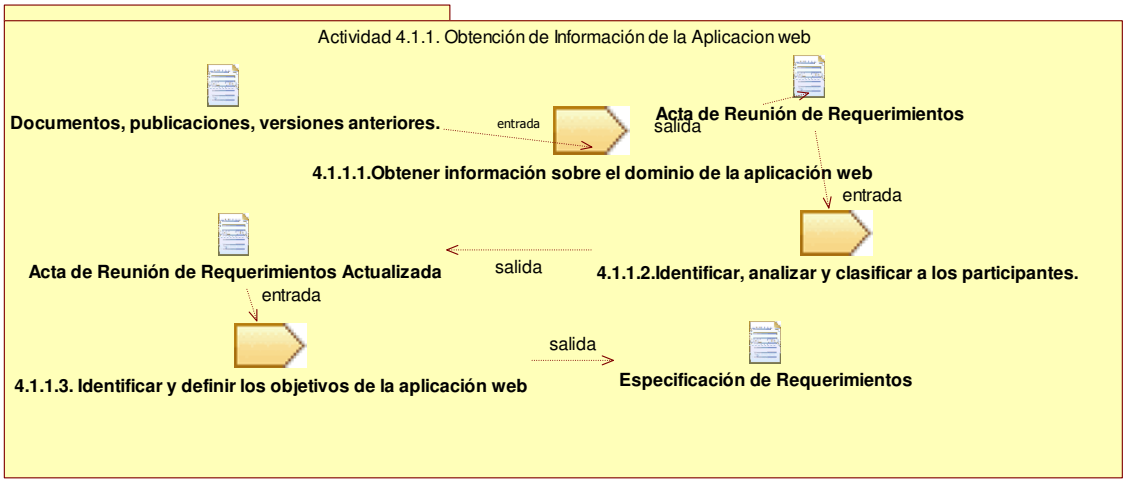


Figura 4 Obtención de Información de la Aplicación Web

En la figura 4 se aprecian las tareas correspondientes a ésta actividad con sus correspondientes entradas y salidas.
En esta actividad se obtendrá la diferente información de la aplicación Web a desarrollar, tal como la información sobre: el dominio del problema que resuelve la aplicación Web, los participantes, y los objetivos de la aplicación. Está información es necesaria para un buen inicio del proceso de desarrollo.

Objetivos:

- Conocer el dominio del problema.
- Identificar, analizar y clasificar las diferentes personas involucradas en el proyecto.
- Identificar los objetivos generales que se pretenden alcanzar con el desarrollo de la aplicación Web.

Tareas

1. Obtener información sobre el dominio de la aplicación Web.
2. Identificar, analizar y clasificar a los participantes.
3. Identificar y definir los objetivos de la aplicación Web.

Entradas

- Artículos, documentos, publicaciones, etc. sobre el campo específico del negocio.
- Si hay, versiones anteriores o predecesoras a la aplicación Web que se va a desarrollar en el negocio.

Salidas:

- Acta de Reunión de Requerimientos.
- Especificación de Requerimientos.

Responsable: Analista

Participantes:

- Cliente y/o Usuarios
- Equipo de desarrollo.

4.1.1.1 Obtener Información sobre el dominio de la Aplicación Web

Esta tarea es especialmente crítica cuando la aplicación Web es pionera en el entorno donde se va implantar, puesto que los clientes y/o usuarios no tienen experiencias previas que ayuden al entendimiento con el equipo de trabajo a la hora de indicar las necesidades. Y por otro lado, tampoco el equipo de desarrollo es experto en el tema [90].

Objetivos:

- Conocer el dominio del problema, antes de comenzar las reuniones y entrevistas con los clientes.
- Conocer y familiarizarse con la terminología con las que los clientes y/o usuarios están acostumbrados a trabajar.

Entradas:

- Artículos, documentos, publicaciones, etc. sobre el campo específico del negocio.
- Si hay, versiones anteriores o predecesoras a la aplicación Web que se va a desarrollar en el negocio.

Como llevar a cabo esta tarea:

- Se debe realizar una recopilación de artículos, documentos, publicaciones, etc. sobre el campo específico del negocio, ya sean internos o externos al sistema.
- Si hay versiones anteriores o predecesoras a la aplicación Web que se va a desarrollar en el negocio, se recomienda solicitar la información respectiva, puesto que las experiencias sobre estas van a ser las bases sobre las que los clientes van a realizar sus peticiones.
- Cuando el entorno del negocio sea muy específico, se aconseja la realización de entrevistas o visitas previas para conocer el dominio de la aplicación a realizar.

- Registrar la información obtenida en esta tarea, en la plantilla Acta de Reunión de Requerimientos (Anexo 6).

Salidas:

- Acta de Reunión de Requerimientos (Anexo 6).

Nota: *Por* otro lado, cuando el equipo de desarrollo conoce el dominio de la aplicación, esta tarea no es necesaria. Realmente sólo es necesaria en el desarrollo de sistemas de información para entornos no conocidos por el equipo.

4.1.1.2 Identificar, analizar y clasificar a los participantes.

Los participantes son personas u organizaciones que serán afectadas por la aplicación Web y que tienen influencia directa o indirecta en los requerimientos de la misma [91]. Existen varias propuestas para clasificar los participantes de un sistema interactivo, entre las cuales se tiene, la que se basa en dividirlos entre aquellos que utilizarán la aplicación Web directa o indirectamente [92]:

- Directamente:
  - Ingenieros de software responsables del desarrollo.
  - Usuarios finales.
- Indirectamente:
  - Directores o personas que son responsables del trabajo de los usuarios finales.
  - Y los que están relacionados con el desarrollo del sistema.
  - Socios y/o proveedores tecnológicos.

Objetivos:

- Descubrir y clasificar todos los participantes, incluso aquellos que puedan influir negativamente en el proyecto.

Entradas:

- Acta de Reunión de Requerimientos (Anexo 6)

Como llevar a cabo esta tarea:

- Utilizar técnicas participativas como la Observación de Campo [74], puesto que no es lo mismo intentar identificar dichos usuarios en el lugar donde la acción se realiza que fuera de ella.
- Identificar los perfiles de usuario: El perfil de usuario responde a criterios de tipos de usuarios en cuanto a sus capacidades y habilidades dando lugar a los perfiles de usuarios que agrupan características similares. Existen varios métodos para obtener el perfil de los usuarios, siendo los cuestionarios y las entrevistas los más utilizados [74]. Para las entrevistas y los cuestionarios se recomienda tener en cuenta aspectos como: el grado de conocimiento/uso de equipos/programas informáticos que tienen los usuarios, la experiencia profesional, el nivel de estudios, la experiencia en el puesto o tipo de trabajo, el entorno social, etc.
- Identificar los roles de los usuarios: Los roles indican clases de usuarios que tienen asignados ciertos subconjuntos de tareas, ya sea por elección propia o como resultado de la organización en la que se encuentran [93]. Por definición, los roles están orientados a las funcionalidades de la aplicación. Se debe tener en cuenta que más de un usuario puede estar involucrado en un mismo rol y un mismo usuario puede tener varios roles al mismo tiempo.
- Utilizar la plantilla Acta de Reunión de Requerimientos (Anexo 6).

Salida:

- Acta de Reunión de Requerimientos Actualizada (Anexo 6)

Nota: Existe una estrecha relación entre los perfiles de usuarios y los roles que éstos desenvuelven. La relación entre ambas clasificaciones es del tipo *n.m*, o sea, que uno o más perfiles de usuarios puede estar asociado a uno o más roles y viceversa.

Es importante tener en cuenta:

- Los perfiles de usuario más adelante se plasmarán en los actores y las tareas que realizan los usuarios estarán asociadas a los roles en los casos de uso [94].
- Si se logra clasificar acertadamente a los usuarios finales de acuerdo al perfil y al rol, la disposición de las funcionalidades estará adaptada al modelo mental de los distintos perfiles de usuarios, favoreciendo la usabilidad de la aplicación Web [94].

4.1.1.3 Identificar y definir los objetivos de la aplicación Web.

Tras haber realizado las entrevistas a los clientes y usuarios, es necesario destacar cuáles son los objetivos que se pretenden conseguir cuando el sistema se encuentre en la etapa de implantación. A medida que se va desarrollando la especificación de requerimientos, los objetivos se pueden ir refinando y concretando, de manera que cada vez se vayan identificando mejor los requerimientos del sistema. Se debe recordar que un requisito no es más que una necesidad que el sistema debe cubrir para poder alcanzar uno o varios objetivos impuestos por el usuario [95], [96]

Objetivos:

- Determinar los objetivos del sistema.
- Determinar los objetivos prioritarios del sistema.
- Revisar en caso que haya conflictos, los objetivos previamente identificados.

Entradas

- Acta de Reunión de Requerimientos

Como llevar a cabo esta tarea:

- Utilizar la plantilla para definir objetivos del sistema que se encuentra en Especificación de Requerimientos (Anexo 7) propuesta por [97].

Salidas

- Especificación de Requerimientos (Anexo 7) propuesta por [97].

Nota: Si no fuera posible realizar la reunión con los implicados la información puede recogerse mediante entrevistas individuales a los implicados identificados. En este caso, el principal inconveniente es que no se tendrá ninguna oportunidad de establecer un consenso entre todos ellos.

4.1.2 Actividad Especificación de Requerimientos

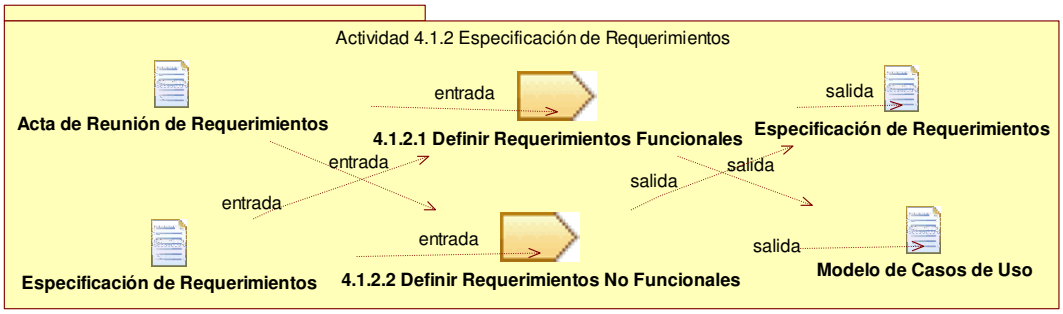


Figura 1 Especificación de Requerimientos

En la figura 4 se aprecian las tareas correspondientes a ésta actividad con sus correspondientes entradas y salidas
En esta actividad se debe entender qué desea el Cliente y/o Usuario, que haga el producto y generar un acta con los requerimientos [98].

La Especificación de Requerimientos y el Modelo de Casos de Uso sirven de contrato sobre la funcionalidad del sistema, entre el cliente, los usuarios y el equipo de desarrollo. Esto permite a los clientes y/o usuarios validar que la aplicación será como ellos esperan, y al equipo de desarrollo construir lo que se espera.
Después de haber hecho un estudio de los requerimientos que se mencionan en las diferentes metodologías para el desarrollo de aplicaciones Web, se han considerado los mas pertinentes para el desarrollo de aplicaciones Web. Además se ha introducido uno nuevo los requerimientos de usabilidad; debido a que el proceso de desarrollo expuesto en este capítulo está centrado en tener en cuenta aspectos de usabilidad, los requerimientos de usabilidad se tratarán con más detalle, sin embargo se proporcionará una breve explicación de los demás.

1 **Requerimientos funcionales:** Recogen qué debe hacer el sistema de forma interna, sin incluir aspectos de interfaz o interacción. También son conocidos en el ambiente Web como requerimientos de servicios [17].
   1.1 **Requerimientos de interacción:** Definen cómo se muestra la información, cómo se podrá navegar en la aplicación Web, los criterios de recuperación que se ofrecen y la funcionalidad a la que se les permite tener acceso a los actores [99], [100].
2 **Requerimientos no funcionales**: Recogen otros requerimientos del sistema, por ejemplo: requerimientos de comunicaciones del sistema, de fiabilidad, de entorno de desarrollo, de portabilidad, entre otros. Además estos imponen restricciones a los requerimientos funcionales relacionadas con la eficiencia, consistencia, reusabilidad, flexibilidad, adecuación a estándares, etc. [101].
   2.1 **Requerimientos de usabilidad**: Determinan qué se va a entender por un nivel aceptable de utilización y de aceptación del producto final por parte del usuario [60].
   2.2 **Requerimientos de almacenamiento de información**: Estos requerimientos responden a la pregunta: ¿Qué información debe almacenar y administrar la aplicación Web?.

Objetivo

- Especificar los requerimientos de la aplicación Web a construir.
- Proporcionar las bases para la planificación de cada iteración.
- Proporcionar las bases para estimar costo y tiempo de desarrollo.

Nota: Este proceso depende en muchos casos de la intuición personal del analista y del conocimiento que posea sobre el dominio de la aplicación; además, de la facilidad del cliente para transmitir sus necesidades resulta de gran importancia, dado que en ocasiones el analista no conoce en profundidad el dominio de la aplicación Web a desarrollar y/o el cliente no presenta la habilidad de saber transmitir correctamente sus necesidades [102].

Tareas:

1 Definir requerimientos funcionales.
   1.1 Definir requerimientos de interacción
2 Definir requerimientos no funcionales.
   2.1 Definir requerimientos de usabilidad.
   2.2 Definir requerimientos de almacenamiento de información.

Entradas:

- Acta de Reunión de Requerimientos.
- Especificación de Requerimientos.

Salidas:

- Especificación de Requerimientos Actualizados.
- Modelo de Casos de Uso

Responsable: Analista
Participantes:

- Cliente y/o usuario, Equipo de desarrollo

4.1.2.1 Definir Requerimientos funcionales.

Los requerimientos funcionales responden a la pregunta ¿qué se puede hacer en la aplicación Web?, es decir que se debe definir qué se va a poder hacer con la información de la aplicación Web y las posibles funcionalidades [90].

Objetivos:

- Definir los requerimientos funcionales, expresados como casos de uso, que deberá cumplir la aplicación Web a desarrollar.
- Identificación de conflictos, en los requerimientos funcionales definidos.

Entradas:

- Acta de Reunión de Requerimientos.
- Especificación de Requerimientos.

Como llevar a cabo esta tarea:

- Se deben definir los actores de la aplicación Web, estos pueden ser definidos con base en los perfiles de usuario y los roles identificados en la actividad 2. Además se recomienda utilizar la plantilla para la definición de actores que se encuentra en Modelo de Casos de Uso, ver Anexo 8.
- Para la parte gráfica utilizar Los diagramas de casos de uso [10] . En ellos aparecen dos elementos importantes, el caso de uso en si y los actores.
- Utilizar la plantilla para casos de uso que se encuentra en Modelo de Casos de Uso, ver Anexo 8.

Salidas:

- Especificación de Requerimientos (Anexo 7)
- Modelo de Casos de Uso (Anexo 8).

4.1.2.1.1 Definir Requerimientos de interacción

Los requerimientos de interacción se refieren a la manera como los actores van a interactuar con la aplicación Web durante la navegación. Incluyendo varios aspectos como la forma en la que se visualizaran los datos, las posibilidades de navegación y de ejecución de la funcionalidad o la manera en la que se recupera la información [90].

Los Requerimientos de interacción vienen dados por dos aspectos: las Frases y Prototipos de Visualización. Las frases representan un criterio de recuperación de información en el sistema. Por otra parte, los prototipos de visualización hacen referencia a qué datos se le muestran a cada uno de los actores y qué funcionalidad se le asocia a cada módulo de presentación de la información. Para entender mejor los conceptos se puede ver el ejemplo en el anexo 7.

Objetivos:

- Identificar y definir las frases
- Identificar y definir los prototipos de visualización

Entradas:

- Acta de Reunión de Requerimientos.
- Especificación de Requerimientos.

Como llevar a cabo esta tarea:

- Utilizar la plantilla para la definición de frases que se encuentra en Especificación de Requerimientos ver anexo 7.
- Utilizar la plantilla para recolección de prototipos de visualización que se encuentra en Especificación de Requerimientos ver anexo 7.

Salidas:

- Especificación de Requerimientos (Anexo 7)

4.1.2.2 Definir Requerimientos no funcionales.

En cualquier aplicación Web aparecen una serie de necesidades que no se pueden catalogar entre los requerimientos anteriores. En esta tarea se identifican y recogen todas estas necesidades que han quedado fuera de las clasificaciones anteriores.
Dentro de los requerimientos no funcionales de interés para aplicaciones Web, se pueden incluir las siguientes: [97]:

Requerimientos de comunicaciones del sistema: Son Requerimientos de carácter técnico relativos a las comunicaciones que deberá soportar la aplicación Web a desarrollar. Por ejemplo: la aplicación deberá utilizar el protocolo TCP/IP para las comunicaciones con otros sistemas
Requerimientos de fiabilidad: Son los que establecen los factores que se requieren para la fiabilidad del software en tiempo de explotación. La fiabilidad mide la probabilidad del sistema de producir una respuesta satisfactoria a las demandas del usuario. Por ejemplo, determinar qué tasa de fallos máxima por meses se permite.
Requerimientos de entorno de desarrollo: Describen si existen restricciones en las herramientas, lenguajes de programación, sistemas operativos, etc. que se van a usar en el desarrollo del sistema.
Requerimientos de portabilidad: Definen qué características debe tener el software para que sea fácil de usar en otro entorno.
Además en esta categoría también se encuentran los requerimientos de usabilidad y de almacenamiento de información los cuales se describen en detalle más adelante.

Objetivos:

- Identificar los requerimientos no funcionales de la aplicación Web a desarrollar.
- Revisar, en el caso de que haya conflictos, los requerimientos no funcionales previamente identificados.

Entradas:

- Acta de Reunión de Requerimientos.
- Especificación de Requerimientos.

Como llevar a cabo esta tarea:

- Utilizar la plantilla para Requerimientos no funcionales que se encuentra en Especificación de Requerimientos ver anexo 7.

Salidas*:*

- Especificación de Requerimientos (Anexo 7)

4.1.2.2.1 Definir Requerimientos de usabilidad.

La usabilidad es vista generalmente para asegurar que los productos interactivos sean fáciles de aprender, efectivos y agradables para sus usuarios. Para cumplir con tal característica se debe tener en cuenta requerimientos de usabilidad al inicio del desarrollo de la aplicación Web, debido a que la usabilidad no debe tenerse en cuenta solo al final del sistema sino también en las etapas iníciales.

Además la suficiencia o carencia de usabilidad en una aplicación Web contribuirá al éxito o al fracaso de la misma, es por todo esto que a los requerimientos de usabilidad se les debe dar una prioridad alta, y son tan importantes como los demás requerimientos.
La definición de los requerimientos de usabilidad se ha realizado a partir de las propiedades de usabilidad presentadas en el capítulo 3, debido a que estas expresan o representan las heurísticas y los principios de diseño que los investigadores en el campo de usabilidad han encontrado, tienen una influencia directa sobre la usabilidad de una aplicación.

Objetivo:

- Especificar los requerimientos de usabilidad.
- Contrastar los requerimientos de usabilidad con los demás para analizar si alguno entra en conflicto.

Entradas:

- Acta de Reunión de Requerimientos.
- Especificación de Requerimientos.

Como llevar a cabo esta tarea:

- Utilizar la plantilla para Requerimientos de usabilidad la cual se encuentra en Especificación de Requerimientos (Anexo 7).

Salidas:

- Especificación de Requerimientos (Anexo 7)

Nota: Un aspecto importante a considerar, tal como se ha mencionado antes, es que suele darse el caso de que algún objetivo funcional entra en conflicto con otro de usabilidad.
Esta situación se resuelve, por ejemplo, con un preciso análisis de los pros y contras, en una reunión con los participantes de la aplicación Web.

4.1.2.2.2 Definir Requerimientos de Almacenamiento de Información.

A partir de las entrevistas realizadas con los diferentes usuarios y los objetivos identificados en la Actividad Obtención de Información de la Aplicación Web se definirán los requerimientos de almacenamiento de información.

Cada requisito de almacenamiento de información representa un concepto relevante para el que es necesario almacenar información, así como su estructura, significado y las restricciones o reglas de negocio que deberá cumplir dicha información, y a la vez se debe identificar, o revisar si existen conflictos [16].

***Objetivos*:**

- Identificar los requerimientos de almacenamiento de información que deberá cumplir el sistema.
- Identificar los requerimientos de restricciones de información o reglas de negocio que deberá cumplir el sistema software.
- Revisar, en el caso de que haya conflictos, los Requerimientos de almacenamiento y/o de restricciones de información previamente identificados.

Entradas:

- Acta de Reunión de Requerimientos.
- Especificación de Requerimientos.

Como llevar a cabo esta tarea:

- Utilizando la plantilla para requerimientos de almacenamiento de información, la cual se encuentra en Especificación de Requerimientos (Anexo 7), en este se explican los campos que pueden ser difíciles de entender y se proporciona un ejemplo.

Salidas:

- Especificación de Requerimientos (Anexo 7)

4.1.3 Actividad Validación y Priorización de Requerimientos

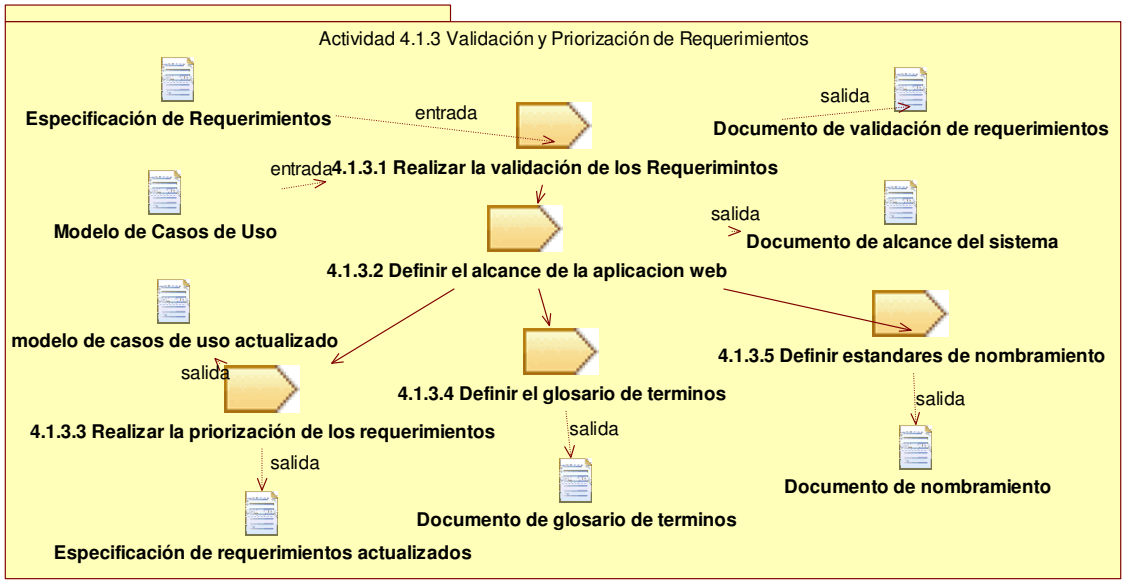


Figura 5 Validación y Priorización de Requerimientos

En la figura 5 se aprecian las tareas correspondientes a ésta actividad con sus correspondientes entradas y salidas.
Después de especificar los requerimientos de la aplicación Web es necesario validarlos con el cliente y/o usuario, para confirmar si los Requerimientos corresponden con lo que el usuario desea de la aplicación. Además de detectar y resolver incongruencias, Incompatibilidades, mal entendidos o errores [90], [98].

Luego se puede definir el alcance del sistema, y seguir con la priorización de los requerimientos, adicionalmente se puede generar el documento de glosario de términos y estándares.

Objetivo

- Realizar la validación y priorización de todos los requerimientos de la aplicación Web a construir.
- Describir el alcance del sistema
- Definir el glosario de términos.
- Definir los estándares de nombramiento.

Tareas:

1 Realizar la validación de los Requerimientos.
2 Definir el alcance de la aplicación Web.
3 Realizar la priorización de los Requerimientos.
4 Definir el glosario de términos.
5 Definir estándares de nombramiento

Entradas:

- Especificación de Requerimientos
- Modelo de Casos de Uso

Salidas:

- Documento de Validación de Requerimientos
- Documento de Alcance del Sistema
- Documento de Especificación de Requerimientos actualizado con la priorización de Requerimientos
- Documento de Glosario de términos
- Documento de Nombramiento.

Rol responsable: Analista
Participantes:

- Cliente y/o Usuario; Líder del proyecto o Gerente de proyecto; Arquitecto; Ingeniero de calidad.

4.1.3.1 Realizar la validación de los requerimientos

Objetivo:

- Validar los requerimientos especificados con el cliente y/o usuario.

Entradas:

- Especificación de Requerimientos
- Modelo de Casos de Uso

Como llevar a cabo esta tarea:

- Realizar reuniones con el cliente y/o usuario para validar los requerimientos que se especificaron, utilizando por ejemplo una lista de chequeo en la que aparezca los requerimientos y una columna que indique si es conforme o no con lo que el cliente desea.
- Realizar una matriz de rastreabilidad o trazabilidad, esta es una técnica que permite validar los resultados obtenidos en la especificación de Requerimientos [97]. La matriz de rastreabilidad es una tabla en la que se presentan enfrentados los objetivos y los Requerimientos, de forma que se pueda detectar cuales son los objetivos que son alcanzados, bien parcialmente bien en su totalidad, al cumplirse un requisito del sistema. Para esto se puede hacer uso de la plantilla matriz de rastreabilidad que se presenta en Validación de Requerimientos, ver Anexo 9.

Salidas

- Documento de Validación de Requerimientos.

Nota: De la reunión de Validación de Requerimientos pueden surgir cambios en los requerimientos que impliquen que se realicen nuevamente la actividad de Especificación de Requerimientos.

4.1.3.2 Definir el Alcance de la Aplicación Web

Dada la especificación de los requerimientos y la aprobación de los mismos por el Cliente y/o usuario se procede a establecer cuáles de los requerimientos especificados se implementarán [98], definiendo de esta manera el alcance de la aplicación Web.

Objetivo

- Definir el alcance de la aplicación Web a construir.

Entradas:

- Especificación de Requerimientos
- Modelo de Casos de Uso
- Documento de Validación de Requerimientos.

Como llevar a cabo esta tarea:

- Realizar un documento donde se especifique, en lenguaje natural, lo que comprende y de que trata el sistema a construir.
- Utilizar el documento de Alcance del Sistema, ver Anexo 10.

Salidas

- Documento de Alcance del Sistema.

4.1.3.3 Realizar la priorización de los requerimientos.

Esta tarea es primordial ya que la priorización de los requerimientos determinara como se comenzara a desarrollar la aplicación Web a construir [98]. Es necesario que los requerimientos de la aplicación Web se prioricen, teniendo en cuenta las necesidades y expectativas del cliente.

Objetivo

- Priorizar los requerimientos de la aplicación Web a construir.

Entradas:

- Especificación de Requerimientos
- Modelo de Casos de Uso
- Documento de Validación de Requerimientos.
- Documento de Alcance del Sistema.

Como llevar a cabo esta tarea:

- Enumerar los requerimientos que se especificaron.
- Definir en cuantas iteraciones se realizará la aplicación.
- Asignar prioridades a los requerimientos, utilizando el campo prioridad de cada una de las plantillas utilizadas en los anexos 7 y 8: Especificación de Requerimientos y Modelo de Casos de Uso.
- Agrupar requerimientos por cada iteración de acuerdo a la prioridad asignada.

Salidas*:*

- Especificación de Requerimientos Actualizados
- Modelo de Casos de Uso Actualizado.

4.1.3.4 Definir el glosario de términos.

El Glosario es un documento complementario que define la terminología común usada en todo el proyecto. A medida que se van generando documentos, todos los términos utilizados deben ser explicados [98], para el correcto entendimiento entre los participantes.

Objetivo

- Definir los términos utilizados en el proyecto para que sea entendible la lectura de los mismos.

Entradas:

- Especificación de Requerimientos
- Modelo de Casos de Uso
- Documento de Validación de Requerimientos.
- Documento de Alcance del Sistema.

Como llevar a cabo esta tarea:

- Hacer una lista con los conceptos claves, proporcionando la descripción de cada uno de ellos. Es importante tener en cuenta que esta lista de conceptos claves puede ayudar a definir algunas actividades tales como el modelo conceptual y modelo navegacional que se describirán mas adelante en la etapa de diseño. Ya que sirven como medio para encontrar clases, nodos y contextos navegacionales [94].
- Utilizar plantilla para glosario, ver Anexo 11.

Salidas:

- Documento de Glosario de términos

4.1.3.5 Definir los estándares de nombramiento

Se deben estandarizar el nombramiento y definición de los elementos que componen la aplicación Web tales como: paquetes, clases, atributos, métodos / funciones, variables, tablas e índices que se usarán para el proyecto.

Objetivo

- Definir los estándares a utilizar en el proyecto de los elementos que componen el sistema.

Entradas:

- Especificación de Requerimientos
- Modelo de Casos de Uso
- Documento de Validación de Requerimientos.
- Documento de Alcance del Sistema.

Como llevar a cabo esta tarea:

- Utilizar la plantilla de Nombramiento, ver Anexo 12.

Salida:

- Documento de Nombramiento.

4.1.4 Actividad Definir la Arquitectura Candidata

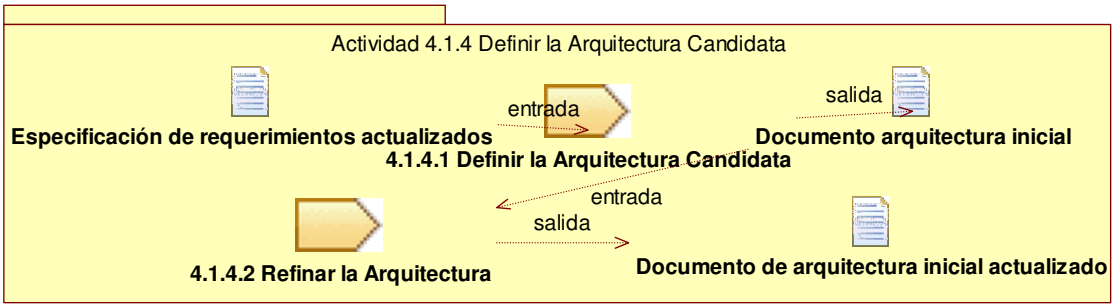


Figura 6 Definir la Arquitectura Candidata

En la figura 6 se aprecian las tareas correspondientes a ésta actividad con sus correspondientes entradas y salidas.

La arquitectura del software define un marco común donde describir los elementos importantes de la aplicación Web de forma que sirva como base para ir construyendo el resto de los elementos del desarrollo. Se puede ver como un esqueleto que ayuda a que todos los elementos del software se mantengan estables durante los numerosos cambios que se producen en el desarrollo [103].

Otro aspecto importante de la arquitectura es que indica la importancia de cada uno de los elementos del desarrollo, esta información se puede utilizar para planificar el orden en que sería mejor abordar el desarrollo, así como para controlarlo y guiarlo.

Dentro de la arquitectura del software se van a definir los siguientes elementos:

- Organización de la aplicación Web
- Conjunto de elementos estructurales y las colaboraciones que se definen entre ellos.
- Composición del sistema en subsistemas
- Estilos arquitectónicos que se utilizaran(elementos, interfaces, colaboraciones y composiciones)

Objetivo

- Crear una propuesta inicial de la arquitectura del software.
- Refinar la arquitectura.

Tareas

1 Analizar la Arquitectura Candidata
2 Refinar la Arquitectura

Entrada

- Documento de Especificación de Requerimientos actualizado con la priorización de Requerimientos.

Salida:

- Documento Arquitectura Inicial

Rol responsable: Arquitecto

Participantes:

- Analista; Diseñador; Arquitecto; Ingeniero de calidad; Equipo de desarrollo.

4.1.4.1 Analizar la Arquitectura Candidata

Durante el desarrollo de la aplicación Web la arquitectura va a ir evolucionando, ya que se irán incorporando elementos relevantes para la arquitectura definidos en cada una de las actividades del proceso de ingeniería esto va a originar un **conjunto** de vistas complementarias de la aplicación. El documento de arquitectura se irá completando en la medida en la que la aplicación crezca y según avance el desarrollo del software.

Objetivo

- Crear una propuesta inicial de la arquitectura del software.

Entradas*:*

- Documento de Especificación de Requerimientos actualizado con la priorización de Requerimientos

Cómo llevar a cabo la tarea:

- Identificar los Casos de Uso significativos para la arquitectura, esto viene desde la priorización de requerimientos.
- De acuerdo con los recursos (personas, herramientas, tiempo, etc.) y la complejidad definir el estilo arquitectónico (ver capítulo 2) que se va a adoptar.
- Identificar las clases para los casos de uso significativos.

Salidas:

- Documento Arquitectura Inicial

4.1.4.2 Refinar la Arquitectura

En esta tarea se completará la arquitectura, identificando elementos y mecanismos de diseño como los patrones, organizando el modelo de implementación y manteniendo la consistencia e integridad de la arquitectura para garantizar la integración de los elementos de diseño preexistentes con los nuevos y permitir la máxima reutilización de componentes todo esto con la experiencia de los expertos [104]

Objetivo

- Refinar la arquitectura inicial.

Entrada

- Documento Arquitectura Inicial

Cómo llevar a cabo la tarea:

- Realizar varias sesiones, identificando primero elementos a diseñar y mecanismos de diseño, e incorporando nuevos elementos en cada iteración.
- Después se introducirán temas relativos a concurrencia y distribución, si se requieren. También se refinará el diseño con decisiones sobre la arquitectura.
- Poner especial atención si la arquitectura requiere tecnología específica en la Etapa de implementación.

Salidas:

- Documento Arquitectura Inicial Actualizado.

4.2 Etapa de Diseño

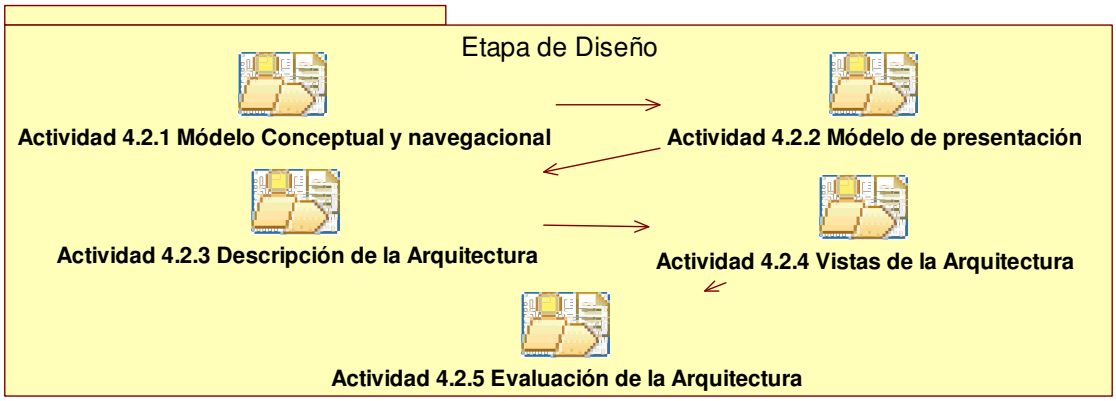


Figura 7 Etapa de Diseño

En la figura 7 se pueden apreciar las actividades que corresponden a la etapa de diseño.

El diseño es una tarea muy importante en el desarrollo de una aplicación Web, debido a que es en esta etapa donde se toman las decisiones estratégicas y tácticas, para poder modelar de una manera correcta los requerimientos funcionales, de interacción, de usabilidad y no funcionales requeridos en un una aplicaciones Web. Además es primordial debido a que los modelos obtenidos en esta etapa serán usados en la etapa de implementación de la aplicación Web.
En esta etapa se diseña el modelo conceptual, navegacional, de presentación, se hace la descripción de la arquitectura, las vistas de la arquitectura y evaluación de la arquitectura.
Es importante anotar que en esta etapa se hace uso intensivo de los patrones mencionados en el capítulo 3.

4.2.1 Actividad Modelo Conceptual y Navegacional

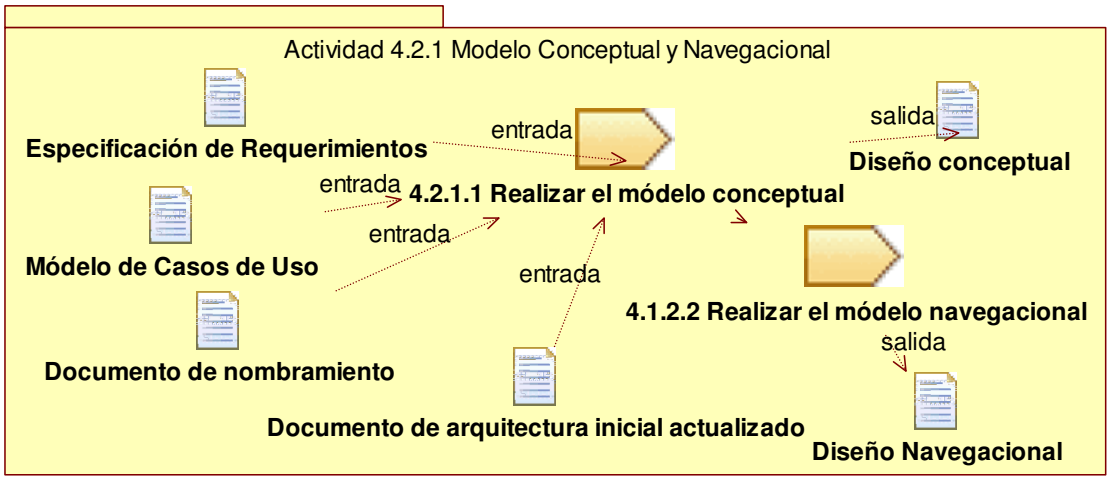


Figura 8 Modelo Conceptual y Navegacional

En la figura 8 se puede se aprecian las tareas correspondientes a ésta actividad con sus correspondientes entradas y salidas.

Después de haber realizado la etapa de requerimientos y análisis, se diseña el modelo conceptual y navegacional, los cuales serán usados en la etapa de implementación de la aplicación.

Objetivo

- Modelar y describir la información que maneja la aplicación Web, además de su estructura.

Tareas

1 Realizar el modelo conceptual
2 Realizar el modelo navegacional

Entradas

- Especificación de Requerimientos
- Modelo de Casos de Uso.
- Documento de Nombramiento
- Arquitectura inicial (candidata refinada)

Salida

- Diseño Conceptual.
- Diseño Navegacional.

Rol responsable: Diseñador

Participantes:

- Arquitecto; Analista; Equipo de desarrollo.

Patrones Relacionados

Para realizar esta tarea es aconsejable se haga uso de los patrones que se muestran en la siguiente tabla, los cuales ayudarán a realizar un buen diseño navegacional y a considerar detalles relevantes para tener una aplicación Web usable.

Tabla 60 Patrones de Hipermedia Relacionados con el Modelo Navegacional

Patrones de Presentación Nivel Intermedio		
Diseño Navegacional		**Sistemas Hipermedia**
Nodo como Unidad única	Contexto Navegacional	Enlace como Relación entre Vistas
Método crear Nodo y Crear enlace	Referencia Activa	Observador Navegacional

4.2.1.1 Realizar el Modelo Conceptual

El modelo conceptual representa la estructura estática del sistema [16], este permite modelar la información que se maneja en la aplicación Web y representar las relaciones que se establecen entre ellas. Este modelo es muy importante ya que a partir de el se puede diseñar el modelo entidad relación y el navegacional.

El modelo conceptual viene representado por dos elementos:

- El diagrama de clases conceptuales, que se representa mediante un diagrama de clases usando la notación UML[10].
- El diccionario de datos: Describe las clases y las relaciones o asociaciones entre las clases.

***Objetivo*:**

- Realizar el diagrama de clases.
- Realizar el modelo entidad relación.

Entradas:

- Especificación de Requerimientos
- Modelo de Casos de Uso.
- Documento de Nombramiento
- Arquitectura inicial Actualizado(candidata refinida)

Como llevar a cabo esta tarea:

Para realizar el diagrama de clases es necesario tener en cuenta:

- Las plantillas de los requerimientos de almacenamiento de información y sus respectivas naturalezas ya que básicamente a partir de cada requisito de almacenamiento se genera una clase.
- A partir de las naturalezas de los requerimientos de almacenamiento, los datos se traducen en atributos de la clase.
- Se recomienda agrupar las clases comunes en paquetes.
- Utilizar la plantilla para describir las clases que se encuentra en el documento de Diseño, ver anexo 13.
- Utilizar la plantilla para describir las asociaciones que se encuentra en el documento de Diseño, ver anexo 13.

Para realizar el diagrama entidad – relación se recomienda lo siguiente:

- Revisar el diagrama de clases, debido a que: Cada clase puede ser mapeada en una entidad, cada atributo de la clase puede pasar a ser un atributo de una entidad.
- Colocar el diagrama entidad – relación en el documento de Diseño, Ver anexo 13.

Salidas*:*

- Diseño Conceptual.

4.2.1.2 Realizar el Modelo de Navegación

En OOHDM [13], (Método de Diseño Hipermedia Orientado a Objetos) la navegación es fundamental en el diseño de aplicaciones. Un modelo navegacional es construido como una vista sobre un diseño conceptual, permitiendo construir modelos diferentes para los diferentes perfiles de usuario. El diseño de navegación es expresado en dos esquemas: el esquema de clases navegacionales y el esquema de contextos navegacionales. En OOHDM existe un conjunto de tipos predefinidos de clases navegacionales: nodos, enlaces y estructuras de acceso. La semántica de los nodos y los enlaces son los que tradicionalmente tienen las aplicaciones hipermedia, y las estructuras de acceso, tales como índices o recorridos guiados, representan los posibles caminos de acceso a los nodos [105].

En este modelo se representan tres aspectos básicamente [90]:

- Cómo se va a poder navegar a través de la información conceptual, por lo que realmente el modelo navegacional va a ser una vista del modelo conceptual.
- Qué elementos (información, funcionalidad, posibilidades de navegación, etc.) van a aparecer en esa navegación y cómo se van a adaptar al usuario que interactúa con el sistema.
- Las relaciones que aparecen entre dichos elementos de la navegación.

Objetivos:

- Definir, a partir de las especificaciones realizadas en el modelo conceptual, un modelo de navegación coherente y sin errores.
- Establecer la estructura de enlaces y las herramientas de navegación.
- Definir un modelo de navegación que se adecue a las necesidades particulares de cada rol.
- Describir qué elementos (información, funcionalidad, posibilidades de navegación, etc.) van a aparecer en la navegación, las relaciones que aparecen entre dichos elementos de la navegación y cómo se va a adaptar al usuario que interactúa con el sistema.

- Definir cómo se le proporcionará a cada usuario del sistema el acceso a la información y la funcionalidad que le es relevante para llevar a cabo su tarea dentro del sistema y qué secuencias de caminos deberán seguir para conseguirlo [106].

Entradas:

- Especificación de Requerimientos
- Modelo de Casos de Uso.
- Documento de Nombramiento
- Arquitectura inicial (candidata refinida)
- Diseño Conceptual.

Como llevar a cabo esta tarea:

Debido a que el modelo navegacional es prioritario en las aplicaciones Web, se explicará en detalle cada una de las sub-tareas que son necesarias para obtener un excelente resultado.

Subtareas:

1 Clasificación e identificación de Actores Navegacionales
2 Construcción de los Mapas de Navegación

Salidas:

- Diseño Navegacional.

4.2.1.2.1 Clasificación e identificación de Actores Navegacionales [106]

En esta tarea se especifican qué tipos de usuarios van a poder interactuar con la aplicación Web, considerando cada perfil de usuario identificado previamente, qué interrelaciones existen entre ellos y cuál va a ser su modo de acceso a la aplicación Web; Se distinguen tres tipos de usuario, en función del tipo de acceso que tengan con la aplicación:

- Anónimos:
  - Al conectarse al sistema, estos usuarios no necesitan identificarse.
  - Habitualmente sus permisos con el sistema son muy reducidos.
  - No se pueden establecer políticas de personalización individuales
- Registrados:
  - Estos usuarios necesitan identificarse al conectarse con la aplicación Web.
  - Habitualmente, gestionan la funcionalidad del sistema y la información.
  - Se pueden establecer políticas de personalización individuales.
- Abstracto:
  - Se utilizan para expresar responsabilidades comunes entre usuarios o sea para los mecanismos de especialización.

Los actores con un sistema de navegación similar forman parte de un mismo grupo. El resto de actividades de análisis se aplican para cada grupo de actores.

Entradas:

- Especificación de Requerimientos
- Modelo de Casos de Uso.
- Documento de Nombramiento
- Arquitectura inicial (candidata refinida)
- Diseño Conceptual.

Como llevar a cabo esta tarea:

- Se debe realizar un estudio para detectar los potenciales tipos de usuarios interactivos con el sistema, las interrelaciones entre ellos y crear un diagrama de usuarios, dicho diagrama permite expresar: los tipos de usuarios que pueden usar el sistema y su accesibilidad a la aplicación, así como las relaciones entre los usuarios. Para entender cómo construir el diagrama de usuarios en el anexo 13 se proporciona un ejemplo.

Salidas*:*

- Diseño Navegacional con Diagrama Actores Navegacionales

4.2.1.2.2 Construcción de los Mapas de Navegación

Las propiedades navegacionales de una aplicación Web se describen asociando un mapa navegacional para cada tipo de usuario. Un mapa navegacional está compuesto por contextos navegacionales y vínculos navegacionales y definirá la estructura global de navegación de la aplicación Web. El contexto donde se inicia la navegación corresponde al Home o a la página de inicio.

Entradas:

- Especificación de Requerimientos
- Modelo de Casos de Uso.
- Documento de Nombramiento
- Arquitectura inicial (candidata refinida)
- Diseño Conceptual.
- Diseño Navegacional con Diagrama Actores Navegacionales

Como llevar a cabo esta tarea [106]

- **Detectar los contextos navegacionales:** Un contexto navegacional es una Unidad de Interacción Abstracta que representa una vista sobre un conjunto de datos y/o servicios (del esquema conceptual) accesible para un usuario en un determinado momento. Es una Unidad porque constituye el elemento lógico básico de creación de la navegación permitida en los mapas navegacionales. De Interacción porque representa una interacción con el usuario (espera una acción/ respuesta por parte del usuario, bien de navegación, bien de activación de un servicio), y Abstracta porque sólo se especifica qué datos y/o servicios se visualizarán en el contexto, pero no cómo se presentarán [106]. Se compone de un conjunto de clases navegacionales (clase directora y clase complementaria) y relaciones entre ellas.
- **Identificar los Vínculos de Navegación:** Los cuales indican la navegación entre contextos (relación de alcanzabilidad entre contextos de navegación). Esta navegación se define a partir de relaciones navegacionales definidas dentro de los contextos.
- **Establecer las Clases Navegacionales:** Se definen como una proyección (vista) que se especifica sobre una clase del diagrama de clases definido en el Modelo Conceptual. Cada clase navegacional se genera a partir de un prototipo de visualización y cada atributo desde los atributos de los prototipos de visualización. En el contexto navegacional debe aparecer al menos una clase navegacional principal, llamada clase directora y opcionalmente otras que complementan la información de esta clase, llamadas clases complementarias. La Clase Directora: Es la clase principal de un contexto, existe una única por contexto (obligatoria) y de ella surge toda la navegación. Las Clases Complementarias: Complementan la información de la clase directora. Pueden aparecer varias por contexto (no son obligatorias).

- **Establecer las relaciones binarias unidireccionales**: Las clases navegacionales están unidas entre sí por relaciones binarias unidireccionales que pueden ser definidas sobre una relación de agregación o de especialización/generalización existente entre las dos clases en el diagrama de clases. Se define dos tipos de relaciones entre clases navegacionales:
- **Relación de contexto**, define un vínculo navegacional entre contextos, indica la dirección de navegación, implica necesariamente la existencia de un contexto navegacional (destino) en el que la clase directora es la clase destino de la relación, se realiza a través de la relación de agregación o especialización/generalización sobre la que está definida la relación. Se utiliza para eliminar la ambigüedad en caso de existencia de más de una relación entre las dos clases. Es opcional si entre las dos clases existe una única relación definida en el diagrama de clases.
- **Relación de Dependencia Contextual** que indica la existencia de una relación entre dos clases de un contexto, pero no define una semántica navegacional entre ellas. Se utiliza para proporcionar información complementaria de la clase directora.

Salidas:

- Diseño Navegacional con Diagrama Actores Navegacionales y Mapas de Navegación.

4.2.2 Actividad Modelo de Presentación

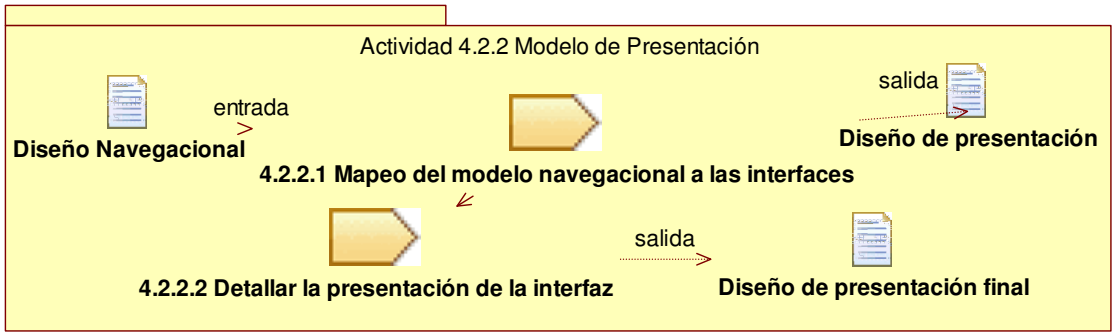


Figura 9 Modelo de Presentación

En la figura 9 se puede se aprecian las tareas correspondientes a ésta actividad con sus correspondientes entradas y salidas

Después de que las estructuras navegacionales son definidas, se deben especificar los aspectos de interfaz. Esto significa definir la forma en la cual los objetos navegacionales pueden aparecer, cómo los objetos de interfaz activarán la navegación y el resto de la funcionalidad de la aplicación, qué transformaciones de la interfaz son pertinentes y cuándo es necesario realizarlas [105].

Objetivos:

- Definir la estructura lógica de presentación de los objetos navegacionales en la interfaz de usuario.
- Especificar las características de presentación del sistema.

Tareas

1 Mapeo del modelo navegacional a las interfaces.
2 Detallar la presentación de la interfaz.

Entradas:

- Diseño navegacional

Salida:

- Diseño de presentación

Rol responsable: Diseñador

Participantes:

- Diseñador grafico; ingeniero de calidad; cliente y/o usuario

Patrones Relacionados: En la siguiente tabla se relacionan los patrones que pueden ser utilizados, de acuerdo a los requerimientos del usuario, para la interfaz de la aplicación Web.

Tabla 61 Patrones de hipermedia y de interacción relacionados con la presentación de la interfaz de la aplicación Web

Patrones para Organizar y Mostrar la Estructura de la Interfaz de la Aplicación Web		
Patrones de Hipermedia de interfaz	**Patrones de Interacción**	
Información solicitada	Narrativa	Visión global en detalle
Desacoplar la interacción de la información	despliegue de información de alta densidad	Grupos pequeños de cosas reaccionadas
agrupación conductual	Espacios navegacionales	Conjunto tabulador
	Conjunto de jerarquías	Detalle Opcional en Demanda
	Framework repetido	Superficie activa central
	Superficie activa en mosaico	

Patrones que Ayudan a Organizar y Manejar los Controles de la Aplicación Web			
Patrones de Interacción			
Panel de control	Apariencia de Fondo	Superficie activa en mosaico	Acciones de Localización de Objetos
Desactivar Cosas No Pertinente	Superficie Activa central	Pila de superficies activas	Acciones para múltiples objetos
Espacio del Objeto Personal	Superficie activa en mosaico	Acciones Convenientes de Ambiente	Barra de herramientas

Patrones para Proporcionar Realimentación y facilidad de Recordar en la Aplicación Web			
Patrones de Interacción			
Instrucciones paso a paso	Bueno por defecto	Indicador de progreso	Opción de un conjunto pequeño
Mostrar el Indicador de Acceso	Estado recordado	Mensaje importante	Colección editable
Descripción Corta	Historia de la interacción	Demostración	Absolver entrada de texto
			Entrada de texto estructurado

Patrones que Ayudan a Definir como Proporcionar Orientación al Usuario		
Patrones de Interacción		
Demostración	Puntos de entrada Claros	Ir un paso atrás
Mapa de espacios navegacionales	Secciones de color codificado	Ir a un lugar seguro

Patrones para desarrollar una aplicación Web adaptable al usuario		
Patrones de Interacción		
Preferencias de usuario	Secuencia de Acción escrita	Marcador de libros
Espacio del Objeto Personal	anotaciones de usuario	Chequeo real

Patrones para Utilizar Comandos en la Aplicación Web
Patrones de Interacción
Comando compuesto

4.2.2.1 Realizar Mapeo del modelo navegacional a las interfaces.

El énfasis de esta tarea está en la organización estructural de los elementos de acuerdo a los elementos del modelo navegacional. En esta tarea se utiliza los nodos o contextos navegacionales como entidades básicas para definir las propiedades de presentación [30].

Objetivo:

- Obtener el modelo de interfaz a partir del modelo navegacional.

Entradas:

- Diseño navegacional

Como llevar a cabo esta tarea:

- Por cada nodo o contexto navegacional existente en el diseño navegacional construir una interfaz.
- Además, para cada relación de contexto definida en el modelo navegacional se implementa un enlace a la interfaz Web que representa el contexto o nodo destino.
- Incluir en cada diseño de interfaz todas las opciones de navegación definidas para el nodo, en el diseño navegacional.
- Utilizar la plantilla para Describir las Interfaces que se encuentra en el documento de Diseño, ver anexo 13.

Salidas:

- Diseño de presentación

4.2.2.2 Detallar la presentación de la interfaz.

La forma en que se estructure la presentación de la aplicación puede determinar la usabilidad del sistema.

Objetivo:

- Dar soporte a la percepción, la interpretación y la comprensión de la información de las aplicaciones Web, aspectos relacionados con la parte física de la interacción (colores, fuente, organización de la información, elementos…), el lenguaje (visual para las interfaces visuales, auditivo para las auditivas…), los modelos de la información, la consistencia, la coherencia, la retroalimentación.

Entradas:

- Diseño navegacional
- Diseño de presentación

Como llevar a cabo esta tarea:

- Organizar la información de manera que sea útil y comprensible para los usuarios del sistema,
- Utilizar patrones que se presentan a continuación en la tabla 3, de acuerdo a las necesidades que se requiera cumplir con la aplicación Web, la descripción de estos patrones se encuentra en el capítulo 3.

Salidas:

- Diseño de presentación final

4.2.3 Actividad Descripción de la Arquitectura

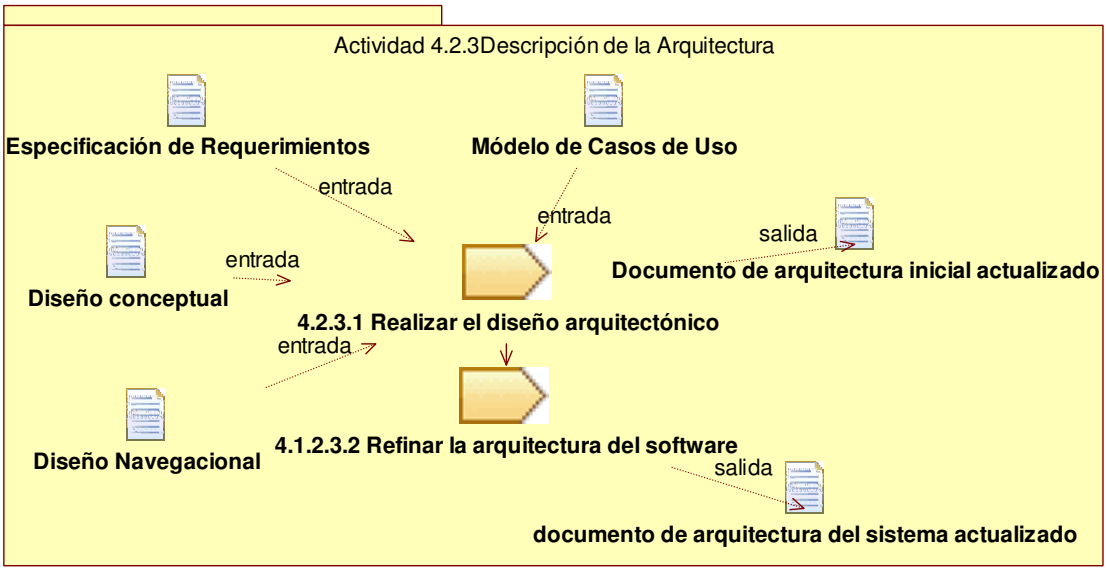


Figura 10 Descripción de la Arquitectura

En la figura 10 se puede se aprecian las tareas correspondientes a ésta actividad con sus correspondientes entradas y salidas

Objetivo:

Realizar la descripción de la Arquitectura, con base en los documentos generados en la etapa Requerimientos y Análisis y al Modelo de Diseño, que muestra la traza entre requerimientos y diseño, el modelo de implementación (módulos y submódulos) y la traza entre diseño e implementación, así como también el modelo de distribución (servidores, terminales, comunicaciones, etc.).

Todo esto se especifica en el documento Descripción de la Arquitectura [107].

Tareas:

1 Realizar el Diseño Arquitectónico.
2 Refinar la Arquitectura del Software.

Entradas

- Especificación de Requerimientos
- Modelo De Casos de uso
- Documento Arquitectura Candidata
- Diseño Conceptual
- Diseño Navegacional

Salida:

- Documento de Arquitectura del Sistema

Rol responsable: Arquitecto

Participantes:

- Analista; Ingeniero de Calidad.

Patrones Relacionados

Según el análisis realizado de los patrones en el capítulo 3, determinar cuáles patrones utilizar de acuerdo a los requerimientos se requieran cumplir con la aplicación Web.

A continuación en la tabla 62 se presenta una clasificación de los patrones arquitecturales, dependiendo de su utilidad; la descripción de estos patrones se encuentra en el capítulo 3.

Tabla 62 Patrones Arquitecturales

Patrones para la Lógica de Dominio		
Capas	Pizarra	Micro Kerne
Reflexión	Mediador	
Patrones de Acceso a Datos		
Transacción Script	Patrón Modelo del dominio	Patrón Modulo de la tabla
Patrones Arquitecturales para Organizar la Presentación de la Aplicación Web		
MVC	**PAC**	
Patrones para el controlador	**Patrones para la vista**	
Controlador de página	Plantilla de la vista	Vista en dos pasos
Controlador principal	Transformador de la vista	Controlador de la aplicación

4.2.3.1 Realizar el diseño arquitectónico

Objetivo:

- Definir la arquitectura del software a desarrollar.

Entradas:

- Especificación de Requerimientos
- Modelo De Casos de uso
- Documento Arquitectura inicial actualizado
- Diseño Conceptual
- Diseño Navegacional

Cómo llevar a cabo la tarea:

- Con base a la experiencia adquirida por el grupo en el desarrollo se definen una serie de elementos básicos que describen una primera visión de la arquitectura que tendrá el sistema, tanto a nivel físico como lógico.
- Conceptualmente, este modelo describe a grandes rasgos la solución propuesta, mostrando en grandes bloques los subsistemas de que se compondrá, así como la funcionalidad más importante a tener en cuenta.
- Utilizar la plantilla Arquitectura del sistema que se encuentra en el Anexo 14.

Salida:

- Documento de Arquitectura del Sistema Actualizado con el Diseño Arquitectonico

4.2.3.2 Refinar la Arquitectura del Software

Objetivos:

- Completar la arquitectura del software, añadiendo nuevas clases al modelo de diseño en cada iteración realizada.
- Refinar los casos de uso, incluyendo más detalles de la interfaz de usuario y de la base de datos.
- Asegurar que las clases de diseño definidas ofrecen la funcionalidad que las realizaciones de los casos de uso precisan, y que se ofrece la información suficiente para las clases de implementación.

Entradas:

- Especificación de Requerimientos
- Modelo de Casos de uso
- Documento Arquitectura Candidata

- Diseño Conceptual
- Diseño Navegacional

Como llevar a cabo la tarea:

- Identificar nuevas clases de diseño para las clases de análisis, cada vez con más detalle.
- Identificar métodos de las clases de diseño en relación a la funcionalidad encontrada en el modelo de análisis.
- Identificar atributos de las clases de diseño.
- Se comienza a organizar el modelo de implementación, para facilitar el paso del diseño a la codificación.

En esta actividad se debe mantener la consistencia y la integridad de la arquitectura del software, asegurando que:

- los nuevos elementos de diseño identificados en cada iteración se integran correctamente con el modelo de diseño existente hasta el momento, y
- se intenta reutilizar al máximo componentes de diseño existentes para minimizar el esfuerzo.

Salida:

- Documento de Arquitectura del Sistema Actualizado con el Diseño Arquitectónico Refinado.

4.2.4 Actividad Vistas de la Arquitectura:

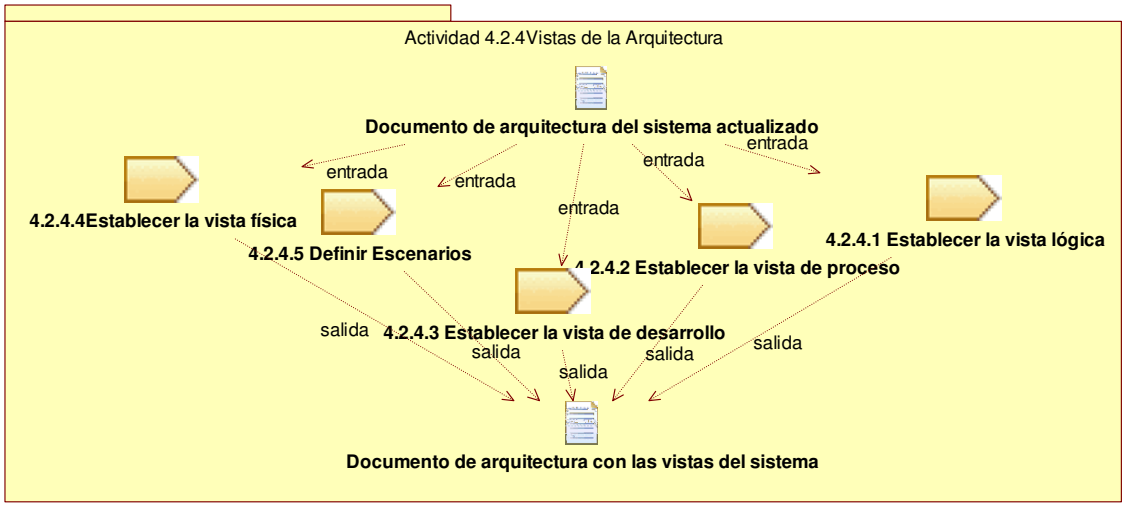


Figura 11 Vistas de la Arquitectura

En la figura 11 se puede se aprecian las tareas correspondientes a ésta actividad con sus correspondientes entradas y salidas

Descripción: La arquitectura del software se trata de abstracciones, de descomposición y composición, de estilos y estética. También tiene relación con el diseño y la implementación de la estructura de alto nivel del software [108].
Los diseñadores construyen la arquitectura usando varios elementos arquitectónicos elegidos apropiadamente.
Estos elementos satisfacen la mayor parte de los requisitos de funcionalidad y rendimiento del sistema, así como también otros requisitos no funcionales tales como confiabilidad, escalabilidad, portabilidad, usabilidad y disponibilidad del sistema.

El modelo 4+1 describe la arquitectura del software usando cinco vistas concurrentes. Tal como se muestra en la Figura 12, cada vista se refiere a un conjunto de intereses de diferentes participantes del sistema[108].

- Vista lógica: Comprende las abstracciones fundamentales del sistema a partir del dominio de problemas.
- Vista de proceso: Comprende el conjunto de procesos de ejecución independiente a partir de las abstracciones anteriores.
- Vista de desarrollo: Organización estática de módulos en el entorno de desarrollo.
- Vista física: un mapeado22 del software sobre el hardware.
- El quinto elemento considera todos los anteriores en el contexto de escenarios los cuales se describen por medio de los casos de uso[24].

Los diseñadores de software pueden organizar la descripción de sus decisiones de arquitectura en estas cuatro vistas, y luego ilustrarlas con un conjunto reducido de casos de uso o escenarios, los cuales constituyen la quinta vista.

La arquitectura evoluciona parcialmente a partir de estos escenarios.

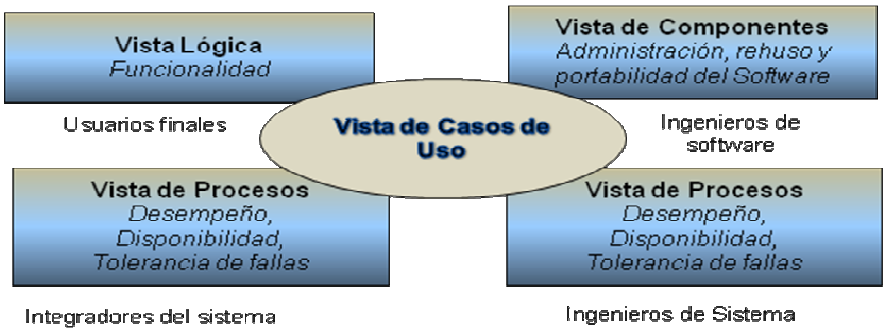


Figura 12 Modelo 4+1 Vistas

Tareas:

- Establecer la Vista Lógica
- Establecer la Vista de Proceso.
- Establecer la Vista de Desarrollo.
- Establecer la Vista Física
- Definir Escenarios.

Entrada:

- Documento de Arquitectura del Sistema Actualizado con el Diseño Arquitectónico Refinado.

Salida:

- Documento de Arquitectura del Sistema Con la Descripción y Definición de Cada una de las Vistas

Rol responsable: Arquitecto

Participantes:

- Analista; Ingeniero de Calidad; Diseñador; Equipo de desarrollo

[22] Mapeado viene de ingles mapping que significa traducción, en el resto del documento se utilizará el termino “Mapeado” para referirnos a “Mapping”.

4.2.4.1 Establecer la Vista Lógica:

La arquitectura lógica apoya principalmente los requisitos funcionales lo que el sistema debe brindar en términos de servicios a sus usuarios. El sistema se descompone en una serie de abstracciones, tomadas (principalmente) del dominio del problema en la forma de *objetos* o *clases de objetos*. Aquí se aplican los principios de abstracción, encapsulamiento y herencia [108]. Esta descomposición no solo se hace para potenciar el análisis funcional, sino también sirve para identificar mecanismos y elementos de diseño comunes a diversas partes del sistema.

***Objetivo**:*

- Definir la vista lógica arquitectura de la aplicación Web.

Como llevar a cabo la tarea:

- Se basa en el modelo conceptual y se utiliza el enfoque de Booch/Rational para representar la arquitectura lógica, mediante diagramas de clases y templates de clases. Un diagrama de clases muestra un conjunto de clases y sus relaciones lógicas: asociaciones, uso, composición y herencia. Grupos de clases relacionadas se agrupan en categorías de clases. Los templates de clases se centran en cada clase individual; enfatizan las operaciones principales de la clase, e identifican las principales características del objeto. Si es necesario definir el comportamiento interno de un objeto, esto ser realiza con un diagrama de transición de estados o diagrama de estados. Los mecanismos y servicios comunes se definen como utilities de la clase.
- Utilizar la notación para la vista lógica que corresponde a los diagramas de: Clases, Estados y Colaboración, en UML [10]:
- Utilizar una herramienta para modelar los componentes.
- Adicionar esta vista en la plantilla Arquitectura del sistema que se encuentra en el Anexo 14.

4.2.4.2 Establecer la Vista de Proceso:

La arquitectura de procesos toma en cuenta algunos requisitos no funcionales tales como la rendimiento y la disponibilidad. Se enfoca en asuntos de concurrencia y distribución, integridad del sistema, de tolerancia a fallas. Esta vista se describe en varios niveles de abstracción, donde cada nivel se refiere a distintos intereses. El nivel más alto la arquitectura de procesos puede verse como un conjunto de redes lógicas de programas comunicantes (llamados “procesos”) ejecutándose en forma independiente, y distribuidos a lo largo de un conjunto de recursos de hardware conectados mediante un bus, una LAN o WAN [108].

Un proceso es una agrupación de tareas que forman una unidad ejecutable. Los procesos representan el nivel al que la arquitectura de procesos puede ser controlada tácticamente (i.e., comenzar, recuperar, reconfigurar, y detener). Además, los procesos pueden replicarse para aumentar la distribución de la carga de procesamiento, o para mejorar la disponibilidad [108].

Objetivo:

- Definir la vista de procesos de la arquitectura.

Como llevar a cabo la tarea:

- Particionar el software en un conjunto de tareas independientes: hilo de control separado que puede planificarse para su ejecución independiente en un nodo de procesamiento. Se puede distinguir entre:
    - tareas mayores las cuales involucran elementos arquitectónicos que se comunican a través de un conjunto bien definido de mecanismos de comunicación inter-tarea: servicios de comunicación sincrónicos y

asincrónicos basados en mensajes, llamados a procedimientos remotos, difusión de eventos, etc.
    - tareas menores son tareas adicionales introducidas localmente por motivos de implementación tales como actividades cíclicas, almacenamiento en un buffer, time-out, etc.).
- Utilizar la notación para la vista de procesos que corresponde a los diagramas de: Actividad, Estados y Secuencia propuestos en UML:
- Adicionar esta vista en la plantilla Arquitectura del sistema que se encuentra en el Anexo 14.

4.2.4.3 Establecer la Vista de Desarrollo.

La vista de desarrollo se centra en la organización real de los módulos de software en el ambiente de desarrollo del software. El software se empaqueta en partes pequeñas – bibliotecas de programas o subsistemas que pueden ser desarrollados por uno o un grupo pequeño de desarrolladores. Los subsistemas se organizan en una jerarquía de *capas*, cada una de las cuales brinda una interfaz estrecha y bien definida hacia las capas superiores [108].
La vista de desarrollo tiene en cuenta los requisitos internos relativos a la facilidad de desarrollo, administración del software, reutilización y elementos comunes, y restricciones impuestas por las herramientas o el lenguaje de programación que se use. Esta vista apoya la asignación de requisitos y trabajo al equipo de desarrollo, y apoya la evaluación de costos, la planificación, el monitoreo de progreso del proyecto, y también como base para analizar la reutilización, portabilidad y seguridad. Es la base para establecer una línea de productos.

***Objetivo**:*

- Definir la vista de desarrollo de la arquitectura.

***Como llevar a cabo la tarea*:**

- La vista de desarrollo contiene la implementación de las clases de diseño significativas arquitectónicamente. Se incluyen las clases que implementan mecanismos genéricos de los que dependen otros componentes de la implementación. Está formada por un conjunto de componentes de implementación y sus relaciones.
- Se realiza una organización del modelo de la implementación en subsistemas de implementación. Esto facilita la implementación y la prueba de estos elementos de forma independiente.
- Los subsistemas de implementación están muy ligados a los de diseño pero se manifiestan de forma distinta dependiendo del lenguaje de programación utilizado para la implementación (en java se usan paquetes, en visual Basic se usan proyectos, en C++ se usan directorios, etc.)
- Utilizar Componente: lo que hace referencia al empaquetamiento físico de los elementos del modelo de la implementación (ejecutables, fichero de datos, librerías, tablas, etc.). Algunos elementos que se pueden representar como componentes son:
- Se puede describir la relación entre componentes usando la notación para diagrama de componentes propuesta por UML [10].
- Adicionar esta vista en la plantilla Arquitectura del sistema que se encuentra en el Anexo 14.

4.2.4.4 Establecer la Vista Física.

La arquitectura física toma en cuenta los requisitos no funcionales del sistema tales como la disponibilidad, confiabilidad (tolerancia a fallas), rendimiento (throughput), y

escalabilidad. El software ejecuta sobre una red de computadores o nodos de procesamiento (o tan solo nodos).

Objetivo:

- Definir la vista física de la arquitectura.

Como llevar a cabo la tarea

- Se describe como los componentes definidos en la vista de implementación se mapean a la plataforma hardware existente. Se realiza una visión de la forma física del sistema usando la interconexión de nodos. Un nodo es un elemento de la arquitectura en el que se ejecuta uno o más componentes de implementación (Ordenador, Red,…).
- En esta vista es donde los desarrolladores realizan un proceso de ingeniería de sistemas. Hay que volver a introducir todos los elementos desarrollados dentro del nuevo sistema.
- Se incluyen aspectos como la instalación de los diferentes elementos tanto software como hardware, la carga de las bases de datos, etc.
- Se puede utilizar la notación propuesta por UML [10]. para diagrama de despliegue.
- Adicionar esta vista en la plantilla Arquitectura del sistema que se encuentra en el Anexo 14.

4.2.4.5 Definir Escenarios.

Los elementos de las cuatro vistas trabajan conjuntamente en forma natural mediante el uso de un conjunto pequeño de escenarios relevantes –instancias de casos de uso más generales– para los cuales se describen scripts correspondientes (secuencias de interacciones entre objetos y entre procesos) tal como lo describen Rubin y Goldberg [110]. Los escenarios son una abstracción de los requisitos más importantes.

Como llevar a cabo la tarea

- Definir los escenarios a partir de los casos de uso, definiendo para cada escenario un estímulo, un ambiente y una respuesta, un ejemplo de un escenario es el siguiente: Un usuario remoto de Web requiere un reporte de base de datos en hora pico y lo recibe dentro de los 5 segundos.
- Expresar el diseño mediante el uso de diagramas de escenarios y diagramas de interacción de objetos propuestos por UML [10].

4.2.5 Actividad Evaluación de la Arquitectura

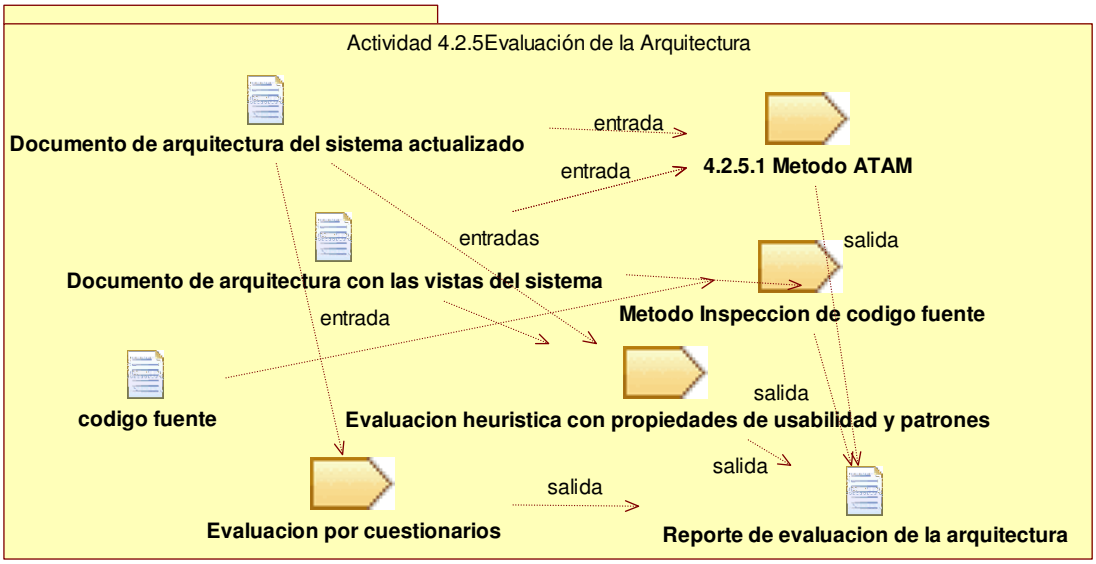


Figura 13 Evaluación de la Arquitectura

Existen una serie de métodos para evaluar la arquitectura entre los cuales se pueden destacar:

- Método de Análisis de Acuerdos de Arquitectura (Architecture Trade-off Analysis Method, ATAM.
- Método Inspección de código fuente.

Objetivo:

- Evaluar la arquitectura de la aplicación Web.

Métodos:

1 Método ATAM.
2 Método para Evaluar y Comparar Arquitecturas
3 Método Inspección de código fuente
4 Método para Evaluar la Usabilidad de una Aplicación Web a partir de la Arquitectura.

Entradas

- Descripción de la Arquitectura
- Documento de vistas de la arquitectura
- Código fuente

Salida:

- Reporte de la evaluación de la arquitectura

Rol responsable:

- Ingeniero de pruebas; Arquitecto

Participantes:

- Analista; Diseño; Equipo de desarrollo.

4.2.5.1 Método ATAM

El nombre del método ATAM surge del hecho que revela la forma en que una arquitectura específica satisface ciertos atributos de calidad, y provee una visión de cómo los atributos de calidad interactúan con otros; esto es, los tipos de acuerdos que se establecen entre ellos.

El método recomendado para realizar la evaluación de la arquitectura es: El Método de Análisis de Acuerdos de Arquitectura (Architecture Trade-off Analysis Method, ATAM).

Objetivo

- Identificar los estilos arquitectónicos.

Como llevar a cabo la tarea

El método de evaluación ATAM comprende nueve pasos, agrupados en cuatro fases. La Tabla 63 presenta las fases y sus pasos enumerados, junto con su descripción.

Tabla 63 Pasos del Método de Evaluación ATAM

Fase 1. Presentación	
1. Presentación del ATAM	El líder de evaluación describe el método a los participantes, trata de establecer las expectativas y responde las preguntas propuestas.
2. Presentación de las metas del negocio	Se realiza la descripción de las metas del negocio que motivan el esfuerzo, y aclara que se persiguen objetivos de tipo arquitectónico.
3. Presentación de la Arquitectura.	El arquitecto describe la arquitectura, enfocándose en cómo ésta cumple con los objetivos del negocio.
Fase 2: Investigación y análisis	
4. Identificación de los enfoques arquitectónicos	Estos elementos son detectados, pero no analizados.
5. Generación del *Utility Tree*	Se elicitan los atributos de calidad que engloban la "utilidad" del sistema (desempeño, disponibilidad, seguridad, modificable, usabilidad, etc.), especificados en forma de escenarios. Se anotan los estímulos y respuestas, así como se establece la prioridad entre ellos
6. Análisis de los enfoques arquitectónicos	Con base en los resultados del establecimiento de prioridades del paso anterior, se analizan los elementos del paso 4. En este paso se identifican riesgos arquitectónicos, puntos de sensibilidad y puntos de balance.
Fase 3: Pruebas	
7. Lluvia de ideas y establecimiento de prioridad de escenarios.	Con la colaboración de todos los involucrados, se complementa el conjunto de escenarios.
8. Análisis de los enfoques arquitectónicos	Este paso repite las actividades del paso 6, haciendo uso de los resultados del paso 7. Los escenarios son considerados como casos de prueba para confirmar el análisis realizado hasta el momento.
Fase 4: Reporte	
9. Presentación de los resultados	Basado en la información recolectada a lo largo de la evaluación del ATAM, se presentan los hallazgos a los participantes.

4.2.5.2 Método para Evaluar y Comparar Arquitecturas

Losavio [41] proponen un método para evaluar y comparar arquitecturas de software candidatas, que hace uso del modelo de especificación de atributos de calidad adaptado del modelo ISO/IEC 9126. En este trabajo plantean que la especificación de los atributos de calidad al utilizar un modelo basado en estándares internacionales ofrece una vista amplia y global de los atributos de calidad, tanto a usuarios como arquitectos del sistema, para efectos de la evaluación.

El método describe las siguientes actividades:

Actividades

1 Analizar los requerimientos funcionales y no funcionales principales del sistema, para establecer las metas de calidad

2 Utilizar el modelo de calidad ISO/IEC 9126 adaptado para arquitecturas de software. Algunas métricas pueden definirse con mayor nivel de detalle

3 Presentar las arquitecturas candidatas iníciales
4 Construir la tabla comparativa para las arquitecturas candidatas
5 Establecer prioridades para las sub-características de calidad tomando en cuenta los requerimientos de calidad del sistema
6 Analizar los resultados obtenidos y resumidos en la tabla, de acuerdo con las prioridades establecidas

4.2.5.3 Método Inspección de código fuente.

Este método se utiliza cuando la aplicación Web ya está desarrollada, y se requiere evaluar la arquitectura implementada, por lo que se procede a revisar el código.
La evaluación se puede hacer teniendo como referencia los patrones presentados en el capítulo 3.

***Objetivo**:*

- Evaluar la arquitectura de una aplicación Web.

Como llevar a cabo la tarea

Para evaluar la arquitectura utilizando este método se propone:

- Determinar en qué lenguaje de programación ha sido construida la aplicación.
- Si el lenguaje no se conoce, se debe documentarse acerca de las especificaciones del lenguaje, sus características, etc.
- Tomar como referencia los patrones arquitecturales presentados en el capítulo 3. y determinar los que apliquen de acuerdo al dominio de la aplicación Web.
- Realizar una matriz con una serie de preguntas que sirvan para identificar si se están cumpliendo dichos patrones.

4.2.5.4 Método para Evaluar la Usabilidad de una Aplicación Web a partir de la Arquitectura.

Este método es la propuesta que se hace en este trabajo de grado para evaluar la usabilidad de una aplicación Web a partir de su arquitectura.

Objetivo:

- Evaluar la usabilidad de una aplicación Web a partir de su arquitectura.

Como llevar a cabo la tarea

Para evaluar la arquitectura utilizando este método se propone:

- Analizar cuales de las propiedades debe cumplir la aplicación Web a evaluar.
- Con base en las propiedades que debe cumplir se analizan que patrones contribuyen a que se cumpla con dichas propiedades.
- Construir tablas similares a la que se presentan en los anexos 15, 16 y 17 o en su defecto utilizar dichas tablas.

4.3 Conclusiones del capítulo.

El presente capitulo aporta una metodología para un procesos de desarrollo en sus etapas iníciales, el cual se ha definido de tal forma que sea claro para los integrantes de pequeñas empresas que tienen que mezclar roles para cumplir con el desarrollo de un proyecto. Para seguir el proceso de una manera lógica, ordenada y entendible se dividió el proceso inicial en dos etapas una que corresponde a requerimientos y análisis y otra que es la de diseño; para cada etapa se definieron actividades, las cuales se describen de forma concisa y contienen objetivos, entradas, tareas, salidas, responsables, y participantes involucrados, así como también los patrones que se aconseja utilizar dependiendo de la actividad, ya que en algunas actividades no es necesario utilizar los patrones. Por otro lado se llega a la granularidad de describir cada tarea con sus entradas, salidas y con la descripción de cómo se debe o se puede llevar a cabo. La definición puntual de los objetivos y productos a entregar.

Este proceso permite enfocar la aplicación a las necesidades de los usuarios finales del sistema gracias al manejo de requisitos de usabilidad.
Cabe aclarar que es un proceso complementario a otras propuestas existentes para el desarrollo de aplicaciones Web,

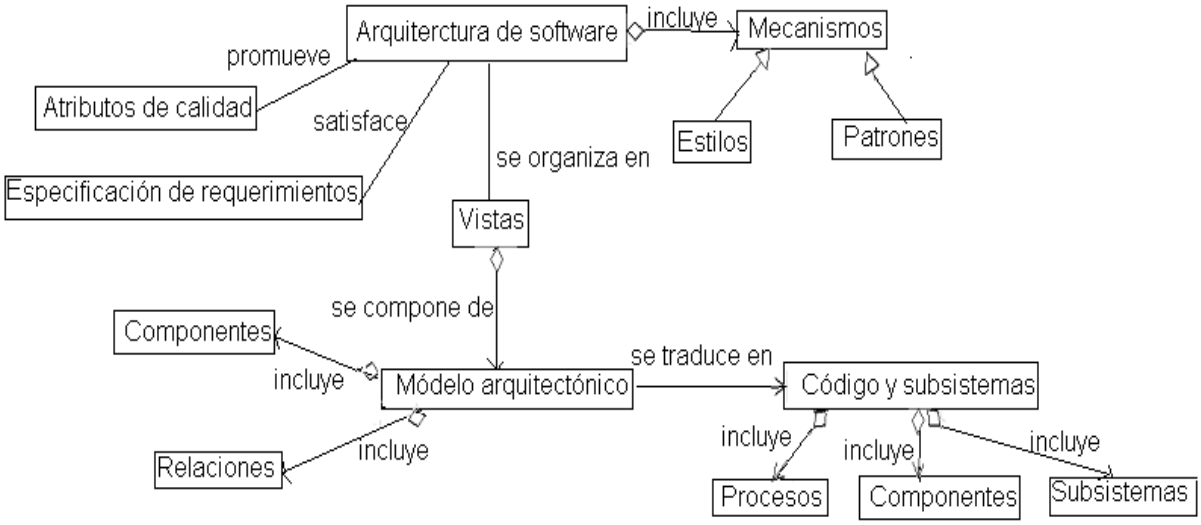


Figura 14 Diagrama de Arquitectura

Capítulo 5 CASO DE ESTUDIO DEL MARCO DE REFERENCIA CENTRADO EN LA ARQUITECTURA PARA LA MEJORA DE CARACTERÍSTICAS DE USABILIDAD EN EL DESARROLLO DE APLICACIONES WEB CONSTRUIDAS POR MIPYMES

Debido a la importancia que presenta la relación entre la academia y la industria para la retroalimentación de conceptos entre lo teórico y lo práctico se presenta el siguiente caso de estudio el cual corresponde a la aplicación del Marco de Referencia en dos aplicaciones Web: "Servicio Web agropecuario" y "Compromiso" las cuales han sido desarrolladas por las MIPYMEs: IKERNELL Aplicaciones Software e INPUT Technologies Ltda., respectivamente, las cuales hacen parte de la incubadora de empresas Parquesoft Popayán.

Sobre los mencionados productos se realizó una evaluación para determinar la utilización de los patrones de presentación y de lógica.

Este capítulo se estructura de la siguiente manera en la primera sección se describe las características de una MIPYME y se detallan las empresas IKERNELL e INPUT, en la sección 5.2 se describe el caso de estudio y los resultados obtenidos de la aplicación del Marco de Referencia al módulo avícola de la empresa IKERNELL, y en la sección 5.3 se describe el caso de estudio y los resultados obtenidos de la aplicación del Marco de Referencia al módulo no conformidades de la empresa INPUT.

5.1 MIPYMEs

Las MIPYMEs de Colombia se clasifican de acuerdo con el número de trabajadores y el valor de sus activos totales según articulo 2º ley 590 de 2000. Definiéndose para las microempresas hasta 10 trabajadores y un valor de activo total de hasta 500 salarios mínimos mensuales legales vigentes. En el mercado de desarrollo de software estas MIPYMEs se diferencian de las medianas y grandes empresas no solo en el número de trabajadores sino también en los procesos de desarrollo ya que en la mayoría de los casos no los tienen bien definidos, lo que conlleva a que los roles de las personas dentro de la organización no se establezcan o se cumplan a cabalidad, llegando a tener una misma persona más de un rol, por ejemplo: un analista puede desempeñarse también como desarrollador, arquitecto, etc.
La mayoría de países latinoamericanos como por ejemplo: México, Honduras, Ecuador, Republica Dominicana, entre otros clasifican a estas empresas de acuerdo al número de empleados y el capital que generan; además proyectos como Competisoft **¡Error! No se encuentra el origen de la referencia.** demuestran que las falencias mencionadas que caracterizan a este tipo de empresas son comunes a todos los países latinoamericanos, debido a los pocos recursos y el mínimo apoyo del gobierno.
Lo expuesto anteriormente, puede traer consecuencias en la usabilidad de las aplicaciones, debido a que en la mayoría de las grandes empresas, existen personas encargadas específicamente de probar la aplicación antes de que esta sea implantada, teniendo roles como el de ingeniero de pruebas o ingeniero de calidad. Esto generalmente no ocurre en las pequeñas empresas, las cuales en este caso se ven obligadas a contratar a terceros que se encarguen de estas pruebas, incurriendo en costos adicionales que a veces las pequeñas empresas no pueden solventar, dándose el caso que algunas MIPYMEs no prueben sus aplicaciones antes de ponerlas en producción en

el entorno final, lo que trae como consecuencia errores en la aplicación que posteriormente conducirán a que la empresa deba cumplir con una póliza de garantía o a que se extiendan los tiempos y los problemas en soporte, lo que implica a largo tiempo la pérdida de clientes y de credibilidad en el mercado.

Para contribuir a la solución de este problema el marco de referencia expuesto en este trabajo de grado, brinda un proceso de desarrollo, en este caso para aplicaciones Web, sencillo y ágil; el cual tiene en cuenta en las primeras etapas del proceso de desarrollo una de las principales características de calidad: La usabilidad, dicho proceso de desarrollo tiene su mayor aporte y su soporte en los patrones de presentación y lógica a los cuales en el capítulo 3, se les realizó un análisis exhaustivo para conocer su impacto sobre la usabilidad. Todo esto con la finalidad de que estas empresas puedan tener aplicaciones más confiables y que cumplan con las expectativas del usuario final.

Cabe anotar que el capítulo 3 expuesto en este trabajo de grado puede ser utilizado tanto por pequeñas, medianas o grandes empresas, debido a que la utilización de los patrones y su impacto de usabilidad es tratado de la misma manera para toda empresa. El capítulo 4 fue definido pensando más en las pequeñas empresas, para lo cual se analizó y se seleccionó de las diferentes metodologías para aplicaciones Web, las tareas más relevantes en cuanto a la usabilidad y calidad que debe cumplir todo sistema, definiéndose además nuevas tareas y estableciéndose una serie de plantillas que faciliten el desarrollo de las actividades expuestas en el proceso. Aunque no se descarta la utilización de este proceso de desarrollo por las grandes empresas.

Para validar el marco de referencia y contribuir a la mejora de usabilidad en las aplicaciones Web, se tomó dos aplicaciones Web “Servicio Web Agropecuario” y “Compromiso” , los cuales han sido desarrollados por las empresas MIPYMEs de Parquesoft Popayán IKERNELL Aplicaciones Software e INPUT Technologies Ltda. respectivamente. Estas empresas se describen a continuación:

IKERNELL Aplicaciones Software: Empresa del sector solidario que tiene como principal objeto la construcción de aplicaciones informáticas a la medida, que respondan a las necesidades de competitividad y mejora de los procesos de gestión de la información de aquellos sectores y comunidades que hasta el momento no podían acceder a componentes tecnológicos que por tradición habían sido costosos y complejos de manejar. IKERNELL cuenta con un portafolio de servicios, entre los que se encuentran los de asesoría, consultoría, configuración, diseño, implantación y desarrollo de aplicaciones Web.

IKERNELL actualmente está constituida por seis personas, 5 trabajando tiempo parcial en la empresa y solo una de tiempo completo, la cual está encargada de gran parte de las funciones y responsabilidades. Asumiendo varios roles a la vez como el de desarrollador, analista, administrador, entre otros. Estas características, y además por el activo total que maneja que es menos de 500 salarios mínimos legales, hacen que IKERNELL sea considerada como una MIPYME.

INPUT Technologies Ltda.: Grupo empresarial dedicado a brindar soluciones tecnológicas integrales que contribuyan al mejoramiento continuo de los procesos y faciliten la obtención de la calidad de las empresas en general. INPUT es una organización que está dispuesta a los constantes retos de renovación de los procesos productivos de las organizaciones modernas.

5.2 Caso de Estudio "Servicio Web Agropecuario"

La empresa IKERNELL ha desarrollado varios productos, entre ellos el Sistema Web Agropecuario (SWA). El cual tiene como objetivo contribuir a la competitividad del sector agropecuario de Colombia y en general de Latinoamérica.

El SWA está conformado por dos grandes componentes: el Portal Web y un sistema de información para la administración de granjas agropecuarias (Sistema Agropecuario Integral S.A.I).
El SAI es una herramienta donde reposa la información de los procesos productivos de las granjas dedicadas a la explotación agropecuaria, en el cual se integra los sectores de: Agricultura, Avicultura, Ganadería y Piscicultura.
Actualmente el SAI cuenta con los siguientes módulos:

- Módulo de Agricultura (ModAgro): Manejo y administración de granjas dedicadas a la explotación agrícola (gestión de cultivos).
- Módulo de Avicultura (ModAvi): Manejo y administración de granjas dedicadas a la explotación de gallinas ponedoras.
- Módulo de Ganadería (ModGan): Administración de granjas bovinas para la producción de carne y leche.
- Módulo de Piscicultura (Módulo Piscícola): Administración de granjas dedicadas a la explotación piscícola.

5.2.1 Módulo de Avicultura ModAvi

El módulo denominado ModAvi, es una alternativa de solución para la industria Avícola, que busca brindar a los administradores de las granjas avícolas una plataforma de gestión donde puedan llevar confiablemente la información de las labores que se presentan diariamente al interior de las mismas.

Mediante un menú iterativo, el usuario puede manipular flexiblemente la información y con él elaborar operaciones básicas como registros, consultas, actualizaciones, reportes planos y gráficos para la estructuración y manipulación de los datos.

Actualmente ModAvi se encuentra constituida por una serie de submódulos encargados de la operatividad técnica (diligenciamientos de formularios) y administrativa (presupuestos) de las granjas del sector.

Cabe anotar que la versión actual del módulo que es la primera, tiene un tiempo de desarrollo de año y medio, del cual se han encargado dos desarrolladores, contando con la asesoría de algunas personas de la parte agropecuaria, las cuales brindaron información acerca de la gestión de las granjas avícolas. En este momento el módulo y en si el servicio Web agropecuario no cuenta con usuarios finales, pero existe en Panamá una persona que está comercializando el servicio, quien lo está explorando, para así poder ofrecerlo.

La figura 15 muestra a manera general las operaciones más sobresalientes en este módulo, que a continuación se describen:

- GESTIÓN DE GALPONES Y LOTES: Registra la llegada de lotes de aves a la granja. El alojamiento de las aves en las instalaciones (galpones), los traslados

de aves de un galpón a otro, los retiros de las aves de los galpones en el momento de finalizar su etapa productiva. Ver figuras 16 y 17.

- GESTIÓN DE CONTROL: En este se registra la producción diaria de huevos clasificada según su tipo, la mortalidad en los galpones, con su respectiva causa de muerte (enfermedades), los descartes encontrados en los diferentes galpones debido a la baja producción de las aves, la ganancia semanal de peso de las aves, el consumo alimenticio de las aves, el programa de vacunación para cada galpón, la gallinaza producida por galpón. Ver figuras 18 y 19.
- *INVENTARIO DE PRODUCCIÓN:* Es este submódulo se lleva el control de la producción diaria de: Huevos, gallinaza, descartes, ver figura 19.
- PROVEEDORES: Registra los insumos necesarios para la producción avícola, el catálogo de proveedores, especificando los productos que ellos ofrecen con sus respectivos precios de venta, registros de las compras de insumos adquiridos para la explotación avícola. Ver figura 20.

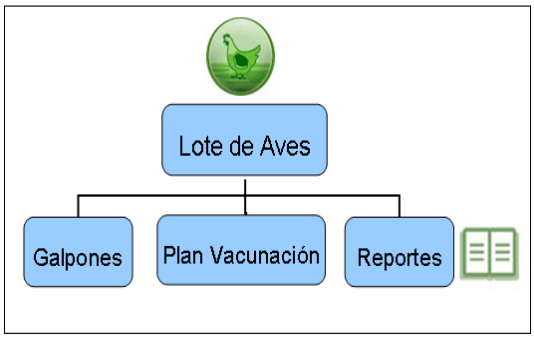


Figura 15 Secciones Generales de ModAvi

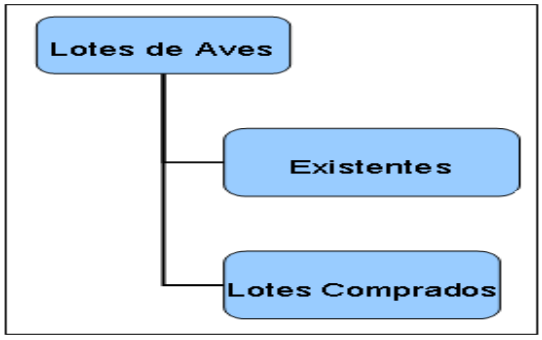


Figura 16 Sección Lotes de Aves

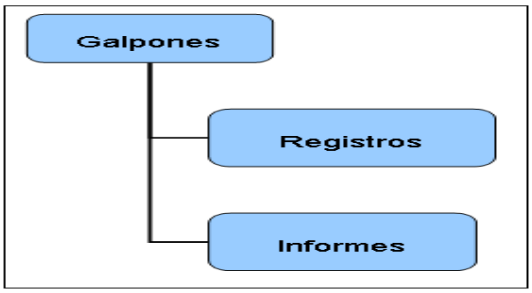


Figura 17 Sección Galpones

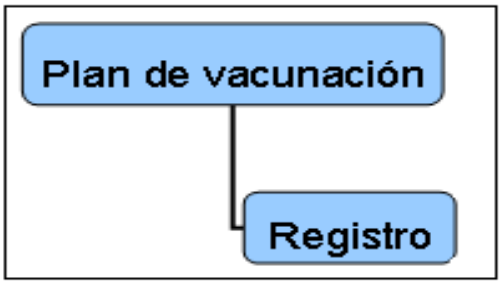


Figura 18 Sección plan de vacunación

Figura 19 Sección Reportes Generales

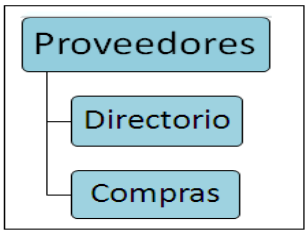


Figura 20 Sección Proveedores

A continuación se describe el proceso y los resultados obtenidos de la aplicación del Marco de Referencia al módulo avícola ModAvi, para el cual se tomaron en cuenta las siguientes secciones: lote de aves con sus ítems existentes y lotes comprados, galpones con sus ítems registros e informes, y proveedores con el ítem compras.

5.2.2 Evaluación de los patrones de Presentación

Después de haber definido el módulo del servicio Web agropecuario a utilizar en el caso de estudio, se procedió a revisar el módulo en general, y luego se realizó una evaluación exhaustiva del módulo teniendo como referencia los patrones de presentación, de lógica de dominio y datos, los cuales aparecen descritos en el capítulo 3.

Para evaluar el módulo se generó, para cada patrón, una serie de preguntas que permitieran verificar realmente si la aplicación estaba cumpliendo con dichos patrones. Definiéndose una matriz con un conjunto de preguntas para cada patrón, la cual consta de los siguientes campos:

Tabla 64 Formato de Matriz

ID	El número de cada pregunta.		
Pregunta	Descripción de la pregunta, definida para el patrón evaluado. Por patrón se tiene una o más preguntas.		
Patrón relacionado	Nombre del patrón a evaluar.		
Ruta	La ruta o el camino donde se encontró el hallazgo o la no conformidad.		
Tipo de Respuesta			
C		**NC**	**NA**
Conforme (C), significa que el módulo cumple con la pregunta.		No conforme (NC), significa que el módulo no cumple con la pregunta.	No Aplica (NA), cuando la pregunta no era pertinente para este tipo de aplicación. Por la lógica de negocio manejada.
Hallazgo	Descripción de la no conformidad o hallazgo encontrado.		
Observaciones/ Recomendación	Algunas observaciones o recomendaciones dadas para solucionar la no conformidad		

A continuación se da un ejemplo de algunas de las preguntas realizadas, en este caso para el patrón narrativa, el cual aparece descrito en el anexo 4.

Tabla 65 Ejemplo del formato para evaluación con base en las propiedades y los patrones

ID	Interfaz	Patrón relacionado	Ruta	C	NC	NA	Hallazgo	Observaciones/ Recomendación
1	¿Está la información presentada en lenguaje natural, en cuanto a redacción?	**Narrativa**	Avícola/ lotes de aves/Existentes. Avícola/lotes de aves/Lotes comprados		x		El texto no expresa lo que realmente se hace en los enlaces existente y lotes comprados, falta mas detalle y claridad.	Se recomienda cambiar el enlace de lotes comprados por lotes no asignados, ya que este nombre da más claridad acerca de lo que se hace en estos enlaces.

2	¿Se usan colores, fuentes y espacios en blanco para enfatizar los puntos de interés?		Avícola		x		El color de la letra del menú avícola le falta más color, que resalte ya que este es un contenido importante.	Se recomienda colocarle negrilla al texto o escoger un color mas fuerte.

En la tabla 65 se puede ver que al tomar como referencia el patrón narrativa se encontró que en el módulo avícola en las rutas: Avícola/ lotes de aves/Existentes, Avícola/lotes de aves/Lotes comprados, presentaba una no conformidad la cual hace referencia a: El texto no expresa lo que realmente se hace en los enlaces existente y lotes comprados, falta más detalle y claridad en los mismos. Para resolver esta no conformidad se dio la siguiente recomendación: cambiar el enlace: lotes comprados por lotes no asignados, ya que este nombre es más intuitivo para el usuario.
Las preguntas de todos los patrones pueden verse detalladamente en los Anexos 15, 16 y 17.

Teniendo la matriz se realizó primeramente la evaluación del módulo a nivel de interfaz y de navegación, con el fin de evidenciar el cumplimiento o no de cada pregunta de los patrones de presentación. Para esta evaluación se utilizó la evaluación heurística descrita en el capítulo 4, teniendo como principios heurísticos las propiedades de usabilidad definidas en el capítulo 2, y los patrones de presentación definidos en el capítulo 3. Como personas expertas las tres integrantes de este trabajo de grado, quienes por las materias vistas como lo son: usabilidad y calidad de software; además del conocimiento adquirido a lo largo de la carrera y por la literatura leída a lo largo de este trabajo de grado, se consideraron hábiles para realizar la evaluación.
Luego se realizó la valoración a nivel lógico, para comprobar la satisfacción o no de las preguntas realizadas para los patrones.

5.2.2.1 Resultados de la Evaluación

Para la evaluación se definió la siguiente escala:

Tabla 66 Escala para la evaluación

Grado de cumplimiento	Descripción
NU No Utilizado	Se considera que un patrón no es utilizado cuando no es conforme con las preguntas respectivas de cada patrón.
PU Parcialmente Utilizado	Si la aplicación es conforme con algunas de las preguntas realizadas para el patrón.
AU Altamente Utilizado	Si es conforme con la mayoría de preguntas realizadas para el patrón, más de la mitad de estas.
CU Completamente Utilizado	Si es conforme con todas las preguntas realizadas para el patrón.

Los resultados obtenidos en la evaluación realizada al módulo avícola, de los patrones de presentación, de lógica y datos se describen en las Tablas 68, 69 y 70.

El formato que se manejará de aquí en adelante en todas las Tablas es el siguiente:

Tabla 67 Formato Tablas

Nombre del Patrón	No de Preguntas	Hallazgos Encontrados	Grado de Cumplimiento
Nombre del patrón	Número de preguntas definidas para el patrón.	Número de no conformidades, es decir total de preguntas del patrón, con las cuales el módulo no cumple	Hace referencia a la escala definida en la tabla 5.1.

Tabla 68 Resultados de la Evaluación de los Patrones de Presentación

Patrones Presentación	No de Preguntas	Hallazgos encontrados	Grado de cumplimiento
Alto Nivel			
MVC	11	0	CU
PAC	4	2	PU
Nivel Intermedio			
Controlador de Página	5	0	CU
Controlador principal	3	3	NU
Controlador de la Aplicación	1	0	CU
Plantilla de Vista	2	2	NU
Transformador de Vista	2	2	NU
Dos Fases por Vista	1	1	NU
Enlace como Relación entre Vistas	1	0	CU
Observador de Navegación	1	1	NU
Nodo como una Unidad única	1	0	CU
Método Crear Nodo y Crear Enlace	1	0	CU
Contexto Navegacional	1	0	CU
Referencia Activa	1	0	CU
Bajo Nivel o de Interfaz de usuario			
Información Solicitada	3	1	AU
Desacoplar la Interacción de la Información	2	0	CU
Agrupación Conductual	2	1	PU
Anticipación Conducta	1	2	NU
Proceso de Retroalimentación	2	1	PU
Narrativa	2	2	NU
Despliegue de Información de Alta Densidad	4	2	PU
Formulario	7	2	AU
Panel de Control	3	0	CU
Editor WYSIWYG	2	0	CU
Comando Compuesto	2	2	NU
Espacios Navegacionales	1	1	NU
Visión Global en Detalle	3	0	CU
Grupos Pequeños de Cosas Relacionadas	1	0	CU
Conjunto de Jerarquías	2	0	CU
Conjunto Tabulador	2	0	CU
Mapa o Grafica	1	1	NU
Detalle Opcional en Demanda	1	1	NU

Desactivar Cosas No Pertinente	2	1	PU
Mostrar el Indicador de Acceso	1	0	CU
Descripción corta	1	0	CU
Apariencia de Fondo	1	0	CU
Superficie Activa Central	2	0	CU
Montón de Superficies Activas	1	0	CU
Mapa de Espacios Navegacionales	1	1	NU
Puntos Claros de Entrada	1	0	CU
Secciones de Color Codificado	1	0	CU
Regresar a un Lugar Seguro o Punto de Control	1	0	CU
Acciones Convenientes de Ambiente	1	1	NU
Acciones de Localización del Objeto	1	0	CU
Acciones para Objetos Múltiples	1	0	CU
Colección Editable	1	0	CU
Absolver Entrada de Texto	1	0	CU
Entrada de Texto Estructurado	1	0	CU
Barra de Herramientas	1	0	CU
Preferencias de Usuario	1	1	NU
Anotaciones de Usuario	1	1	NU
Framework Repetido	1	0	CU
Buenas por Defecto	1	1	NU
Estado Recordado	1	0	CU
Historia de la Interacción	1	1	NU
Mensaje Importante	2	0	CU
Chequeo Real	1	1	NU
Demostración	1	1	NU
Total	105	36	

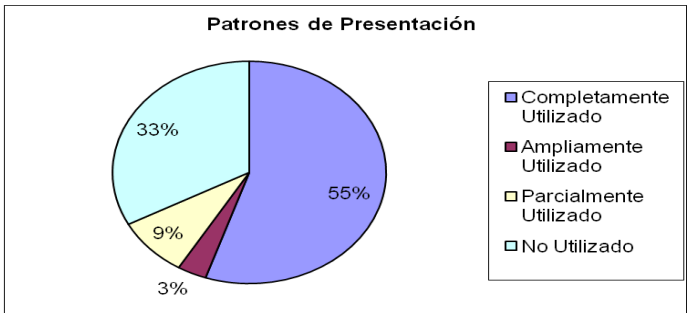


Figura 21Porcentaje de Utilización de los Patrones de Presentación

Como puede observarse en la figura 21, los patrones de presentación, se están utilizando completamente en un 55%, altamente utilizados un 3%, parcialmente utilizados un 9% y No utilizados un 33%.

Debido a los resultados anteriores se sugirió realizar los siguientes cambios en orden de importancia:

1. Resaltar con color el enlace donde el usuario actualmente se encuentra.
2. Colocar un texto de ayuda a los íconos que no son intuitivos.

3. Poner en el mismo orden los enlaces en los diferentes formularios.
4. Cambiar los nombres de algunos enlaces.
5. Organizar de una manera organizada y fácilmente visible formularios como el de factura.
6. Realizar un mapa del sitio.
7. Mostrar los reportes en Figuras.
8. Registrar la historia de navegación.
9. Proporcionar comandos con los cuales se pueda acceder a la funcionalidad de la aplicación, como por ejemplo ofrecer una manera de hacer auto completación.

Los anteriores cambios se tendrán en cuenta en la evaluación que se presenta en la sección 5.2.4.1.
Las **propiedades** que se encuentran comprometidas al realizar estos cambios son: consistencia, ayuda u orientación, minimizar la carga cognitiva, facilidad de navegación, experiencia de usuario, gestionando el error, lenguaje de los usuarios, facilidad de recordar el sistema, realimentación y personalización.

Los **patrones** que ayudan a dar solución a estas falencias son: Proceso de realimentación, agrupación conductal, anticipación de conducta y observador de navegación, narrativa, despliegue de la información de alta densidad, espacios navegacionales, comando compuesto, mapa o grafica y preferencias de usuario.

Debido a lo anterior puede concluirse que el módulo tiene varias falencias en usabilidad, por el número de propiedades que estos patrones afectan y por el impacto alto que tienen. Es importante anotar que el módulo también fue evaluado por dos personas expertas en la lógica de dominio de la aplicación, las granjas avícolas, los cuales corroboraron los resultados obtenidos.

Para ver en detalle los hallazgos encontrados así como las recomendaciones encontradas, puede dirigirse al Anexo 15.

Cabe anotar que hubo varios patrones que no aplicaron para el módulo avícola, estos son:

- Instrucciones paso a paso, el cual hace referencia a aspectos como el de guiar al usuario a través de una tarea compleja, lo cual no aplicó porque en el módulo no se tienen tareas que haya la necesidad de explicar paso a paso, como por ejemplo la instalación de un software, para lo cual si sería conveniente, utilizar un wizard.
- Pila de superficies activas, tiene que ver con organizar las superficies o pantallas en pilas. No se consideró pertinente para el módulo debido a que esta organización amerita cuando, como en el caso de Word, se pueden abrir varias ventanas y apilarlas, sin que la una afecte la otra, en este caso no sería conveniente porque al trabajar varias interfaces a la vez, puede ocasionar daños en la integridad de los datos, ya que gran parte de éstas dependen entre si.
- Opción de un conjunto pequeño, indica al usuario el tiempo que se demora en cargar o descargar un archivo, lo cual no aplicó debido a que en este módulo no se realizan estas acciones. Además cuando se está cargando la aplicación, el navegador, muestra una barra de progreso.
- Personal object space, tiene que ver con permitirle al usuario poner, mover, agrupar y ordenar las cosas como ellos deseen. Este patrón no se consideró para esta versión aunque no se descarta su utilización si es necesario en futuras

versiones, lo que contribuirá a propiedades como explicito control de usuario y adaptabilidad.

- Marcador de libros, permite al usuario guardar un registro de los lugares y sus puntos de interés, este patrón no se considero pertinente debido a que la navegabilidad de este módulo es sencilla, sin muchos enlaces, pero sin embargo puede usarse en otras versiones, si la navegación se vuelve compleja, lo cual ayudaría a propiedades como ayuda u orientación, facilidad de navegación y facilidad de recordar el sistema.
- Indicador de progreso, tiene que ver con mostrarle al usuario el progreso de la operación en el tiempo. No aplicó, ya que el módulo actualmente no tiene operaciones que tarden tiempo en ejecutarse o cargarse.

5.2.3 Evaluación de los Patrones de Lógica de Dominio y Datos

Luego de realizar la evaluación al módulo "Avícola" teniendo como referencia los patrones de presentación , se evaluó la arquitectura de la aplicación, utilizando el método de inspección, descrito en el capítulo 4, teniendo como base los patrones de lógica y de datos, expuestos en el capítulo 3.

Antes de empezar la evaluación, fue necesario consultar las características del lenguaje de programación usado, en este caso php.

Al revisar la aplicación se encontró que la aplicación IKERNELL tiene una arquitectura cliente/ servidor, de tres capas, capa de presentación, lógica y datos, con la mayoría del trabajo pesado en el servidor y el cliente(s) se ocupa de la interacción con el usuario y de algunas validaciones. El módulo consta de varios tipos de archivos los cuales son: Archivos "js", Archivo "php", Archivo "tpl", que es el archivo que tiene el html.

- Archivo "js": El cual contiene código en Javascript, este archivo se encarga de la parte de control, y existe uno de estos por cada página y uno general para las validaciones comunes.
- Archivo "php": En este archivo se encuentra toda la lógica de la aplicación, se hacen llamados a los métodos de la clase SWA que es la que ejecuta los procedimientos de la base de datos. Algo de resaltar en este archivo es que se encontró una librería de la cual no se tenía conocimiento, para lo cual fue necesario investigar de qué se trataba, Smarty. Llegando entonces a comprender que Smarty es una plantilla que sirve para separar el código PHP, que tiene la lógica de negocios, del código HTML, la lógica de la presentación. La utilización de esta plantilla es muy importante debido a que contribuye a que la aplicación Web cumpla con patrones como el de capas y el MVC.
- Archivos "tpl": En este archivo se encuentra el código html, el cual se encuentra separado del código de Javascript, lo que se conoce como Javascript no obstructivo, lo cual es muy favorable, además se encuentra separado del código css, es decir de las hojas de estilo, que se encargan de definir la presentación de un documento html.

En la figura 22 se muestran los componentes mencionados y las relaciones que entre ellos existen, se puede ver que la relación entre los diferentes archivos, aunque la relación más fuerte se tiene entre los archivos js con tpl, js con php, debido a que los archivos js

son los controladores en la aplicación los cuales cuando llega una solicitud realizan una petición a los archivos php los cuales contienen la lógica de dominio, cuando obtienen una respuesta la pasan a los tpl los cuales conforman las vistas de la aplicación; debido a que los js son los que, por lo general, envían información a los tpl se puede concluir que la relación más débil se presenta de los archivos tpl a los js y de los archivos php a los tpl. Por la misma razón.

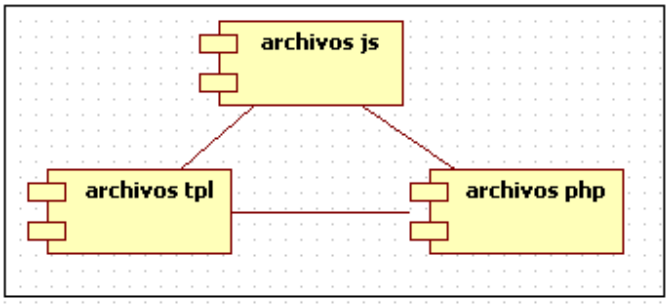


Figura 22 Componentes del Módulo Avícola

Tabla 69 Resultados de la evaluación de los patrones de Lógica de Dominio

Patrones Lógica de Dominio	No de Preguntas	Hallazgos encontrados	Grado de cumplimiento
Alto Nivel			
Capas	6	1	AU
Nivel Intermedio			
Pizarra	4	2	PU
Mediador	5	3	PU
Microkernell	6	6	NU
Reflexión	5	5	NU
Total	26	17	

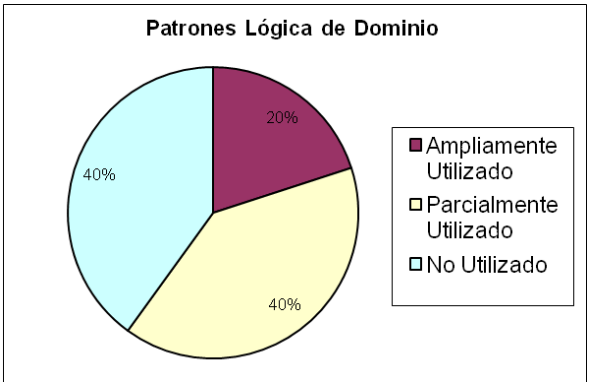


Figura 23 Porcentajes de utilización de los Patrones de Lógica de Dominio

Como puede observarse en la figura 23 el 40% corresponde al porcentaje de No utilización de los patrones definidos en la tabla 69, el cual es un porcentaje bastante alto. El 60% es el porcentaje de utilización de los patrones y cabe resaltar que dicho porcentaje

corresponde a la aplicación de patrones que no eran conocidos por los desarrolladores de la empresa, tales como son el de capas, mediador y pizarra, los cuales como se describió en el capítulo 3, tienen un impacto de usabilidad alto y medio positivamente en propiedades como la consistencia, gestionando el error, facilidad de navegación, accesibilidad, realimentación, facilidad de navegación y adaptabilidad: personalización. Aunque la aplicación cumple con varios de los patrones de la tabla 69, cabe anotar que existe un gran porcentaje de no utilización, por lo que se les recomendó seguir el proceso de desarrollo descrito en el capítulo 4, y estudiar los patrones descritos en el capítulo 3, debido a que, de esta manera, podrán tener una mejor comprensión de estos y a la vez les guiará en que etapa del desarrollo utilizarlos. Teniendo una aplicación con un grado de usabilidad mayor, debido a las implicaciones que dichos patrones tienen en la usabilidad.
Además, se sugirió separar algunos archivos que todavía tenían el código php y html embebido, porque aunque gran parte de estos archivos tenían esta separación gracias a la utilización de la plantilla smarty, aún hacían falta algunos. Esto con el fin de ayudar al cumplimiento de patrones como el de capas.
Para ver en detalle los hallazgos encontrados así como las recomendaciones encontradas, puede dirigirse al Anexo 16.

Tabla 70 Resultados de la Evaluación de los Patrones de Datos

Patrones de Datos	No de Preguntas	Hallazgos encontrados	Grado de cumplimiento
Acceso a Datos			
Script de Transacción	1	0	CU
Modelo del Dominio	1	1	NU
Modulo de Tabla	1	1	NU
Total	3	2	

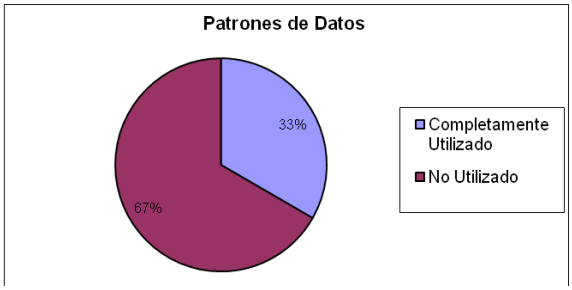


Figura 24 Porcentaje de utilización de los patrones de Datos

De acuerdo a la figura 24 se puede afirmar que en el módulo Avícola ModAvi los patrones de datos se aplican completamente en un 33 % y no se utilizan en un 67%.
Aunque la aplicación cumple con uno de los patrones de la tabla 70, cabe anotar que existe un gran porcentaje de no utilización, debido a estos resultados se sugirió estudiar patrones como el modelo de dominio y el modulo de tabla, los cuales se encuentran explicados en el capítulo 3 de este trabajo de grado, con el fin de brindar mas opciones para acceder a los datos de la aplicación.
Para ver en detalle los hallazgos encontrados así como las recomendaciones encontradas, puede dirigirse al Anexo 17.

5.2.4 Resultados Finales

Después de dar a conocer los hallazgos encontrados al desarrollador, y que él realizara los cambios pertinentes, se procedió a evaluar nuevamente la aplicación en base a la matriz mencionada en la sección 5.2.2, obteniéndose los siguientes resultados.

5.2.4.1 Resultados de los patrones de Presentación

Tabla 71 Resultados de la Evaluación de los Patrones de Presentación

Patrones Presentación	No de Preguntas	Hallazgos encontrados	Grado de cumplimiento
Alto Nivel			
MVC	11	0	CU
PAC	4	2	PU
Nivel Intermedio			
Controlador de Página	5	0	CU
Controlador principal	3	3	NU
Controlador de la Aplicación	1	0	CU
Plantilla de Vista	2	1	PU
Transformador de Vista	2	1	PU
Dos Fases por Vista	1	1	NU
Enlace como Relación entre Vistas	1	0	CU
Observador de Navegación	1	0	CU
Nodo como una Unidad única	1	0	CU
Método Crear Nodo y Crear Enlace	1	0	CU
Contexto Navegacional	1	0	CU
Referencia Activa	1	0	CU
Bajo Nivel o de Interfaz de Usuario			
Información Solicitada	3	0	CU
Desacoplar la Interacción de la Información	2	0	CU
Agrupación Conductual	2	0	CU
Anticipación Conducta	1	0	CU
Proceso de Retroalimentación	2	1	PU
Narrativa	2	0	CU
Despliegue de Información de Alta Densidad	4	0	CU
Formulario	7	1	AU
Panel de Control	3	0	CU
Editor WYSIWYG	2	0	CU
Comando Compuesto	2	2	NU
Espacios Navegacionales	1	0	CU
Visión Global en Detalle	3	0	CU
Grupos Pequeños de Cosas Relacionadas	1	0	CU
Conjunto de Jerarquías	2	0	CU
Conjunto Tabulador	2	0	CU
Mapa o Grafica	1	0	CU
Detalle Opcional en Demanda	1	0	CU
Desactivar Cosas No Pertinente	2	0	CU
Mostrar el Indicador de Acceso	1	0	CU
Descripción corta	1	0	CU
Apariencia de Fondo	1	0	CU

Superficie Activa Central	2	0	CU
Montón de Superficies Activas	3	0	CU
Mapa de Espacios Navegacionales	1	1	NU
Puntos Claros de Entrada	1	0	CU
Secciones de Color Codificado	1	0	CU
Regresar a un Lugar Seguro o Punto de Control	1	0	CU
Acciones Convenientes de Ambiente	1	0	CU
Acciones de Localización del Objeto	1	0	CU
Acciones para Objetos Múltiples	1	0	CU
Colección Editable	1	0	CU
Absolver Entrada de Texto	1	0	CU
Entrada de Texto Estructurado	1	0	CU
Barra de Herramientas	1	0	CU
Preferencias de Usuario	1	1	NU
Anotaciones de Usuario	1	1	NU
Framewok Repetido	1	0	CU
Buenas por Defecto	1	0	CU
Estado Recordado	1	0	CU
Historia de la Interacción	1	0	CU
Mensaje Importante	2	0	CU
Chequeo Real	1	0	CU
Demostración	1	1	NU
Total	107	16	

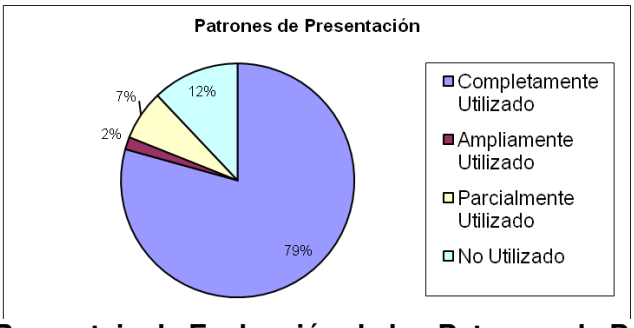


Figura 25 Porcentaje de Evaluación de los Patrones de Presentación

De acuerdo a la figura 25 se puede afirmar que en el módulo Avícola ModAvi gran parte de los hallazgos encontrados fueron corregidos, de un 55% de patrones completamente utilizados se paso a un 79%, con un 2% de patrones ampliamente utilizados, 7% de patrones parcialmente utilizados y un 12% de no utilizados, el cual era anteriormente de un 33%.

Los anteriores resultados muestran una mejora en la usabilidad gracias a la utilización de los **patrones**: agrupación conductal, anticipación de conducta, observador de navegación, narrativa, despliegue de la información de alta densidad, espacios navegacionales y preferencias de usuario.

La mejora de la usabilidad se hace notoria ya que se cumplen las **propiedades:** consistencia, ayuda u orientación, minimizar la carga cognitiva, facilidad de navegación, experiencia de usuario, lenguaje de los usuarios, facilidad de recordar y realimentación.

Entre los cambios realizados en el módulo se encuentran:

- Resaltar con color el enlace donde el usuario actualmente se encuentra.
- Colocar un texto de ayuda a los iconos que no eran intuitivos.
- Poner en el mismo orden los enlaces en los diferentes formularios.
- La utilización de nombres de algunos enlaces acorde con lo realizado.
- Mejora en la organización y visibilidad de formularios como el de factura.
- Presentación de los reportes en gráficos no solo en plantillas.
- Desactivación de varios campos en los formularios, con el fin de brindar una mejor comprensión al usuario.

Los cambios mencionados fueron muy importantes para mejorar la interacción y la navegación, obteniendo como resultado una aplicación con un mayor grado de usabilidad de la que se tenía inicialmente.

Faltaron por corregir algunos hallazgos, especialmente el de registrar la historia de navegación y un error en el formulario factura, los cuales están pendientes para futuras versiones de la aplicación. Ya que no se puede olvidar que la mejora de un producto software es un proceso continuo e incremental, en el que influyen diversos factores como el tiempo, costos, etc., por lo que, a veces no todos los cambios se realizan en un solo ciclo. Para este caso en particular, el factor determinante fue el tiempo del desarrollador, el cual solo trabaja tiempo parcial en la empresa y además tiene a su cargo otras responsabilidades. Pero cabe anotar que se logró el compromiso por parte de él, para implementar éste y los otros cambios restantes, en la próxima versión del módulo.

Para ver en detalle los hallazgos corregidos y los que faltan por realizar, puede dirigirse al Anexo 18.

5.2.4.2 Resultados de los Patrones de Lógica de Dominio y Datos

Tabla 72 Resultados de la Evaluación a los Patrones de Lógica de Dominio

Patrones Lógica de Dominio	No de Preguntas	Hallazgos encontrados	Grado de cumplimiento
Alto Nivel			
Capas	6	0	CU
Nivel Intermedio			
Pizarra	4	2	PU
Mediador	5	3	PU
Microkernell	6	6	NU
Reflexión	5	5	NU
Total	26	16	

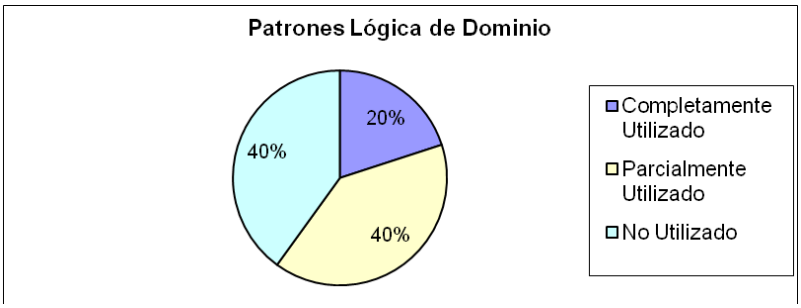


Figura 26 Porcentajes de Evaluación de los Patrones de Lógica de Dominio

De acuerdo a la figura 26 se puede afirmar que en el módulo Avícola ModAvi se mejoró la utilización de los patrones definidos en la tabla 72, de un 20% de patrones altamente utilizados se paso a un 20% de patrones completamente utilizados. Mejorándose patrones como el de capas.

Los cambios realizados fueron la separación del código php y el html, de archivos como buscar lote. Obteniendo como resultado una mejora en la utilización del patrón capas y a la vez en algunos patrones de presentación como el MVC, vista de plantilla y vista de transformación. Además de una arquitectura que contribuye a las propiedades de usabilidad: gestionando el error, accesibilidad, y consistencia, debido al impacto que éstos patrones tienen en la usabilidad, lo cual fue explicado en el capítulo 3.

Los patrones pendientes por implementar, se tendrán en cuenta para futuras versiones de la aplicación, o para futuros desarrollos, esto debido al factor tiempo mencionado anteriormente, que afecta sustancialmente un proceso de mejora.

Cabe anotar que a la empresa se le entregó el material con todos estos patrones y el proceso de desarrollo, además se realizó una exposición de patrones como el MVC, capas, plantilla de vista, transformador de vista, script de transacción, entre otros, mostrándose un gran interés, por parte del equipo, de desarrollo en utilizar estos patrones en las próximas aplicaciones.

Para ver en detalle los hallazgos corregidos puede dirigirse al Anexo 19.

Tabla 73 Resultados de la Evaluación de los Patrones de Datos

Patrones de Datos	No de Preguntas	Hallazgos encontrados	Grado de cumplimiento
Acceso a Datos			
Script de Transacción	1	0	CU
Modelo del Dominio	1	1	NU
Modulo de Tabla	1	1	NU
Total	3	2	

Figura 27 Porcentajes de Evaluación de los Patrones de Datos

Como puede observarse en la figura 27 no hubo cambios en el porcentaje de utilización de los patrones de datos, ello se debe a que el equipo de desarrollo tuvo como prioridad los cambios sugeridos para los patrones de presentación y de lógica. Aunque, se logro el interés por parte de la empresa de aplicar éstos patrones en próximas aplicaciones.

Para ver en detalle los hallazgos pendientes por corregir puede dirigirse al Anexo 20.

5.2.4.3 Resultados de la Percepción de Personas Externas

Para corroborar los resultados presentados en la sección 5.2.4, se tomó una muestra de estudiantes (7 estudiantes) con conocimientos sobre usabilidad, a los cuales se les pidió que evaluaran la aplicación antes y después de los cambios, proporcionándoles una encuesta basada en 12 preguntas, las cuales tenían una calificación de 1 a 4, siendo:

1= Malo, la pregunta no es satisfecha por el módulo.
2= Medio Malo, la pregunta es parcialmente satisfecha por el módulo.
3= Medio Bueno la pregunta es ampliamente satisfecha por el módulo.
4=Muy Bueno: la pregunta es completamente satisfecha por el módulo.

En todo este proceso se uso el Método para Evaluar la Usabilidad de una Aplicación Web a partir de la Arquitectura, presentado en la sección 4.2.5.4 teniendo como heurísticas las propiedades de usabilidad, las cuales fueron explicadas a los estudiantes, y también se usó la evaluación por cuestionarios. Estos Métodos se encuentran explicados en el capítulo 4 de este trabajo de grado.

La ponderación de los resultados se realizó de la siguiente manera:
Se sumaron todos los valores iguales, por ejemplo si 3 estudiantes dieron el valor de 1 a cualquiera de las preguntas se consideraba que el módulo tenía un total de 3, dando como resultado una calificación Mala.
Los resultados obtenidos pueden verse en las figuras 32 y 33:

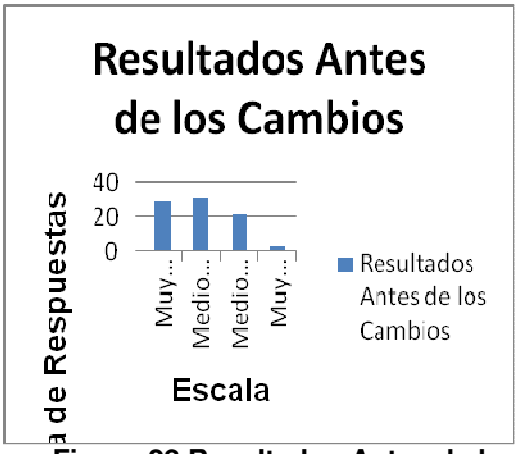


Figura 28 Resultados Antes de los Cambios

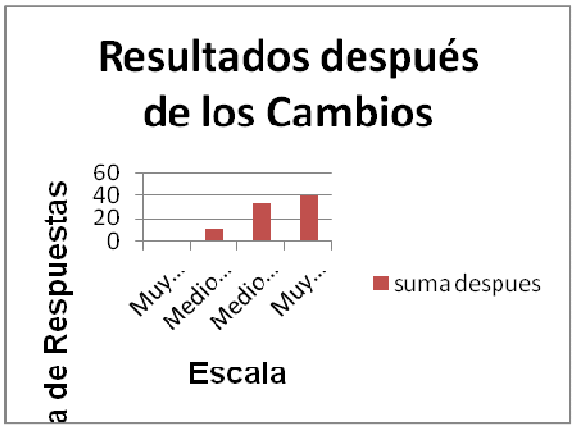


Figura 29 Resultados Después de los Cambios

Como puede verse en la figura 28 los valores muy malo y medio malo tienen un porcentaje de 35 % y 37 %, después de los cambios realizados tienen un porcentaje de 0% y 13 %. Esto indica una notable diferencia, entre lo que se tenía antes y después, lo que significa que la percepción de los evaluadores externos coincide con lo expuesto en la sección 5.2.4. Gran parte de las falencias encontradas fueron solucionadas, quedando solo un pequeño porcentaje por corregirse en el módulo Avícola. Para los valores medio bueno y muy bueno respectivamente, antes se tenía un porcentaje de 25% y 3%, y después se obtuvo un porcentaje de 39% y 48 %, por lo que, se puede afirmar que los evaluadores concluyen que antes el módulo era bueno pero que después la usabilidad del módulo mejoró. Estos resultados además corroboran las hipótesis que los **patrones**: agrupación conductual, anticipación de conducta, observador de navegación, narrativa, despliegue de la información de alta densidad, espacios navegacionales y preferencias de usuario, mejoran las **propiedades** de usabilidad: mapeo natural, minimizar la carga cognitiva, realimentación, ayuda u orientación, facilidad de recordar el sistema, lenguaje de los usuarios, consistencia, facilidad de navegación, experiencia de usuario.

Los resultados encontrados en la sección 5.2.4, también se ratifican con los comentarios realizados por los evaluadores, como los siguientes:

- "El módulo mejoro después de los cambios realizados."
- "El módulo ahora cumple con una gran cantidad de propiedades de usabilidad."
- "En general la aplicación mejoro, pues hay mayor cumplimiento de las propiedades de usabilidad y es más intuitiva para el usuario respecto a la versión anterior"

El considerar personas externas proporciona una mayor objetividad a los resultados obtenidos en este caso de estudio, dando validez a los resultados expuestos en la parte 5.2.4.
Para ver de manera visual algunos de los cambios realizados en el módulo avícola, tanto en la interfaz como en la arquitectura, resultado de evaluar los patrones de Presentación y de lógica, puede dirigirse al anexo 27

5.3 Caso de estudio Compromiso

Compromiso es una herramienta software basada en las normas ISO 9000 con el objetivo de apoyar el sistema de gestión de la calidad, brindando la posibilidad de hacer un seguimiento completo de los procesos y requisitos necesarios para cumplir con dicho estándar. Este producto va dirigido a las empresas certificadas, empresas en proceso de certificación o empresas que busquen organizar sus procesos internos como la comunicación y divulgación del Sistema de Gestión de Calidad, Administración de la Documentación, Seguimiento de No Conformidades y Gestión de Equipos (equipos de computo, maquinaria, instrumentos de medición y demás).

El sistema se encuentra en etapa de desarrollo, y una vez terminado contara con 9 módulos: Comunicación interna, Documentos, Actas de calidad, Recurso humano, No conformidades, Objetivos de calidad, Auditoria, Gestión de equipos, Configuración; de los cuales se encuentran implementados hasta el momento el módulo de configuración, documentos, gestión de equipos, no conformidades y comunicación interna (sujetos a posibles cambios y ajustes).

Configuración: Permite paramétrizar y adaptar el software Compromiso de acuerdo a las necesidades de la empresa.
Documentos: Permite la publicación y control a través de una interfaz Web de los documentos actualizados del sistema de gestión de calidad. La funcionalidad de este módulo está enfocada en el proceso de elaboración, edición y control de cambios de estos documentos.
Gestión de equipos: Permite mantener el registro y control de todos los equipos que afectan de alguna forma el nivel de calidad de los productos o servicios de la empresa como maquinaria, vehículos, equipos de computo y demás, detallando los procesos de planeación de mantenimiento y/o calibración preventiva, ejecución de las tareas de mantenimiento y/o calibración preventiva y correctiva para cada equipo.
No conformidades: Permite registrar y hacer gestión de las No Conformidades, Acciones Correctivas y Preventivas incluyendo las quejas y reclamos tanto internos como externos de la organización dándoles el trámite correspondiente.
Comunicación interna: Permite difundir, comunicar, evaluar el sistema de gestión de calidad en general, promueve la divulgación y creación de cultura de mejoramiento continuo dentro de la organización.
Auditoria: Permite realizar gestión de las auditorías internas desde la plantación, planeación hasta el registro de los hallazgos de auditoría (No Conformidades) facilitando su seguimiento.

5.3.1 Módulo No Conformidades

Este módulo permite registrar y hacer gestión de las No Conformidades que se presentan en los procesos de la empresa, determinando las causas e implementando Acciones Correctivas y Preventivas para asegurarse que la “No Conformidad”, no vuelva a ocurrir, contribuyendo así, con la mejora continua de los procesos. También incluye las quejas y reclamos tanto internos como externos de la organización dándoles el trámite correspondiente.
Este módulo tienen un tiempo de desarrollo de un año, lo lleva a cabo un desarrollador con asesorías de algunos ingenieros industriales quienes ofrecen información de los procesos de la norma ISO 9000 y diseñadores gráficos para la parte de interfaz. La

versión actual es la primera que se ha tenido. Este módulo está en proceso de venta a una empresa de Popayán, por lo tanto, no tiene usuarios finales por el momento.

En las figuras siguientes se muestra las acciones que este módulo permite hacer como lo son:

- Registrar no conformidades por los integrantes de la organización y personal externo a ella.
- Procesar la información de las no conformidades (Causas, actividades correctivas, responsables).
- Realizar seguimiento sobre las actividades y desarrollo de ellas mediante indicadores de cumplimiento.
- Evaluar los resultados y eficacia de las acciones correctivas tomadas para solucionar la no conformidad.
- Llevar registro sobre todos los acontecimientos que generaron no conformidades y las acciones correctivas que se tomaron frente a ellas.

Para el caso de estudio se tomó en cuenta los componentes: procesar la información, registrar una no conformidad, realizar el seguimiento, en el cual el usuario registra una no conformidad, las causas relacionadas, las actividades, los planes de acción por cada actividad y los responsables de dar solución a la no conformidad.
El proceso seguido en el caso de estudio se describe a continuación.

5.3.1.1 Evaluación de los Patrones de Presentación

Después de haber definido el módulo a evaluar en el caso de estudio del producto "Compromiso", se realizó el mismo procedimiento descrito en la sección 5.2.2, donde se especifica el proceso de evaluación y se da un ejemplo en la Tabla 65 de las preguntas que se hicieron por cada patrón analizado en el capítulo 3, todas las preguntas se pueden ver con más detalle en los Anexos 21,22 y 23 las cuales siguen el mismo formato dado en la Tabla 64.

5.3.1.1.1 Resultados de la Evaluación

Los resultados obtenidos en la evaluación realizada al módulo "No conformidades", en cuanto a interfaz y navegación de la aplicación Web, con base en los patrones de presentación teniendo en cuenta la escala de la Tabla 66 y el formato de la Tabla 67 se describen a continuación:

Tabla 74 Resultados de la evaluación de los patrones de presentación

Patrones Presentación	No de Preguntas	Hallazgos encontrados	Grado de cumplimiento
Alto Nivel			
MVC	13	1	AU
PAC	4	4	NU
Nivel Intermedio			
Controlador de Pagina	6	1	AU
Controlador de Frente	3	3	NU
Controlador de la Aplicación	1	0	CU
Plantilla de Vista	2	2	NU
Transformador de Vista	2	2	NU
Dos Fases por Vista	1	1	NU
Enlace como Relación entre Vistas	1	0	CU

Observador de Navegación	1	0	CU
Nodo como una Unidad única	1	0	CU
Método Crear Nodo y Crear Enlace	1	0	CU
Contexto Navegacional	1	0	CU
Referencia Activa	1	0	CU
Bajo Nivel o de Interfaz de usuario			
Información Solicitada	3	1	AU
Desacoplar la Interacción de la Información	2	0	CU
Agrupación Conductual	2	0	CU
Anticipación Conducta	1	1	NU
Proceso de Retroalimentación	2	1	PU
Narrativa	2	2	NU
Despliegue de Información de Alta Densidad	4	2	PU
Formulario	7	4	PU
Panel de Control	3	1	AU
Editor WYSIWYG	2	1	PU
Comando Compuesto	2	2	NU
Espacios Navegacionales	1	0	CU
Visión Global en Detalle	3	0	CU
Grupos Pequeños de Cosas Relacionadas	1	0	CU
Conjunto de Jerarquías	2	1	PU
Conjunto Tabulador	2	1	PU
Mapa o Grafica	1	1	NU
Detalle Opcional en Demanda	1	1	NU
Desactivar Cosas No Pertinente	2	1	PU
Mostrar el Indicador de Acceso	1	0	CU
Descripción corta	1	1	NU
Apariencia de Fondo	1	1	NU
Superficie Activa Central	2	1	PU
Mosaico de superficies Activas	3	1	AU
Montón de Superficies Activas	1	1	NU
Mapa de Espacios Navegacionales	1	1	NU
Puntos Claros de Entrada	1	1	NU
Secciones de Color Codificado	1	0	CU
Regresar a un Lugar Seguro o Punto de Control	1	1	NU
Acciones Convenientes de Ambiente	1	0	CU
Acciones de Localización del Objeto	1	0	CU
Acciones para Objetos Múltiples	1	1	NU
Colección Editable	1	0	CU
Absolver Entrada de Texto	1	0	CU
Entrada de Texto Estructurado	1	1	NU
Barra de Herramientas	1	1	NU
Preferencias de Usuario	1	1	NU
Anotaciones de Usuario	1	0	CU
Framewok Repetido	1	0	CU
Buenas por Defecto	1	1	NU
Estado Recordado	1	0	CU
Historia de la Interación	1	1	UN

Mensaje Importante	2	0	CU
Chequeo Real	1	1	NU
Demostración	1	1	NU
Total	**112**	**50**	

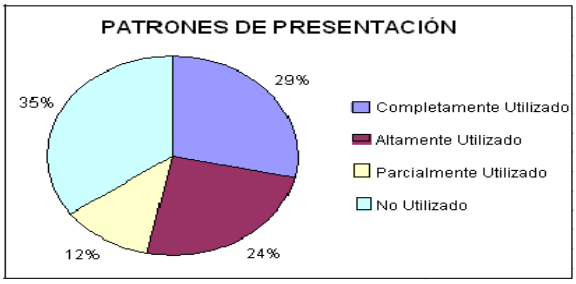


Figura 30 Porcentajes de utilización de los Patrones de Presentación

La figura 30 muestra que los patrones de presentación se aplican completamente en un 29%; teniendo un porcentaje alto en la no utilización de los patrones el cual corresponde a un 35%.
Estos resultados traen como consecuencia que la aplicación Web no cumpla con las expectativas del usuario en cuanto a las características de usabilidad; teniendo falencias en las **propiedades de usabilidad**: Realimentación, Ayuda u orientación, Lenguaje de los usuarios, Consistencia: Visual y Funcional, que tienen un impacto alto positivamente **en los patrones:** Narrativa, despliegue de información de alta densidad, formulario, comando compuesto, descripción corta, puntos claros de entrada, regresar a un punto de control, los cuales según la Tabla 74 no son utilizados.

Por lo tanto, se hicieron recomendaciones en orden de prioridad:

1. Colocarle a los enlaces nombres claros para el usuario; hacer una buena distribución de la información para que se vea en un solo pantallazo.
2. Proporcionar ayuda para llenar los formularios y una descripción corta de la función de los botones.

Existen algunos **patrones que no aplicaron** para el módulo “No conformidades” como por ejemplo: Instrucciones paso a paso, pila de superficies activas, opción de un conjunto pequeño, preferencia de usuario, espacio de objeto personal, secuencia de acción escrita, marcadores de libros, indicador de progreso, historia de la interacción. Estos patrones son utilizados para guiar al usuario paso a paso en actividades como la instalación del software, lo cual el módulo no lo requiere; tener varias superficies o vistas en pila, tampoco aplica por que la información que se presenta no requiere abrir varias ventanas; indicar al usuario el tiempo que se demora en cargar o descargar un archivo, el módulo no tiene un indicador pero cuando se carga un archivo el navegador Web muestra la barra de progreso, se podría tener su propio indicador para futuras versiones si se requiere; los otros patrones son para personalizar la aplicación y guardar el recorrido de navegación; se pueden implementar estos patrones a medida que la aplicación va evolucionando en futuras versiones. Las propiedades que mejorarían porque tienen un impacto alto positivamente en los patrones que no aplican al módulo son: Consistencia: Visual, Realimentación, Ayuda u orientación, Explicito control de usuario.

Al implementar la mayoría de los patrones de presentación en la aplicación Web, se logrará mayor interacción con el usuario, teniendo una mejor aceptación por parte del mismo.

Para ver en detalle los hallazgos puede dirigirse al Anexo 21.

5.3.1.2 Evaluación de los Patrones de Lógica de Dominio y Datos

Luego de la evaluación realizada al módulo "No conformidades", en cuanto a interfaz y navegación de la aplicación Web de los patrones de presentación se evaluó la arquitectura del software, con el método de inspección de código descrito en la sección 4.2.5.3 del capítulo 4; teniendo como base los patrones lógicos.

El producto de software "Compromiso" está desarrollado en el lenguaje de programación php, en la evaluación se tuvo en cuenta las características de este lenguaje para realizar una correcta valoración de los patrones.

Al revisar el código de la aplicación se encontró que tiene una arquitectura cliente/ servidor de tres capas, presentación, lógica y de datos con un cliente ligero que se ocupa de la interacción con el usuario y la mayoría del trabajo pesado esta en el servidor. La estructura consta de:

- El control, en el cual se encuentran todas las funciones comunes de los módulos.
- El modelo, contiene la parte que gestiona la lógica de la aplicación, pero no se encuentra la gestión de los datos. Esto se encuentra en el control.
- La Vista, contiene los formularios del módulo.
- El Script, donde están los controles de las vistas del módulo.

La estructura que tiene el módulo "No conformidades" contribuye al cumplimiento de los patrones Capas y MVC que complementan la arquitectura cliente / servidor.

En la figura 37 se muestran los componentes mencionados y las asociaciones entre ellos. De esta forma al llegar una solicitud el script realiza un llamado al modelo para poder dar respuesta a la solicitud, el modelo da la respuesta al script y este se comunica con la vista para mostrar los resultados al usuario.

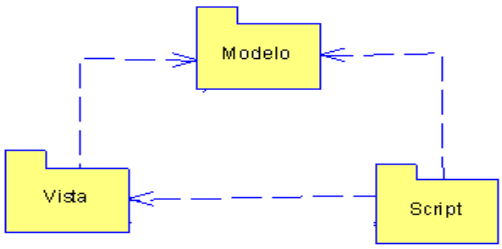


Figura 31Componentes del módulo

Tabla 75 Resultados de la Evaluación de los Patrones de Lógica de Dominio

Patrones Lógica de Dominio	No de Preguntas	Hallazgos encontrados	Grado de cumplimiento
Alto Nivel			
Capas	6	0	CU
Nivel Intermedio			

Pizarra	4	2	PU
Micro-Kernel	6	6	NU
Reflexión	6	6	NU
Mediador	5	4	PU
Total	27	18	

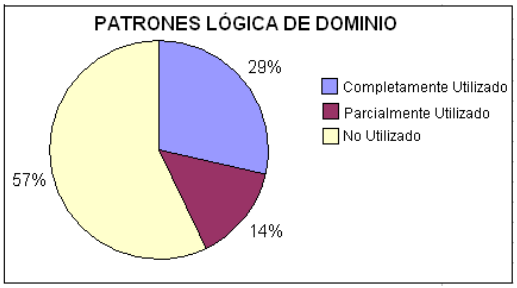


Figura 32 Porcentajes de Utilización de los Patrones de Lógica de Dominio

La evaluación realizada al código fuente demostró que el módulo no utiliza los patrones lógicos en un 57% y un 29% corresponde a patrones completamente utilizados, este porcentaje se debe a la utilización del patrón capas como se puede ver en la Tabla 75.
Por lo anterior, **se recomienda** implementar patrones como el Micro-Kernel y reflexión para así garantizar **las propiedades de usabilidad**: Gestionando el error, Consistencia: Funcional, Evolutiva, Mapeo Natural, Accesibilidad y Adaptabilidad; que tienen un impacto alto y medio positivamente.

Para ver en detalle los hallazgos corregidos dirigirse al Anexo 22.

Tabla 76 Resultados de la Evaluación de los Patrones de Datos

Patrones de Acceso a Datos	No de Preguntas	Hallazgos encontrados	Grado de cumplimiento
Transacción de Script	1	1	NU
Modelo del Dominio	1	1	NU
Modulo de la Tabla	1	1	NU
Total	30	21	

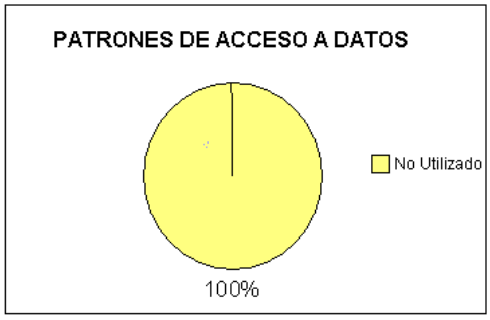


Figura 33 Porcentajes de Utilización de los Patrones de Datos

De acuerdo a la figura se puede afirmar que en el módulo los patrones de datos no se aplican en un 100 %. Debido a este resultado se sugirió estudiar los patrones Transacción de script, modelo de dominio y modulo de tabla, los cuales se encuentran explicados en el capítulo 3 de este trabajo de grado.

Para ver en detalle los hallazgos encontrados así como las recomendaciones encontradas, puede dirigirse al Anexo 23.

5.3.2 Resultados Finales

Después de realizar la evaluación al módulo "No conformidades" se dio a conocer al desarrollador los hallazgos encontrados y las recomendaciones. Al implementar los cambios sugeridos se evaluó nuevamente la aplicación como se hizo anteriormente, obteniendo los siguientes resultados.

5.3.2.1 Resultados de la evaluación de los Patrones de Presentación

Tabla 77 Resultados Evaluación de los Patrones de Presentación

Patrones Presentacion	No de Preguntas	Hallazgos encontrados	Grado de cumplimiento
Alto Nivel			
MVC	13	1	AU
PAC	4	4	NU
Nivel Intermedio			
Controlador de Pagina	6	0	CU
Controlador de Frente	3	3	NU
Controlador de la Aplicación	1	0	CU
Plantilla de Vista	2	0	CU
Transformador de Vista	2	2	NU
Dos Fases por Vista	1	1	NU
Enlace como Relación entre Vistas	1	0	CU
Observador de Navegación	1	0	CU
Nodo como una Unidad única	1	0	CU
Método Crear Nodo y Crear Enlace	1	0	CU
Contexto Navegacional	1	0	CU
Referencia Activa	1	0	CU
Bajo Nivel o de Interfaz de usuario			
Información Solicitada	3	1	AU
Desacoplar la Interaccion de la Informacion	2	0	CU
Agrupación Conductual	2	0	CU
Anticipación Conducta	1	1	NU
Proceso de Retroalimentación	2	1	PU
Narrativa	2	0	CU
Despliegue de Información de Alta Densidad	4	1	AU
Formulario	7	2	AU
Panel de Control	3	0	CU
Editor WYSIWYG	2	0	CU
Comando Compuesto	2	2	NU
Espacios Navegacionales	1	0	CU
Visión Global en Detalle	3	0	CU
Grupos Pequeños de Cosas Relacionadas	1	0	CU
Conjunto de Jerarquías	2	0	CU
Conjunto Tabulador	2	0	CU
Mapa o Grafica	1	1	NU
Detalle Opcional en Demanda	1	0	CU

Desactivar Cosas No Pertinente	2	1	PU
Mostrar el Indicador de Acceso	1	0	CU
Descripción corta	1	0	CU
Apariencia de Fondo	1	1	NU
Superficie Activa Central	2	0	CU
Mosaico de superficies Activas	3	1	AU
Montón de Superficies Activas	1	1	NU
Mapa de Espacios Navegacionales	1	0	CU
Puntos Claros de Entrada	1	0	CU
Secciones de Color Codificado	1	0	CU
Regresar a un Lugar Seguro o Punto de Control	1	0	CU
Acciones Convenientes de Ambiente	1	0	CU
Acciones de Localización del Objeto	1	0	CU
Acciones para Objetos Múltiples	1	0	CU
Colección Editable	1	0	CU
Absolver Entrada de Texto	1	0	CU
Entrada de Texto Estructurado	1	1	NU
Barra de Herramientas	1	0	CU
Preferencias de Usuario	1	1	NU
Anotaciones de Usuario	1	0	CU
Framewok Repetido	1	0	CU
Buenas por Defecto	1	1	NU
Estado Recordado	1	0	CU
Historia de la Interación	1	1	UN
Mensaje Importante	2	0	CU
Chequeo Real	1	0	CU
Demostración	1	0	CU
Total	**112**	**28**	

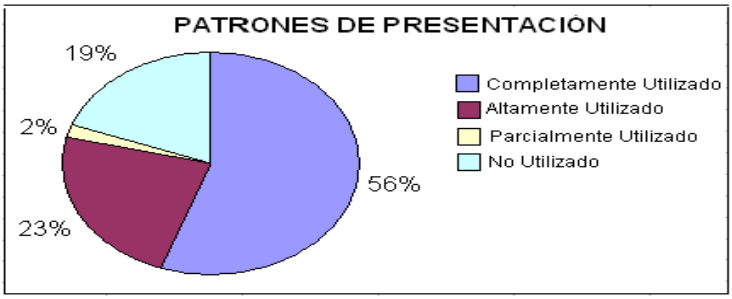


Figura 34 Porcentajes de la Evaluación de los patrones de Presentación

En la figura 34 se puede apreciar que aumentó el porcentaje de los patrones completamente utilizados de un 29% a un 56%, esto quiere decir, que la mayoría de los hallazgos encontrados en el módulo “No conformidades” en la evaluación anterior se corrigieron, reduciendo el porcentaje de no utilizado de un 35% a un 19%, mejorando la usabilidad en la interfaz de usuario, proporcionando una interacción aceptable de la aplicación Web.

El módulo mejoró al implementar los **patrones:** narrativa, formulario, panel de control, editor WYSIWYG, descripción corta, superficie activa central, mapa de espacios navegacionales, puntos claros de entrada, regresar a un lugar seguro, demostración; los cuales tienen un impacto alto positivamente en las **propiedades de usabilidad**: Lenguaje de los usuarios, Realimentación, Ayuda u orientación, Facilidad de navegación. Estas propiedades se reflejan en la presentación de la información, en el nombre de los enlaces e iconos en un lenguaje familiar al usuario, en la organización, ayuda y orientación para ingresar la información en los formularios.
Los patrones que no se alcanzaron a cumplir totalmente se dejarán para futuras versiones de la aplicación debido a que el equipo de desarrollo priorizo los cambios de acuerdo a su conveniencia, tiempo y esfuerzo.

Para ver en detalle los hallazgos que fueron corregidos así como los que faltan por corregir, puede dirigirse al Anexo 24.

5.3.2.2 Resultados de la Evaluación de los Patrones de Lógica de Dominio y Datos

Tabla 78 Resultados de la Evaluación de los Patrones de Lógica de Dominio

Patrones Arquitecturales	No de Preguntas	Hallazgos encontrados	Grado de cumplimiento
Alto Nivel			
Capas	6	0	CU
Nivel Intermedio			
Pizarra	4	2	PU
Micro-Kernel	6	6	NU
Reflexión	6	6	NU
Mediador	5	4	PU
Total	27	18	

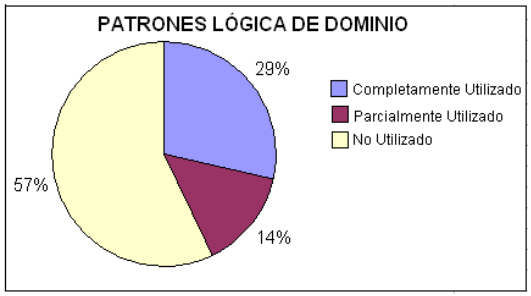


Figura 35 Porcentajes de Evaluación de los Patrones de Lógica de Dominio

Tabla 79 Resultados de la Evaluación de los Patrones de Datos

Patrones de Datos			
Transacción de Script	1	1	NU
Modelo del Dominio	1	1	NU
Modulo de la Tabla	1	1	NU
Total	30	21	

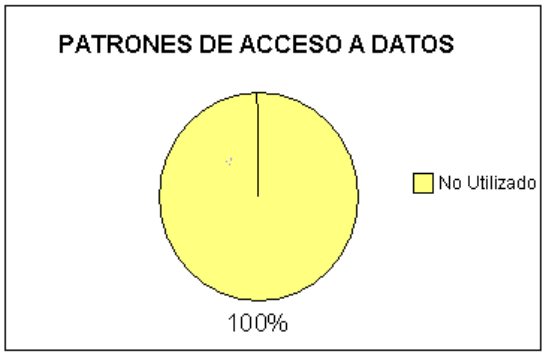


Figura 36 Porcentajes de Evaluación de los Patrones de Datos

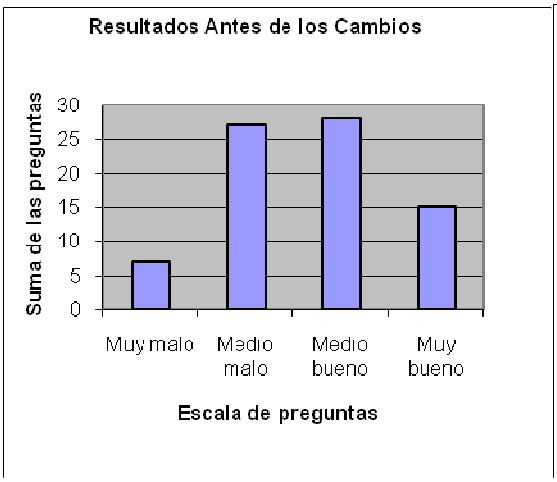
En cuanto a los resultados obtenidos en la evaluación final de los patrones lógicos y de datos, los cuales pueden verse en las figuras 35 y 36, donde no se aprecia ningún cambio con respecto a las figuras iniciales. Una de las causas es la priorización de actividades en la empresa y la limitación de tiempo. También hay que tener en cuenta que la implementación de estos patrones es una mejora continua que se hace notoria a largo plazo por su misma naturaleza. Debido a lo anterior los hallazgos que quedaron por corregir en este módulo se implementarán en futuras versiones de la aplicación, o en otros módulos,

Para ver en detalle los hallazgos pendientes por corregir puede dirigirse al Anexo 25 y 26.

5.3.2.3 Resultados de la Percepción de Personas Externas

Se realizó una evaluación con estudiantes que vieron la electiva HCI (interacción hombre máquina), para ratificar el análisis realizado en la sección 5.3.2; teniendo como resultado lo siguiente:

Como puede apreciarse en las figuras 37 y 38 respectivamente, al analizar los resultados de las preguntas de la encuesta realizada a los estudiantes antes de implementar las recomendaciones establecidas, se puede notar que las características de usabilidad del módulo "No conformidades" tenían un porcentaje de 9% en la escala de muy malo y 35% en medio malo, 36% en la escala de medio bueno y 20% en muy bueno; como se puede notar se tiene un porcentaje muy alto en medio malo, lo cual indica que la aplicación tiene deficiencias en las propiedades de usabilidad, como se estableció en la sección 5.3.1.1.1. Luego de aplicar los patrones de presentación el porcentaje cambio de un 9% a 2.6% en la escala de muy malo, y de un 35% a un 7.8% en la escala de medio malo, aquí es donde se ve que el cambio fue muy notorio, aumentando el porcentaje de medio bueno que paso de un 35% a un 53% y muy bueno paso de un 20% a un 36.6%.; por lo tanto la usabilidad del módulo mejoró con la utilización de los patrones; esto quiere decir, que la percepción de los estudiantes ratifica los resultados analizados en la sección 5.3.2.



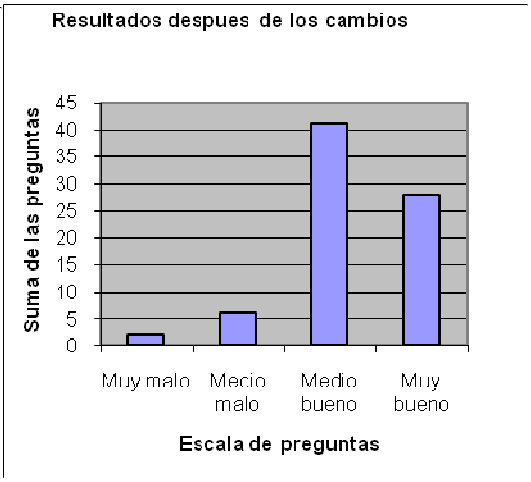


Figura 37Antes de los cambios **Figura 38 Después de los cambios**

En la figura 39 se muestran los resultados de la encuesta de acuerdo a la escala establecida comparando el antes y después de aplicar los patrones de presentación según la percepción de los estudiantes.

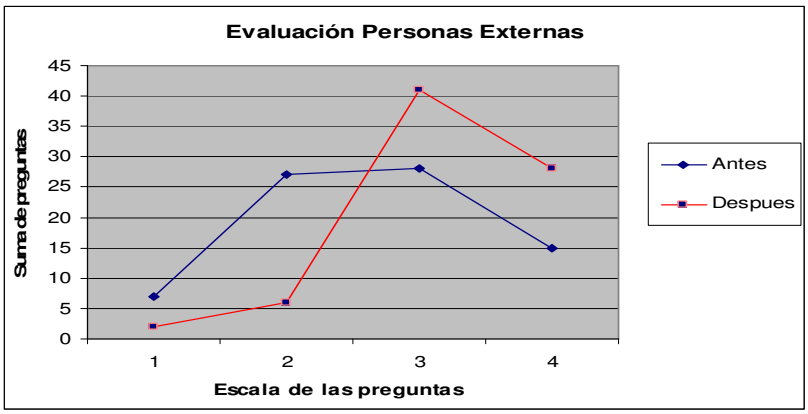


Figura 39 Comparación del Antes y Después de implementar los patrones

Las figuras que se describen en el Anexo 28 muestran las evidencias de los hallazgos encontrados y corregidos en la interfaz teniendo en cuenta las recomendaciones hechas según los patrones de presentación.

5.4 Aplicación de algunos de los Procesos definidos en el capítulo 4 de ingeniería para el desarrollo de aplicaciones en la Web, teniendo en cuenta aspectos de Usabilidad

En la empresa INPUT Tecnologies Ltda. Se aplicó el proceso de desarrollo que se describe en el capítulo IV, aunque no se ha implementado completamente debido a que el módulo "Actas" está en proceso de desarrollo, las etapas que hasta el momento se han abarcado son las de requerimientos y análisis, donde el líder de desarrollo Ricardo Ledezma y el desarrollador Javier Castillo dan su opinión acerca de este proceso: "*El*

desarrollo del módulo de actas de calidad que se lleva a cabo bajo la metodología definida en el Marco de Referencia descrito en el capítulo IV; ha sido una herramienta fundamental en la planeación del proyecto, debido a la claridad y definición puntual de los objetivos y productos a entregar. Esta metodología define de manera muy concreta puntos que son esencial para el desarrollo de aplicaciones Web de forma ágil, tales como la definición de tareas, técnica a utilizar y productos a entregar logrando una mejora considerable respecto a otras metodologías usadas en el desarrollo de aplicaciones para la Web. Además, permite enfocar la aplicación a las necesidades de los usuarios finales del sistema gracias al manejo de requisitos de usabilidad

Con esta metodología, se obtiene una visión clara sobre la funcionalidad de la aplicación, lo que permite planear de forma efectiva el desarrollo, sin perder tiempo por causa de un equivocado planeamiento y definición del proyecto.

De acuerdo a lo expuesto anteriormente optamos por seguir esta metodología para el desarrollo de futuras aplicaciones para la Web".

También dieron su opinión sobre lo que no les parecía: *"La plantilla sugerida para la captura de requisitos de usabilidad, solícita información sobre aspectos que en nuestro concepto podría limitar los requisitos de usabilidad a ciertos objetivos o requisitos de la aplicación y consideramos que los requisitos de usabilidad deben ser tenidos en cuenta en toda la aplicación; específicamente los campos objetivos asociados, requisitos asociados e intervalo temporal".*

Con respecto a la plantilla para la captura de requisitos se confirmará lo dicho a medida que se aplique el proceso a diferentes dominios y se obtenga varias opiniones según la experiencia de su implementación. Esto tiene una verificación a largo plazo.

5.5 Conclusiones

Al terminar este capítulo se puede afirmar que el caso de estudio realizado en dos MIPYMES de Parquesoft Popayán, Ikernell e Imput, sirvió para contribuir en la validación del marco de referencia expuesto en este trabajo de grado. Ayudando a corroborar la influencia que tienen los patrones expuestos en el capítulo 3 en la usabilidad de las aplicaciones Web.

La hipótesis sobre la cual se planteó este trabajo de grado la cual es que la usabilidad es una característica que debe tenerse en cuenta desde etapas tempranas del proceso de ingeniería, se ratifica con los resultados obtenidos en los casos de estudio.

Finalmente cabe decir que los resultados obtenidos en este caso de estudio fueron ratificados por la apreciación dada por los usuarios externos que evaluaron la aplicación, lo cual valida que los diferentes métodos y los patrones usados son notablemente buenos para evaluar la usabilidad de una aplicación Web y recomendar las posibles mejoras.

Capítulo 6 CONCLUSIONES Y TRABAJOS FUTUROS

Finalizada la exposición del marco propuesto y de los resultados obtenidos en la aplicación del mismo, en este capítulo se valoran las conclusiones obtenidas de la realización del presente trabajo de grado, así como las oportunidades que deja abiertas para continuar la investigación en nuevas direcciones.

La usabilidad se percibe cada vez más como uno de los atributos de calidad más importantes en las aplicaciones Web, sin embargo, la mayoría de los trabajos que se encuentran entorno al tema de usabilidad y aplicaciones Web se enfocan a la evaluación de este atributo al final de los procesos de desarrollo, por lo que no estudian el impacto que tienen las decisiones arquitecturales en la usabilidad de una aplicación Web, en otros trabajos se encuentran estudios de la relación entre la usabilidad y las arquitecturas de software sin apuntar a un tipo de aplicaciones especificas.

En cuanto a las metodologías que se han propuesto para guiar los procesos de desarrollo de aplicaciones Web, no se ha encontrado evidencia en la literatura de un trabajo que indique que tipo de decisiones arquitecturales se deben tener en cuenta para impactar de manera positiva o negativa la usabilidad. Existen aportes significativos en cuanto al planteamiento de patrones para aplicaciones hipermedia, así como patrones HCI, o patrones navegacionales; sin embargo ninguno de ellos relaciona la arquitectura de software y su impacto en la usabilidad de aplicaciones Web.

La aplicación de las investigaciones que existen alrededor de temas como: Usabilidad y Arquitecturas, Arquitecturas y Aplicaciones Web, Usabilidad en Aplicaciones Web y Metodologías para Proceso de desarrollo de aplicaciones Web, se han realizado en empresas de desarrollo de software que no cumplen con las características que tienen las MiPymes en Colombia, lo que hace que la aplicación de dichas investigaciones no se adapte positivamente a las MiPymes que construyen aplicaciones Web en el departamento del Cauca.

Por lo anterior se considera que los aportes principales que realiza el presente trabajo de investigación se han realizado en el proceso de construcción del marco propuesto, y son las siguientes:

- Se tomaron una serie de patrones existentes y ampliamente conocidos por las comunidades de ingeniería de software para el desarrollo de aplicaciones en general, sobre los cuales se realizo un estudio minucioso para determinar cuales y de que manera afectaban la usabilidad en las aplicaciones Web construidas por MiPymes.
- Con base en las heurísticas existentes para evaluar la usabilidad en aplicaciones interactivas se definieron las propiedades de usabilidad para medir el impacto de los patrones sobre la usabilidad.
- Se planteó un proceso de desarrollo para aplicaciones Web centrado en la arquitectura y teniendo en cuanta aspectos de usabilidad, para esto se estudiaron las diferentes metodologías que existen para el desarrollo de aplicaciones Hipermedia o Web, así como los trabajos que se han realizado entorno a la relación entre ingeniería de software y la Interacción Persona Ordenador. En dicho proceso se plantearon que actividades y tareas se deben realizar y de que

manera, para que una aplicación Web cumpla con el atributo de usabilidad desde las etapas iníciales de un proceso de desarrollo.
- Se relacionaron los patrones planteados con las etapas del proceso de desarrollo indicando que tipo de patrones debían utilizarse en un momento determinado.
- Se generaron una serie de plantillas para poder documentar las etapas del proceso de desarrollo descrito en el capítulo 4, las cuales se relacionan en lo Anexos que van del 6 al 14.
- Se aplico el marco en dos productos: Servicio Web Agropecuario desarrollado por la empresa IKERNELL, y Compromiso desarrollado por la empresa INPUT, estas empresas pertenecientes a la incubadora de empresas de software Parquesoft, ubicada en la ciudad de Popayán en el departamento del Cauca. Dichas empresas cumplen con las características de una MiPyme, expuestas en le capítulo 5, lo cual es la ejercitación de una propuesta para hacer su verificación y validación en un ambiente real.

En la aplicación de la propuesta se ha podido comprobar su viabilidad práctica en un entorno industrial.

Trabajos futuros

Para dar continuidad a la investigación reflejada en el presente trabajo las siguientes líneas parecen prometedoras:

- Ampliar el marco de referencia para empresas que tengan procesos de desarrollo definidos y organizados, o que cuenten con un sistema de gestión de calidad definido e implementado.
- Desarrollar una Herramienta que permita modelar el proceso de desarrollo planteado con base en los patrones propuestos, esto con el ánimo de automatizar los procesos propuestos para lograr una disminución en el modelo de análisis y diseño de una aplicación Web.
- Ampliar el marco de referencia a otros dominios de aplicación, como pueden ser las aplicaciones Web Móviles, entre otras.
- Ampliar el marco de referencia para incluir otros aspectos de calidad relevantes en las aplicaciones Web, por ejemplo: Fiabilidad, Eficiencia, Fiabilidad, entre otras.
- Ampliar y refinar el proceso de desarrollo de software incluyendo etapas posteriores, puede ser llevarlo hasta la etapa de implantación en el ambiente de usuario.
- Ampliar el Marco para el estudio para arquitecturas orientadas a servicios (SOA), las cuales tienen características diferentes que las que se estudiaron.
- Ampliar el marco de referencia con otros patrones que pueden no haberse tenido en cuenta.

Capítulo 7 BIBLIOGRAFIA

[1] Mayer & Bunge Informática LTDA. "Panorama de la industria latinoamericana de software". Brasil, 2004. p. 97, *Congreso Colombiano de Computación - CCC* 2007.

[2] F. J. Pino, F. García, F. Ruiz, M. Piattini, "Adaptación de las normas ISO/IEC 12207:2002 e ISO/IEC 15504:2003 para la evaluación de la madurez de procesos software en países en desarrollo", Colombia. Disponible en: http://ieeexplore.ieee.org/iel5/9907/34417/01642455.pdf.

[3] J. Batista, A. Figueiredo, "SPI in very small team: a case with CMM. Software Process Improvement and Practice 5 (4)", *DOCIS Documents in Computing and Information Science*, pp. 243-250, 2000

[4] R. Pressman, *Ingeniería de Software: Un Enfoque Práctico*. Quinta Edición. McGraw Hill, 2002

[5] J. Bosch, *Design and Use of Software Architectures: Adopting and Evolving a Product Line Approach*, Pearson Education, Addison-Wesley, 2000.

[6] X. Ferré, N. Juristo, H. Windl, L. Constantine, "Usability Basics for Software Developers". *IEEE Software*, vol 18 (11), pp. 22-30

[7] J. Zapopan, Informática Milenium, S.A. de C.V. Disponible: http://www.informaticamilenium.com.mx/Páginas/espanol/sitioWeb.htm

[8] J. ZHOU, "A history of Web portals and their development in libraries" *Department, California State University Library-Sacramento, ETATS-UNIS*

[9] J. Conallen, *Building Web Applications with UML*, Addison-Wesley, 2000.

[10] I. Jacobson, G. Booch, J. Rumbaugh, *El Proceso Unificado de Desarrollo de Software*, Addison Wesley 2000.

[11] O. M. F. De Troyer, C. J. Leune, "WSDM: A User Centered Design Method for Web Sites". Tilburg University, Infolab. Belgium. 1997. *Published in Computer Networks and ISDN systems, Proceedings of the 7th International World Wide Web Conference, Elsevier,* pp. 85 - 94, 1998

[12] L. Olsina, "Building a Web-based information system applying the hipermedia flexible process modelling strategy", 1st International Workshop on Hipermedia Development, Hypertext 1998.

[13] G. Rossi, "An Object Oriented Method for Designing Hipermedia Applications". Ph.D. Thesis, Departamento de Informática, PUC-Rio, Brazil, 1996.

[14] N. Koch, "Software Engineering for Adaptative Hipermedia Applications". Ph. Thesis, FAST Reihe Softwaretechnik Vol(12), Uni-Druck Publishing Company, Munich. Germany 2001.

[15] P. Kruchten, *The Rational Unified Process.* Addison Wesley 1998.

[16] M. J. Escalona, J. Torres, M. C. Mejías, "Modelos y técnicas para la especificación y el análisis de la navegación en sistemas software", Ph.D. Tesis, Departamento de Lenguajes y Sistemas Informáticos, Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla, Octubre de 2004.

[17] M. J. Escalona, N. Koch, "Ingeniería de requisitos en aplicaciones para la Web: Un estudio comparativo", Universidad de Sevilla, Lenguajes y Sistemas Informáticos, España, *Universidad de Munich y F.A.S.T. GmbH, Munich, Alemania,* 2000

BIBLIOGRAFIA

[18] R. Mollineda, “Arquitectura del software: Arte y oficio”; *Revista del Instituto Tecnológico de Informática,* disponible en: http://Web.iti.upv.es/actualidadtic/2005/02/2005-02-arquitectura.pdf, último acceso 30-04-2007

[19] P. Clements, “A survey of architecture description languages”. Proceedings of the International Workshop on Software Specification and Design, Alemania, 1996.

[20] D. Garlan, “Software architecture: A roadmap”. *En Anthony Finkelstein (ed.), The future of software engineering, ACM Press*, 2000.

[21] F. Buschman, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, *Pattern-Oriented Software Architecture – A System of Patterns*; John Wiley & Sons Ltd. Chichester, England, 1996.

[22] L. Bass, P. Clements, R. Kazman. *Software Architecture in Practice SEI Series in Software Engineering*. Addison-Wesley, 1998.

[23] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, J. Carriere. “The architecture tradeoff analysis method”, *Proceedings of the International Conference on Engineering of Complex Computer Systems*, Monterey, CA, 1998

[24] P. Kruchten, “Architectural blueprints--The 4+1 view model of software architecture”, *IEEE Software, Institute of Electrical and Electronics Engineers*. pp. 42-50, 1995

[25] E. Folmer, “Arquitectura de software análisis de usabilidad” Ph.D. Memoria de la tesis doctoral. Para grado de Doctor en Matemáticas y ciencias naturales en la Universidad pública Groningen, 2005.

[26] R. Taylor, N. Medvidovic, K. Anderson, J. Whitehead, J. Robbins, K. Nies, P. Oreizy, y D. Dubrow, “A component- and message-based architectural style for GUI software”. *Proceedings of the 17th International Conference on Software Engineering*, *Seattle, ACM* Press, 23 al 30 de Abril de 1995.

[27] M. Shaw, R. Deline, D. Klein, T. L. Ross, D. M. Young, G. Zelesnik, "Abstractions for software architecture and tools to support them", *IEEE Transactions on Software Engineering* 4, 1995.

[28] G. Booch, *Object-Oriented Design With Applications*, Benjamin Cummings, 1991.

[29] G. J. S. Castejón, “Arquitectura y diseño de sistemas Web modernos”. Secretario del CIIRM. *Revista de Ingeniería Informática del CIIRM*. 2004.

[30] P. Valderas, M. Ruiz, J. Fons, M. Albert, V. Pelechado, “Aplicación de un Método de Modelado de Aplicaciones Web para el desarrollo de un Portal Web Universitario”, *Departamento de Sistemas Informáticos y Computación* Universidad Politécnica de Valencia Camino de Vera s/n 46022 Valencia, España.

[31] Core J2EE Patterns: Patterns index page, disponible en: http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html

[32] T. R. Fielding, ”Architectural styles and the design of network-based software architectures”, Tesis doctoral, University of California, Irvine, 2000.

[33] Carnegie Mellon, Software Engeenering Institute, Client/Server software architectures-- An overview, Carnegie Mellon University. Consulta Septiembre 2003 disponible en: http://www.sei.cmu.edu/str/descriptions/clientserver_body.html

[34] R. Orfali, D. Harkey y J. Edwards, *Essential Client/lServer Survival Guide,* 3.* ed., Wiley, 1999.

[35] C. I. Silva, “Modelamiento de sistema de monitoreo de rendimiento de servicios sobre arquitectura SOA”, Memoria para optar el titulo de ingeniero. Universidad de Chile 2007

[36] Gartner. ”Service oriented architectures”, Part 1 y SSA Research Note SPA-401-068, 401-069, , ”Service oriented architectures”, Part 2, 1996.

[37] H. Hao, What is Service-Oriented Architecture, disponible en: http://Webservices.xml.com, http://www.xml.com/pub/a/ws/2003/09/30/soa.html septiembre, 2003. último acceso 08-01-2008.

[38] N. Bieberstein, et al. Service-oriented architecture compass, Pearson 2006, ISBN: 0-13-187002-5. Disponible en: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconchoosingcommunicationoptionsinnet.asp; .

[39] M. Shaw, D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall, 1996.

[40] J. L. Arciniegas, “Memoria de la tesis doctoral Contribution to quality driven evolutionary software development for service oriented architecture”. Universidad Politécnica de Madrid. 2006

[41] F. Losavio, L. Chirinos, N. Lévy, & Ramdane-Cherif, “A. quality characteristics for software architecture”. A ser publicado en el *JOT* 2003.

[42] F. P. Jr. Brooks, *The Mythical Man-Month: Essays on Software Engineering, Twentieth Anniversary* Edition, Addison-Wesley, 1995.

[43] M. Svahnberg, C. Wohlin, L. Lundberg, and M. Mattson, “A method for understanding quality attributes in software architecture structures”, *Proceedings of the 14th international conference on Software engineering and knowledge engineering, SESSION: Workshop on software engineering decision support*, Ischia, Italy, 2000.

[44] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns Elements of Reusable Object-Orientated Software*, Addison-Wesley, 1995

[45] M. Klein, y R. Kazman, “Attribute-based architectural styles”. *Technical Report, CMU/SEI-99-TR-022, ESC-TR-99-022,* Carnegie Mellon University, Octubre de 1999.

[46] G. Rossi, A. Garrido, and S. Carvalho, *Design Pattern for Object-Oriented Hipermedia Applications, Pattern Languages of Program Design*, Vol. 2, chapter 11, Vlissides, Coplien y Kerth editors, Addison-Wesley, 1996, pp. 177- 191

[47] G. Rossi, D. Schwabe and A. Garrido, “Towards a pattern language for hipermedia applications”, The 3rd. *Pattern Languages of Programming Conference* (Washington University technical report #WUCS-97-07), February, 1997.

[48] G. Rossi, D. Schwabe, A. Garrido, “Design reuse in hipermedia application development”, *Proceedings of The Eight ACM Conference on Hypertext*, Hypertext’97. Southampton, United Kingdom, April 1997.

[49] R. Allen, y D. Garlan, “The Wright Architectural Description Language”, *Technical Report*, Carnegie Mellon University. Verano de 1996.

[50] F. Montero, M. D. Lozano, “Integración de Calidad y Experiencia en el Desarrollo de Interfaces de Usuario”, Tesis doctoral elaborada para optar al título de Doctor en Informática, Departamento de sistemas informáticos, universidad de castilla-la mancha 2005

BIBLIOGRAFIA

[51] R. S. Pressman, *Ingeniería del Software: Un enfoque práctico*; Cuarta edición. McGraw-Hill, México DF, 1998.

[52] I. Sommerville, *Software engineering*; Quinta edición. Addison-Wesley, 1998

[53] S. Lawrence, *Software Engineering: Theory and Practice*. Prentice Hall. 1998

[54] P. Clements, R. Kazman, y M. Klein, "Evaluating software architectures: Methods and case studies", *The SEI Series in Software Engineering*, 2001.

[55] ATAM

[56] Lung, C. Bot, S, Kalaichelvan, K and Kazman, R: "An approach to software architecture analysis for evolution and reusability", *Proc. of CASCON (Centre for Advanced Studies Conf.),* Toronto, Canada, pp. 144-154. Nov. 1997

[57] *ISO 9241-11. International Standard. Ergonomic requirements for office work with visual display terminals (VDTs)-Part 11: Guidance on usability*. 1998.

[58] *ISO/IEC 9126-1, International Standard, Software engineering-Product quality-Part 1*: Quality model. 2001.

[59] J. Nielsen, *Usability Engineering. Academic Press Professional*, Boston, MA. 1993.

[60] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey, *Human-Computer Interaction*, Addison Wesley. 1994.

[61] J. Redish, "Are we really entering a post-usability era". *ACM SIGDOC Asterisk Journal of Computer Documentation*, vol. 19 (1), pp. 18-24, 1995.

[62] W. Quesenbery, "What does usability mean: Looking beyond 'Ease of use'", *Proceedings of the 48th Annual Conference, Society for Technical Communication,* 2001. Disponible en http://www.wqusability.com/articles/more-than-ease-of-use.html.

[63] L. L. Constantine, and L. A. D. Lockwood, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*, Addison-Wesley, New York NY. 1999.

[64] D. Hix, and H. R Hartson, *Developing User Interfaces: Ensuring Usability Through Product and Process*, John Wiley and Sons. 1993.

[65] B. Shackel, "Usability - context, framework, design and evaluation, Human factors for informatics usability", *Cambridge University* Press. 1991.

[66] B. Shneiderman, *Designing the User Interfaz: Strategies for Effective Human-Computer Interaction*, Addison-Wesley. 1998.

[67] D. Wixon, and C. Wilson, *The usability engineering framework for product design and evaluation* I, in: M. G. Helander (eds.), In Handbook of Human-Computer Interaction, Elsevier North-Holland, 1997, pp. 653-688.

[68] L. Trenner, J. Bawa, *The Politics of Usability: A Practical Guide to Designing Usable Systems in Industry*. Springer Verlag. 1998.

[69] D. J. Mayhew, M. A. Mantei, *Basic Framework for Cost-justifying Usability Engineering,* Academic press, Inc. Orlando, FL, USA 1994.

[70] A. M Lund, "Another approach to justifying the cost of usability", *ACM Interactions*, junio 1997.

[71] S. Dray, *The Importance of Designing Usable Systems.* Interactions, V. 2, issue 1, ACM Press, 1995, pp. 17-20..

[72] J. Battey, “IBM's redesign results in a kinder, simpler Web site”, *Revista InfoWorld* 1999.

[73] S. Atxondo, J. Casanovas, A. Guersenzvaig, “Proyecto Intranet” *La Caixa* 2002. *Sesión de apertura del Congreso Interacción* 2003.

[74] T. Granollers, I. Saltiveri. MPUI+a. “Una metodología que integra la ingeniería del software, la interacción persona ordenador y la accesibilidad en el Contexto de equipos de desarrollo multidisciplinares” *Memoria de la tesis doctoral para grado de doctor en Informática, especialidad en Interacción Persona-Ordenador*, por la Universitat de Lleida, 2004.

[75] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, McGraw-Hill, NY. 1992.

[76] T. K. Landauer, *The Trouble with Computers: Usefulness, Usability and Productivity,* MIT Press. 1995.

[77] D. L. Cuomo, and C. D. Bowen, "Understanding usability issues addressed by three user-system interfaz evaluation techniques", *Interacting with Computers* 6(1), pp. 86-108. 1994.

[78] J. Nielsen, *Heuristic Evaluation, in: J. Nielsen and Mack, R. L. (eds.), Usability Inspection Methods*, John Wiley and Sons, New York, N.Y. 1994

[79] M. Fowler, D. Rice, E. Foemmel, E. Hieatt, R. Mee, R. Stafford, *Patterns of Enterprise Application Architecture*, Addison Wesley, 2002.

[80] B. Meyer, “Object-Oriented Software Construction, Gang of four. T. Winograd: Architectures for context: human-computer-interaction”, 16(2), *HI Jcournal,* December 2001.

[81] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software* Addison Wesley, 2003

[82] Layered application. Version 1.0.1. Microsoft Patterns & Practices, http://msdn.microsoft.com/architecture/patterns/default.aspx?pull=/library/en-us/dnpatterns/html/ArclayeredApplication.asp , 2004.

[83] G. Kiczales, “Towards a new model of abstraction in software engineering”, *Proceedings of IMSA’92*, pp. 1-11

[84] M. Lang, “Hipermedia system development, Do we really need new Methods”, *Site-Where Parallels Intersect. Informing Science*. pp. 883-891. June 2002.

[85] L. Baresi, F. Garzotto, P. Paolini, “Extending UML for modelling Web applications”, *Annual Hawaii International Conference on System Sciences*. pp. 1285 - 1294. Maui, USA. Enero 2001.

[86] H. Shubin, M. M. Meehan, “Navigation in Web applications”. *ACM Interactions Magazine. IV*. Noviembre 1997.

[87] S. Buckingham & C. McKnight, “World Wide Web usability: introduction to this special issue” *International Journal of Human-Computer Studies* . 47(1), 1-4. 1997.

[88] C. Barry, M. Lang, “A survey of multimedia and Web development techniques and methodology usage”. *IEEE Multimedia.* pp. 52-56. Abril-Junio 2001.

[89] Software Process Engineering Meta-Model, version 2.0 http://www.omg.org/technology/documents/formal/spem.htm

[90] M. J. Escalona, M. Mejías, J. Torres, A. M. Reina. “The NDT Development process”, *Proceedings of IV International Conferences on Web Engineering. ICWE* 2003. LNCS 2722. pp. 463-467. Springer Verlag 2003.

BIBLIOGRAFIA

[91] G. Kotonya, I. Sommerville, *Requirements Engineering. Processes and Techniques,* JohnWiley. 1997

[92] W. M. Newman, M. G. Lamming, *Interactive System Design*. Addison-Wesley. 1995.

[93] G. C. Veer, van der; B. F. Lenting, B. A. J. Bergevoet, *GTA:* "Groupware task analysis - modeling complexity acta psychologica". *CTIT, Twente University of Technology, Enschede, NL, # Dept. Computer Science,* 91, pp. 297-322. 1996.

[94] L. A. Olsina, "Modelo de proceso flexible para el soporte sistemático al desarrollo de aplicaciones de hipermedia" Tesis presentada al Departamento de Informática de la UNLP como parte de los requisitos para la obtención del título de Master en Ingeniería de Software La Plata, Agosto de 1997; revisada en Febrero de 1998.

[95] A. Dardenne, S. Fickas, A. Van Lamsweerde, "Goal-directed concept acquisition in requirements elicitation". *6th International workshop on Software specification and design*. pp. 25-26. Como, Italy 1991.

[96] A. Dardenne, S. Fickas, A. Van Lamsweerde, "Goal-directed requirements acquisition", *Science of computer Programming*, 20. pp. 3-50. 1993.

[97] A. Durán, B. Bernandez, "Metodología para la elicitación de Requisitos de Sistemas software". *Departamento de Lenguajes y Sistemas Informáticos*. Universidad de Sevilla. Sevilla, 2002.

[98] X. Romano, D. Correa, MoDSGX: Modelo de proceso para desarrollo de software con Genexus.

[99] C. M. J. Escalona, J. Torres, M. M. C. Mejìas, L. L. F. Jurado, "Navigational Development Techniques", *Departamento de Lenguajes y Sistemas Informáticos,* Universidad de Sevilla.

[100] N. R. Brisaboa, R. Penabad, S. Places, J. Rodríguez, "A documental database query language", *String Proccessing and Information Retrieval -SPIRE* 2001.

[101] I. Aedo, P. Díaz, S. Montero, M. Castro, "Ingeniería de la Web y patrones de diseño", *Laboratorio DEI.* Departamento de Informática, UC3M y Departamento de Ingeniería Eléctrica, Electrónica y de Control, Universidad Nacional de Educación a Distancia.

[102] P. J. Valderas, Aranda, "Especificación de requisitos en el desarrollo de aplicaciones Web" *Departamento de Sistemas Informáticos y Computación* Universidad Politécnica de Valencia.

[103] R. Soto De Giorgis, W. P. Muñoz, S. Roncagliolo De la Horra, "Propuesta de un modelo navegacional para el desarrollo de aplicaciones basadas en OOHDM" *Escuela de Ingeniería Informática*, Universidad Católica de Valparaíso, Chile. E-mail: rsoto@inf.ucv.cl, wenceslao.palma@ucv.cl, silvana@ucv.cl.

[104] A. Díez, "IRqA y el desarrollo de proyectos: Experiencias Prácticas", *I Jornadas de Ingeniería de Requisitos Aplicadas*. Seville, Spain. JIRA 2001.

[105] D. A. Silva, B. Mercerat, "Construyendo aplicaciones Web con una metodología de diseño orientada a objetos", *LIFIA, Laboratorio de Investigación y Formación en Informática Avanzada,* Facultad de Informática, Universidad Nacional de La Plata, La Plata, Provincia de Buenos Aires, República Argentina.

[106] J. Fons, O. Pastor, P. Valderas, M. Ruiz, "Un Método de Producción de Software en Ambientes Web", *Departamento de Sistemas Informáticos y Computación* Universidad Politécnica de Valencia Camino de Vera s/n 46022 Valencia, Spain.

[107] P. Valderas, J. Aranda, "Especificación de Requisitos en el Desarrollo de Aplicaciones Web", *Departamento de Sistemas Informáticos y Computación* Universidad Politécnica de Valencia.

[108] P. Kruchten, "Planos arquitectónicos: El Modelo de "4+1" vistas de la arquitectura del software", *IEEE Software* 12(6), Noviembre 1995, *Traducido por María Cecilia Bastarrica* en Marzo 2006

[109] G. Booch. *Object-Oriented Analysis and Design with Applications, Benjamin-Cummings* Pub. Co., Redwood City, California, 2nd edition, 1993

[110] S. Kenneth, R and A. Goldberg, "Object behavior analysis" *Communications of the ACM,* 35(9):48–62,1992.