ANÁLISIS COMPARATIVO DE ESTRATEGIAS ARQUITECTÓNICAS DE USABILIDAD WEB

JHONNY ANDRES PAGUAY YAPUD
IVAN ANDRES YANDUN VILLA

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA DE SISTEMAS
PASTO
2017

ANÁLISIS COMPARATIVO DE ESTRATEGIAS ARQUITECTÓNICAS DE USABILIDAD WEB

JHONNY ANDRES PAGUAY YAPUD
IVAN ANDRES YANDUN VILLA

Trabajo de grado presentado como requisito parcial para optar al título de ingeniero de sistemas

Director:
M, Sc. GIOVANNI ALBEIRO HERNANDEZ PANTOJA

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA DE SISTEMAS
PASTO
2017

NOTA DE RESPONSABILIDAD

Las ideas y conclusiones aportadas en el siguiente trabajo, son responsabilidad exclusiva de los autores.

Artículo 1ro del Acuerdo No. 324 de octubre 11 de 1966 emanado del Honorable Consejo Directivo de la Universidad de Nariño.

“La Universidad de Nariño no se hace responsable de las opiniones o resultados obtenidos en el presente trabajo y para su publicación priman las normas sobre el derecho de autor”.

Artículo 13, Acuerdo N. 005 de 2010 emanado del Honorable Consejo Académico.

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

San Juan de Pasto, 11 de septiembre de 2017.

AGRADECIMIENTOS

Los integrantes de este trabajo de grado queremos expresar nuestros más sinceros agradecimientos:

Al magister Giovanni Hernández, por sus ideas, asesorías y recomendaciones, por toda su sabiduría compartida en el proceso del desarrollo del proyecto.

Queremos agradecer a todas las personas que compartieron sus conocimientos; a la Universidad de Nariño, profesores, estudiantes, amigos y demás conocidos.

DEDICATORIA

En esta etapa muy importante en mi vida, quisiera empezar por agradecer al creador de las cosas a DIOS, el cual me ha dado la vida y la oportunidad para llegar a conquistar cada día una nueva etapa.

A mis padres, Francisco Alfredo Paguay y María del Carmen Yapud Pantoja, quienes con su esfuerzo me inspiraron a esta gran etapa de mi vida.

A mis hermanos que siempre estuvieron junto a mí, a mi sobrina Dayanit Marcela Paguay Chávez, quien, me brindó su apoyo absoluto, alentándome para salir adelante, y continuar superándome.

JHONNY ANDRES PAGUAY YAPUD

DEDICATORIA

A mis padres, porque a pesar de las grandes dificultades, siempre me alentaron a continuar adelante ya que sin el apoyo incondicional de ellos no hubiera alcanzado este objetivo.

A mi hijo, que fue el principal motor de la lucha diaria, a mi mujer, por su compañía, por su gran apoyo incondicional, alentándome para salir adelante, para continuar superándome y poder brindar un futuro mejor para ellos.

A mi tío, José María Yandun, gracias a su apoyo incondicional en todo el transcurso de mi vida personal y académica.

A mis profesores, gracias por su tiempo, por su apoyo, así como por la sabiduría que me transmitieron en el desarrollo de mi formación profesional.

IVAN ANDRES YANDUN VILLA

RESUMEN

En el desarrollo de este trabajo, se realizó un estudio de las estrategias arquitectónicas de usabilidad web que se encuentran estandarizadas. El proyecto a desarrollar lleva por nombre “ANÁLISIS COMPARATIVO DE ESTRATEGIAS ARQUITECTÓNICAS DE USABILIDAD WEB” tiene como objetivo aportar a la selección de una estrategia arquitectónica de usabilidad web, mediante un análisis comparativo, para dar respuesta al requisito de calidad de usabilidad.

El presente documento se organiza en cinco capítulos. El capítulo 1, hace referencia al marco teórico de la presente investigación. El capítulo 2, hace referencia a la metodología de esta investigación. El capítulo 3, trata sobre los resultados obtenidos en esta investigación, que se conforma por tres secciones, en la primera sección, se caracteriza las diferentes estrategias arquitectónicas de usabilidad web que se utilizan en la construcción de software, en la segunda sección, se hace un análisis de las estrategias arquitectónicas de usabilidad web encontradas donde se describe de manera comparativa las ventajas y desventajas, en la última sección, se aplica la estrategia arquitectónica con la mayor valorativa resultado del estudio comparativo, al diseño de una aplicación web una tienda online. El capítulo 4, hace referencia a las conclusiones de la presente investigación. Finalmente, el capítulo 5, está conformado por las recomendaciones las cuales se obtuvieron del análisis realizado a las conclusiones.

ABSTRACT

In the development of this work, a study of the architectural strategies of web usability that were standardized. The project to be developed is entitled "COMPARATIVE ANALYSIS OF ARCHITECTURAL STRATEGIES OF WEB USABILITY" aims to contribute to the selection of an architectural strategy of web usability, through a comparative analysis, to meet the requirement of quality of usability.

This document is organized in five chapters. Chapter 1, refers to the theoretical framework of the present investigation. Chapter 2, refers to the methodology of this research. Chapter 3 deals with the results obtained in this research, which is formed by three sections, in the first section, characterizes the different architectural strategies of web usability that are being used in the construction of software, in the second section, makes an analysis of the architectural strategies of web usability found where comparative describes the advantages and disadvantages, in the last section, the architectural strategy is applied with the highest value of the comparative study, the design of a web application an online store . Chapter 4 refers to the conclusions of the present investigation. Finally, chapter 5, is made up of the recommendations that were obtained from the analysis made to the conclusions.

CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS

pág.

GLOSARIO

CALIDAD: conjunto de propiedades y características de un producto o servicio, que le confieren la aptitud para satisfacer necesidades explícitas o implícitas.

CLIENTE: persona o personas encargadas de describir las historias de usuario, es decir, lo que necesita que el aplicativo software haga.

COMANDO: instrucción u orden que el usuario proporciona a un sistema informático.

FEEDBACK: significa retroalimentación. Proceso mediante el cual se realiza un intercambio de datos e información.

HTML: es el lenguaje de marcas de hipertexto, hace referencia al lenguaje de marcado para la elaboración de páginas web.

INTERACCIÓN: describe una acción que se desarrolla de modo recíproco entre personas y objetos.

INTERFAZ: es el diseño de pantallas. Zona de comunicación, en la que se realiza la interacción entre el usuario y el programa.

IPO: interacción persona ordenador.

MECANISMO: característica funcional del sistema**.**

MODELO: son representaciones que describen procesos, problemas y el sistema a desarrollar.

MVC: modelo vista controlador.

PAC: presentación abstracción control.

PROCESO: conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados.

PRODUCTO: es el resultado de un proceso.

REQUISITO: necesidad o expectativa establecida, generalmente implícita u obligatoria.

SISTEMA: conjunto de elementos mutuamente relacionados o que interactúan.

UI: interacción con el usuario.

USUARIO: es la persona que hace uso de las funcionalidades del software.

INTRODUCCIÓN

La gran mayoría de información está disponible en internet, así como la complejidad y heterogeneidad de esta, el gran número de usuarios que acceden a un sitio web las 24 horas del día, juntó con las diferentes características de los usuarios que acceden a dicha información en todo momento, hace crítico y necesario el desarrollo de aplicaciones web usables, que contemplen aspectos de usabilidad como atributo de software especial y de gran prioridad en el desarrollo de software, cada enfoque del atributo de usabilidad provoca que los sistemas de software tengan un nivel de profesionalismo para dar facilidad de aprendizaje, eficiencia, memorabilidad, satisfacción y disminución de errores en sitios web. La usabilidad en sitios web mejora la experiencia del usuario y tienen ventajas económicamente tangibles.

Es el objetivo primordial de este trabajo, aportar a la selección de una estrategia arquitectónica de usabilidad web desde un estudio teórico, para dar respuesta al requisito de calidad de usabilidad. Para alcanzar este fin, se realizará una caracterización de las estrategias arquitectónicas de usabilidad web que se utilizan en la construcción de software, con el propósito de identificar las ventajas y desventajas.

Este trabajo se realiza para seleccionar una estrategia arquitectónica de usabilidad web que resultará de la identificar y comparar las ventajas y desventajas que existen entre estas estrategias arquitectónicas. El análisis comparativo permitirá de manera teórica identificar los aspectos que tienen mayor relevancia a favor encontrados en las estrategias arquitectónicas de usabilidad web, antes de ser seleccionadas. Al seleccionar una estrategia arquitectónica de usabilidad con mayor valoración es muy útil para lograr desarrollar un producto software de calidad, que satisfaga las necesidades del usuario. A partir de los resultados de la comparación, se construirá un producto software aplicando la estrategia arquitectónica de usabilidad web con mayor valoración, para transferir el conocimiento teórico en práctico.

ELEMENTOS DEL PROCESO INVESTIGATIVO

TÍTULO

ANÁLISIS COMPARATIVO DE ESTRATEGIAS ARQUITECTÓNICAS DE USABILIDAD WEB.

MODALIDAD

La modalidad corresponde con un trabajo de investigación, que se desarrolló bajo la línea de software y manejo de información.

LÍNEA DE INVESTIGACIÓN

Línea software y manejo de información. Tiene como objetivo, planificar, analizar, diseñar, implantar, administrar sistemas complejos de información y de conocimiento.

DESCRIPCIÓN DEL PROBLEMA

FORMULACIÓN DEL PROBLEMA

La industria de software en Colombia, se ha caracterizado por ser de alto crecimiento y ha logrado acumular experiencia, conocimiento y capacidades para la producción y prestación de servicios informáticos en diferentes sectores. Según el Ministerio de Comercio, Industria y Turismo[1], esta industria ha crecido en los últimos 5 años en un 230%. Además, cuenta con una infraestructura instalada capaz de soportar operaciones de talla mundial. Para Heshusius Rodríguez[2], esta industria

[1] COLCIENCIAS. Generación de estrategias para el desarrollo tecnológica del sector software y servicios de ti mediante la aplicación de vigilancia. Bogotá: Colciencias, 2011. p.7.

[2] HESHUSIUS RODRÍGUEZ, Karen. Colombia: Desafíos de una Industria en Formación. En: BASTOS TIGRE, Paulo y SILVEIRA MARQUES, Felipe. Desafíos y oportunidades de la Industria de Software en América Latina. Bogotá, Colombia: Mayol Ediciones S.A., 2009. p. 143-144.

está dominada por microempresas y pequeñas empresas, que se dedican de manera concreta al desarrollo de software a la medida, a la intermediación entre las

multinacionales y los clientes finales, a la compra venta de equipos y a la oferta de diferentes servicios de TI.

Las empresas de la industria de software, en cuanto a su operación o funcionamiento, han venido avanzando en los diferentes factores que intervienen al momento de construir software, aspecto relevante para determinar el éxito o fracaso de los trabajos. Según el Standish Group, en el informe que presenta en el chao manifiesto, para el año 2012 del total de trabajos de construcción de software evaluados para identificar su porcentaje de éxito, el 39% finalizan con éxito, un 43% son catalogados como cuestionables, ya que tuvieron desfases en el tiempo, presupuesto, en las características y funcionamiento del software, o en alguna combinación de las anteriores; y un 18% fracasaron. El panorama para el país y la región, está en relación con los datos del chao manifiesto y puede ser más desalentador.

Por otra parte, aspectos como las nuevas maneras de hacer negocios, la tecnología como un factor determinante en la competitividad de las organizaciones, el amplio dominio de la red internet y la diversidad de dispositivos a través de los cuales, se puede acceder a ella, han hecho que actualmente, existan unas necesidades crecientes de software y que aparezcan nuevos desafíos cuando se desea construirlo[3]. Entre los desafíos, se encuentra el incremento en la complejidad de los requerimientos no funcionales. Según Sommerville[4], un requisito es la descripción de lo que el producto software realizará, los servicios que este proporcionará y las restricciones que tendrá en su operación. Los requisitos o requerimientos no funcionales corresponden a las restricciones de los servicios o funciones ofrecidas por el software. Actualmente, estos requisitos, se consideran elementos determinantes en la calidad de un producto y se los denomina atributos de calidad.

Existen diferentes clasificaciones y agrupaciones de atributos de calidad, pero entre las más representativas, se encuentran en los estándares ISO-9126 Software Quality Model e IEEE 1061[5]. Por ejemplo, seguridad, confiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad, entre otros. Para describir la importancia

[3] Casallas, Rubby. ¿Aún en crisis? Algunos mitos y desafíos de la Ingeniería de Software. 2007, Sistemas, p. 8-17.
[4] Sommerville, Ian. Ingeniería de Software. s.l.: Pearson Education, 2011. p. 83.
[5] DÁVILA, Abraham; MELENDEZ, Karin; FLORES, Luis. Determinación de los requerimientos de calidad del producto software basados en normas internacionales. IEEE Latin America Transactions, 2006, vol. 4, no 2, p. 100-106.

que han tomado los atributos de calidad, se presenta el caso de la aplicación web que permite la búsqueda de contenido en internet como lo es google.com. Para el año 2012, Google, dio respuesta a 1.2 billones de consultas, es decir, que cada segundo se realizaba 3805 búsquedas. La funcionalidad de la aplicación, relativamente no es compleja. No obstante, se debe pensar en la forma como este software atiende más de 3800 consultas, cada segundo, que debe estar disponible las 24 horas del día, los 7 días a la semana, los 365 días del año. Otro caso es Amazon, una organización dedicada a la comercialización de productos que utiliza la red internet. Para el año 2008, Amazon estaba facturando aproximadamente 500 dólares cada segundo, es decir, 30.000 dólares cada minuto. Así, se podría seguir citando casos de la forma como la complejidad de los requerimientos no funcionales o atributos de la calidad del software han cobrado un nivel alto de importancia al momento de determinar la calidad del producto software.

Por lo anteriormente descrito, cobra relevancia establecer de manera sistemática un estudio que permita abordar la forma como se puede dar respuesta a los atributos que determinan la calidad de un producto software, qué estrategias se pueden implementar, cuál es el desempeño de las estrategias en diferentes tecnologías.

PLANTEAMIENTO DEL PROBLEMA

¿Cómo aportar a la selección de una estrategia arquitectónica de usabilidad web?

OBJETIVOS

OBJETIVO GENERAL

Aportar a la selección de una estrategia arquitectónica de usabilidad web, mediante un estudio comparativo.

OBJETIVOS ESPECÍFICOS

A continuación, se relacionan los fines específicos que deberán cumplirse para lograr en conjunto el objeto del presente estudio:

- Caracterizar las estrategias arquitectónicas de usabilidad web que se utiliza en la construcción de software.
- Describir de manera comparativa las ventajas y desventajas de las estrategias arquitectónicas identificadas de usabilidad web que se utiliza en la construcción de software.
- Construir un producto software aplicando la estrategia arquitectónica de usabilidad con mayor valoración en la comparativa.

JUSTIFICACIÓN

Este trabajo es interesante porque, al establecer las estrategias arquitectónicas de usabilidad permitió de manera sistemática definir cuales existen y qué se ha descrito sobre ellas de forma teórica. Además, el comparar las estrategias arquitectónicas de usabilidad permitió de manera teórica identificar los aspectos que tienen a favor y en contra, antes de ser seleccionadas. Al construir un producto software utilizando la estrategia arquitectónica que obtenga la mayor valoración en el proceso de evaluación, permitió contrastar los resultados de la comparativa teórica y lo obtenido en la implementación de la estrategia de forma práctica.

Este trabajo también es útil porque con la identificación de estrategias arquitectónicas de usabilidad para aplicaciones web, estableciendo las ventajas y desventajas que tienen, se logró optimizar los costos de diseño, rediseño y mantenimiento de las aplicaciones web, además de aumentar la tasa de conversión de visitantes a clientes de un sitio web y mejorar la imagen de una aplicación web. Todos estos beneficios implican una reducción y optimización general de los costes de producción, así como un aumento en la productividad. La usabilidad permite mayor rapidez en la realización de tareas y reduce las pérdidas de tiempo. Además, la usabilidad, se ha aplicado consistentemente e integrado a los procesos de desarrollo de software primero con la ergonomía y ahora ampliándose a los sistemas de pantalla, con el auge de aplicaciones para tabletas, smartphone y nuevas tecnologías que han permitido su amplia difusión.

En cuanto al diseño de sitios web, la usabilidad tiene relación con la facilidad de uso de estos, así como la capacidad de que los usuarios y visitantes pueden navegar por el mismo, de una forma práctica, útil y sencilla. No obstante, todos los sitios en la web no son usables, algunas páginas hacen cometer errores e incluso causan no volver a visitar más la página, esto justifica que se integre estrategias arquitectónicas de usabilidad al realizar un proceso de desarrollo de software para que así el resultado de un producto de desarrollo de software cumpla con unos criterios de calidad para los usuarios finales.

Por lo tanto, se puede definir que el objetivo de esta investigación está ubicado en un marco novedoso y poco trabajado, pero no exento de importancia para el desarrollo y generación de sistemas de calidad, mediante la aplicación de una estrategia arquitectónica de usabilidad en la web. Por lo tanto, este estudio es un trabajo novedoso y de alto valor para los desarrolladores de software, ya que los resultados son un aporte a la selección de una estrategia arquitectónica de usabilidad dentro del proceso de desarrollo de software.

Debido a la importancia y relevancia del modelamiento en la construcción de software, este estudio permitió aplicar un método basado en principios, procesos y conceptos de la arquitectura de software, analizando la usabilidad como un atributo de calidad para la construcción de aplicaciones web.

ALCANCE Y DELIMITACIÓN

Este trabajo tiene como fin realizar un análisis comparativo de las estrategias arquitectónicas de usabilidad web, identificando ventajas y desventajas para establecer un orden de favorabilidad, a partir del cual, se selecciona una estrategia, la cual se aplicó en la construcción de un producto software. El análisis fue efectuado por los integrantes de la investigación, a partir de la revisión documental de las estrategias de mayor uso y difusión que estén documentadas. Los beneficiarios, son todos los interesados en la construcción de software orientado a la web que definen la usabilidad como un atributo de la calidad del producto, es decir, empresas de la industria de software y programas académicos de Educación Superior que forman técnicos, tecnólogos y profesionales en la construcción de software. Consiste en definir con toda claridad cuál es el alcance o extensión de la propuesta de trabajo, describe los temas/módulos que involucra el trabajo; el escenario en el cual se realizará; beneficiarios de los resultados. El alcance y la delimitación deben estar reflejados en los objetivos tanto generales como específicos.

1. MARCO TEÓRICO

1.1 ANTECEDENTES

Para la realización del presente trabajo se tuvo en cuenta los siguientes antecedentes:

En el trabajo investigativo realizado por Moreno[6], el cual tuvo como objetivo el desarrollo de técnicas y procedimientos a incorporar durante el diseño de un sistema software con el fin de conseguir mejoras en la usabilidad. Las principales conclusiones obtenidas en la investigación, se orientó en seleccionar un patrón de diseño para que pueda ser incorporado y cómo éste impacta sobre otros atributos de calidad. La principal similitud de este antecedente con el estudio que se realizará, se establece en la selección y uso de un patrón arquitectónico de usabilidad, como estrategia en el diseño de software de calidad. Las principales diferencias de esta investigación en relación con el antecedente, se encuentran al buscar identificar más de una estrategia arquitectónica de usabilidad web y establecer elementos de comparación que permita determinar ventajas y desventajas entre ellas.

En el trabajo investigativo presentado por Arciniega et al[7], el cual tuvo como objetivo principal establecer un proceso para la definición de la arquitectura considerando aspectos de usabilidad en el desarrollo de aplicaciones web. La principal conclusión de este antecedente corresponde con el uso de una metodología basada en el enfoque de desarrollo ágil y centrado en el punto de vista de la usabilidad, en 2 pequeñas y medianas empresas del departamento del Cauca – Colombia. La principal similitud del antecedente con este estudio, se orientó en identificar formas de usabilidad para aplicaciones web, desde un enfoque de la arquitectura. La principal diferencia obedece a que, en este trabajo, se identifica estrategias arquitectónicas de usabilidad, orientadas a la web para evaluarlas determinando ventajas y desventajas.

---

[6] MORENO, Ana. patrones de usabilidad: Mejora de la usabilidad del software desde el momento arquitectónico. VIII Jornadas de ingeniería del software y bases de datos (JISBD 03). Alicante, noviembre 2013.

[7] ARCINIEGAS, José L; FERNÁNDEZ, Verónica; HORMIGA, Amparo; TULANDE, ALEYDA; URBANO, Fernando A; COLLAZOS, César A. Proceso de requerimiento y análisis para la definición de la arquitectura desde la perspectiva de usabilidad para el desarrollo de aplicaciones en la Web, Revista Avances en Sistemas e Informática, vol. 6, núm. 2, septiembre, 2009, p 8.

En el trabajo investigativo realizado por Panach[8], el cual tuvo como objetivo presentar un método (llamado MIMAT) para incorporar las características funcionales de usabilidad dentro de un método de desarrollo dirigido por modelos. El estudio estableció como principal conclusión, un problema de un analista de software, es incorporar la usabilidad a los sistemas ya desarrollados esto conlleva a un alto coste de tiempo al no tratar la usabilidad durante el desarrollo de software. La incorporación de estrategias de usabilidad durante el desarrollo acorta estos tiempos. Esta investigación se relaciona con la investigación planteada, ya que muestra cómo incorporar la usabilidad dentro de los procesos de producción de software lo cual resultó un aporte importante, para la selección de una estrategia arquitectónica de usabilidad. La principal diferencia es que en esta investigación se realiza comparación y caracterización de estrategias de usabilidad en la web.

En el trabajo investigativo realizado por Ferre[9], el cual tuvo como objetivo ayudar a los ingenieros de software a seleccionar las técnicas y actividades IPO (Interacción Persona-ordenador) más apropiadas para integrar un proceso de desarrollo, de modo que se trate la usabilidad en el producto software. La conclusión principal a la que arribó este estudio es, la usabilidad se percibe cada vez más por las organizaciones de desarrollo de software como un objetivo estratégico además en los últimos años la usabilidad ha pasado de ser un atributo a hacer un objetivo relevante en el desarrollo de software. La principal similitud de este antecedente con el estudio que se realizó son la de identificar estrategias arquitectónicas de usabilidad, al comparar con el antecedente se concluye que este estudio es de gran importancia para esta investigación ya que este antecedente muestra cómo los atributos de calidad en especial el de usabilidad determina la calidad de un producto software al integrar la usabilidad como un objetivo prioritario en el desarrollo de software. La principal diferencia en relación con el antecedente es la caracterización de estrategias de usabilidad en la web.

En la investigación realizada por Mauro, Castillo y Fernández[10], el cual tuvo como objetivo implementar una herramienta para la gestión de requisitos con el fin de implementar atributos de calidad de usabilidad, la principal conclusión obtenida en la investigación se orientó a satisfacer las necesidades tanto de los usuarios como

[8] PANACH, José Ignacio. Incorporación de mecanismos de usabilidad en un entorno de producción de software dirigido por modelos. Trabajo de grado. Valencia: Universidad Politécnica de Valencia. Departamento de Sistemas Informáticos y Computación, 2010. 570 p.

[9] FERRE, Javier. Principios Básicos de Usabilidad para Ingenieros Software. Investigación. Madrid: Universidad Politécnica de Madrid. Facultad de Informática, 2005. 8 p. Disponible en Internet: <http://is.ls.fi.upm.es/xavier/papers/usabilidad.pdf>

[10] CALLEJAS CUERVO, Mauro; CASTILLO ESTUPIÑAN, Luz Yadira; FERNÁNDEZ Álvarez, Ruby Mónica. HELER: UNA HERRAMIENTA PARA LA INGENIERÍA DE REQUISITOS AUTOMATIZADA. Entramado, vol. 6, núm. 2, julio-23 diciembre, 2010, p. 184-200.Disponible en Internet: <http://www.redalyc.org/pdf/2654/265419645014.pdf>

el de los clientes dentro del atributo de calidad de usabilidad, la principal similitud de este antecedente con el estudio que se realizó es una investigación e implementación del atributo de calidad usabilidad por medio de una aplicación de página web como estrategia en el diseño de software de calidad, las principales diferencias de esta investigación en relación con el antecedente se encuentra en identificar más de una estrategia arquitectónica de usabilidad por medio de una aplicación web y observar las diferentes ventajas y desventajas que se puede dar durante este proceso y aplicación del atributo de calidad de usabilidad web.

1.2 SUPUESTOS TEÓRICOS

A continuación, se describe los elementos teórico-conceptuales que apoyan el desarrollo de este trabajo investigativo.

1.2.1 Arquitectura de software. La arquitectura de software se comprende como una guía de desarrollo para un sistema de software, esta define y describe la estructura del sistema de software que se va a implementar, así como, funcionamiento, con interacción y coordinación entre los componentes o piezas del mismo. De acuerdo con Buschmann y Sommerland[11], las componentes de un sistema de software representan las decisiones de diseño, los resultados de la organización y configuración de las componentes de la estructura del sistema, que garantiza que se satisfaga las necesidades de los clientes y usuarios.

Dentro del desarrollo de un sistema de software, se establece la definición de la arquitectura. Según Hofmeister, Nord y Soni[12], este propósito se logra con las siguientes etapas. La primera etapa, corresponde al levantamiento de requerimientos, donde se muestra qué elementos y funciones son necesarias para el desarrollo de software, así como la captura, documentación y priorización de las necesidades que influencian la arquitectura. La segunda etapa, es el diseño, la cual, determina las piezas y estructuras que conformarán el producto software. En esta etapa, además, se establece un marco de trabajo para gestión y control de los componentes que integran la arquitectura con base en tácticas de diseño que involucra la toma de decisiones. La tercera etapa, es la documentación, donde se abarca todo el sistema e involucra la representación de las estructuras que son representadas a través de distintas vistas las cuales especifican los distintos

[11] Buschmann, F., Meunier, R., Rohnert, H., Sommerland, P., Pattern oriented software architecture: a system of patterns. John Wiley & Sons, 1996.

[12] Christine Hofmeister, Robert Nord y Dilip Soni. Describing software architecture with uml. En proceedings of the first working conference on software architecture, IEEE Computer Society Press, San Antonio, febrero de 1999. p. 145-160.

aspectos técnicos y funcionales. Finalmente, se realiza la etapa de evaluación, la cual establece, un estudio factible que permite identificar posibles problemas y riesgos y establecer recomendaciones. Por otro lado, las ventajas de evaluar el diseño la cual es una actividad que permite saber si el sistema cumple con las necesidades de los interesados, con esto se busca disminuir costos de corrección de defectos, ya que este es mucho menor a tener que corregirlos cuando el sistema haya sido terminado.

A sí mismo, la arquitectura de software es primordial en el desarrollo de un sistema software, porque determina su estructura y como este satisface los requisitos no funcionales o de calidad establecidos por los interesados, brindando una solución a un problema. Además, el adoptar un diseño arquitectónico permite la reutilización al crear sistemas distintos, esto tiene una ventaja muy importante debido a que reduce costos, aumenta la calidad, sobre todo si el diseño ha resultado ser exitoso en otros sistemas.

1.2.2 Atributo de calidad. La calidad del software se comprende como el conjunto de cualidades o características implícitas que se esperan de todo software, lo caracterizan y determinan su utilidad y existencia. Barbacci y Weinstock[13], definen la calidad de software, como el grado en el cual el software posee un conjunto de atributos o cualidades que le permiten ser lo que dice que se es. Por lo tanto, la calidad se entiende que son las cualidades, características adicionales que posee un sistema de software, estas características o atributos se definen como las propiedades de un servicio que presta el sistema el cual al realizar un proceso cumple con los requerimientos y satisface las necesidades de los usuarios.

Bass y Kazman[14], clasifican los atributos de calidad en dos categorías. La primera, corresponde a los atributos observables, los cuales son atributos que se determinan en el tiempo de ejecución de un sistema. Por ejemplo, desempeño, seguridad, disponibilidad, funcionalidad, usabilidad entre otros. La segunda, es la de los no observables, los cuales son atributos que se determinan en tiempo de desarrollo de un sistema. Por ejemplo, configurabilidad, integralidad, interoperabilidad, modificabilidad, mantenibilidad, portabilidad, reusabilidad, escalabilidad, testeabilidad, entre otros. Por otra parte, los atributos de calidad también son considerados funciones u objetivos específicos, los cuales proporcionan los resultados esperados por un usuario en el proceso de interacción con el producto software.

[13] BARBACCI, Mario; MARK, Klein; THOMAS A; LONGSTAFF, CHARLES, B; WEINSTOCK, Technical report, CMU/SEI-95-TR-021, ESC-TR-95-021, Quality Attributes, Diciembre 1995.
[14] BASS, Len; CLEMENTS y KAZMAN, Software architecture in practice, Third Edition, 2013. Capítulo 4: Understanding quality attributes.

1.2.3 Usabilidad. El término usabilidad es algo novedoso surgió poco después del internet en los años 90 aunque se la ha conocido por otros nombres, como, por ejemplo, diseño centrado en el usuario o (HCI) por sus siglas en inglés. Para Nielsen[15], la usabilidad es un atributo de calidad del software que se refiere a la sencillez de uso, porque facilita que un producto software sea entendido, aprendido, usado, accesible y resulte ser atractivo para el usuario. A continuación, se describe la definición de algunos autores y estándares ISO:

Nielsen[16], considerado el padre de la usabilidad, la definió como un atributo de calidad que mide el nivel de facilidad de las interfaces de usuario al ser utilizadas. La palabra usabilidad también se refiere a métodos para mejorar la facilidad de uso durante el proceso de diseño.

Nielsen y Ben Shneiderman[17], han escrito por separado sobre un marco de trabajo de aceptabilidad del sistema, donde la usabilidad es parte de la utilidad y se compone de las siguientes características. La primera, es la facilidad de aprendizaje, se refiere a funcionalidad y comportamiento del sistema, esta define en cuánto tiempo un usuario que nunca ha visto una interfaz, puede aprender a usarla bien y realizar operaciones básicas. La segunda, es eficiencia de uso, esta involucra alcanzar el nivel de productividad requerido, una vez que el usuario ha aprendido a usar el sistema, está determina la rapidez con que se puede desarrollar las tareas. La tercera, es retención sobre el tiempo, esta describe cuando un usuario ha utilizado un sistema tiempo atrás y tiene la necesidad de utilizarlo de nuevo, la curva de aprendizaje debe ser significativamente menor que el caso del usuario que nunca haya utilizado dicho sistema. Esto es de primordial importancia para aplicaciones usadas intermitentemente. Finalmente, la última característica es tasas de error, es la capacidad del sistema para ofrecer una tasa baja de errores, apoyar a los usuarios a cometer pocos errores durante el uso del sistema y en caso de que cometan errores ayudarles a recuperarse fácilmente.

Bohmann[18], el experto en usabilidad define que, los usuarios puedan terminar sus tareas en particular, los usuarios y sus principales necesidades son conocidas y detalladas es decir los usuarios pueden terminar sus tareas sin demora o errores y pueden disfrutar de la experiencia.

---

[15] NIELSEN, Jakob, Usability engineering. AP Professional, Boston, MA, 1993.
[16] NIELSEN, Jakob, [en línea] <http://www.useit.com/alertbox/20030825.htm> [citado el 20 de abril de 2016].
[17] NIELSEN, Jakob. (1999). [en línea] <http://www.useit.com/alertbox/20030825.html. > [citado el 20 de abril de 2016].
[18] Bohmann, Pensando en el usuario. 2001. Disponible en: <http://www.bohmann.dk/observations/2001oct12.html 2001>. [citado el de febrero 2017].

ISO/IEC 9126, la usabilidad es la capacidad del producto software para ser entendido, aprendido, usado y resulte atractivo para el usuario, cuando se usa bajo determinadas condiciones[19]. Esta característica se subdivide a su vez en las siguientes subcaracterísticas. La primera subcaracterística es la capacidad para reconocer su adecuación, consiste en la capacidad del producto que permite al usuario entender si el software es adecuado para sus necesidades. La segunda sub característica es la capacidad de aprendizaje, es la capacidad del producto que permite al usuario aprender su aplicación. La tercera subcaracterística es la capacidad para ser usado, esta permite al usuario operarlo y controlarlo con facilidad. La cuarta subcaracterística es la protección contra errores de usuario, es la capacidad del sistema para proteger a los usuarios de hacer errores. La quinta subcaracterística es estética de la interfaz de usuario, se refiere la capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario. Finalmente, la última subcaracterística es accesibilidad, es la capacidad del producto que permite que sea utilizado por usuarios con determinadas características especiales y discapacidades.

ISO 9241, la usabilidad se refiere a la efectividad, eficiencia y satisfacción con la que usuarios concretos pueden abarcar unos objetivos específicos en un entorno en particular[20]. El primer factor la efectividad, es la exactitud e integridad con la que los usuarios alcanzan los objetivos especificados y, por tanto, implica la facilidad de aprendizaje; la ausencia de errores del sistema o la facilidad del mismo, para ser recordado. El segundo factor la eficiencia, son los recursos empleados, por ejemplo, esfuerzo, tiempo entre otros; en relación con la exactitud e integridad con la que los usuarios alcanzan los objetivos especificados. Finalmente, el último factor es la satisfacción, un factor subjetivo que implica una actitud positiva en el uso del producto.

1.2.4 Usabilidad en la web. La web se ha convertido en un elemento esencial tanto en el desarrollo de las organizaciones como de las instituciones, ofrece información y una amplia gama de servicios. El acceso a la web se ha hecho indispensable hoy en día, los usuarios acceden las 24 horas del día a un sitio o página web, por esto la satisfacción de un sitio web depende directamente de su facilidad de uso es decir de su usabilidad. En la actualidad la usabilidad se aplica básicamente al diseño de páginas web si bien se puede integrar a cualquier aplicación o desarrollo de software

[19] ISO, 9126. Information Technology-software Product Evaluation-Quality Characteristics and Guidelines for their use. ISO, ISO, 1991. Disponible en: <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16883>. [citado el 06 de junio 2016].

[20] ISO, 9241. Ergonomics requirements for office work with visual display terminals.ISO, 1998. Disponible en: <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16883>. [citado el 06 de junio 2016].

ya que es un atributo de calidad clave en el desarrollo web. Para Murray[21], la clave en la usabilidad de un sitio web es asegurarse de que el sitio sea útil y usable para la audiencia objetivo; quizás una de las decisiones más importantes al momento de comenzar a desarrollar un sitio web sea precisamente el definir los usuarios que van a tener acceso o cuales se va a manejar.

Hassan[22], define la usabilidad dentro del campo del desarrollo web como la disciplina que estudia la forma de diseñar sitios web para que los usuarios puedan interactuar con ellos de la forma más fácil, cómoda e intuitiva posible. La mejor forma de crear un sitio web usable es realizar un diseño centrado en el usuario, diseñado para y por el usuario, en contraposición a lo que podría ser un diseño centrado en la tecnología o uno centrado en la creatividad u originalidad.

1.2.5 Estrategia de arquitectura. Para Grant[23], una estrategia es un concepto que se aplica a diferentes ámbitos como, por ejemplo, el militar, el empresarial, el deportivo, entre otros. Esta desea alcanzar un fin mediante el desarrollo de un conjunto de acciones que se implementan en un contexto determinado. En la arquitectura de software, una estrategia es una guía que contiene un conjunto de tácticas de arquitectura de software para satisfacer los atributos de calidad y cumplir con uno o varios objetivos. En una estrategia de arquitectura, se planea, organiza, ejecuta y evalúa; se es consciente de las acciones que se realizan y de las decisiones que se toman, con el fin último de diseñar un producto software de calidad.

1.2.6 Táctica de arquitectura. Una táctica se comprende que es un plan a ejecutar, para lograr un objetivo y dar una solución exitosa y satisfacer una respuesta a un atributo de calidad. Las tácticas de arquitectura tienen como objetivo lograr satisfacer atributos de calidad en un sistema de software, se considera una arquitectura de software bien diseñada aquella que integra tácticas que permite satisfacer las necesidades del sistema.

Por otra parte, para Gaitán, una táctica consiste en aplicar una determinada estrategia que permite organizar y optimizar uno o varios componentes del sistema.

---

[21] MURRAY, Tom. Authoring Tools for Advanced Technology Learning Environments, Dordrecht: Kluwer Academic Publishers.2003.

[22] HASSAN, Y. Introducción a la usabilidad. En: No Solo Usabilidad, noviembre 1, 2002. [citado el 06 de junio de 2016] <nosolousabilidad.com>.
Disponible en <http://www.nosolousabilidad.com/articulos/introduccion_usabilidad.htm>.

[23] GRANT, R. Dirección estratégica. Conceptos, técnicas y aplicaciones.2004 Ed. THOMSON – CIVITAS.

En la arquitectura de software para Bass[24], las tácticas son decisiones de diseño que influyen en el control de la respuesta de un atributo no funcional estas se asocian con los escenarios y preocupaciones; se enfocan principalmente en los atributos de calidad específicos como, por ejemplo, desempeño, disponibilidad, modificabilidad, seguridad, testeabilidad, usabilidad, entre otros.

1.2.7 Estrategia arquitectónica de usabilidad web. La táctica de usabilidad web, se puede combinar con otras tácticas o refinar en un orden jerárquico que dan lugar a una estrategia la cual conlleva que la usabilidad en sitios web mejore la experiencia del usuario y provoque ventajas económicamente tangibles. Una estrategia arquitectónica de usabilidad web da respuesta al requisito de calidad de usabilidad en el desarrollo de aplicaciones web, porque está integra un conjunto de tácticas que hacen que cada enfoque del atributo de usabilidad provoque que un sitio web, tenga un nivel de profesionalismo alto para dar facilidad de aprendizaje, eficiencia, memorabilidad, satisfacción, y disminución de errores entre otras.

Por lo anterior, una estrategia de arquitectura de usabilidad, web se comprende como una guía de diseño arquitectónico integrado por tácticas, para satisfacer el atributo de usabilidad a un alto nivel de calidad. En el diseño de sitios web que sigan como guía estrategias arquitectónicas de usabilidad provoca que sean usables, porque estos contemplan aspectos de usabilidad como un atributo de software especial.

[24] BASS, L., CLEMENTS, P., y KAZMAN, R. Software Architecture in practice. 2003. Addison-Wesley.

2. METODOLOGÍA

2.1 PARADIGMA, ENFOQUE Y TIPO

Esta investigación es de corte cuantitativo, porque según Sampieri, y Baptista[25], representa un conjunto de procesos secuenciales y probatorios, por lo tanto, no se puede saltar los pasos debido a que el orden es riguroso que parte de una idea que una vez delimitada, se deriva en objetivos y preguntas de investigación, de las preguntas se establecen hipótesis y se determinan variables y la validez de los resultados se soportará a través de un proceso estadístico descriptivo debido a que se realizan mediciones, se utilizan técnicas de recolección y se analizan los datos procediendo de manera inductiva donde se obtendrán conclusiones empíricas.

El enfoque para esta investigación es Empírico-analítico[26], debido a que se basa en las experiencias propias para aportar a la selección de una estrategia arquitectónica de usabilidad web, identificando las estrategias arquitectónicas, describiendo de manera comparativa las ventajas y desventajas de las estrategias arquitectónicas y construyendo un producto software aplicando una estrategia arquitectónica de usabilidad con mayor valoración en la comparativa.

El tipo de investigación es descriptiva porque, se pretende a través de las situaciones y de las características de la propuesta planteada recolectar datos sobre la hipótesis y de esta manera describir las características que poseen las estrategias arquitectónicas de usabilidad web, para posteriormente establecer una forma de comparación y aplicarla identificando y representando las ventajas y desventajas que puede tener cada una.

2.2 POBLACIÓN Y MUESTRA

La población corresponde a las estrategias arquitectónicas de usabilidad web que han sido estandarizadas.

[25] HERNÁNDEZ Sampieri, Roberto, FERNÁNDEZ, Carlos y BAPTISTA, Pilar. Fundamentos de metodología de la investigación. S.l.: MCGRAW-HILL, 2007.
[26] LANDEAU, Rebeca. Elaboración de trabajos de investigación. S.l.: ALFA, 2007.

El muestreo es no probabilístico de tipo intencional, debido a que, para el estudio, se utilizará aquellas estrategias arquitectónicas de usabilidad web que se encuentren claramente definidas y documentadas.

2.3 PROCESO DE INVESTIGACIÓN

Tabla 1. Operacionalización de los objetivos.

Objetivos específicos	**Fuente**	**Técnica de recolección**	**Instrumento**	**Técnica de Procesamiento**	**Resultado**
Caracterizar las estrategias arquitectónicas de usabilidad web que se están se utilizan en la construcción de software.	Documentación de las estrategias arquitectónicas de usabilidad web a caracterizar.	Revisión documental.	Ficha de revisión documental.	Análisis documental.	Documento con la caracterización de las estrategias arquitectónicas de usabilidad web.

Objetivos específicos	**Fuente**	**Técnica de recolección**	**Instrumento**	**Técnica de Procesamiento**	**Resultado**
Describir de manera comparativa las ventajas y desventajas de las estrategias arquitectónicas identificadas de usabilidad web que se están utilizando en la construcción de software.	Documento con la caracterización de las estrategias arquitectónicas de usabilidad web.	Revisión documental.	Matriz de revisión documental.	Estadística descriptiva.	Matriz de ventajas y desventajas de las estrategias arquitectónicas de usabilidad web.

3. DESARROLLO

En este capítulo se describe los resultados encontrados en esta investigación. Como primer punto, se describe el contexto del objetivo a evaluar, continuando con una descripción de las variables analizadas y finalmente se realiza una síntesis de los resultados para cada uno de los objetivos.

3.1 CARACTERIZACIÓN DE LAS ESTRATEGIAS ARQUITECTÓNICAS

En esta sección se caracteriza las estrategias arquitectónicas de usabilidad web que se utilizan en la construcción de software. Para alcanzar este objetivo, se tuvo como fuente de información de las estrategias arquitectónicas de usabilidad web que han sido estandarizadas. El muestreo es no probabilístico de tipo intencional, debido a que, para el estudio, se utilizará aquellas estrategias arquitectónicas de usabilidad web que se encuentren claramente definidas y documentadas. La técnica que se utilizó para la recolección de información fue la revisión documental, para el análisis de la información se utilizó como técnica el análisis documental. Las variables analizadas fueron: las estrategias arquitectónicas de usabilidad web.

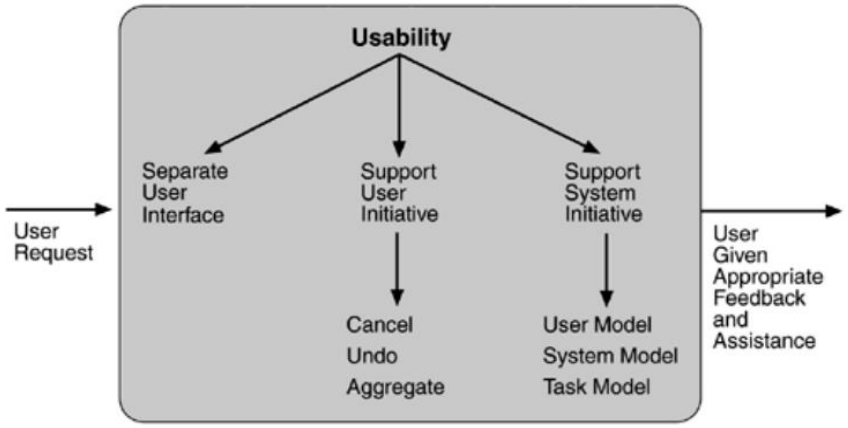
Para realizar la caracterización de las diferentes estrategias se centra en:

- Los trabajos que integran tácticas de usabilidad durante la construcción del diseño arquitectónico.
- Las tácticas que se complementan con los modelos que tratan la usabilidad.
- Los patrones de usabilidad que dan solución a problemas de usabilidad en un sistema de software, los cuales integran un conjunto de tácticas.
- Las tácticas que se complementan con herramientas que tratan la usabilidad dentro de un método de diseño arquitectónico.
- Directrices de usabilidad y frameworks.

Una vez presentado el contexto y la forma como se realizó el análisis de los datos, se procede a mostrar los resultados encontrados.

Una estrategia de usabilidad web integra un conjunto de tácticas, en la figura 1, se detallan las tácticas de usabilidad propuestas por Bass.

Figura 1. Tácticas de usabilidad



Fuente: (ArquitecturaSWjrpolanco38, 2012)
[En línea]. <http://etutorials.org/shared/images/tutorials/tutorial_51/05fig03.gif>

En la actualidad, se desarrolla trabajos de sistemas web utilizando directamente herramientas de diseño complejas, que permite construir vistosas y elaboradas interfaces para los usuarios, sin embargo, se omite el importante proceso previo de análisis y diseño arquitectónico, el cual satisface los requerimientos funcionales y no funcionales establecidos por los interesados como, por ejemplo, usuarios y clientes. Esto genera como consecuencia aplicaciones que presentan problemas de calidad al no satisfacer algunos de los atributos como es el de usabilidad el cual se refiere a la eficiencia, satisfacción y a la sencillez de uso. Por esta razón, se dio origen a estrategias arquitectónicas de usabilidad web, en esta investigación las estrategias que se han considerado son:

- **Tácticas en tiempo de diseño**: apoyan al diseño de la interfaz e interacción con el usuario.
- **Tácticas en tiempo de ejecución**: son aquellas que dependen de quién toma la iniciativa en la interacción entre el usuario y el sistema, estas apoyan al usuario durante la ejecución del sistema.

Cabe resaltar que un conjunto de tácticas, permite seguir un guía para el diseño arquitectónico de sistemas, lo cual da lugar a la estrategia; y sirve como guía para

lograr satisfacer el atributo de usabilidad. A continuación, se hace un análisis detallado de las tácticas anteriormente mencionadas.

3.1.1 Tácticas en tiempo de diseño. Son consideradas decisiones de diseño, las cuales permiten al usuario interactuar con el sistema. Las tácticas en tiempo de diseño que se han considerado para esta investigación, son:

- Separar la interfaz del resto de la aplicación.
- Diseño de interfaces de usuario.
- Guías de estilo y frameworks.

Estas tácticas, se puede combinar con otras tácticas o refinar en un orden jerárquico que dan lugar a una estrategia la cual conlleva que la usabilidad en sitios web mejore la interacción y experiencia del usuario.

3.1.1.1 Separar la interfaz del resto de la aplicación. Según Bass, la interfaz de usuario se espera que cambie frecuentemente durante el desarrollo y después de la implementación, mantener el código de interfaz de usuario por separado localizara variaciones en él. Los patrones de arquitectura de software para poner en práctica esta táctica son los siguientes:

- Modelo-vista-controlador.
- Presentación-abstracción-control.
- Seeheim.
- Arch/Slinky.

A continuación, se hace un análisis detallado de los diferentes patrones anteriormente mencionados.

• **Modelo vista controlador.** El modelo vista controlador -MVC, para Álvarez[27], es una propuesta de diseño arquitectónico de software, utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario que separa la lógica de una aplicación de la presentación o interfaz de usuario.

[27] ALVAREZ, Miguel. ¿Qué es MVC? 2016. [en línea]. < https://desarrolloweb.com/articulos/que-es-mvc.html>.[citado el 01 de diciembre de 2016].

Este modelo, surge de la necesidad de crear software más robusto y ordenado con un ciclo de vida más apropiado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos. Este modelo, se comprende que está indicado especialmente para el diseño de arquitecturas de aplicaciones que requieren de una gran interactividad con los usuarios, como es el caso de aplicaciones web.

Por otra parte, este modelo, se considera una guía para el diseño de arquitecturas de aplicaciones, que ofrece una fuerte interactividad con los usuarios. Este surge con el objetivo de reducir el esfuerzo de programación, necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, a partir de estandarizar el diseño de las aplicaciones. Este patrón organiza la aplicación en tres modelos separados. El primero es un modelo, que representa los datos de la aplicación y sus reglas de negocio. El segundo, es un conjunto de vistas que representa los formularios de entrada y salida de información. El tercero, es un conjunto de controladores que procesa las peticiones de los usuarios y controla el flujo de ejecución del sistema, que permite la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en lapso reducido de tiempo.

Según Pavón[28], el modelo - MVC se estructura en tres capas. A continuación, se analiza las tres capas del modelo vista controlador:

La primera capa es la de modelos, es la capa donde se trabaja con los datos, esta representa la parte de la aplicación que implementa la lógica de negocio, por tanto, contiene mecanismos para acceder a la información y también para actualizar su estado. Los elementos de información se los encuentra habitualmente en una base de datos, por lo que en los modelos se tienen todas las funciones que accederán a las tablas. Por ejemplo, en Facebook, la capa de modelo cumple la función de tareas como guardar información del usuario, tales como, asociaciones con amigos almacenamiento de recuperación de información, fotos, videos, encontrar sugerencias de nuevos amigos entre otras, mientras que el objeto del modelo puede ser considerado como, amigo, usuario, comentario, foto y video. Estos ejemplos generalmente están en una base de datos, por lo que en los modelos se tiene todas las funciones que accederán a las tablas y harán los correspondientes selects, updates, inserts entre otras funcionalidades.

[28] PAVÓN, Juan. Estructura de las aplicaciones orientadas a objetos: El patrón modelo-vista-controlador. Universidad Complutense Madrid.2008 Disponible en: <https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.>. [citado el 31 de agosto de 2016].

La segunda capa es vistas, para Peinado[29], es lo que utilizan los usuarios para interactuar con la aplicación como, por ejemplo, los gestores de plantillas pertenecen a esta capa. Esta capa, contiene el código de la aplicación que va a producir la visualización de las interfaces de usuario. En las vistas nada más se tiene los códigos como, por ejemplo, HTML que permiten mostrar la salida. Esta capa es responsable del uso de la información de la cual dispone para producir cualquier interfaz de presentación de cualquier petición que se presente.

En la vista generalmente se trabaja con los datos, sin embargo, no se realiza un acceso directo a éstos. Las vistas necesitan los datos a los modelos y en ellas se produce la salida, tal como la aplicación la requiera. Por ejemplo, como la capa de modelo devuelve un conjunto de datos, la vista los usaría para hacer una página HTML que los contenga o brindar un resultado con formato XML para que otras aplicaciones puedan consumir. Por otra parte, la capa de la vista no se restringe únicamente a HTML o texto que represente los datos, sino que puede ser utilizada para brindar una amplia variedad de formatos en función de sus necesidades tales como videos, música, documentos y cualquier otro formato.

La tercera capa es control, esta capa contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como por ejemplo visualizar un elemento, realizar una compra, una búsqueda de información, entre otras como ejemplo, más detallado en Symfony las peticiones se canalizan a través de los controladores frontales como index.php y frontend_dev.php. Por otra parte, es una capa que sirve de enlace entre las vistas y los modelos, esta realiza llamadas al modelo para obtener los datos y se los pasa a la vista para que muestre al usuario, responde a los mecanismos que puedan requerirse para implementar las necesidades de la aplicación. Sin embargo, su responsabilidad no es manipular directamente datos, ni mostrar ningún tipo de salida, sino servir de enlace entre los modelos y las vistas para implementar las diversas necesidades del desarrollo. Para Cake Software Foundation[30], los controladores pueden ser vistos como administradores los cuales cuidan de que todos los recursos necesarios para completar una tarea se deleguen a los trabajadores más adecuados, esta espera peticiones de los clientes, comprueba su validez de acuerdo con normas de autenticación o autorización, delega la búsqueda de datos al modelo y selecciona el tipo de respuesta más adecuado según las preferencias del cliente, finalmente delega este proceso de presentación a la capa de la vista.

---

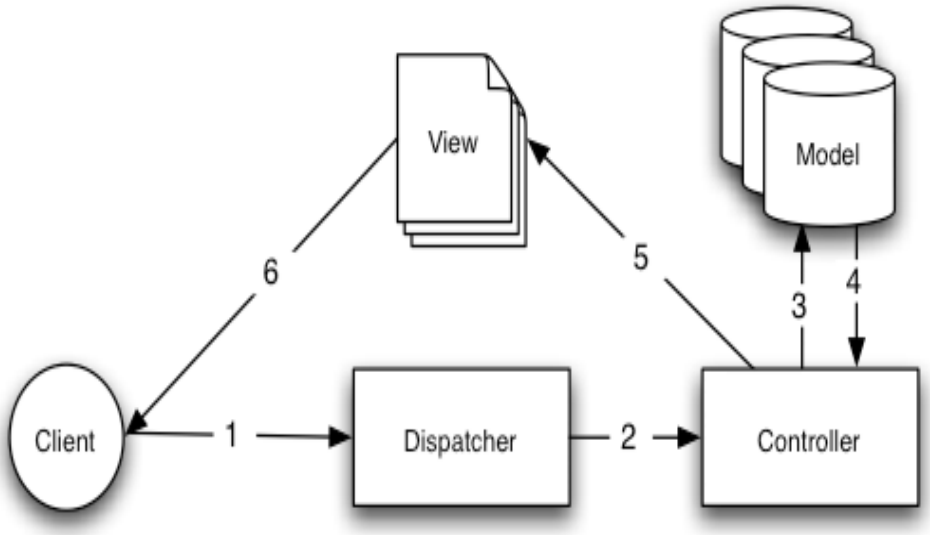
[29] PEINADO, F. Patrón Modelo-Vista-Controlador. Universidad Complutense Madrid. Disponible en <https://www.fdi.ucm.es/profesor/fpeinado/courses/oop/LPS-14ModeloVistaControlador.pdf >. [citado el 31 de agosto de 2016].

[30] CAKE SOFTWARE FOUNDATION. Entendiendo Modelo-Vista-Controlador [en línea]. < https://book.cakephp.org/1.2/es/The-Manual/Beginning-With-CakePHP/Understanding-Model-View-Controller.html >. [citado el 04 de febrero 2017].

La figura 2, representa la arquitectura de aplicación del -MVC, que describe cómo interactúan las tres capas que conforman la arquitectura de desarrollo de software en el patrón - MVC, donde se presenta una clara separación entre las distintas responsabilidades de la aplicación web.

En la figura 2, las flechas representan cómo se da la interacción entre las diferentes capas de los distintos elementos que componen una aplicación - MVC, junto con el usuario. Los controladores, con su lógica de negocio, hacen vínculo entre la capa de modelo y la capa de vistas. Pero además en algunos casos los modelos pueden enviar datos a las vistas. A continuación, a modo de ejemplo, se realiza un análisis cómo interactúan las capas del - MVC. Como primera acción, el usuario accede a un sitio web la solicitud que hace el usuario llega a la capa de controlador, este hace una comunicación tanto con la capa de modelos como la de vistas; a la capa de modelos se le hace una solicitud o actualización de datos, en cambio a la capa de vistas se le da la salida pertinente, una vez se hayan realizado las operaciones correspondientes según la lógica del negocio. Para realizar la salida, en ocasiones las vistas pueden solicitar más información a los modelos.

Figura 2. Diagrama modelo vista controlador



Fuente: CAKE SOFTWARE FOUNDATION. Entendiendo Modelo-Vista-Controlador [En línea]. < https://book.cakephp.org/2.0/es/_images/basic_mvc.png>

• **Presentación-abstracción-control.** Para Buschmann et al.[31] PAC es un patrón de arquitectura de software que define una estructura para sistemas interactivos, este se basa en una jerarquía de agentes cooperantes en la que, al igual que el modelo vista controlador - MVC, se divide en un sistema interactivo, donde cada agente es responsable de algún aspecto de la funcionalidad, está compuesto por tres componentes o capas: presentación, abstracción y control. Esta subdivisión separa los aspectos de interacción con el usuario-sistema del agente de su funcionalidad principal y de la comunicación con otros agentes. A continuación, se analiza las tres capas del patrón - PAC:

La primera capa es presentación, es la comunicación visual del agente es decir brinda la información y la lógica de operaciones en un formato previamente asignado, que corresponda con el tipo de información, por ejemplo, audio y video entre otros, que sea adecuado para interactuar. Además, es la combinación de las vistas y modelos que corresponden a las componentes del - MVC. La segunda capa es abstracción, esta capa es la encarga de procesar los datos y provee operaciones de funcionalidad, además, corresponde a la componente modelo en el - MVC. La tercera capa es control, esta conecta los componentes de presentación y abstracción de cada agente, se encarga de asuntos como el flujo de control y de la comunicación entre las otras dos componentes y además le agrega la funcionalidad necesaria para comunicarse con otros agentes. Todos los agentes se comunican sólo a través de este tipo de componente, esta capa no tiene equivalente en el - MVC.

Por otra parte, para dar una descripción sobre los agentes en el contexto de presentación-abstracción-control los cuales según Catalini[32], son unidades básicas de funcionalidad que procesan información y la comparten con otros agentes. Para realizar tal procesamiento, las responsabilidades del agente deben ser semánticamente coherentes desde el momento en que el agente no requiera interactuar con otros para procesar la información, sino sólo para la recepción y transmisión de los datos. Cada agente es responsable de encapsular las implementaciones privadas que posea. Existe cierto escalamiento de niveles de abstracción en este patrón, dado el simple hecho de tratarse de un sistema jerárquico de distribución de agentes (similar a lo que ocurre con el patrón en capas). Los niveles superiores en la jerarquía suponen un nivel de abstracción

[31] BUSCHMANN, Frank; MEUNIER, Regine; ROHNERT, Hans; SOMMERLAD, Peter Y STAL, Michael. Pattern-oriented software architecture: A System of Patterns. Vol 1: WILEY, John & Sons. 1996. p. 145-168. ISBN 0-471-95869-7.

[32] Catalini, E. Arquitectura Modelo/Vista/Controlador. [en línea]. <https://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador/>. [citado el 04 de febrero de 2017].

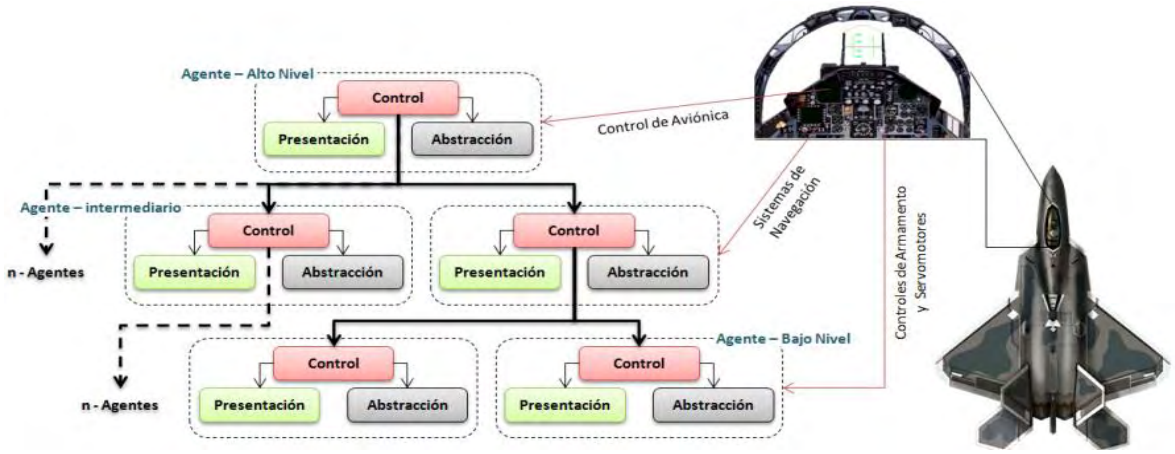
mayor correspondiente a los agentes ubicados en niveles inferiores. PAC es un modelo jerárquico en el cual cada capa agente en cierto nivel de la jerarquía actúa como intermediario entre su propio agente padre e hijo. PAC organiza la estructura del sistema, jerárquicamente, según 3 niveles de agentes: alto nivel, intermedio y bajo nivel. A continuación, se analiza los tres niveles:

Como primer nivel se encuentran los agentes de alto nivel, estos proveen el núcleo de la funcionalidad del sistema. Este tipo de agente es quien coordina y estructura la funcionalidad del sistema como, por ejemplo, un menú, barras de herramientas, entre otras. Como segundo se encuentran los agentes intermedios, los cuales representan combinaciones o relaciones entre agentes de bajo nivel, por ejemplo, este tipo de agentes puede representar diversas vistas sobre los datos. Finalmente se encuentran los agentes de bajo nivel, representan conceptos auto contenidos, sobre los cuales los usuarios del sistema actúan. Típicamente estos agentes soportan las operaciones de los usuarios.

A cada agente se le asigna un área específica como, por ejemplo, para algunos agentes como son los de nivel medio estos no requieren de las presentaciones interactivas, por lo tanto, no tienen componente de presentación, en cambio la componente de control se la requiere para todos los agentes es por esto que a través de esta le sirve de puente para la comunicación entre sí.

A modo de ejemplo, en la figura 3, se muestra la implementación de este patrón, como también los niveles de abstracción:

Figura 3. Implementación del modelo PAC



Fuente. (Gaitán ,2014)

Este modelo se puede aplicar para las siguientes aplicaciones como primer ejemplo, es seguro para un sistema distribuido donde todos los agentes están remotamente distribuidos y donde cada uno de ellos le corresponde funcionalidades con los datos y la interfaz. Como segundo ejemplo, es oportuno para aplicaciones con componentes GUI, donde cada uno de ellos mantiene su propia interfaz de datos actuales e interactivos que requieren comunicarse con distintos componentes.

• **Modelo de Seeheim.** Este modelo, es el resultado de un taller que trató sobre herramientas para software de interfaces, realizado en Alemania en una ciudad llamada con el mismo nombre Seeheim, en el año de 1983[33]. Este modelo describe la interfaz de usuario como la capa exterior del sistema además afirma que la comunicación entre hombre-máquina se compone en tres niveles basado en aspectos lingüísticos.

El primer nivel es el de presentación, él es la parte estática y visible de la interfaz, este controla la entrada y salida que representan los datos e interpreta las acciones del usuario, además gestiona el aspecto semántico por medio de una capa de adaptación, este se construye sobre sistemas de ventanas y cajas de herramientas, como ejemplo, un léxico de una interfaz.

El segundo nivel es el de gestión de diálogo, este maneja los eventos o mensajes, así como el canal de entrada y salida de información que se realizan como resultado de las acciones de interacción del usuario con la interfaz, además, constituye un enlace de comunicación entre el nivel de presentación y de aplicación, este se asemeja a la sintaxis de comunicación entre los otros dos niveles; procesamiento sintáctico de componente léxico.

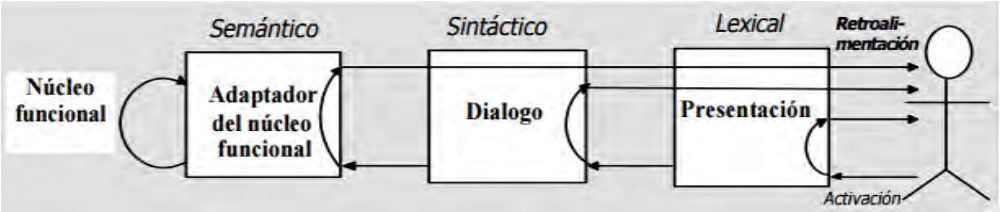
Finalmente, el tercer nivel es la interfaz de comunicación esta determina la interfaz entre la componente interactiva y el resto del software provee la retroalimentación semántica para el chequeo de la validez de los inputs, equivaldría a la semántica de la aplicación ya que gestiona el aspecto sintáctico del sistema.

La división en capas facilita el tratamiento de cada una por separado y además promueve la reutilización y la portabilidad, con lo que se posibilita el desarrollo rápido de prototipos, tarea fundamental para un diseñador. En la figura 4, se describe los tres niveles del modelo Seeheim:

[33] ABASCAL, Julio; AEDO, Ignacio; CAÑAS, José; GEA, Miguel; GIL, Ana; LORE, Jesús; MARTÍNEZ; Ana, ORTEGA, Manuel; VALERO, Pedro y VELEZ, Manuel. La interacción persona ordenador. Primera edición. Lleida: LORES, Jesús. 2001. 20 p.

En la figura 4, se describe el resultado de interacción de los tres niveles que muestran la retroalimentación de un sistema de información. En 1983, una de las principales preocupaciones fue la sustitución de las interfaces hombre-máquina de texto con las interfaces gráficas. El modelo Seeheim, por lo tanto, es el proyecto de organización del software que se utiliza para cambiar las interfaces sin cambiar los núcleos funcionales.

Figura 4. Niveles del modelo Seeheim



Fuente: (Abascal, 2001)
[En línea]. http://ccc.inaoep.mx/~grodrig/ /InteraPatternToCIC.pdf

• **Modelo Arch/Slinky.** Este es un patrón de arquitectura de software, para aplicaciones interactivas este modelo tiene una descomposición similar al modelo de Seeheim, pero este adiciona una serie de mejoras como por ejemplo: una identificación más clara del nivel de abstracción de cada componente; una definición explícita de las estructuras de datos intercambiado entre los componentes; los adaptadores los cuales mejoran la modificabilidad y portabilidad; y el meta-modelo Slinky para equilibrar la asignación de funciones a través del sistema; todas estas mejoras incluyen una refinamiento en el nivel de abstracción de cada componente, además de una definición más concreta de las estructuras de los datos intercambiados entre sus componentes principales.

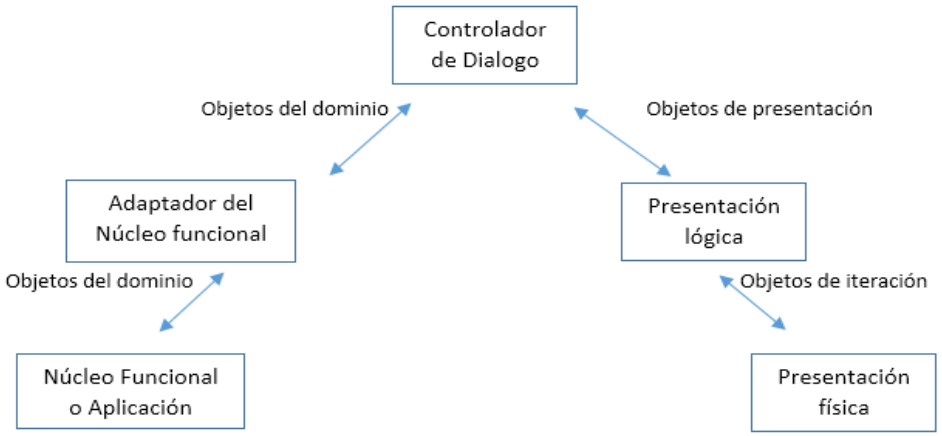
Según Stephen[34], el modelo Arch/Slinky establece que una aplicación interactiva consta de cinco componentes los cuales son:

- Controlador de diálogo.
- Adaptador del núcleo funcional.
- Núcleo funcional o aplicación.
- Presentación lógica.
- Presentación física.

[34] Stephen, Albin. The Art of Software Architecture: Design Methods and Techniques. 2007. p 224-226.ISBN 0471228869.

En la figura 5, se puede observar la arquitectura del modelo Arch/Slinky.

Figura 5. Arquitectura del modelo Arch/Slinky



En la figura 5, las flechas indican el flujo de datos entre las componentes y las cajas corresponden a las componentes funcionales.

Por otra parte, en el modelo Arch/Slinky, los principales componentes funcionales de un sistema interactivo, es decir, la aplicación, el diálogo y la presentación, no intercambian datos directamente, sino que se lleva a cabo el intercambio principal de datos a través de dos adaptadores los cuales son; adaptador del núcleo funcional y la lógica de presentación. A continuación, se analiza los adaptadores de intercambio de datos ya que son de gran prioridad para este modelo:

El adaptador del núcleo funcional, este es el encargado de adaptarse a diversas formas de incoherencia que se da entre los datos de la aplicación y la interfaz de usuario, además es una capa media entre el núcleo funcional y el controlador de diálogo. El objetivo es gestionar las diferencias entre el núcleo y la interfaz funcional. Tal como se muestra en la figura 5, las transferencias de los datos se hacen a través del adaptador del núcleo funcional el cual se lleva a cabo en términos de objetos de dominio los cuales son estructuras de datos de alto nivel, por ejemplo, un número real para modelar la noción de calor, en el contexto un objeto de dominio es una entidad que el diseñador del sistema, quiere hacer perceptible para y manipulable por el usuario. En muchos de los casos, los objetos específicos de dominio pueden ser implementados de manera inapropiada para los usuarios, por lo que estos objetos necesitan ser adaptados.

El segundo adaptador es la presentación lógica, la cual aísla la prestación de los objetos de dominio del conjunto de herramientas de interacción representadas por las facilidades provistas por la plataforma subyacente. Este adaptador se expresa en términos de los objetos de presentación lógicos que son proporcionados por un conjunto de herramientas virtuales, de esta manera, el cambio a un diferente conjunto de herramientas de interacción física requiere la reestructuración de las de reglas de asignación, pero los objetos de presentación lógicos permanecen sin cambios. Kit de herramientas de ventana o AWT por sus siglas en inglés y kit de herramientas extensibles o XVT por sus siglas en inglés estas son ejemplos de conjuntos de herramientas virtuales.

Como se describió anteriormente el objetivo principal de estos adaptadores es garantizar la independencia de las otras componentes. Como se muestra en la figura 5, en un lado del modelo, la componente de aplicación comprende el dominio el cual depende de conceptos y funciones además esta contiene los datos y el procesamiento manejados como por ejemplos, el texto y el diseño en un procesador de textos, y en el otro lado está la componente de presentación la cual gestiona los conceptos de dominio además incluye los dispositivos de interacción hardware y la interacción de los componentes de software, como por ejemplo, ratones y cuadros de diálogo. El componente de diálogo juega un papel importante en la gestión de la secuencia de tareas, este recibe peticiones sin conocer la forma en que se produjeron, selecciona qué datos del núcleo funcional que se representarán en un momento dado.

3.1.1.2 Diseño de interfaces de usuario. Cada vez está adquiriendo mayor importancia al momento diseñar las interfaces de las aplicaciones informáticas más fáciles de usar. Los avances tecnológicos han cambiado la manera como las personas interactúan con los sistemas informáticos, eso justifica que el diseño de interfaces juegue un papel fundamental en el desarrollo de aplicaciones web.

El diseño de interfaces es un proceso iterativo que permite juntar la tecnología y los usuarios en la interfaz, su papel es decisivo en la percepción de calidad de un sitio web, que involucra los aspectos visuales y estéticos que influyen en la confianza y la credibilidad de un sitio web. Para cualquier organización, el diseño de su sitio web es un elemento esencial de su identidad institucional.

Por otra parte, el diseño de la interfaz de usuario se define como la apariencia y el comportamiento de un sistema de información o como la herramienta con la que el usuario interactúa directamente con el ordenador.

La interfaz es el recurso mediante el cual se lleva a cabo la interacción entre usuario y la aplicación, además incluye todos los dispositivos que ponen en contacto al hombre con la máquina[35]. Existen dos tipos de dispositivos de interfaz. El primero son los dispositivos de entrada, es decir, aquellos que permiten al hombre introducir datos en el ordenador como, por ejemplo, ratón, teclado entre otros dispositivos. El segundo son los dispositivos de manipulación, es decir, los elementos que existen en un programa determinado, y que se utilizan para alterar el estado de la información y navegar por la aplicación.

Por otro lado, cuando se emplea el término Look and Feel, una expresión inglesa que tiene muchos significados esto depende generalmente del contexto donde se utilice como, por ejemplo, puede ser en la indumentaria que significa textualmente el aspecto y el tacto que hace referencia a características como el color y la textura, en cuanto al diseño gráfico características del color, tamaño y tipo de letra de un documento. Algunos expertos en usabilidad describen al Look and Feel como la apariencia una capa decorativa de aplicación, como se van a presentar los diferentes componentes como botones, campos de texto, combos, pestañas, texto entre otros, con el fin de buscar dar otras opciones visuales. El diseño y la usabilidad pueden y deben complementarse como disciplinas. Las ventajas pueden verse claramente desde ambos lados**.** Un buen diseño visual no excluye a la usabilidad, sino que la favorece, del mismo modo que un alto grado de usabilidad y el respeto por los usuarios contribuyen a mejorar la imagen de cualquier empresa o institución[36].

Varias aportaciones que se han propuesto para poder implementar esta táctica en aplicaciones web y proyectos software con la usabilidad de los mismos como objetivo principal, por esta razón se integra diferentes aportaciones que se han hecho en la literatura existente, con el objetivo de conseguir documentación más completa, más orientada a la usabilidad y más inteligible. A continuación, se presentan aquellas que se consideró más relevantes con el fin de crear diseños usables que además agreguen un buen diseño visual un ‘Look and Feel’ para la interfaz del usuario y garantizar su consistencia a través de toda la aplicación y que han servido de referencia importante para esta investigación:

[35] HASSAN, Y. Introducción a la usabilidad. En: No Solo Usabilidad, noviembre 1, 2002.Disponible en<http://www.nosolousabilidad.com/articulos/introduccion_usabilidad.htm>. [citado el 06 de junio de 2016].

[36]Lovera, Eduardo. Web taller. En: Diseño y usabilidad. [en línea]. <http://www.webtaller.com/maletin/articulos/como-hacer-copias-seguridad.php/diseno_y_usabilidad.php> [citado el 03 de mayo de 2016].

• **Trabajo de Perzel y Kane**. El trabajo propuesto por los autores, describe cómo ha ido evolucionando el mundo de World Wide, y cómo esto ha cambiado radicalmente el Look and Feel de las interfaces gráficas de usuario[37]. Agregan además que se ha generado nuevos problemas de usabilidad introducidos por el uso de la web, y aunque no especifican cuales son esos problemas si señalan que existen millones de sitios web y solamente miles de ingenieros de usabilidad y diseñadores de UI señalan además, que se han propuesto patrones que buscan ser directrices útiles a fin que les sirva a los desarrolladores para mejorar la usabilidad de los sitios web, lo que implica satisfacer las expectativas de los usuarios y mejora su productividad y satisfacción y facilidad de uso. Estos patrones documentan las implicaciones sobre el factor humano de elementos de IU de uso frecuente de aplicaciones web.

Los autores exponen una colección inicial de este tipo de patrones. Además, destacan la diferencia que existe entre un sitio web y una aplicación web. Un sitio web requiere básicamente que el usuario envíe información de navegación, por ejemplo, los sitios web de muchos periódicos online, mientras que una aplicación requiere una gama mucha más amplia de interacción con el usuario, por ejemplo, un sitio de comercio electrónico, servicios de mainframe, cartelera de anuncios entre otros. Aunque cabe resaltar que algunas páginas web combinan los elementos tanto de un sitio web y una aplicación web, por ejemplo, una colección de páginas de contenido combinado con una función de registro. Cabe mencionar que los patrones que proponen los autores se entienden que se centran en el contexto de aplicaciones web.

Por otra parte, los autores definen que los patrones propuestos se pueden clasificar como diseño de interfaces o basados en el sistema, estos patrones mejoran los atributos de usabilidad. Para que, por ejemplo, la aplicación web se pueda aprender, sea fácil de recordar eficiente fiable y flexible y automatizada, comprensible y subjetivamente satisfactoria. Los autores admiten que el tema de usabilidad web es bastante amplio, digno de todo un lenguaje de patrones, pero esperan que los patrones propuestos constituyan un primer paso útil. Los patrones propuestos en este trabajo, son:

- La zanahoria y un palo.
- Declaración de política.
- Marcadores de campos obligatorios.
- Lo que se ve es lo que se obtiene.
- Plan b.

[37] Perzel, K., Kane, D. Usability Patterns for Applications on the World Wide Web.1999. PloP 99 Conference.

A continuación, se hace un análisis detallado de los patrones anteriormente mencionados:

Figura 6. La zanahoria y el palo

Fuente. (Perzel et al., 1999).

o **La zanahoria y un palo**, La figura 6, representa gráficamente este patrón de llegar al usuario brindándole una pequeña parte de zanahoria, para llegar a los usuarios finales los cuales se rehúsan a suministrar información. El desarrollador de una aplicación de un sitio web utiliza formas para recopilar información del usuario, estas formas contienen campos, por ejemplo, cajas de texto, las cuales requieren entradas del usuario con el fin de poder avanzar en la aplicación y obtener algún valor particular requerido, algunos de estos campos son para enviar información personal. Los autores clasifican este patrón como basado en el sistema.
Este patrón propone una solución que determine lo que los usuarios consideren que es una zanahoria algo valiosa para ellos. Para eso se propone las siguientes pautas:

- ✓ Ofrecer al usuario final una parte de esa zanahoria antes de solicitar información personal.
- ✓ El contenido debe de ser retenido "el palo" hasta que la información solicitada sea prevista por parte del usuario.
- ✓ Cuando se solicita la información personal, explicar exactamente cómo se utilizará dicha información.
- ✓ Limitar la cantidad de información sensible reunir lo más útil, como, por ejemplo, en muchos casos, esta será la dirección de correo electrónico y código postal.
- ✓ Después de haber reunido la información clave, a continuación, conceder al usuario el acceso al resto de la zanahoria.
- ✓ Si los usuarios perciben que van a recibir algo de valor para su información, a continuación, estos estarán dispuestos a compartir o brindar información. Para muchas aplicaciones web, la zanahoria del sitio tiene que ofrecer es la información. Sin embargo, a diferencia de un mercado tradicional, es difícil para

el usuario inspeccionar información antes de decidir “SI” para intercambiar información personal con el contenido del sitio. Es por eso que es importante ofrecer algo de valor primero de forma gratuita para dar motivación al usuario, con el fin de establecer una base para el intercambio de información y la creación de una relación con el usuario. A si el usuario puede determinar si un intercambio de este tipo es ventajoso para él.

o **Declaración de política**, este patrón proporciona una solución al problema para establecer la suficiente confianza con los usuarios para que puedan proporcionar información personal. Si el desarrollador de la construcción de una aplicación web utiliza formularios para recopilar información el usuario podría considerar dicha información como determinante y confidencial para él, como, por ejemplo, la dirección de correo electrónico, teléfono, número, edad, sexo entre otros datos. Los autores clasifican este patrón como basado en el sistema.

La solución que propone este patrón, es utilizar una declaración detallada de cómo utilizar la información suministrada por el usuario. Escribir una declaración de modo que sea consistente con sus prácticas reales es decir garantizar la capacidad de confidencialidad de información presentada, esto conlleva a que se debe asegurar de que se comunique en la declaración de política cómo se va a utilizar la información del usuario. Colocar estas declaraciones en un lugar prominente en los formularios constituye una buena política de comunicado, esta varía en función del contexto, pero en muchos casos da a los usuarios la opción de solicitar que la información no pueda ser compartida con otras empresas o sitios. Otras dan a los usuarios la capacidad de controlar si recibieron solicitaciones como resultado de compartir su información. A sí mismo, el centro para la democracia y la tecnología han publicado algunas directrices para la evaluación de políticas de privacidad.

Razón fundamental por la que se debe implementar una declaración de políticas es que a menudo los usuarios son sensibles a la protección de su privacidad en línea, y en particular estos están preocupados por los datos que son utilizados por un sitio y en ocasiones son vendidos a terceros. Sin embargo, también se entiende y se aprecia que existen razones legítimas de negocios para los sitios web para recoger datos acerca de sus usuarios. De acuerdo con una encuesta, más del 72% de los usuarios dijeron que compartirán datos con un sitio si sólo los sitios proporcionen una declaración acerca de cómo se utilizará la información[38].

[38] HOFFMAN, Donna; NOVAK, Thomas; PERALT, Marcos. Building consumer trust in Online Environments. En: The case for information privacy. 1999. p. 80-85.

Si un sitio provee una declaración de política que aborde inquietudes de los usuarios sobre la protección de su privacidad de su información, esto genera a que el sitio aumente el número de usuarios a compartir su información.

o **Marcadores de campos obligatorios**, este patrón proporciona una solución al problema de cómo asegurar de que el usuario final suministre información esencial para el uso de una aplicación web. Si el desarrollador de la construcción de una aplicación web utiliza formularios para recopilar información del usuario. Los formularios contienen ciertos campos que requieren entradas del usuario con el fin de avanzar y obtener el valor de la aplicación, mientras que otro puede ser opcional. Los autores clasifican este patrón como diseño de interfaces. La solución que propone este patrón es marcar de una forma clara los campos que los usuarios están obligados a brindar información, con el fin de operar el sitio con eficacia. Para esto se debe seguir las siguientes pautas:

- ✓ Se debe asegurar que todos los campos sean realmente necesarios.
- ✓ Los campos necesarios se deben marcar utilizando varios enfoques tales como utilizar un icono gráfico, por ejemplo, un asterisco, un identificador verbal como puede ser la palabra “necesario”.
- ✓ Los campos requeridos se los debe agrupar en una sección específica del formulario.
- ✓ La mejor opción es adicionar un elemento gráfico si hay un número significativo de los campos obligatorios.
- ✓ Los elementos gráficos no deben basarse exclusivamente en el color o en la familia de fuentes.
- ✓ La parte superior del formulario es clave para explicar la notación de campos requeridos.
- ✓ Si hay sólo uno o unos pocos de estos campos requeridos, es acorde agregar el término "necesario" a la etiqueta.
- ✓ Razón fundamental por la que se debe implementar marcadores de campos obligatorios es porque se asegura de que la información sea completa antes de que se envíe el formulario. Esto puede ahorrar tiempo al usuario y la reanudación potencial del formulario. Por otra parte, se debe resaltar que este patrón no evitará que el usuario envíe los datos erróneos para eludir requerido.

o **Lo que se ve es lo que se obtiene,** este patrón proporciona una solución al problema de cómo asegurar que el usuario vea todo lo que él necesita ver en un formulario web. En la construcción de sitios web en el que se utilizan formularios para recopilar información del usuario. Estos formularios requieren la entrada del usuario con el fin de avanzar y obtener el valor de la aplicación. Existen muchos campos en los formularios. Los autores clasifican este patrón como diseño de interfaces.

La solución que propone este patrón, es que se debe diseñar el contenido de la página web para mostrar los elementos más críticos en la parte superior izquierda, los elementos críticos son los campos de entrada y botones de comando que se necesita que el usuario seleccione para continuar utilizando la aplicación. También son críticos las pistas visuales que se comunican cuando hay más información a la parte inferior o el derecho de la pantalla. Para implementar esta solución se debe seguir las siguientes pautas:

- ✓ Mantener toda la información importante dentro de los parámetros de 535 píxeles de ancho y 295 píxeles de largo.
- ✓ Si el formulario requiere que el usuario utilice el scroll, se debe proporcionar pistas visuales que estimulen al usuario desplazarse hasta encontrar un pulsador.
- ✓ Consistentemente se debe presentar los botones como enviar “submit” en el mismo lugar con el fin de que los usuarios lo puedan localizar fácilmente en un formulario.
- ✓ Limitar las formas de consulta a una pantalla.
- ✓ Permitir scroll a los formularios de entrada de datos.
- ✓ Limitar el número de campos críticos en el formulario para evitar fatigar al usuario.

o **Plan b,** este patrón proporciona una solución al problema de cómo se debe apoyar a los usuarios que no son capaces de percibir gráficos de páginas web. Si el desarrollador de la construcción de una aplicación web tiene previsto utilizar gráficos, iconos y la tipografía para mejorar el sitio. La solución que propone este patrón es que se debe crear una alternativa para comunicar información sin gráficos siempre que la información se comunica de forma gráfica en una página. Esta solución puede adoptar la de incluir la etiqueta alt de HTML para especificar un texto equivalente.

El objetivo de la propiedad alt, es facilitar una descripción de los contenidos de un archivo de imagen; proporcionar texto para los usuarios que no pueden ver imágenes en sus navegadores es uno de los usos más citados de la propiedad alt. Esto incluye a los usuarios que utilizan navegadores que no pueden presentar imágenes o tienen vista de imágenes con discapacidad, a los usuarios con discapacidad visual, y aquellos usuarios que emplean lectores de pantalla. Si un usuario puede ver imágenes, los atributos alt también se mostrará cuando el usuario se desplaza sobre su imagen[39]. Google recomienda que se haga lo más útil posible para los usuarios finales.

[39] SEVENMARKETING. ¿Porque es importante el atributo ALT en etiqueta de imágenes? 2014. Disponible en <http://www.sevenmarketing.pe/por-que-e>. [citado 31 de junio de 2016].

Siguiendo las recomendaciones de Google, el cual es el motor de búsqueda más grande y más usado en la actualidad, se toma sus recomendaciones. A continuación, se presentan algunas recomendaciones para la creación de propiedades eficaces alt:

- ✓ Mantenlo simple.
- ✓ Considere el texto que rodea la imagen.
- ✓ Utilice palabras claves para ayudar a su ranking en búsquedas.

• **Trabajo de Hernández, Álvarez y Arteaga.** En este trabajo los autores proponen el uso de patrones de interacción para el diseño de interfaces de usuario para la web, debido a que estos patrones se han usado en las buenas prácticas de diseño además estos especifican claramente la forma de implementación y cómo establecerlos en el contexto en el que pueden ser aplicados e incluso las consecuencias de su uso, sin descuidar que estos patrones de interacción son clave porque se basan sobre principios de usabilidad[40].

Los autores presentan investigaciones basadas en patrones de interacción para la web, como los trabajos de Welie, Tidwell y Muñoz, en donde estos trabajos presentan un catálogo de patrones que han sido basados por las buenas prácticas de programadores y diseñadores expertos en el desarrollo de interfaces usables, donde se centran en las buenas prácticas de una serie de guías o recomendaciones, que puedan ser usadas por los desarrolladores novatos con el fin de que tengan la habilidad de diseñar interfaces usables.

Los autores definen a los patrones de interacción, como soluciones prácticas a problemas frecuentes, así mismo, promueven la reutilización de los buenos diseños, disminuyen el tiempo cuando un diseñador está adquiriendo experiencia, son excelentes opciones para el diseño de interfaces usables, en virtud de que se definen a partir de los criterios de usabilidad. Cabe resaltar, que presentan una dificultad tal razón es, porque no se cuenta con un catálogo de patrones universalmente aceptada, esto puede llegar a ser una la lista inimaginablemente grande de patrones en tanto no se formalice su producción.

Por otra parte, los autores no proponen patrones como tal, sino que mencionan a modo de ejemplo, la estructura de patrones de otras investigaciones. Por lo que este

[40] HERNANDEZ, Helena; ALVAREZ, Guillermo, ARTEAGA, Muñoz. Patrones de interacción para el diseño de Interfaces web usables. Puebla México. 2003. Instituto Nacional de Astrofísica Óptica y Electrónica.

trabajo se lo ha tomado como análisis recopilatorio sobre patrones interacción persona ordenador que involucra definiciones y usos. Al ser un trabajo recopilatorio, se puede deducir que es una investigación empírica deductiva, así mismo, cabe resaltar que este trabajo muestra cómo la usabilidad es un factor fundamental en el diseño arquitectónico.

- **Trabajo de Seffah y Taleb**. En esta propuesta planteada por los autores, se realiza un análisis acerca de cómo pueden ser utilizados los patrones IPO (interacción persona ordenador), como artefactos en el proceso de diseño[41]. También se plantean temas como el proceso de diseño centrado en el usuario, el diseño dirigido por patrones y el lenguaje de patrones así mismo, se realiza una comparación de las diferentes propuestas establecidas por los patrones de interacción que se ha utilizado para transmitir las buenas prácticas de diseño como, por ejemplo, guías, directrices, declaraciones y patrones.

Los autores presentan investigaciones basadas en patrones, y describen cómo la experiencia del usuario puede ser incluida en el proceso de selección de patrones a través del uso de variables de comportamiento, atributos de patrones y relaciones asociadas, también en su investigación incluyen ejemplos de cómo se puede aplicar a los patrones orientados al diseño por sus siglas en inglés POD, estos emplean un lenguaje de patrones para aplicaciones web que muestre las relaciones existentes, los autores también señalan el lenguaje UPDATE[42], el cual clasifica los patrones en tres categorías que corresponde con varios pasos que se lleva a cabo en el proceso de diseño de aplicaciones web: arquitectura, estructura y soporte a la navegación.

Los autores hacen una síntesis acerca del uso de los patrones en otras dos aproximaciones de diseño las cuales son: el diseño basado en componentes y el diseño dirigido por modelos. Se resalta que estas dos aproximaciones reducen la abertura entre el diseño y la implementación de los sistemas de software. En este trabajo no se plantea patrones como tal, por lo tanto, no se puede asumir recomendación IPO, por tal razón se asigna a este criterio el valor de carencia.

También se indica que esta propuesta está orientada hacia el diseño, por lo tanto, se puede decir que se centra en el diseño detallado y no se necesitaría código

[41] SEFFAH, Ahmed; TALEB, Mohamed. Tracing the evolution of HCI patterns as an interaction design tool. London.2 de diciembre de 2011.p 1-18.

[42] Seffah y Taleb. Entorno de diseño asistido por patrón de usabilidad (UPDATE) es lenguaje Web Y el ambiente de diseño desarrollado por la Universidad Concordia. Grupo de ingeniería de software centrado en el ser humano.2008

asociado, este trabajo es de tipo recopilatorio y de análisis de definiciones y usos de patrones y lenguaje de patrones.

• **Trabajo de Welie.** Presenta un catálogo de patrones de diseño publicado en su sitio web. Este catálogo es una colección que contiene una gran cantidad de las mejores prácticas en diseño de interacción, los cuales están centrados en resolver los problemas de usabilidad que los usuarios tienen con un sitio web. Además, agrega ejemplos propios de otros autores sobre la aplicabilidad de los patrones y aclara que la solución puede ser exitosa pero también puede fallar en otros contextos. Welie, aclara que su trabajo se puede ver como un punto de referencia o un kit de herramientas para el diseño de experiencias de usuario, en la página web también se encuentran enlaces de otras librerías de patrones como, por ejemplo, Tidwell, Yahoo! entre otros[43].

Welie[44], define los patrones como tareas relacionadas que están categorizadas según el tipo de problema de usabilidad que solucionan este propósito se logra con tres categorías. La primera, son las necesidades de los usuarios en la cual se agrupan los patrones que satisfacen una necesidad directa del usuario. La segunda categoría son las necesidades de las aplicaciones la cual agrupa patrones que ayudan a la aplicación o el diseñador, para interactuar mejor con el usuario. Finalmente, la última categoría está relacionada con el contexto del diseño.

Cada uno de los patrones que propone mejora al menos uno de los atributos de usabilidad tales como facilidad de aprendizaje, facilidad de memorización, velocidad de rendimiento, ratio de errores, satisfacción y grado de completitud de tareas[45]. Esta clasificación va desde aspectos específicos aplicables a muchos sistemas, por ejemplo, navegaciones, interacciones, básicas, búsquedas, manejo de datos, personalización, compras, toma de decisiones, entradas de formularios, mensajes de alerta, realimentación, simplificación de interacción, experiencias hasta clasificaciones muy generales tales como tipos de páginas o tipos de sitios.

Por otra parte, el autor no indica el método utilizado en la recopilación de patrones, sino que se especifica que durante años se ha recogido ejemplos y conocimiento sobre su aplicabilidad los cuales son una guía de diseño para que los sitios web sean usables.

---

[43] RODRÍGUEZ, Francy. Obtención y uso de patrones para la implementación de funcionalidades de usabilidad en aplicaciones web. Tesis de doctorado. Universidad Politécnica de Madrid.

[44] WELIE, M.A Pattern Library for Interaction Design.2008. [en línea]. <http://www.welie.com/> [citado el 30 de agosto de 2016].

[45] PANACH, José. Incorporación de Mecanismos de Usabilidad en un Entorno de Producción de Software Dirigido por Modelos. Tesis doctoral). Universidad Politécnica de Valencia. 40 p.

• **Trabajo de Tidwell.** El autor ha propuesto un lenguaje completo de patrones de interacción para el diseño de interfaces usables para la web, debido a que los patrones especifican claramente la forma de implementación, el contexto en el que pueden ser aplicados e incluso las consecuencias de su uso, sin olvidar que los patrones de interacción se rigen sobre principios de usabilidad. Por otra parte, aclara que los patrones de diseño de interfaces, están destinados a ser utilizados como, ejemplo, en las personas que diseñan interfaces de usuario tradicionales, sitios web, documentación en línea, juegos de vídeo, en otras. Además, el autor señala que ha desarrollado este trabajo de patrones por más de diez años publicados en el sitio web y en el libro (Designing Interfaces) del cual tiene dos ediciones[46].

Según el autor, los patrones de interacción contribuyen en el momento de diseñar un modelo conceptual que hay de detrás de la interfaz. El objetivo previsto de este lenguaje de patrones es deliberadamente amplio para apoyar la interacción de alta calidad entre una persona y un artefacto de software, Tidwell estructura los patrones conforme a los diferentes aspectos del diseño de las interfaces de usuario, algunas de las categorías son: organización de contenido, mecanismos de navegación, disposición de páginas, listas y tablas, acciones y comandos, formularios y controles, diseño móvil, editores y constructores.

Según el autor, estos patrones íntegramente se apoyan en recomendación de interacción persona ordenador, cabe resaltar que algunos de los patrones son bastante específicos para el diseño de las interfaces de usuario, por lo tanto, son usados para el diseño detallado, cada patrón se ha descrito en un formato determinado. La estructura que propone el autor con la que se presentan los patrones tiene la siguiente estructura:

- Nombre: es el nombre que describe el problema que aborda el patrón.
- Problema: descripción del problema.
- Ejemplos: muestra una breve descripción y ejemplos gráficos.
- Porque: explica por qué la solución propuesta es apropiada para el problema.
- Como: representa la parte de la solución.
- Cuando usar: describe el contexto en el que el patrón se puede utilizar.
- En otras librerías: se presentan diferentes enlaces a otros autores.

En algunos casos Tidwell, se apoya en otros patrones para poder describir la estructura del patrón. Cabe resaltar que el trabajo propuesto es una recopilación de

[46]Tidwell, Jennifer. Patrones de diseño de Interacción.2010. [en línea]. <http://www.mit.edu/~jtidwell/interaction_patterns.html>. [citado el 3 de octubre de 2016].

buenas prácticas por lo cual apropia un método empírico por lo que se basa en la lógica empírica.

• **Directrices de usabilidad para sitios web del estado colombiano.** Esta investigación propone seguir un diseño web centrado en el usuario el cual se caracteriza por asumir que todo el proceso de diseño del sitio web debe estar encaminado para el usuario, sus necesidades, características y objetivos por tal razón que cualquier aplicación web debe estar condicionada para la satisfacción del usuario final[47].

Esta investigación es el resultado de un análisis de diversos estudios sobre el campo de la usabilidad en sitios web que se ha realizado en varios países de América y Europa. Cada estudio dio como resultado ideas que tratan la usabilidad en cada directriz. Los autores resaltan que las directrices de Estados Unidos son las que más han influido en su investigación las cuales las puede encontrar en su sitio web, así mismo, este documento se ha sustentado en otras fuentes como son la Guía Web Proexport[48], esta es una entidad oficial colombiana que propuso sus propias directrices para sitios web, también en diferentes documentaciones que tratan la usabilidad y la experiencia de usuario, como Hassan, Nielsen, Morville, entre otros.

Estas directrices están agrupadas en las siguientes categorías; arquitectura de información, diseño de interfaz de usuario, diseño de interacción, búsqueda, pruebas de usabilidad y contenido. Cada directriz está estructurada de la siguiente forma:

- Número y título: el título describe el tema a tratar y el número identifica la directriz.
- Directriz: se comprende como la guía a seguir.
- Impacto: explica la importancia y el cumplimiento de la directriz.
- Evidencia: para el caso colombiano, una parte de las directrices tiene un elemento probatorio a través de evaluaciones con usuarios, realizadas por algunos expertos en usabilidad de Colombia.
- Roles: cada directriz tiene asociados los roles encargados de su cumplimiento.

[47] CARVAJAL, Mario; SAAB, Juan. Documento de análisis de prácticas y recomendaciones mundiales en Usabilidad.23 de agosto de 2010.disponible en:<http://programa.gobiernoenlinea.gov.co/apc-aa-files/5854534aee4eee4102f0bd5ca294791f/GEL108_CINTEL_Documento_de_analisis_de_practicas_y_recomendaciones_mundiales_Usabilidad.pdf>. [citado el 15 de septiembre de 2016].

[48] PROEXPORT. Guía Web Proexport 1.0. 2008. Disponible en:<http://www.brreg.no/elmer/elmer2-english.pdf>. [citado el 14 de octubre de 2016].

- Comentarios: se incluye ejemplos de investigaciones de buenas y malas prácticas y se brinda información detallada de cada directriz.
- Verificación: se determina el cumplimento o no de la matriz mediante una metodología sugerida por parte del evaluador.
- Recursos: esta es una parte es opcional en la cual se determina qué herramientas pueden verificar el cumplimiento de la directriz.
- Fuentes: referencia documental, como, por ejemplo, libros, investigaciones, blog, sitio web entre otros, donde se trata el tema.

Este estudio es fundamental para mejorar la calidad en los sitios web y, ante todo, para mejorar la experiencia para el usuario, porque brinda unos lineamientos para concebir un producto de calidad donde se satisfaga el atributo de usabilidad.

3.1.1.3 Reglas de oro de Nielsen. Nielsen[49], es considerado el gurú de la usabilidad en la web, el autor propone diez principios generales para el diseño de interacción, las cuales se las considera como “heurísticas", ya que son amplias reglas generales y no específicas directrices de usabilidad. A continuación, se describe las diez reglas de oro propuestas por Nielsen para concebir un sistema usable:

La primera regla es la visibilidad del estado del sistema, esta especifica que el sistema siempre debe mantener informados a los usuarios de lo que está ocurriendo y brindarle una respuesta en el menor tiempo posible. La segunda regla es la adecuación entre el sistema y el mundo real, esta especifica que el sistema debe utilizar el lenguaje del usuario, con expresiones y palabras que le resulten familiares, la información debe presentarse en un orden lógico y natural. La tercera regla es control y libertad del usuario, esta especifica que el usuario es quien debe controlar el sistema, el usuario puede, en cualquier momento, abortar una tarea, o deshacer una operación y volver al estado anterior. La cuarta regla es consistencia y estándares, esta especifica que los usuarios no tienen por qué saber que diferentes palabras, situaciones o acciones significan lo mismo. La quinta regla es prevención de errores, esta específica que es importante ayudar al usuario a que no caiga en un error. Los atajos de teclado pueden hacer más rápida la interacción para usuario las situaciones que causan más errores y modificar la interfaz para que no se produzcan estos errores. La sexta regla es reconocimiento en lugar de recordar, esta específica que se debe minimizar la carga de memoria del usuario haciendo visibles los objetos, las acciones y las opciones, el usuario no debe tener que recordar la información de una parte del diálogo a otra, además las

[49] NIELSEN, Jakob. 10 Heuristics for User Interface Design. 2011

instrucciones de uso del sistema deben ser visibles o fácilmente recuperables cuando sea apropiado. La séptima regla es flexibilidad y eficiencia en el uso, esta especifica que para los usuarios con experiencia para llevar a cabo las operaciones más rápidamente, por ejemplo, abreviaturas, teclas de función, haga doble clic en el ratón, la función de nuevo en los sistemas de hipertexto. También se utilizan para recuperar información que es una profundidad en el árbol de navegación de la interfaz principal. La octava regla es diseño estético y minimalista, esta especifica que los diálogos no deben contener información que sea irrelevante o raramente necesaria, cada unidad extra de información en un diálogo compite con las unidades relevantes de información y disminuye su visibilidad relativa. La novena regla es ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores, esta especifica que los mensajes de error se deben entregar en un lenguaje claro y simple, indicando en forma precisa el problema y sugerir una solución constructiva al problema. Finalmente, la décima regla es ayuda y documentación, a pesar de que es mejor si el sistema se puede utilizar sin documentación, puede ser necesario proporcionar ayuda y documentación. Cualquier información de este tipo debe ser fácil de buscar, centrada en la tarea del usuario, enumerar los pasos concretos que se deben llevar a cabo, y no ser demasiado grande.

Estas reglas son fundamentales para poder facilitar el uso de cualquier usuario, no solo dentro de los sitios web, sino en cualquier sistema de información.

3.1.1.4 Implementación de frameworks. Los patrones son de gran importancia para la creación de arquitecturas de software, estas definiciones resultan abstractos y en algunas ocasiones no es tan claro cómo para conectarlos con la tecnología que se usa para el desarrollo. El diseño de arquitecturas de software se basa en la toma de decisiones que involucran la selección de conceptos con el fin de satisfacer los requerimientos que influyen sobre la arquitectura. Dentro de estos conceptos se encuentran los frameworks. Es así que en esta investigación se presentan los frameworks, estos se complementan con los patrones y las tácticas los cuales son clave en la arquitectura, porque son elementos reutilizables de software que proveen funcionalidades genéricas enfocadas a resolver cuestiones recurrentes.

En el desarrollo de software un framework según Galindo y Camps[50], es una estructura conceptual y tecnológica que brinda un soporte a un proyecto de software, este incluye soporte de programas, bibliotecas y un lenguaje interpretado para ayudar a desarrollar y unir las diferentes componentes de un proyecto, estos representan una arquitectura de software que modela las relaciones generales de las entidades del dominio. En general, un framework se comprende como una

[50] GALINDO, M; CAMPS, R. Diseño e implementación de un marco de trabajo (framework) de presentación para aplicaciones JEE.2008.

estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Además, se considera como una aplicación genérica incompleta y configurable a la que se le puede añadir las últimas piezas para construir una aplicación concreta.
Los frameworks típicamente se basan en patrones y tácticas, como objetivo principal ofrecen una funcionalidad definida, auto contenida, siendo construidos usando patrones de diseño. La característica principal es su alta cohesión y bajo acoplamiento. En términos del desarrollo web un framework es un conjunto de componentes que integran un diseño reutilizable que facilita y agiliza el desarrollo de aplicaciones web tales como aplicaciones web dinámicas y servicios web. Los frameworks típicamente se basan en patrones y tácticas. Un ejemplo de ello es el framework Struts2, este es un nuevo framework de presentación para desarrollar aplicaciones web, este sigue las mejores prácticas y patrones de diseño como es el controlador del patrón modelo vista controlador - MVC, además dispone de una gran cantidad de plugin que añaden funcionalidades que le permiten la integración con otros frameworks como puede ser Spring y JSF.

En la tabla 2, se describe las características de los frameworks, los cuales la mayoría comparten las mismas características por tipo, entre las que destacan, son:

Tabla 2. Caracterización de los frameworks

Característica	**Descripción**
Autenticación	Incluye mecanismos para la identificación de usuarios mediante login y contraseñas que permite restringir el acceso y el tipo de permiso.
Controladores	Gestiona eventos, peticiones como una introducción de datos mediante un formulario o el acceso a una página son fácilmente adaptables a las necesidades del proyecto.
Acceso a los datos	Incluye las herramientas e interfaces como, por ejemplo, en archivos texto, lenguaje etiquetado que integra la base de datos.
Internacionalización	Permite incluir varios idiomas en el desarrollo de la aplicación.

Característica	Descripción
Abstracción de URLs y sesiones	No es necesario manipular directamente las URLs ni las sesiones, ya que el framework se encarga de manejarlas.

Existen muchos framework por tal razón es casi imposible cuantificarlos y su uso va de acuerdo a las necesidades de cada proyecto y al gusto del desarrollador.

3.1.2 Tácticas en tiempo de ejecución. Consisten en retroalimentar al usuario sobre lo que hace el sistema, además ayudan al usuario a realizar sus actividades fácilmente. Una vez el sistema se está ejecutando la usabilidad se mejora dando al usuario una respuesta en cuanto a lo que hace proporcionando al usuario la capacidad de emitir comandos basados en usabilidad, como ejemplo, en caso de que el usuario cometa un algún error en sus transacciones el sistema ofrecerá soporte en ellas a través de botones de confirmar, agregar y cancelar, operaciones que el usuario podrá utilizar en el momento que lo necesite o mostrar múltiples vistas ya sea en la corrección de errores o de las operaciones más eficientes.

Por otra parte, cuando se toma los términos interacción hombre máquina se comprende que se intercambia información mediante un software y la persona. Los investigadores en la interacción hombre con máquina han usado los términos iniciativa del sistema, iniciativa del usuario e iniciativa mixta con el fin de poder comprender quién toma la iniciativa en la realización de ciertas acciones y el comportamiento de interacción.

Cuando se da el caso de quien toma la iniciativa es el usuario, se sugiere que el arquitecto de software debe diseñar algún tipo de respuesta del sistema, como si se tratase de una responsabilidad más del sistema, si por lo contrario si es el sistema quien toma la iniciativa resulta necesario proporcionarle al sistema distintos tipos de información para que pueda llevar a cabo su tarea. Esta información se comprende que se trata de modelos relativos al usuario, a la tarea que está realizando el usuario o al estado del sistema. Las tácticas que utiliza el sistema serán aquellas que le permitan identificar los modelos que el sistema necesitará para predecir las intenciones del usuario o su propio comportamiento. Cuando se emplea iniciativa mixta se entiende que existe combinación de iniciativas de ambas perspectivas, como, por ejemplo, cuando se ejecuta un comando cancelar se refiere a iniciativa de usuario, en ese momento el sistema puede poner un indicador de progreso que corresponde a iniciativa del sistema.

Las tácticas en tiempo de ejecución según Bass[51], las cuales se han considerado para esta investigación, son:

- Iniciativa del usuario.
- Iniciativa del sistema.

3.1.2.1 Tácticas de iniciativa del usuario. Si es el usuario quien toma la iniciativa, según Bass, el arquitecto debe diseñar una respuesta por parte del sistema como si se tratase como cualquier otra funcionalidad, cuando éste emita un comando. Además, el arquitecto debe enumerar las responsabilidades del sistema para responder al comando del usuario. A modo de ejemplo, se hace un análisis cuando el usuario presiona el botón de cancelar. El sistema debe estar escuchando, por lo tanto, se debe tener un oyente constante que no está bloqueado por las acciones de lo que sea cancelado. Para implementar esta táctica, se debe incorporar mecanismos de usabilidad como, cancelar deshacer y agregar. A continuación, se hace un análisis de los mecanismos mencionados.

• **Undo y cancel**. La usabilidad es un elemento esencial para el éxito general de un sitio web y de un sistema de software. Incluir características simples de usabilidad puede aumentar considerablemente la funcionalidad percibida del sistema por parte del usuario. La amplia aceptación de funcionalidades, como undo y cancel, muestra que tales características se han convertido en parte esencial de los sistemas interactivos, estos mecanismos añaden características de usabilidad que ayudan al usuario ya que otorgan la capacidad de ser tolerante a los errores[52]. Estas características funcionales de usabilidad tienen como objetivo permitir que un usuario pueda cancelar o rehacer una acción. A continuación, se presenta un análisis detallado por cada mecanismo de usabilidad en los que este se divide: Undo/redo y Abort operation[53].

Undo/redo. Para un sistema la inclusión de esta funcionalidad puede ser primordial o ser una funcionalidad deseable, ya que esta es una de las funcionalidades con características muy utilizadas en aplicaciones como, por ejemplo, los procesadores

[51] BASS, Len; CLEMENTS, Paul; KAZMAN, Rick. Software Architecture in Practice. Second edition. WESLEY, Addison. 2003.ISBN: 0-321-15495-9.

[52] JURISTO, N; MORENO, A; SÁNCHEZ-SEGURA, M; Davis, A. Gathering Usability Information through Elicitation Patterns. 2005.

[53] PANACH, José. Incorporación de mecanismos de Usabilidad en un entorno de producción de software dirigido por modelos. Valencia, 2010, capítulo 9. Tesis doctoral. Universidad Politécnica de Valencia. Centro de Investigación en Métodos de Producción de Software.

de texto, hojas de cálculo, editores gráficos, gestores de correos electrónicos, sistemas de mensajería instantánea entre otros[54].

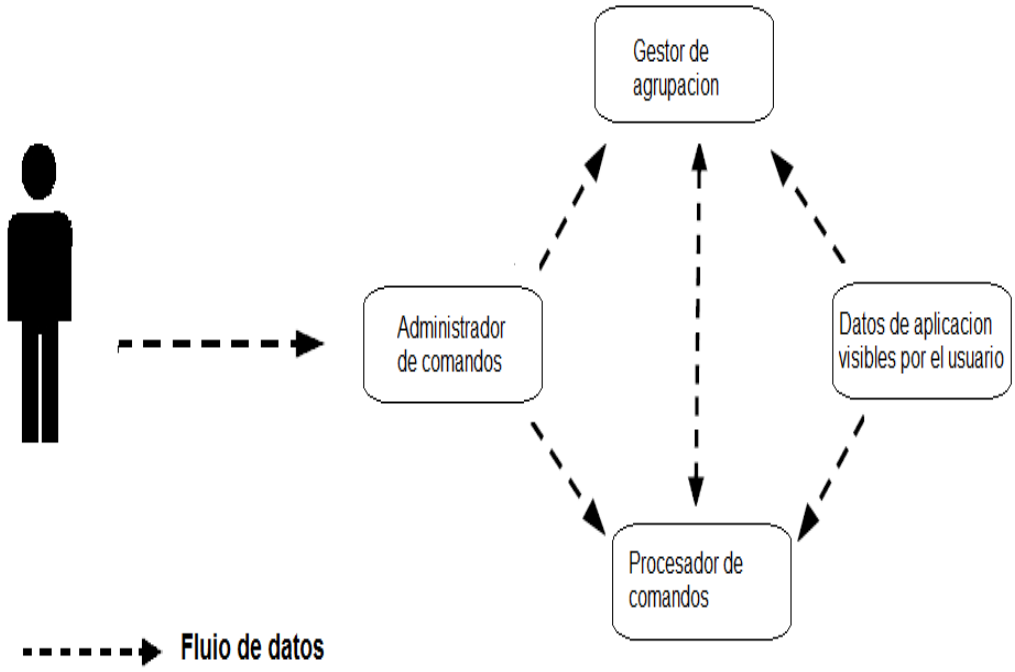
Este mecanismo integra a los sistemas la oportunidad de deshacer acciones realizadas por parte del usuario, en ocasiones los usuarios suelen examinar el sistema sin tener conocimientos de este mismo y, por lo tanto, suelen cometer errores, este mecanismo ayuda al usuario a resolver problemas que hayan surgido por un mal manejo o errores comunes, ya que los errores se convierten de irreversibles a reversibles. Este mecanismo de usabilidad tiene el mismo objetivo que el patrón de usabilidad ir un paso atrás de Tidwell, además se lo puede utilizar para deshacer acciones ejecutadas por error por parte del usuario o rehacer cambios como rehacer acciones previamente que se han modificado.

Abort operation. Este mecanismo de usabilidad integra a los sistemas la funcionalidad de cancelar una acción antes de que esta finalice su ejecución. El usuario tiene la posibilidad de cancelar una acción a mitad de ejecución si la acción requiere mucho tiempo como, por ejemplo, más de 10 segundos. Además, si el usuario ingresa en un escenario en el cual no quiere estar, este pueda salir volviendo al escenario del que procediera esto se conoce como navegación hacia el origen. Esta forma de uso tiene el mismo objetivo que el patrón de usabilidad ir un paso atrás de Tidwell. Por otra parte, este mecanismo tiene las siguientes dos formas de uso, la primera es cancelar durante la ejecución, el usuario puede cancelar un servicio en ejecución que requiera mucho tiempo, la segunda es salir de un escenario, el usuario debe poder salir de una interfaz para poder regresar a la anterior.

- **Agregación**. Bass en su trabajo “Lograr Usabilidad A través del software Arquitectura” expone este patrón de arquitectura el cual lo describe en la siguiente situación. Un usuario puede tener la necesidad de realizar una o más acciones sobre un mismo objeto, por ejemplo, un usuario que utilice adobe Illustrator desee ampliar muchas líneas en un dibujo, para realizar esto necesita hacer una a la vez, esto es un trabajo muy tedioso. El autor afirma que agregación de datos no se puede predecir ya que resulta de los requisitos de cada tarea, por lo tanto, el define que los sistemas deben permitir a los usuarios seleccionar y actuar sobre combinaciones arbitrarias de datos. Este patrón se divide en las siguientes componentes (ver figura 7).

[54] MERLINO, H; DIESTE, P; PESADO; GARCÍA, Martínez, R. Modelo de Inclusión de la Funcionalidad UNDO/REDO. Universidad Nacional de La Plata. Programa de Doctorado en Ciencias Informáticas. Facultad de Informática.

Figura 7. Componentes del patrón agregación



Fuente: (Bass, 2001)

A continuación, se hace una descripción de cada una de las componentes del patrón agregación presentadas en la figura 7:

o El administrador de comandos: este componente gestiona los comandos que genera el usuario, un comando tiene una acción y uno o más temas que proporcionan entrada o aceptan salida desde el comando.
o El gestor de agrupación: este componente gestiona la definición de grupos y la adición y supresión de elementos de datos de un grupo. Este componente tiene como función de crear, eliminar y agregar grupos, tanto los puntos de datos como los grupos deben poder agregarse y eliminarse de los grupos, este además controla la interacción de los comandos a través del procesador de comandos. A través de él se puede acceder a los datos visibles por el usuario y presenta grupos para apoyar los comandos de edición del grupo.
o Datos de aplicación visibles por el usuario**:** este componente proporciona acceso a los datos de la aplicación y es visible para el usuario. Los datos pueden residir en un único repositorio o puede distribuirse a través de una colección de componentes, los datos de aplicación visibles para el usuario están disponibles tanto para aquellos componentes que controlan los datos de presentación como aquellos que manipulan los datos.

3.1.2.2 Tácticas de iniciativa del sistema. Se soportan en un modelo de información, si es el sistema el que toma la iniciativa este depende de cierta información lo cual da lugar a tres tácticas adicionales. La primera es mantener un modelo de tareas, en este caso se determina el contexto para que el sistema pueda tener alguna idea de lo que el usuario quiere hacer y a si el sistema le pueda brindar una guía a seguir como ejemplo, cuando un usuario visita alguna página bancaria y el campo de ingresar el nombre requiere de solo letras minúsculas en este caso se utiliza un corrector de sintaxis. La segunda es mantener un modelo de usuario, en este caso involucra la información que tiene el usuario sobre el sistema y en el comportamiento en términos de tiempo de respuesta esperado. Finalmente, la tercera es mantener un modelo de sistema, en este caso define la expectativa del sistema para así dar una retroalimentación adecuada al usuario, por ejemplo, poder predecir el tiempo adecuado para una actividad. A continuación, se hace un análisis detallado de las tácticas anteriormente propuestas:

- **Mantener un modelo de tareas**. El modelo de tareas analiza y describe el conocimiento que el usuario debe poseer acerca del sistema para su correcta utilización, así mismo, describe tareas que el usuario va a realizar por medio de la interfaz de usuario de la aplicación, estas tareas se aíslan en diferentes acciones las cuales representan una serie de pasos a seguir para alcanzar los objetivos de la tarea. En este modelo también se reúnen los requisitos de calidad de las tareas como, por ejemplo, los requisitos de tiempo de respuesta.

Para especificar el modelo de tareas se recoge distintas notaciones entre las que se destacan métodos basados en un análisis; cognitivo los cuales representan el tipo de conocimiento que tiene el usuario acerca del sistema estos identifican las secuencias del comportamiento; descriptivos estos permiten obtener una descripción completa del sistema a partir de la información obtenida de las tareas; predictivos estos permiten describir el conocimiento que necesita el usuario con el sistema[55]. En la tabla 3, se detallan los métodos de análisis de tareas más relevantes:

Tabla 3. Métodos de análisis de tareas

Método	Tipo	Descripción
CTT	Descriptivo	Herramientas de soporte al análisis y verificación.
HTA	Cognitivo	Modelo de descomposición del conocimiento.

[55] CROSS, Nigel. Ingeniería del diseño. 1999. Capítulo 2.

Método	Tipo	Descripción
GOMS	Cognitivo	Describe la tarea por parte del usuario.
KML	Predictivo	Mide el rendimiento y la capacidad del usuario.

A continuación, se hace un análisis de los modelos de tareas que se han propuesto en la tabla anterior.

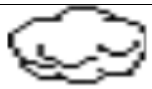
El modelo CTT, es una notación ampliamente difundida en la comunidad de interacción persona ordenador la cual representa actividades interactivas. Este modelo emplea una estructura jerárquica de tareas la cual usa una sintaxis gráfica que le permiten la ordenación temporal de las subtareas. Esta notación está soportada por el lenguaje formal LOTOS[56], el cual expresa aspectos como secuencialidad, concurrencia y recursión.

Por otro lado, este modelo se suele utilizar para el análisis de requisitos de la interfaz ya que para que la aplicación sea aceptada y satisfactoria para el usuario final, la interfaz es una parte clave en el desarrollo de cualquier aplicación. Este modelo de tareas para el diseño y desarrollo de interfaces de usuario, consigue un nivel de abstracción superior, haciendo que el desarrollo de software se convierta en un proceso de ingeniería lo que conlleva a concebir un software de calidad y permite obtener aplicaciones interactivas centradas en el usuario, este modelo propone cuatro categorías de tareas y para cada una de ellas se asocia una representación gráfica. En la tabla 4, se describe cada categoría.

Tabla 4. Realización de las tareas

Tipo de tarea	Notación Gráfica	Descripción
Tareas de usuario		Describe procesos realizados por el usuario usando la información proveniente del entorno.
Tareas de aplicación		Puede obtener información interna del sistema o producir información para el usuario.

[56] PATERNÒ, F. Model–based design and evaluation of interactive application. 2000 Springer–Verlag.

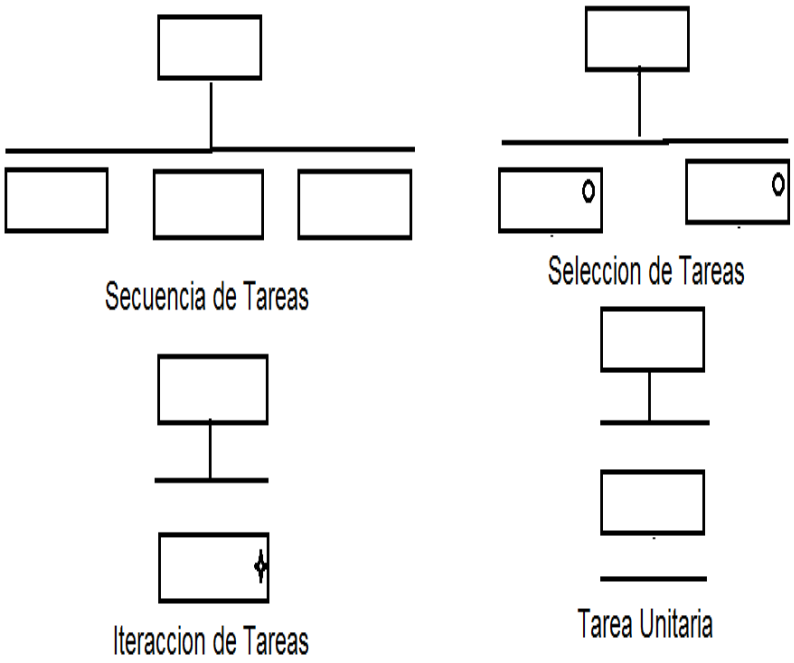
Tipo de tarea	Notación Gráfica	Descripción
Tareas de interacción		Se comprende como las tareas realizadas por el usuario cuando este interactúa con la aplicación.
Tareas abstractas		Son tareas complejas que se pueden descomponer en otras más sencillas.

Esta notación también permite soportar un grupo de criterios para soportar el desarrollo concreto de interfaces de usuario agregándole aspectos de usabilidad.

El modelo HTA, esta notación de análisis de tareas desarrollado por Duncan[57], es una de las más conocidas y antiguas, el cual realiza una descripción de tareas en términos de subtareas. Esta descripción de la información se realiza en forma de tabla o en forma de diagrama de árbol que describe las relaciones entre tareas y subtareas. Este modelo presenta cuatro tipos de descomposición de las subtareas las cuales se describe a continuación en cuatro categorías. La primera categoría es la secuencia, la cual es un conjunto ordenado de una secuencia de tareas. La segunda es selección, la cual es un conjunto donde se tiene que seleccionar entre alguna de ellas. La tercera es la de interacción, la cual es la repetición de un subconjunto de tareas. La última categoría es la tarea unitaria, la cual es una actividad que permite división según el nivel de detalle dado.

[57] Annet, J y Duncan, K. Task analysis and training design. Journal of Occupational Psychology. 1967 41, p 211-221.

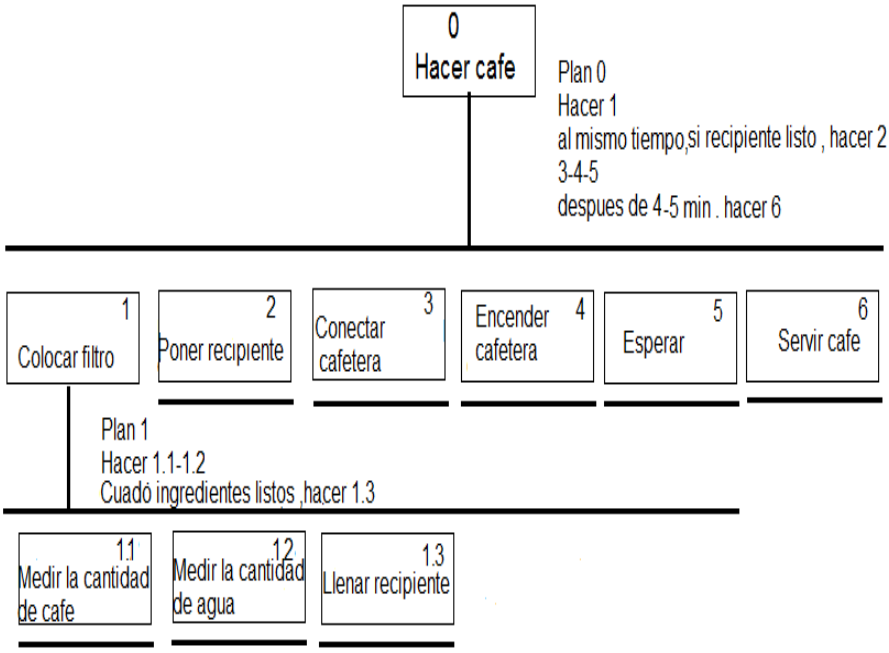
Figura 8. Notación de HTA



Fuente. (Annet, 1967)

La descripción de la información se realiza en forma de tabla o en forma de diagrama de árbol que describe las relaciones entre tareas y subtareas. En la siguiente figura (ver figura 9) se muestra una descripción en HTA como ejemplo, de la preparación del café.

Figura 9. Descripción del HTA



0. Hacer café
1. Colocar Filtro.
    1.1 Medir la cantidad de café.
    1.2 Medir la cantidad de agua.
    1.3 llenar recipiente.
2. Poner recipiente.
3. Conectar cafetera.
4. Encender cafetera.
5. Esperar.
6. Servir café.

Plan 0: hacer 1.
Si recipiente lleno,
Entonces hacer 2 al mismo tiempo.
Hacer 3-4-5
Cuando el café esté listo, hacer 5.
Plan 1: hacer 1,1-1,2
    Cuando ingredientes listos, hacer 1.3

En la fase final se realiza el análisis de la información esta información es útil para la toma de decisiones de diseño y sirve como una guía para las actividades de diseño. La metodología que sigue HTA para el análisis de tareas está divida en tres etapas. La primera es la etapa inicial, esta define la tarea principal que puede ser

dividida en subtareas como, por ejemplo, cuatro y ocho subtareas. La segunda es la etapa intermedia la cual decide el nivel de detalle que se necesita y en qué punto se termina la división. Finalmente, la última etapa es la parte final en esta se revisa y evalúa la consistencia del trabajo que se realizó.

Este modelo de tareas de HTA se comprende de tipo cognitivo de notación gráfica el cual sigue un modelo de descomposición del conocimiento que debe poseer un usuario acerca del uso del sistema.

El modelo GOMS, es un acrónimo que significa objetivo, operadores, métodos y reglas de selección, el cual explica las tareas que el usuario puede realizar con el sistema. Este modelo describe las tareas que el usuario puede llevar a cabo con el sistema o dispositivo, está compuesto por técnicas y es una propuesta para modelar y describir una tarea humana mediante principios físicos y cognitivos[58]. KML-GOMS es una de estas técnicas la cual es una de las más representativas de este modelo ya que analiza y describe en base a operaciones elementales haciendo una simplificación de estas tales, como una pulsación de teclado o el movimiento del ratón, lo que permite obtener predicciones del tiempo que emplea un usuario para realizar una tarea. Cada una de estas operaciones lleva asociado un tiempo de ejecución medio, siendo, por ejemplo, una décima de segundo lo que tardaría un usuario en pulsar un botón del ratón o dos décimas en pulsar una tecla del teclado.

El modelo GOMS, fue uno de los primeros métodos utilizados para el diseño de interfaces de usuario, el cual fue de gran influencia para otras investigaciones. Este es ampliamente utilizado por los especialistas en usabilidad ya que brinda predicciones cuantitativas y cualitativas, de cómo el usuario utilizará un sistema.

- **Mantener un modelo de usuario.** Bass, propone mantener un modelo del usuario donde se determina el conocimiento que el usuario tiene acerca del sistema, así como el comportamiento del usuario que sirve para saber qué tipo de respuesta éste espera. Por eso es necesario primero definir en qué consiste un modelo de usuario. De acuerdo con Fischer[59], un modelo de usuario es la descripción de las características del usuario final de la aplicación, en base a éstas se puede tomar decisiones en la interacción persona ordenador, el cual tiene como objetivo brindar una interfaz de usuario que se adapte a las características y necesidades de cada usuario, esta adaptación se puede dar tanto en tiempo de diseño como de ejecución.

[58] CARD, Stuart; THOMAS, P; MORAN y NEWELL, Allen. The keystroke-level model for user performance time with interactive systems. Communications of the ACM, 23(7), 1980. p 396-410.
[59] FISCHER, Gerhard. User modeling in human-computer interaction. En: User Modeling and User-Adapted Interaction, 11. 2001. p 65-86.

En tiempo de diseño se define por un modelo de usuario el cual representa las tareas y los roles para los diferentes tipos de usuario. En tiempo de ejecución la adaptabilidad es más compleja ya que requiere un modelo de usuario para cada tipo de usuario. En tiempo de ejecución las interfaces de usuario adaptativas brindan al usuario un interfaz que se adapte, como, por ejemplo, en su aspecto o a las tareas que el usuario realiza cuando esté interactuando con el sistema, un ejemplo de adaptabilidad es la disminución de información que el usuario debe procesar para poder realizar una tarea determinada, el sistema debe mostrar únicamente la información prioritaria para la realización de la tarea a realizar por parte del usuario.

Por otra parte, cuando nos referimos a un modelado de usuario se puede obtener convicciones e intereses por parte de los usuarios, también se define clases o perfiles de usuarios en base a atributos comunes. Los atributos sobre los que se hará la clasificación dependen de la información que se tenga de los usuarios, pero normalmente se tratarán de atributos como, ejemplo, necesidades de información, condiciones de acceso, experiencia y conocimientos. Además, cabe aclarar cuando nos referimos al término convicción se comprende que es todo el entorno que rodea al modelado de un usuario, por ejemplo, su estereotipo, los vínculos entre otros, en cambio los intereses son los deseos que resultan de modelar, es decir, el grado de interés que tiene sobre un recurso.

Siguiendo un modelado de usuario, el diseñador tendrá en mente para quién diseña, qué espera encontrar el usuario y en qué forma. Según Hassan[60], el diseño del sitio web debe estar orientado al usuario, organizada y estructurada la información según los modelos definidos de usuarios. El modelado de usuario para Hassan se basa en la definición de arquetipos de usuarios estos representan patrones de conducta, objetivos, necesidades o ejemplo de ideas o conocimiento del cual se derivan otros tantos para modelar los pensamientos y actitudes propias de cada individuo. Estos arquetipos describen al usuario en forma narrativa por lo que son llamados personas a los que se les da una identidad inventada como, por ejemplo, fotografía, nombre en otras, Hassan en su trabajo expone, todos los atributos, características y necesidades del arquetipo deben ser fundamentadas en información real extraída de la audiencia objetiva del sitio web, sino esta técnica no será válida si estos datos fueran inventados, además se propone definir escenarios, estos se crean a partir de un análisis del usuario, los cuales describen situaciones del uso del sitio . Por lo tanto, se comprende que la función principal del modelado del usuario es la de servir de base para la toma de decisiones en el diseño del sitio web, permitiendo desarrollar un diseño centrado en el usuario.

[60] HASSAN MONTERO, Yusef; MARTÍN FERNÁNDEZ, Francisco J. La experiencia del usuario. En: No solo usabilidad, nº 4, 2005. ISSN 1886-8592.

Para poder modelar a un usuario, es necesario tener información inicial que debe ser proporcionada por parte del usuario, ésta es muy importante, porque se toma como base para modelar la información que le podría interesar. A continuación, en la tabla 5, se describe los elementos básicos para poder modelar a un usuario:

Tabla 5. Elementos de modelado de usuario

Elementos	**Descripción**
Información personal	Preguntas básicas, por ejemplo, quien es el usuario, edad, sexo, actividad ocupacional, educación.
Información sobre la experiencia y capacidades de un usuario	Experiencia del usuario en el uso del computador.
Información sobre sus pasatiempos y preferencias	Se refiere a los tópicos que le interesan, dependiendo de su área.
Observación directa	Requiere de la observación de cada usuario en forma individual.
Entrevistas	En esta se detecta el conocimiento y problemas en el uso.
Cuestionarios	Permite un análisis estadístico de los datos.

Hassan[61], describe una situación que un diseñador podría imaginar y preguntarse cuando esté mismo esté usando el sitio, ¿porque el sitio que él mismo creó le resultaría ser difícil de entender e incómodo a los demás en su uso? Por tal razón expone que los arquetipos de usuarios conseguirán precisamente que el diseñador tenga en mente a un usuario real, con limitaciones, habilidades y necesidades reales.

Por otra parte, cuando se habla de mantener un modelo de usuario se comprende que un usuario es muy cambiante, y las convicciones que lo conforman están en constante variación y adaptación, es por eso que es necesario evaluar al usuario constantemente. Rich[62], sugiere utilizar la información sobre el uso del estereotipo que es el grado de experiencia adquirida por el usuario al utilizar un sitio como un indicador de los intereses del usuario, esto mantendrá actualizado el modelo, es por

[61] HASSAN MONTERO, Yusef. Introducción a la usabilidad.01 de noviembre de 2002. [en línea]. <http://www.nosolousabilidad.com/articulos/introduccion_usabilidad.htm> [citado el 15 de diciembre de 2016].

[62] RICH, E. “Stereotypes and User Modelling”, in KOBSA A. And WAHLSTER, W. (eds.) User Models in Dialog System, Springer, Berlin. 1989.

esto que los autores sugieren clasificar la interacción del usuario en las siguientes interacciones:

La primera interacción, es acciones selectivas, esta indica las preferencias e intereses a usuario, por ejemplo, la navegación y seguimiento de las páginas de internet. La segunda, es tiempo de interacción, esta indica cuánto tiempo el usuario estuvo utilizando un recurso, el problema que surge de esta interacción es que no se puede saber si realmente un usuario estuvo utilizando este recurso, o tal vez quedó en el hecho de olvido. La tercera, es evaluaciones son los criterios que presenta el usuario, en una escala generalmente numérica ya establecida, el problema que presenta esta interacción es que el usuario no siempre va a estar dispuesto a colaborar en el aprendizaje del navegador, es por eso que se sugiere que debe estar la opción de desactivar la evaluación como por ejemplo, en el caso del navegador son las preguntas que aparecen periódicamente, en caso de que se desea desactivar la opción por parte del usuario puede influir en el grado de desinterés. Finalmente, la última interacción es confirmación o anulación, cuando el usuario selecciona alguna sugerencia, este permanece cierto tiempo en ella nos confirma que fue exitosa, con esto, se le da un mayor peso a este recurso, en caso contrario esta sugerencia no haya sido seleccionada o no fue vista en cierto lapso de tiempo, puede disminuir el peso del recurso.

- **Mantener un modelo del sistema.** Este modelo se comprende como la información sobre la actividad que el sistema está ejecutando y que es relevante para el usuario. Para Bass, mantener un modelo del sistema, determina el comportamiento esperado como va a reaccionar el sistema para así poder dar una retroalimentación apropiada al usuario, como, por ejemplo, predecir el tiempo necesario para completar una actividad. Por eso es necesario hacer un análisis de cómo se puede incorporar mecanismos de retroalimentación, los cuales tienen como objetivo mantener informado al usuario sobre lo que está ocurriendo en todo momento dentro del sistema. Panach[63], en su trabajo propone incorporarlo como un mecanismo de usabilidad y presenta una subdivisión de este mecanismo en el que se divide en:

o System status feedback
o Interaction feedback
o Progress feedback
o Warning.

[63] PANACH, José. Incorporación de mecanismos de Usabilidad en un entorno de producción de software dirigido por modelos. Valencia, 2010, capítulo 5. Tesis doctoral. Universidad Politécnica de Valencia. Centro de Investigación en Métodos de Producción de Software. Sistemas Informáticos y Computación Centro de Investigación en Métodos de Producción de Software.

A continuación, se describe detalladamente cada mecanismo de retroalimentación:

El primero es **status feedback**, este mecanismo tiene como finalidad comunicar al usuario si hay cambios significativos que se producen en el sistema o si se produce un fallo como puede darse en las siguientes situaciones, mientras un servicio se está ejecutando, antes de la ejecución de un servicio o después de la ejecución de un servicio. Este mecanismo de usabilidad tiene una relación con las heurísticas de Nielsen[64], porque comparte la misma finalidad. Este mecanismo es útil ya que sirve para interactuar con el usuario y se le puede dar las siguientes formas de uso, tales como, informar del éxito o fracaso en la ejecución del servicio, mostrar el estado de la información, mostrar el estado de las acciones, informar de falta de recursos, informar de fallos en dispositivos externos.

La segunda subdivisión es **interaction feedback**, este mecanismo de usabilidad tiene como finalidad informar al usuario de que sus peticiones de interacción han sido recibidas se entiende que actúa cuando se realiza un evento, como por ejemplo, cada vez que el usuario hace una iteración con el sistema, tales como un movimiento del ratón o pulsar una tecla, se le tiene que brindar información por parte del sistema que esta petición ha sido recibida, a este mecanismo se le puede dar la siguiente forma de uso, informar al usuario de que el sistema está procesando la petición de ejecución de un servicio.

La tercera subdivisión es **progress feedback**, este mecanismo tiene como finalidad mostrar el tiempo restante para la finalización de un servicio que es ejecutado por el usuario. Según Tidwell[65], esta funcionalidad está pensada para aquellos servicios cuya ejecución requiere más de dos segundos, en cambio, para aquellos servicios cuya ejecución no sea crítica y su duración estimada sea de más de cinco segundos, el sistema debe dar la posibilidad al usuario de ejecutar otras tareas. A este mecanismo se le asigna la siguiente forma de uso la de informar al usuario del tiempo restante para la finalización de la ejecución de un servicio.

Finalmente, la última subdivisión es **warning**, este mecanismo tiene como finalidad solicitar al usuario confirmación en caso que se cometa algún error y las consecuencias puedan ser irreversibles, de esta forma se evita que el usuario cometa errores. Además de solicitar esta confirmación, el sistema debe informar al usuario, de las consecuencias que tiene la ejecución de la acción. Son varios los

[64] NIELSEN, Jakob. 10 Usability heuristics for user interface design. En: Nielsen norman group. 1995. Disponible en:< https://www.nngroup.com/articles/ten-usability-heuristics/ >. [citado el 17 de diciembre de 2016].
[65] TIDWELL, Jennifer. Designing Interfaces. 2010. O' Reilly Media.

autores que han propuesto la funcionalidad de este mecanismo en forma de patrones de usabilidad, como, por ejemplo, Welie[66], con el patrón llamado escudo. A este mecanismo se le asigna la siguiente forma de uso, la cual es pedir confirmación para la ejecución de servicios con consecuencias irreversibles.

3.1.3 Síntesis de la caracterización de las estrategias. En el desarrollo de este fin, se caracterizó las estrategias arquitectónicas de usabilidad web, donde se presentó los principales trabajos relacionados con el uso de patrones, directrices, frameworks y reglas, con los que se puede que abordar la usabilidad en la construcción de aplicaciones web. Esto permitió conocer qué estrategias son las que se utilizan en la construcción de software, además del uso de tácticas de usabilidad, decisiones de diseño que influyen en la respuesta a este atributo de calidad, las cuales están contenidas en la estrategia y están refinadas en un orden jerárquico, en donde se las sistematizó en dos categorías.

La primera categoría trata los problemas iterativamente en la interfaz de usuario, tácticas en tiempo de diseño, está conformada por separar la interfaz del resto del sistema, diseño de interfaces de usuario reglas de oro de Nielsen e implementación de frameworks. La segunda categoría brinda soporte al usuario durante la ejecución, tácticas en tiempo de ejecución, está conformada por iniciática del usuario e iniciativa del sistema.

[66] WELIE, Martin Van. A Pattern Library for Interaction Design. 2008. Disponible en:< http://www.welie.com/>

3.2 VENTAJAS Y DESVENTAJAS DE LAS ESTRATEGIAS

En esta sección, se describe de manera comparativa las ventajas y desventajas que existen en las estrategias arquitectónicas de usabilidad web que se utilizan en la construcción de software. Para alcanzar este objetivo, se tuvo como fuente de información el documento con la caracterización de las estrategias arquitectónicas de usabilidad web resultado del desarrollo del primer objetivo de este estudio. La técnica que se utilizó para la recolección de información fue la revisión documental. Para el análisis de la información se utilizó como técnica el análisis documental. Las variables analizadas fueron tácticas en tiempo de diseño y tácticas en tiempo de ejecución, con los indicadores patrón, documentación, directriz e implementación.

3.2.1 Definición de indicadores. A continuación, se define los indicadores que fueron establecidos para la recopilación y análisis de la información.

Por patrón, se entiende como un modelo a seguir, el cual brinda una solución. En el mundo del software existen dos clases de patrones. El primero son los patrones de diseño, que es una solución a problemas recurrentes de diseño de software, en otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Un patrón presenta la siguiente estructura: un nombre que lo identifica; una descripción del problema o contexto; una descripción de la solución conceptual; una discusión de las implicaciones del patrón; y su relación con otros patrones[67]. Además, facilitan la reutilización de arquitecturas y diseños de software exitosos. La segunda concepción sobre patrones son los arquitectónicos, los cuales son soluciones a problemas de arquitectura de software en ingeniería de software[68]. Haciendo comparación entre los patrones de diseño y los patrones arquitectónicos, los últimos tienen un nivel de abstracción mayor ya que pueden ser relevantes en la totalidad del sistema.

Por documentación, se comprende como una colección de documentos generales oficiales con la que se prueba o acredita algo. De acuerdo con Definista[69], se puede

[67] Design Patterns. Elements of Reusable Object-Oriented Software - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - Addison Wesley (GoF- Gang of Four).

[68] MORENO, A.M.; SÁNCHEZ-SEGURA, M. Patrones de usabilidad: Mejora de la usabilidad del software desde el momento de arquitectónico. Alicante, noviembre 2003. VIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 03).

[69] Definista. Definición de Documentación.2016 [Entrada de blog]. Disponible en:< http://conceptodefinicion.de/documentacion/> [citado el 14 de enero de 2017].

considerar como una técnica instrumental y auxiliar, que sirve para lograr informar sobre un tema en particular. Se puede considerar como una técnica instrumental y auxiliar, que sirve para lograr informar sobre un tema en particular.

Una directriz, se establece por un conjunto de normas e instrucciones para ejecutar un plan. Además, determina las bases o las formas en que una actividad debe ser realizada.

Por implementación, se define como un proceso que tiene como finalidad poner en práctica, medidas y métodos, entre otros, con el fin de concretar un plan para alcanzar un o varios objetivos. Por ejemplo, la realización de una aplicación, o bien la ejecución de una idea, plan, diseño, modelo científico, estándar, especificación o política, entre otros.

3.2.2 Proceso de análisis de ventajas y desventajas. El análisis se realizó a partir del desarrollo de las actividades que se describe a continuación:

- **Síntesis de la caracterización.** El primer paso para identificar las ventajas y desventajas, se elaboró una síntesis de los resultados obtenidos al caracterizar las estrategias arquitectónicas de usabilidad web que se utilizan en la construcción de software. Para realizar la síntesis, se utilizó las tablas 6 y 7.

Tabla 6. Síntesis de las tácticas en tiempo de diseño

Táctica	**Patrón**	**Documentación**	**Directriz**	**Implementación**
Separar la interfaz del resto de la aplicación	**Nivel Alto.** Sugiere patrones de diseño a poner en práctica tales como Modelo-Vista Controlador, Presentación-Abstracción-Control, Modelo Seeheim, Arch/Slinky.	**Nivel Medio**. No se encuentra suficiente información acerca de cada uno de los patrones que sugiere poner en práctica esta táctica.	**Nivel Bajo**. No presenta directrices a seguir, si no que se puede implementar siguiendo alguno de los patrones que se propone.	**Nivel Alto.** Fácil de implementar en cualquier sistema de software, ya que hoy en día un sin número de framework de desarrollo integra y ayuda a la implementación de alguno de los patrones que sugiere esta táctica. Además, se encuentra documentación con ejemplos prácticos y sencillos de entender.

Táctica	Patrón	Documentación	Directriz	Implementación
Diseño de interfaces de usuario	**Nivel Alto**. Posee una gran variedad de patrones de diseño e interacción, y guías de usabilidad que se encuentran en trabajos propuestos por diferentes autores tales como, Perzel Y kane, Welie, Tidwell, Seffah y Taleb, Hernández Álvarez Y Arteaga.	**Nivel Alto.** Presenta una documentación específica acerca de cada patrón y directriz.	**Nivel Medio alto.** Presenta un conjunto de directrices como guías a seguir las cuales se complementan con los patrones.	**Nivel medio.** La mayoría de los patrones no incluyen ninguna descripción sobre cómo implementarlos y las implicaciones en la arquitectura del sistema que puede tener.
Reglas de oro de Nielsen	**Nivel Bajo.** No sugiere seguir algún tipo de patrón en particular.	**Nivel Alto.** En el sitio web del autor se encuentran artículos completos sobre la teoría en cuanto a principios generales para el diseño.	**Nivel Alto.** Adopta principios de usabilidad los cuales se los considera como directrices guía para conseguir un sistema usable.	**Nivel bajo.** No tiene una facilidad de implementación ya que cada principio se limita a presentar cómo se debe aplicar.
Implementación de frameworks	**Nivel Medio Alto.** Ponen en práctica patrones de diseño	**Nivel Medio Alto.** Incluye información muy detallada para la	**Nivel Bajo.** No menciona directrices a poner en práctica, se	**Nivel Medio**. El usuario debe estudiar qué proporciona el Framework y

Táctica	Patrón	Documentación	Directriz	Implementación
	conocidos, aderezados con funciones que asisten al desarrollador.	implementación.	basa generalmente en patrones de diseño**.**	cómo debe hacer uso de él.

Tabla 7. Síntesis de las tácticas en tiempo de ejecución

Táctica	Patrón	Documentación	Directriz	Implementación
Iniciativa del usuario	**Nivel Medio Alto.** Incorpora mecanismos de usabilidad formalizados como patrones de usabilidad.	**Nivel Alto.** Se encuentra artículos completos de información teórica de cada mecanismo para poner en práctica esta táctica.	**Nivel Medio.** Dentro de estos principios o directrices se encuentran las funcionalidades.	**Nivel Medio Alto**. La implementación de funcionalidades de esta en un sistema nuevo o existente no es un proceso trivial, una de las razones es que su inclusión generalmente se realiza en una etapa avanzada del desarrollo del sistema, generalmente cuando las decisiones claves de diseñado ya han sido tomadas.
Iniciativa del sistema	**Nivel Medio alto.** Se basa principalmente en modelos que utilizan	**Nivel Medio Alto.** Contienen información específica y	**Nivel Alto.** Presenta distintas actividades	**Nivel Medio.** Presenta modelos de facilidad de uso con ejemplos de

Táctica	Patrón	Documentación	Directriz	Implementación
	notaciones que sirven como guía para las actividades de diseño, cabe resaltar que alguno de los modelos utiliza mecanismos de usabilidad considerados como patrones de diseño.	completa por cada modelo.	para su desarrollo.	aplicación, lo que hace que sea aplicable a proyectos reales con aplicaciones de un tamaño mediano-grande y que conllevan especificaciones de cierta complejidad.

• **Modelo matemático para la valoración**. Para determinar la medición de los indicadores, el primer paso fue definir una escala de valoración (ver tabla 8). Posteriormente, con estos juicios de valor, se toma como base la síntesis de las tácticas arquitectónicas de usabilidad, para diligenciar la tabla 9.

Tabla 8. Juicio de favorabilidad

Juicio	Valoración
Muy favorable (MF)	5
Favorable (F)	4
Ni favorable, Ni desfavorable (NF, ND)	3
Desfavorable (D)	2
Muy desfavorable (MD)	1

Tabla 9. Valoración de la táctica arquitectónica de usabilidad

Indicador	Juicio de Favorabilidad				
	MF	F	NF, ND	D	MD
Patrón					
Documentación					
Directriz					
Implementación					

Posteriormente, se debe registrar en la tabla 10, la valoración correspondiente al juicio de favorabilidad (ver tabla 8) y calcular el porcentaje de la frecuencia según el valor asignado.

Tabla 10. Frecuencia de la incidencia por táctica arquitectónica

Táctica Arquitectónica de Usabilidad Web	Patrón		Documentación		Directriz		Implementación	
	FO	%FO	FO	%FO	FO	%FO	FO	%FO
Táctica 1								
Táctica 2								
…								
Táctica m								

Donde FO, es la Frecuencia Observada y corresponde al valor del juicio de favorabilidad, %FO es el porcentaje de la Frecuencia Observada y m es el número total de tácticas.

Una vez se realiza el cálculo de frecuencias, se procede a definir los rangos para establecer el juicio de incidencia de las tácticas arquitectónicas, según los indicadores, como se presenta en la tabla 11.

Tabla 11. Matriz de rangos de favorabilidad para indicadores

Rango	Juicio de Incidencia
(Entre %FO1 y %FO2) (Mayor a %FO1)	Ventaja
(Entre %FO1 y %FO2) (Menor a %FO1)	Desventaja

Finalmente, se procede a asignar el juicio de incidencia, como se presenta en la tabla 12, para las frecuencias obtenidas en la tabla 10, tomando como referente la matriz de rangos de favorabilidad para indicadores.

Tabla 12. Matriz de favorabilidad

Táctica Arquitectónica de Usabilidad Web	Indicador	Porcentaje de la Frecuencia Observada - %FO	Juicio de Incidencia
Táctica 1	Patrón	Porcentaje	(Favorable / Desfavorable)
	Documentación	Porcentaje	(Favorable / Desfavorable)
	Directriz		
	Implementación		
Táctica 2			
…			
Táctica m			

Finalmente, se hace una descripción de las tácticas arquitectónicas según las ventajas y desventajas, de acuerdo con los datos obtenidos en la tabla 12 y se elabora una matriz del perfil de ventajas y desventajas (ver tabla 13).

Tabla 13. Matriz del perfil de ventajas y desventajas por indicador

Táctica Arquitectónica de Usabilidad Web	**Ventajas**	**Desventajas**
Táctica 1	Descripción	Descripción
Táctica 2		
. . .		
Táctica m		

• **Resultados de la aplicación del modelo matemático.** Teniendo como referente el modelo matemático, a continuación, se presenta una síntesis de los resultados obtenidos.

Para determinar la medición de los indicadores, se definió una escala de valoración (ver tabla 9). Posteriormente, con estos juicios de valor, se tomó como base la síntesis de las tácticas arquitectónicas de usabilidad (ver tablas 6 y 7), para diligenciar las tablas 14,15,16,17,18 y 19.

Tabla 14. Valoración de la táctica separar la interfaz del resto de la aplicación

Indicador	**Juicio de Favorabilidad**				
	MF	**F**	**NF, ND**	**D**	**MD**
Patrón	■				
Documentación		■			
Directriz				■	
Implementación	■				

Tabla 15. Valoración de la táctica diseño de interfaces de usuario

Indicador	**Juicio de Favorabilidad**				
	MF	**F**	**NF, ND**	**D**	**MD**
Patrón	■				
Documentación	■				
Directriz		■			
Implementación			■		

Tabla 16. Valoración de la táctica reglas de oro de Nielsen

Indicador	Juicio de Favorabilidad				
	MF	F	NF, ND	D	MD
Patrón				■	
Documentación	■				
Directriz	■				
Implementación				■	

Tabla 17. Valoración de la táctica implementación de frameworks

Indicador	Juicio de Favorabilidad				
	MF	F	NF, ND	D	MD
Patrón		■			
Documentación	■				
Directriz				■	
Implementación		■			

Tabla 18. Valoración de la táctica iniciativa del usuario

Indicador	Juicio de Favorabilidad				
	MF	F	NF, ND	D	MD
Patrón		■			
Documentación			■		
Directriz			■		
Implementación	■				

Tabla 19. Valoración de la táctica iniciativa del sistema

Indicador	Juicio de Favorabilidad				
	MF	F	NF, ND	D	MD
Patrón		■			
Documentación		■			
Directriz	■				
Implementación			■		

Posteriormente, se registró en las tablas 20 y 21, la valoración correspondiente al juicio (Ver tablas 14, 15, 16, 17,18 y 19) donde se calculó el porcentaje de la frecuencia según el valor asignado.

Tabla 20. Frecuencia de la incidencia para tácticas en tiempo de diseño

Táctica arquitectónica de Usabilidad Web	Patrón		Documentación		Directriz		Implementación	
	FO	%FO	FO	%FO	FO	%FO	FO	%FO
Separar la interfaz del resto de la aplicación	5	100%	4	80%	2	40%	5	100%
Diseño de interfaces de usuario	5	100%	5	100%	4	80%	3	60%
Reglas de oro de Nielsen	2	40%	5	100%	5	100%	2	40%
implementación de frameworks	4	80%	5	100%	3	60%	4	80%
Iniciativa del usuario	4	80%	3	60%	3	60%	5	100%
Iniciativa del sistema	4	80%	4	80%	5	100%	3	60%

Tabla 21. Frecuencia de la incidencia para tácticas en tiempo de ejecución

Táctica arquitectónica de usabilidad web	Patrón		Documentación		Directriz		Implementación	
	FO	%FO	FO	%FO	FO	%FO	FO	%FO
Iniciativa del usuario	4	80%	3	60%	3	60%	5	100%
Iniciativa del sistema	4	80%	4	80%	5	100%	3	60%

Una vez se realizó el cálculo de frecuencias, se definió los rangos para establecer el juicio de incidencia de las tácticas arquitectónicas, según los indicadores, como se presenta en la tabla 22.

Tabla 22. Matriz de rangos de favorabilidad

RANGO	JUICIO DE INCIDENCIA
ENTRE 80% Y 100%	VENTAJA
MENOR A %80	DESVENTAJA

Finalmente, se procedió a asignar el juicio de incidencia, como se presenta en las tablas 23 y 24, para las frecuencias obtenidas en las tablas 20 y 21, tomando como referente la matriz de rangos de favorabilidad.

Tabla 23. Matriz de favorabilidad para tácticas en tiempo de diseño

Táctica Arquitectónica de Usabilidad Web	Indicador	Porcentaje de la Frecuencia Observada - %FO	Juicio de Incidencia
Separar la interfaz del resto de la aplicación	Patrón	100%	Ventaja
	Documentación	60%	Desventaja
	Directriz	40%	Desventaja
	Implementación	100%	Ventaja
Diseño de interfaces de usuario	Patrón	100%	Ventaja
	Documentación	80%	Ventaja
	Directriz	80%	Ventaja
	Implementación	60%	Desventaja
Reglas de oro de Nielsen	Patrón	40%	Desventaja
	Documentación	100%	Ventaja
	Directriz	100%	Ventaja
	Implementación	40%	Desventaja
Implementación de frameworks	Patrón	80%	Ventaja
	Documentación	100%	Ventaja
	Directriz	60%	Desventaja
	Implementación	80%	Ventaja

Tabla 24. Matriz de favorabilidad para tácticas en tiempo de ejecución

Táctica Arquitectónica de Usabilidad Web	Indicador	Porcentaje de la Frecuencia Observada - %FO	Juicio de Incidencia
Tácticas de iniciativa del usuario	Patrón	80%	Ventaja
	Documentación	60%	Desventaja
	Directriz	60%	Desventaja
	Implementación	100%	Ventaja
Tácticas de iniciativa del sistema	Patrón	80%	Ventaja
	Documentación	80%	Ventaja
	Directriz	100%	Ventaja
	Implementación	60%	Desventaja

Finalmente, se hizo una descripción de las tácticas arquitectónicas según las ventajas y desventajas, de acuerdo con los datos obtenidos en las tablas 23 y 24. Posteriormente, se elaboró una matriz del perfil de ventajas y desventajas (ver tablas 25 y 26)

Tabla 25. Matriz de ventajas y desventajas para tácticas en tiempo de diseño

Táctica Arquitectónica de Usabilidad Web	Ventajas	Desventajas
Separar la interfaz del resto de la aplicación	• Todos los sistemas complejos experimentan la necesidad de desarrollar y evolucionar porciones del sistema de forma independiente. Por esta razón, los desarrolladores del sistema necesitan una clara y bien documentada separación de las capas, de modo que el módulo del sistema pueda ser desarrollado y mantenido de forma independiente. • Permite la reusabilidad, de modo que la misma implementación,	• Es necesario una mayor dedicación en los tiempos iniciales del desarrollo. • Los patrones que propone esta táctica generalmente exigen al programador desarrollar un mayor número de clases que, en otros entornos de desarrollo. • La distribución de componentes obliga a crear y mantener un mayor número de ficheros.

Táctica Arquitectónica de Usabilidad Web	Ventajas	Desventajas
	pueda ser usada en otras. aplicaciones, sean éstas web o no • Permite una sencilla división de roles. • Permite trabajar en varios niveles de abstracción.	• No todo el sistema se puede estructurar fácilmente como capas.
Diseño de interfaces de usuario	• Permite interactuar al usuario con el sistema de una forma más fácil. • Proporciona mecanismos estándar de control como ventanas y cuadros de diálogo y guías de estilo proporcionando look and Feel.	• Utiliza más recursos del sistema. • Requiere por parte desarrollador una perspectiva más amplia para poder implementarlos.
Reglas de oro de Nielsen	• Se identifican aspectos de usabilidad y componentes del contexto de uso que deben tenerse en cuenta en las fases de especificación de requisitos, diseño y evaluación de usabilidad para posteriormente poderlas mejorar.	• Mejoran la usabilidad del sistema una vez este se ha construido y después iterativamente. • No se especifican como obtener los objetivos propuestos por los principios, por esta razón se limitado al uso práctico.
Implementación de frameworks	• Agilizan el proceso de desarrollo. • El uso de un framework ayuda mucho con la seguridad, reducir tiempo y tener un código optimizado, ordenado y entendible • Generalmente adoptan el patrón MVC que permite un código más limpio y mantenible.	• Fuerza a usar la semántica propia de la framework.

Tabla 26. Matriz de ventajas y desventajas para tácticas en tiempo de ejecución

Táctica Arquitectónica de Usabilidad Web	Ventajas	Desventaja
Iniciativa del usuario	• Incorpora mecanismos que se han denominado patrones de usabilidad que abordan alguna necesidad indicada por una propiedad de usabilidad.	• Las incorporaciones de funcionalidades requieren de más código para su implementación. • Mayor uso de memoria.
Iniciativa del sistema	• Incorpora modelos que contienen métodos que permiten una especificación precisa y sin ambigüedad del diseño a generar. • Al igual que la táctica iniciativa del usuario incorpora mecanismos considerados como patrones que abordan una propiedad especial de usabilidad.	• Variabilidad de las características del usuario, junto con las características de la interacción del usuario con el interfaz que puede cambiar dinámicamente.

3.2.3 Orden de favorabilidad de la táctica. Esta acción permitió hacer el cálculo de la media del porcentaje de la frecuencia observada para cada táctica e indicador y organizar los resultados en orden descendente. Esta operación posibilitó identificar la favorabilidad de cada táctica según los indicadores. Para presentar los resultados, se diligenció la matriz de orden de favorabilidad (ver tablas 27 y 28).

Tabla 27. Matriz de orden de favorabilidad para tácticas en tiempo de diseño

Orden	Táctica Arquitectónica de Usabilidad Web	Media del %FO
1	Diseño de interfaces de usuario	85%
2	Separar la interfaz del resto de la aplicación	80%
3	Implementación de frameworks	75%
4	Reglas de oro de Nielsen	70%

Tabla 28. Matriz de orden de favorabilidad para tácticas en tiempo de ejecución

Orden	Táctica Arquitectónica de Usabilidad Web	Media del %FO
1	Iniciativa del sistema	80%
2	Iniciativa del usuario	75%

3.2.4 Síntesis del análisis de ventajas y desventajas. Finalizado el análisis de ventajas y desventajas, para el indicador patrón en todas las tácticas se obtuvo una valoración superior a 3, excepto reglas de oro de Nielsen ya que esta táctica no pone en práctica ningún patrón en particular para seguir los principios que presenta. Para el indicador documentación, que describe la disponibilidad de la documentación necesaria para poner en práctica las tácticas analizadas, en todas las tácticas se obtuvo una valoración mayor de 3, lo cual se considera favorable, porque cuentan con varios documentos y artículos acerca de la teoría e implementación. Para el indicador directriz, las cuales sirven como guía ya que proporcionan una serie de lineamientos que ayudan a cumplir un objetivo, la táctica reglas de oro de Nielsen obtuvo una valoración de 5, porque cuenta una serie de lineamientos que son de ayuda para concebir un sistema usable, mientras que las tácticas separar la interfaz del resto de la aplicación e implementación de frameworks se obtuvo una valoración de nivel bajo, ya que estás exclusivamente ponen en práctica patrones de diseño para su aplicación y no menciona directrices a seguir en cambio en la táctica diseño de interfaces de usuario obtuvo una valoración favorable, porque es una táctica que incorpora trabajos que integra directrices las cuales se complementan junto con los patrones de diseño, donde se las considera guías de ayuda, con respecto a las demás tácticas se obtuvo una valoración favorable, ya que al igual que la táctica de diseño de interfaces estas integran directrices que se complementan con los modelos y mecanismos. Para el indicador implementación, el cual describe el nivel de facilidad que existe para poder aplicar la táctica, las tácticas separar la interfaz del resto de la aplicación, implementación de frameworks e iniciativa del usuario obtuvieron una valoración muy favorable porque contienen suficiente información para ponerlas en funcionamiento, con respecto a las demás tácticas se obtuvo una valoración poco favorable, ya que no contienen información relativa para su ejecución.

Finalizado el trabajo sistemático, donde se realizó el estudio comparativo de las ventajas y desventajas de las estrategias arquitectónicas de usabilidad web las cuales se analizó las tácticas en tiempo de diseño y las tácticas en tiempo de ejecución, se logró identificar que en las tácticas en tiempo de diseño la que presentó mayor ventaja fue la táctica “diseño de interfaces de usuario” con un porcentaje de favorabilidad de 85 % en la valoración total. Como ventaja, presenta

una documentación completa por cada trabajo propuesto, donde se propone implementar el uso de patrones para lograr que un sistema sea usable, Sin embargo, como desventaja presenta una pequeña dificultad para su implementación, ya que requiere que el analista posea conocimientos básicos en métodos formales para ser capaz de entender la especificación de los patrones.

Por otra parte, en las tácticas en tiempo de ejecución, se logró identificar que la táctica que presentó mayor ventaja fue la táctica iniciática del sistema con un porcentaje de favorabilidad de 80% en la valoración total. Como ventaja presenta modelos y patrones de facilidad de uso, lo que hace que sea aplicable a proyectos reales.

3.3 DESARROLLO DE LA APLICACIÓN

Esta sección, va a enfocarse en el desarrollo de una aplicación web, aplicando la estrategia arquitectónica de usabilidad web, donde se lleva a cabo la práctica de la táctica diseño de interfaces de usuario y la táctica iniciativa del usuario, las cuales fueron las que obtuvieron mayor valoración, resultado del análisis comparativo.

El desarrollo de esta aplicación web se realizó con el fin de aplicar la estrategia arquitectónica de usabilidad web para transferir el conocimiento teórico en práctico y dar respuesta al requisito de calidad de usabilidad.

Para la construcción del producto software de la táctica tiempo de diseño, diseño de interfaces, se seleccionó los patrones propuestos por Tidwell, por estar en continuo análisis y perfeccionamiento y las directrices de usabilidad, las cuales se complementan con los patrones. Para la táctica tiempo de ejecución, iniciativa del sistema, se seleccionó algunas buenas prácticas de interacción con el usuario.

3.3.1 Ejemplos prácticos de patrones de diseño de interfaces de Tidwell. Los patrones de diseño se han introducido como un medio para captar y difundir los mejores conocimientos de diseño y prácticas. En el campo de la interacción computador-humano o (HCI) por sus siglas en inglés, profesionales e investigadores han explorado diferentes vías para usar patrones y lenguajes de patrones como diseño y herramientas. Este documento examina estas vías, utilizando tácticas de diseño de interfaces de usuario para crear un diseño basado en patrones, donde se utilizó los patrones propuestos por Tidwell.

Tidwell, estructura los patrones conforme a los diferentes aspectos del diseño de las interfaces de usuario, algunas de las categorías, son: organización de contenido, mecanismos de navegación, disposición de páginas, listas y tablas, acciones y comandos, formularios y controles, diseño móvil.

A continuación, se presentan los patrones que Tidwell propone en cada categoría los cuales aportan mejoras a la usabilidad del sistema en fase de diseño:

Tabla 29. Recopilación de patrones de Tidwell

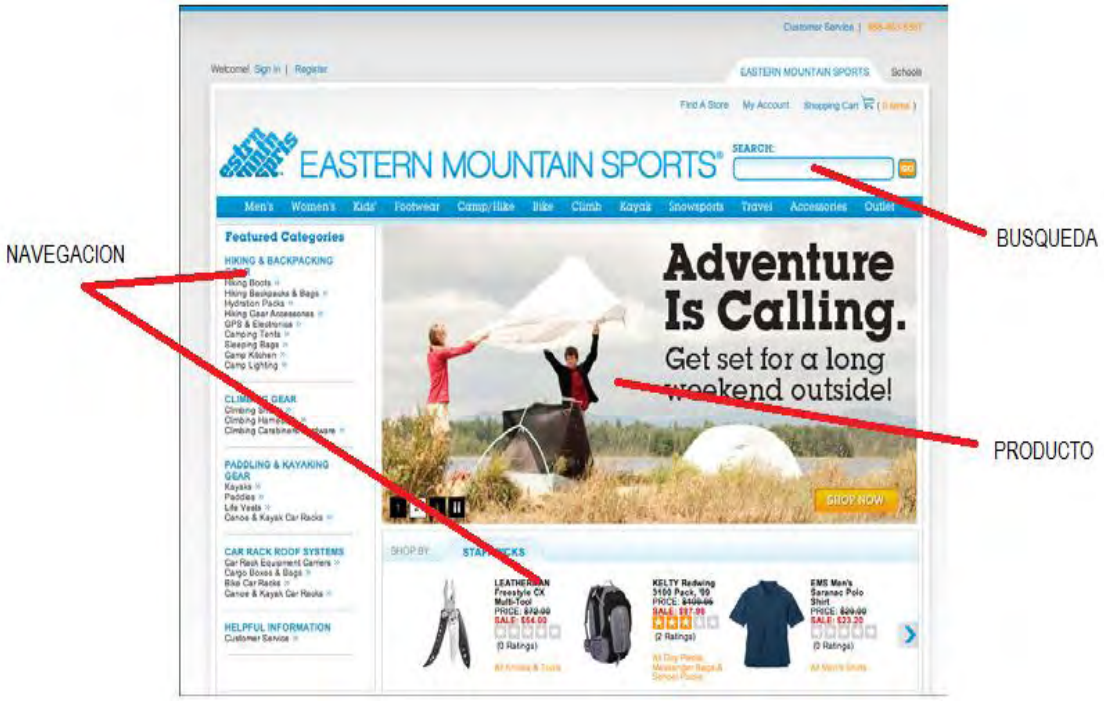
Categoría	**Problema que resuelven**	**Patrones**
Organización de contenido	Estructurar la organización del contenido del sitio web.	• Producto, búsqueda y navegación • Flujo de noticias • Vistas alternativas • Administrador de imágenes • Wizard
Mecanismos de navegación	Desbordamiento cognitivo, desorientación por parte del usuario.	• Migas de pan • Fat menú • Pie de página
Disposición de página	Disposición de los elementos dentro de las páginas para captar la atención del usuario.	• Rejillas iguales • Disposición líquida
Listas y tablas	Organización de los datos dentro del contexto.	• Carrusel • Paginación
Acciones y comandos	Manejo de instrucciones u órdenes que el usuario proporciona al sitio web.	• Grupo de botones • Herramientas flotantes • Botón hecho • Cancelabilidad
Formularios y controles	Control de la información enviada por el usuario.	• Mensajes de error • Contraseña fuerte • Autocompletado
Diseño móvil	Diseñar sitios web optimizados para dispositivos móviles.	• Lista infinita • Botón limpiar texto

A continuación, se presenta un ejemplo práctico de cada patrón organizado por categoría mencionado en la tabla anterior (ver tabla 29).

3.3.1.1 Organización de contenido.

• **Producto, búsqueda y navegación**. El patrón producto, búsqueda y navegación, presenta estos elementos juntos en muchos, sitios exitosos, ya que ofrecen diferentes formas de encontrar los elementos deseados, además brindan facilidad y comodidad al usuario (ver figura 10).

Figura 10. Patrón producto, búsqueda y navegación



Fuente. (Tidwell, 2011)
[En línea]< http://www.ems.com/>

Este patrón se lo puede aplicar a diferentes sitios web tales como medios informativos, sitios educativos y sitios web de comercio electrónico entre otros.

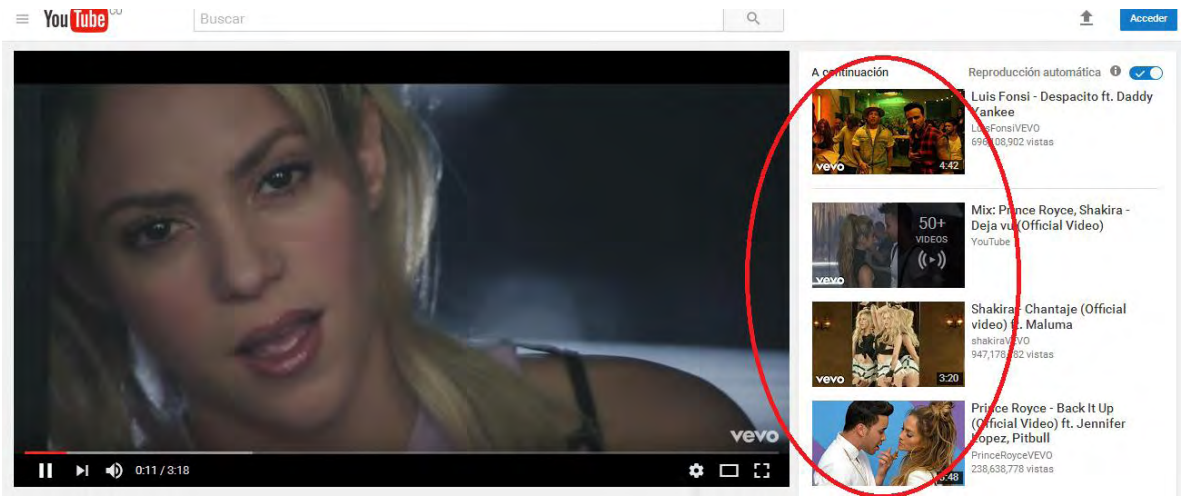
• **Flujo de noticias.** El patrón flujo de noticias muestra los elementos sensibles al tiempo en una lista cronológica inversa, con los últimos artículos en la parte superior. Se actualiza de forma dinámica, y combina los elementos de diferentes fuentes o personas en un solo stream (ver figura 11).

Figura 11. Patrón flujo de noticias

Fuente. (Tidwell, 2011)
[En línea]< http://designinginterfaces.com/patterns/news-stream/>

• **Vistas alternativas.** El patrón vistas alternativas permite al usuario elegir entre los puntos de vista alternativos que son sustancialmente diferente de la vista por defecto, por ejemplo, los sitios de noticias y blogs entre otros, a menudo muestran una gran cantidad de extras en los márgenes alrededor de un artículo, muchos de los cuales están animados o interactivos (ver figura 12).

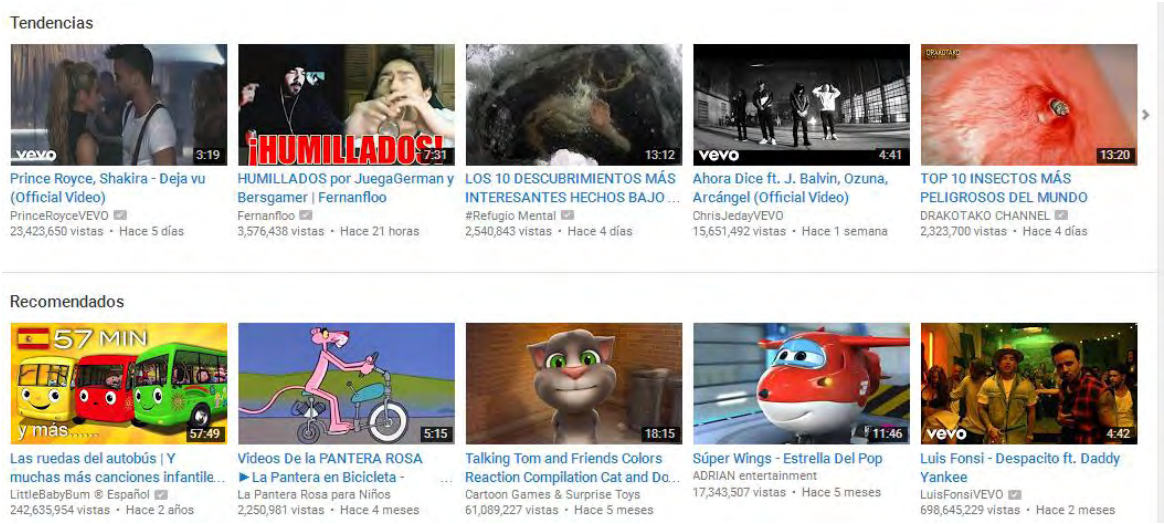
Figura 12. Patrón vistas alternativas



Fuente. (YouTube, 2017)
[En línea]< https://www.youtube.com/>

- **Administrador pictórico.** El patrón administrador de pictórico, utiliza miniaturas en inglés sé cómo thumbnails, que permite la gestión de fotos videos y otros elementos pictóricos ordenadamente (ver figura 13).

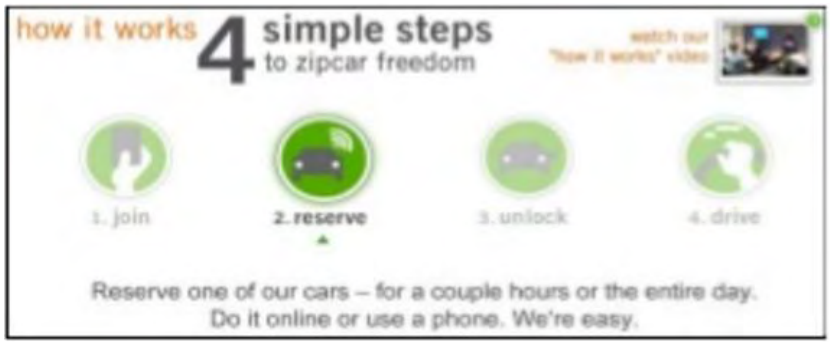
Figura 13. Patrón administrador pictórico



Fuente. (YouTube, 2017)
[En línea]< https://www.youtube.com/>

• **Wizard**. El patrón wizard, sugiere lo siguiente divide y conquistaras. Esto se entiende que al dividir la tarea en una secuencia en trozos, cada uno de los cuales puede ser tratado en un discreto "espacio mental" por el usuario, esto efectivamente simplifica la tarea por parte del usuario, donde se elabora una hoja de ruta pre-planificada a través de la tarea, ahorrando así al usuario el esfuerzo de averiguar la estructura de la tarea, todo lo que necesita el usuario es abordar cada paso a su vez, confiando en que si sigue las instrucciones, las cosas saldrán bien (ver figura 14).

Figura 14. Patrón wizard



Fuente. (Tidwell, 2011)
[En línea]< http://designinginterfaces.com/patterns/wizard/>

3.3.1.2 Mecanismos de navegación.

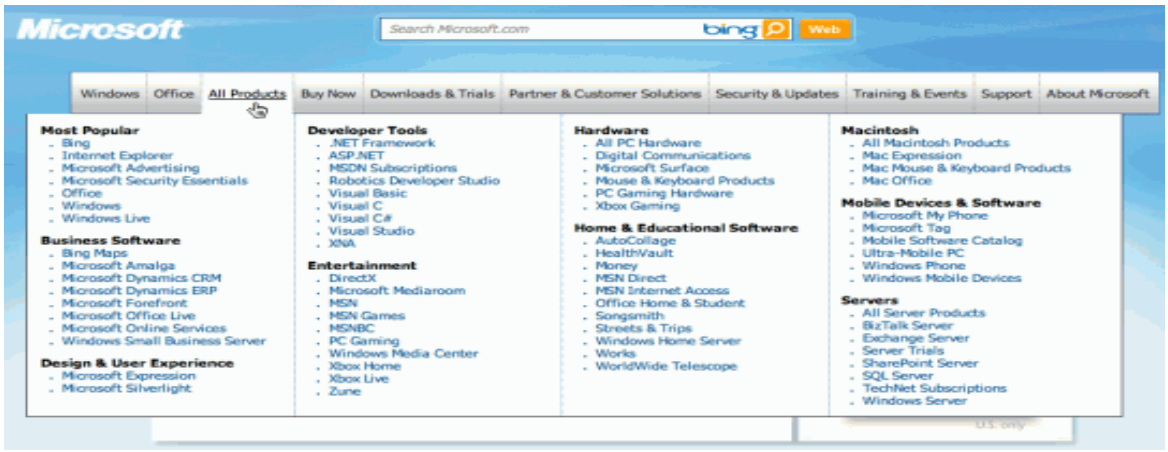
• **Migas de pan.** El patrón migas de pan, muestra una lista ordenada de páginas hasta llegar al origen, brinda una guía para ayudar a mejorar el posicionamiento donde el usuario se encuentre, para que este no se pierda navegando dentro de él sitio, lo que permite una mejor interacción con el sitio web (ver figura 15).

Figura 15. Patrón migas de pan

Fuente. (Alkosto, 2017)
[En línea]< http://www.alkosto.com/tv>

• **Fat menú.** El patrón fat menú, muestra una larga lista de opciones de navegación en los menús, se utiliza para mostrar todas las subpáginas en las secciones del sitio. Se suele utilizar cuando el sitio o aplicación tiene muchas páginas en muchas categorías, posiblemente en una jerarquía con tres o más niveles lo que hace que el sitio sea más descubrible y fácil de acceder a sus distintas secciones (ver figura 16).

Figura 16. Patrón fat menú



Fuente. (Microsoft, 2017)
[En línea] <https://www.microsoft.com/es-co>

• **Pie de página.** El patrón pie de página, hace que un sitio complejo sea más accesible, porque brinda una serie de opciones de navegación que permita al usuario continuar navegando por el sitio web sin tener que volver a buscar el menú principal. (ver figura 17).

Figura 17. Patrón pie de página

Fuente. (Tidwell, 2011)
[En línea]< http://designinginterfaces.com/patterns/sitemap-footer/>

3.3.1.3 Organización de la página

• **Rejillas iguales.** El patrón rejillas iguales, organiza los elementos de contenido en una rejilla o matriz. Cada elemento debe seguir un modelo común y el peso visual de cada elemento debe ser similar (ver figura 18).

Figura 18. Patrón rejillas iguales

Fuente. (Tidwell, 2011)
[En línea]< http://designinginterfaces.com/patterns/grid-of-equals/>

• **Disposición líquida.** El patrón disposición líquida, permite adaptar cada sitio a los diferentes formatos de dispositivos de acceso, siendo que esta sea flexible y adaptable (ver figura 19).

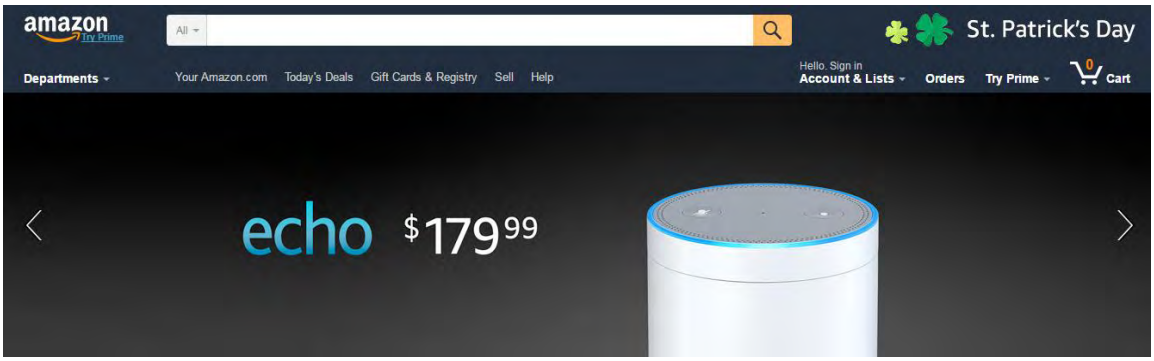
Figura 19. Patrón disposición líquida

Fuente. (Tidwell, 2011)
[En línea]< http://designinginterfaces.com/patterns/liquid-layout/>

3.3.1.4 Listas y tablas

• **Carrusel.** El patrón carrusel, ofrece una atractiva interfaz para navegar por elementos visuales, alentando al usuario, para inspeccionar los elementos que están a la vista y para ver lo que sigue. Un usuario no puede saltar fácilmente a un cierto punto en la lista, por lo que tiene que desplazarse a través de todo este patrón así fomenta la navegación y la curiosidad (ver figura 20).

Figura 20. Patrón carrusel



Fuente. (Tidwell, 2011)
[En línea]< http://designinginterfaces.com/patterns/

• **Paginación.** El patrón paginación, se debe implementar cuando se trata de mostrar grandes cantidades de información en una web, como, por ejemplo, productos de una tienda, resultados de una búsqueda o anuncios en una web de empleo entre otros. La implementación de una correcta interface de paginación garantiza en gran medida una buena experiencia del usuario que busca algo en un mar de opciones (ver figura 21).

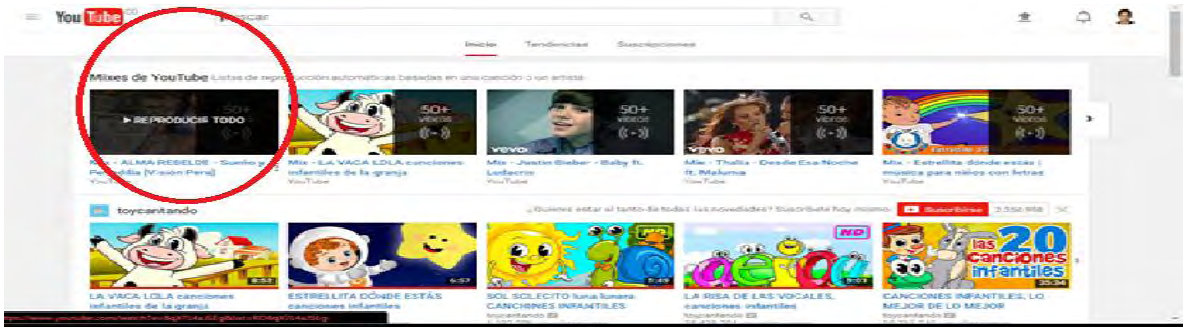
Figura 21. Patrón paginación

Fuente. (Google, 2017)
[En línea]< https://www.google.com.co/>

3.3.1.5 Acciones y comandos

• **Herramientas flotantes.** El patrón herramientas flotantes, se presentan cuando el usuario coloca el puntero sobre algún elemento, pero no necesariamente este está activo, puede al usuario despertar el gesto de llamarle la atención (ver figura 22).

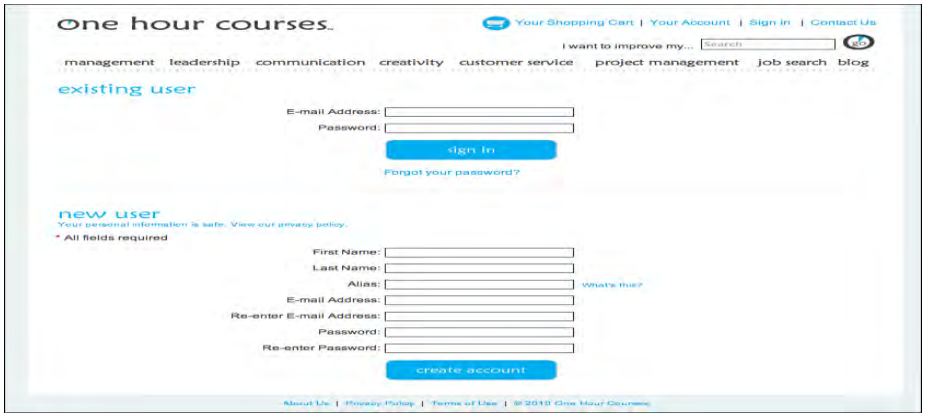
Figura 22. Patrón herramientas flotantes



Fuente. (YouTube, 2017)
[En línea]< https://www.youtube.com/>

• **Botón hecho.** El patrón botón hecho, se coloca un botón que finaliza una transacción al final del flujo visual, este debe ser grande y que llame la atención, donde se puede observar los botones de acción sin siquiera leer las etiquetas, debido al diseño visual (ver figura 23).

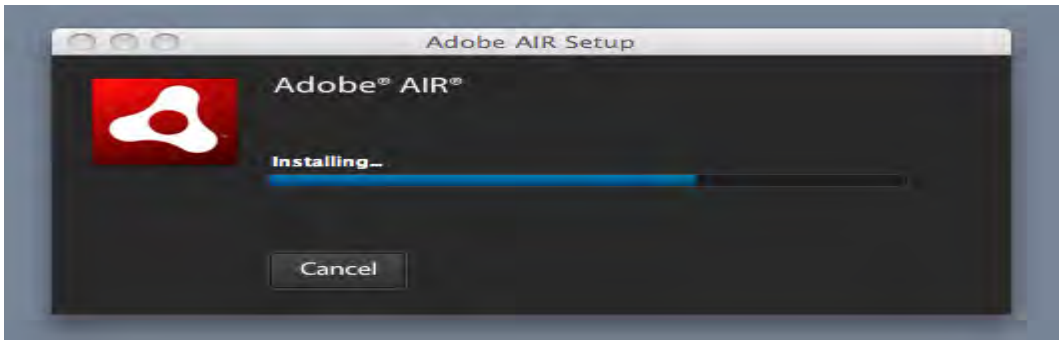
Figura 23. Patrón botón hecho



Fuente. (Twitter, 2017)
[En línea]< https://twitter.com//>

• **Cancelabilidad.** Los usuarios pueden o cambian de idea, una vez que se inicia una operación que requiere mucho tiempo, un usuario puede querer detener una acción, especialmente si un indicador de progreso le dice que va a tomar un tiempo. Este patrón de cancelabilidad sin duda ayuda con la prevención de errores y proporciona la opción de cancelar una transacción en cualquier momento (ver figura 24).

Figura 24. Patrón cancelabilidad



Fuente. (Adobe Air, 2017)
[En línea]<https://get.adobe.com/es/air/>

3.3.1.6 Formularios y controles

• **Contraseña fuerte.** El patrón contraseña fuerte, proporcione al usuario información inmediata sobre la validez y la seguridad de su contraseña (ver figura 25).

Figura 25. Patrón contraseña fuerte

Fuente. (Gmail, 2017)
[En línea]< https://mail.google.com>

• **Mensajes de error.** El patrón mensajes de error, proporciona información a los usuarios cuando estos introducen información a un formulario o envían información (ver figura 26).

Figura 26. Patrón mensajes de error

Fuente. (Twitter, 2017)
[En línea]< https://twitter.com//>

• **Autocompletado.** El patrón de autocompletado es un patrón de búsqueda, que medida que el usuario escribe en un campo de texto, se anticipa las posibles respuestas y muestra una lista seleccionable y se completa automáticamente la entrada cuando corresponda (ver figura 27).

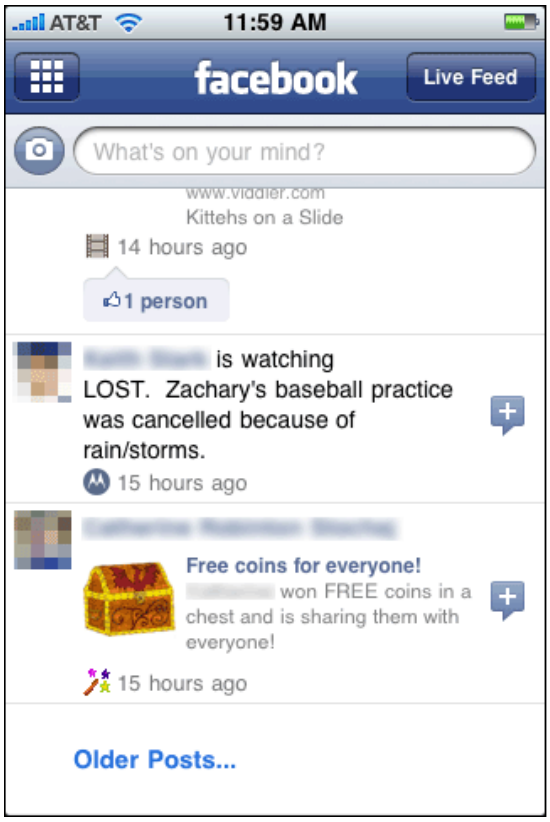
Figura 27. Patrón autocompletado

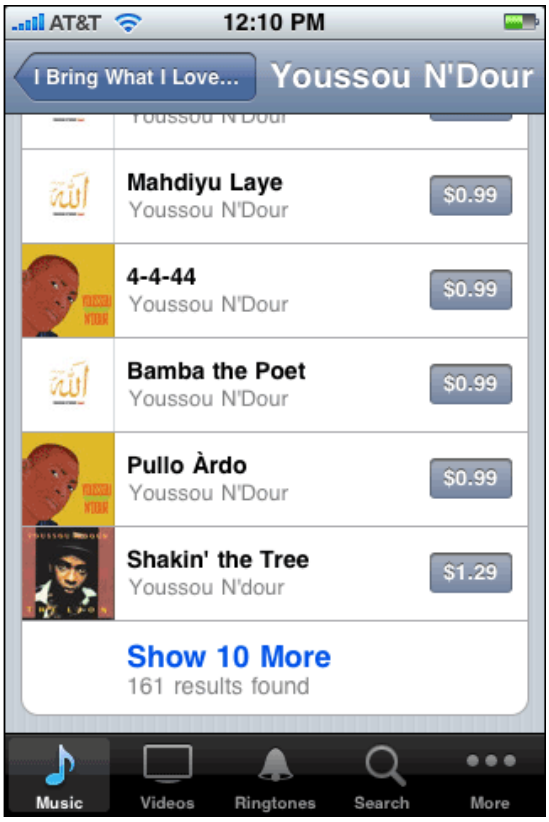
Fuente. (Google, 2017)
[En línea]< https://www.google.com.co/>

3.3.1.7 Diseño móvil

• **Lista infinita.** El patrón lista infinita, se ubica en la parte inferior de una página si hay más elementos se coloca un botón que carga o incremente la lista (ver figura 28).

Figura 28. Patrón lista infinita

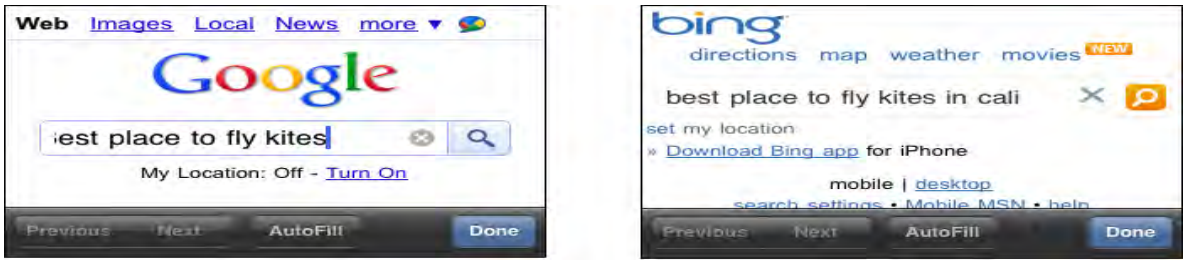




Fuente. (Facebook, 2017)
[En línea] <www.facebook.com>

• **Botón limpiar texto.** El patrón limpiar texto, se ubica un campo en una entrada que con solo al presionar un botón sin necesidad de borrar paso a paso borre toda la entrada (ver figura 29).

Figura 29. Patrón botón limpiar texto

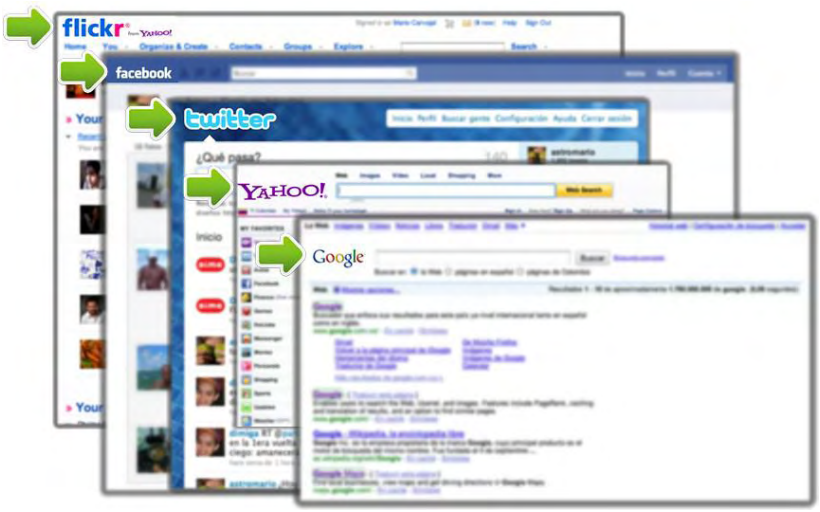


Fuente. (Google, 2017)
[En línea]< https://www.google.com.co/>

3.3.2 Ejemplos prácticos de directrices de usabilidad. Presentan un conjunto de directrices de usabilidad relacionadas con la facilidad de aprendizaje y de uso, para el diseño de aplicaciones web, lo que fortalecer el diseño de interfaz para que contemple aspectos de usabilidad. Las directrices de usabilidad son una serie de lineamientos, se las considera como guías a seguir, las cuales, complementadas con los patrones, ayudan concebir un diseño que contemple el atributo de usabilidad, porque estas siguen un diseño web centrado en el usuario el cual se caracteriza por asumir que todo el proceso de diseño del sitio web debe estar encaminado para el usuario. A continuación, se presenta ejemplos prácticos de directrices de usabilidad.

3.3.2.1 Ubicación del logotipo. Se debe colocar en la parte superior izquierda, en lugar de la derecha, siendo esta la mejor ubicación para colocar el logo en su página web. El logotipo de los sitios más visitados del mundo como son: Facebook, Twitter, Yahoo! y Google. En todos los sitios anteriormente mencionados, el logotipo se encuentra en la parte superior izquierda y tienen un hipervínculo con la página de inicio (ver figura 30).

Figura 30. Directriz ubicación del logotipo



Fuente. (Directrices de usabilidad para sitios web del Estado colombiano, 2010)

3.3.2.2 Contenido que parece publicidad. Se debe evitar que los contenidos y elementos importantes del sitio web sean confundidos con publicidad. Un estudio de Eyetracking de Nielsen Norman Group, en el que se evidencia la ceguera al banner (ver figura 31).

Figura 31. Directriz contenido que parece publicidad

Fuente. (Directrices de usabilidad para sitios web del Estado colombiano, 2010)

3.3.2.3 Contenidos de ejemplo en la página de inicio. Se debe diseñar en la página de inicio contenidos que ejemplifican con claridad el resto del sitio web (ver figura 32).

Figura 32. Directriz contenidos de ejemplo en la página de inicio

Fuente. (Directrices de usabilidad para sitios web del Estado colombiano, 2010)

3.3.2.4 Campos obligatorios. Se debe distinguir claramente los campos obligatorios de los opcionales en un formulario. En el formulario de registro de Netflix se identifican los campos obligatorios con un signo de admiración de color rojo (ver figura 33).

Figura 33. Directriz campos obligatorios

Fuente. (Netflix, 2017)
[En línea]< https://www.netflix.com/co/>

3.3.2.5 Asociación de etiquetas y campos. Se debe asociar claramente las etiquetas con los campos de formulario. Cuando se emplee listas de selección se usa el valor por defecto como etiqueta para el campo. Esta práctica puede reducir el tiempo de lectura y hacer más comprensible y legible el formulario (ver figura 34).

Figura 34. Directriz asociación de etiquetas y campos

3.3.2.6 Motor de búsqueda y ubicación. Se debe proporcionar un motor de búsqueda interno en todas las páginas del sitio web y ubicarlo preferiblemente en la parte superior derecha. El portal de noticias CNN provee un buscador suficientemente ancho y visible, en la parte superior derecha de todas las páginas de su sitio (ver figura 35).

Figura 35. Directriz motor de búsqueda y ubicación

Fuente. (Directrices de usabilidad para sitios web del Estado colombiano, 2010)

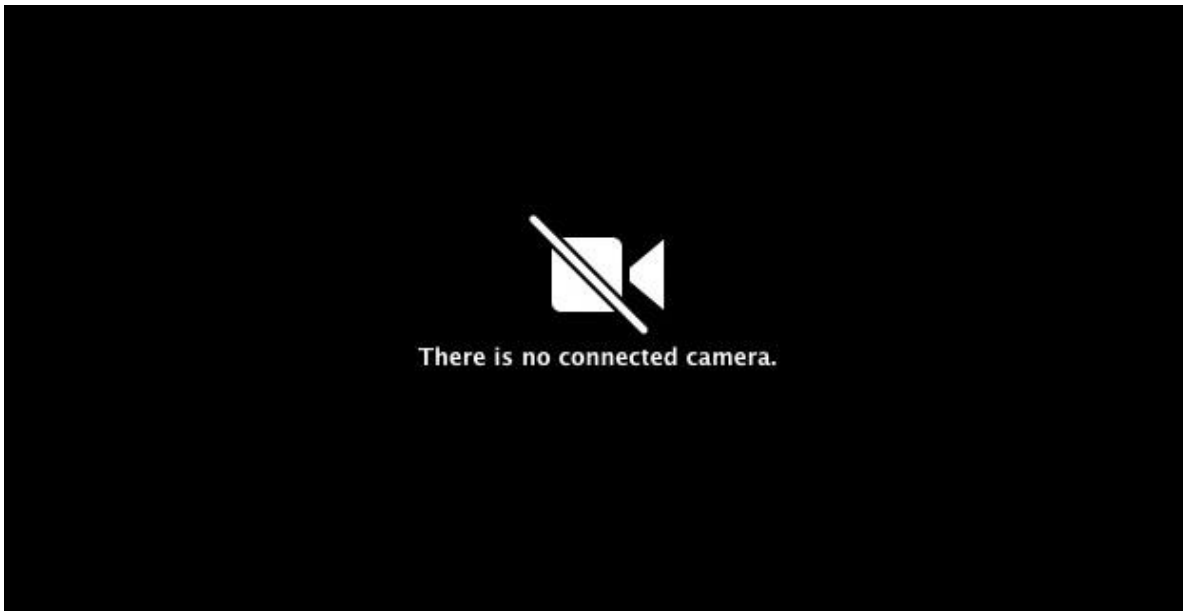
3.3.3 Ejemplos prácticos mantener un modelo del sistema. Estas tácticas consisten en brindar una feedback apropiado por parte del sistema al usuario, lo que se entiende como retroalimentación su objetivo es mantener informado al usuario sobre lo que está ocurriendo en todo momento dentro del sistema. Feedback como mecanismo se divide en:

- System status feedback.
- Interaction feedback.
- Progress feedback.
- Warning.

A continuación, se presenta un ejemplo práctico, por cada subdivisión del mecanismo feedback.

3.3.3.1 System status feedback. Este mecanismo de usabilidad informa al usuario de cambios relevantes que se producen en el sistema o cuando se produce un fallo en los siguientes contextos: durante la ejecución de un servicio; antes de la ejecución de un servicio; después de la ejecución de un servicio, porque no hay suficientes recursos; porque los recursos externos no están trabajando adecuadamente. En hangout cuando falla la comunicación con un dispositivo externo (ver figura 36).

Figura 36. Mecanismo system status feedback



Fuente. (Google, 2017)
[En línea]< https://hangouts.google.com >

3.3.3.2 Progress feedback. El objetivo de este mecanismo de usabilidad es el de mostrar el tiempo restante para la finalización o inicio de un servicio lanzado por el usuario. La página web de mega brinda una barra de progreso circular del tiempo que tarda en cargar su página, así como el tiempo esperado y el porcentaje cuando se realiza una descarga presentada en una barra de progreso (ver figura 37).

Figura 37. Mecanismo progress feedback

Fuente. (Mega, 2017)
[En línea]< https://mega.nz/ >

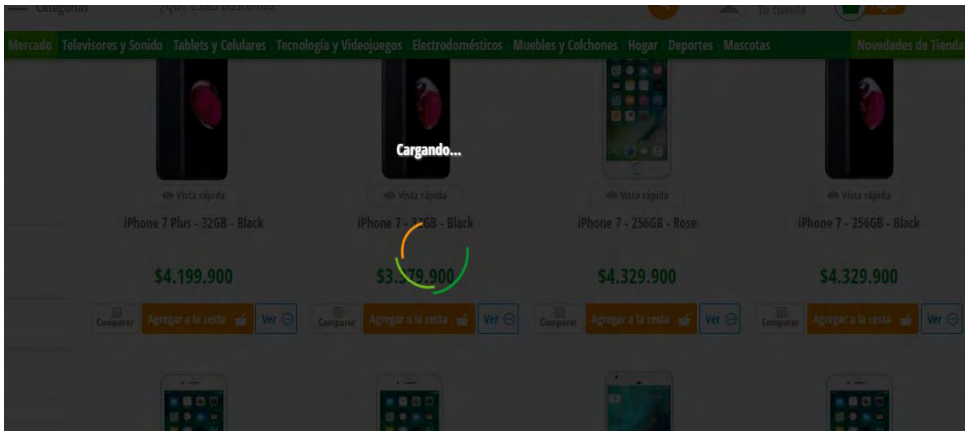
3.3.3.3 Warning. Este mecanismo de usabilidad tiene la funcionalidad de solicitar confirmación al usuario en caso de que el servicio solicitado tenga consecuencias irreversibles. Facebook brinda una alerta antes de eliminar una foto, ofreciendo la posibilidad de cancelar la acción (ver figura 38).

Figura 38. Mecanismo warning

Fuente. (Facebook, 2017)
[En línea] <www.facebook.com>

3.3.3.4 Interaction feedback. Este mecanismo de usabilidad se introduce en los sistemas para que el usuario sea informado de que sus peticiones de interacción han sido recibidas, se comprende cuando se produzca un evento (ver figura 39).

Figura 39. Mecanismo interaction feedback



Fuente. (Tiendas Jumbo, 2017)
[En línea]< http://www.tiendasjumbo.co/>

3.3.4 Construcción del producto software. Para poner en funcionamiento las tácticas establecidas con mayor valoración en el orden de favorabilidad, se construyó un producto software mediante iteraciones e incrementos, dando respuesta a los requerimientos que se muestran en la tabla 30. Los elementos técnicos establecidos para el desarrollo del producto software, se puede observar en el anexo C.

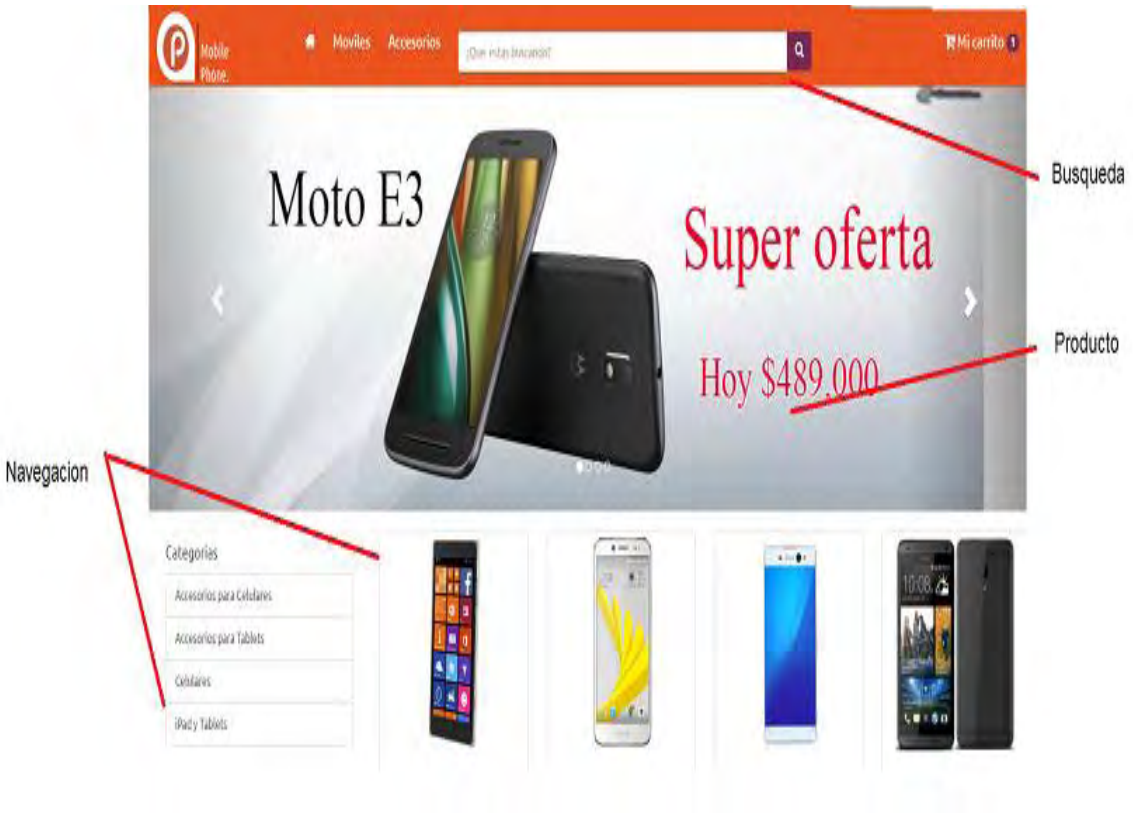
Tabla 30. Requerimientos del producto software

Descripción	Tipo
Como posible comprador quiero buscar un producto por el nombre	Funcional
Como comprador quiero agregar un producto al carrito de compras	Funcional
Como administrador del negocio quiero vender productos a través de Internet	No Funcional
Como administrador del negocio quiero que los datos de las ventas perduren en el tiempo	No Funcional
Como administrador del negocio quiero que la aplicación sea de fácil operación por los usuarios.	No Funcional

A continuación, se presenta la aplicación de las tácticas de mayor favorabilidad identificadas, en el producto software construido.

3.3.4.1 Implementación de las tácticas en tiempo de diseño. Para la construcción del producto software se implementaron, las tácticas diseño de interfaces de usuario la cual fue la que obtuvo mayor valoración resultado del modelo matemático, donde se llevó a la práctica los patrones de Tidwell y las directrices de usabilidad.

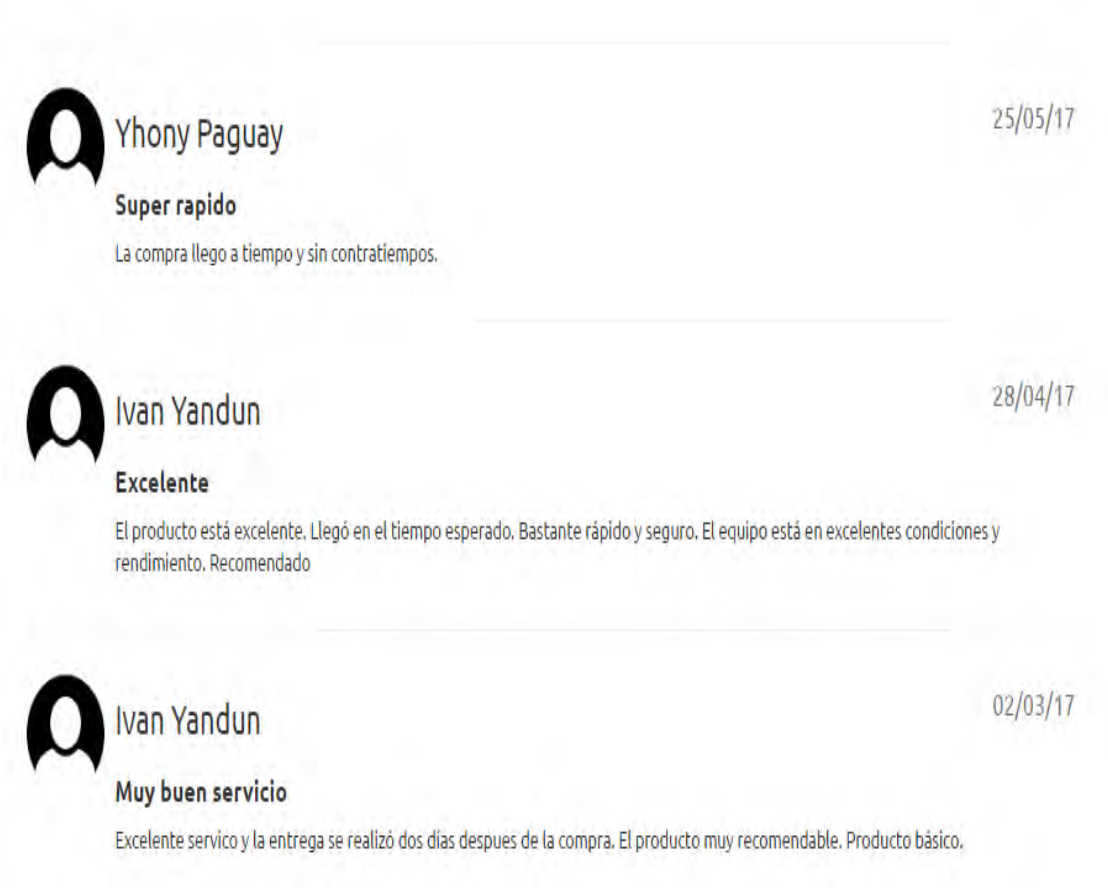
Figura 40. Implementación del patrón producto, búsqueda y navegación



En la figura 40, se muestra la implementación del patrón producto, búsqueda y navegación, que consiste en colocar tres elementos en la página principal del sitio, el cuadro de búsqueda, el panel de navegación y el producto. Este conjunto de elementos resulta para el usuario una manera familiar de navegar, ya que, aunque no conozcan el sitio tienen una clara idea de cómo moverse por él.

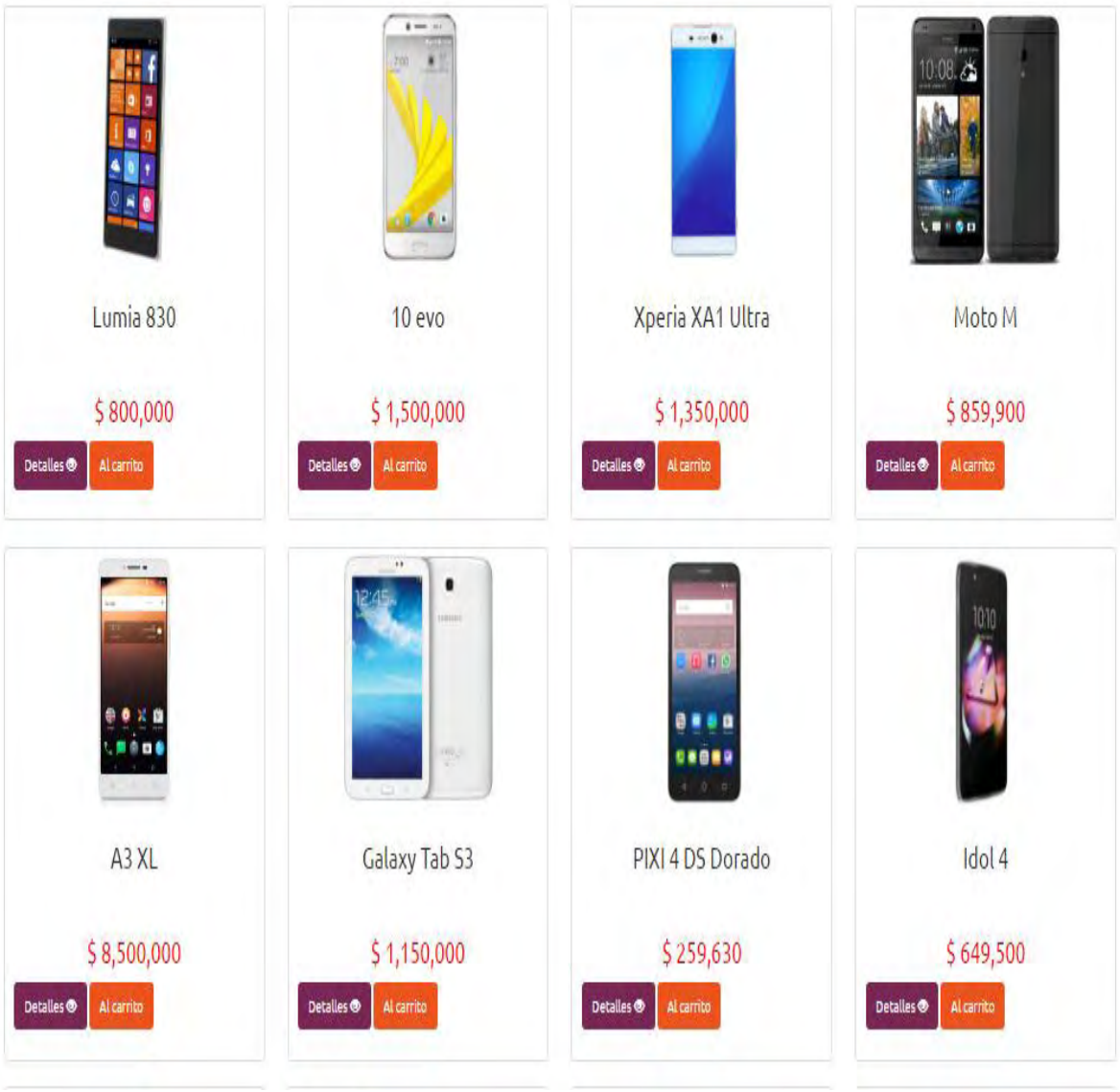
.

Figura 41. Implementación del patrón flujo de noticias



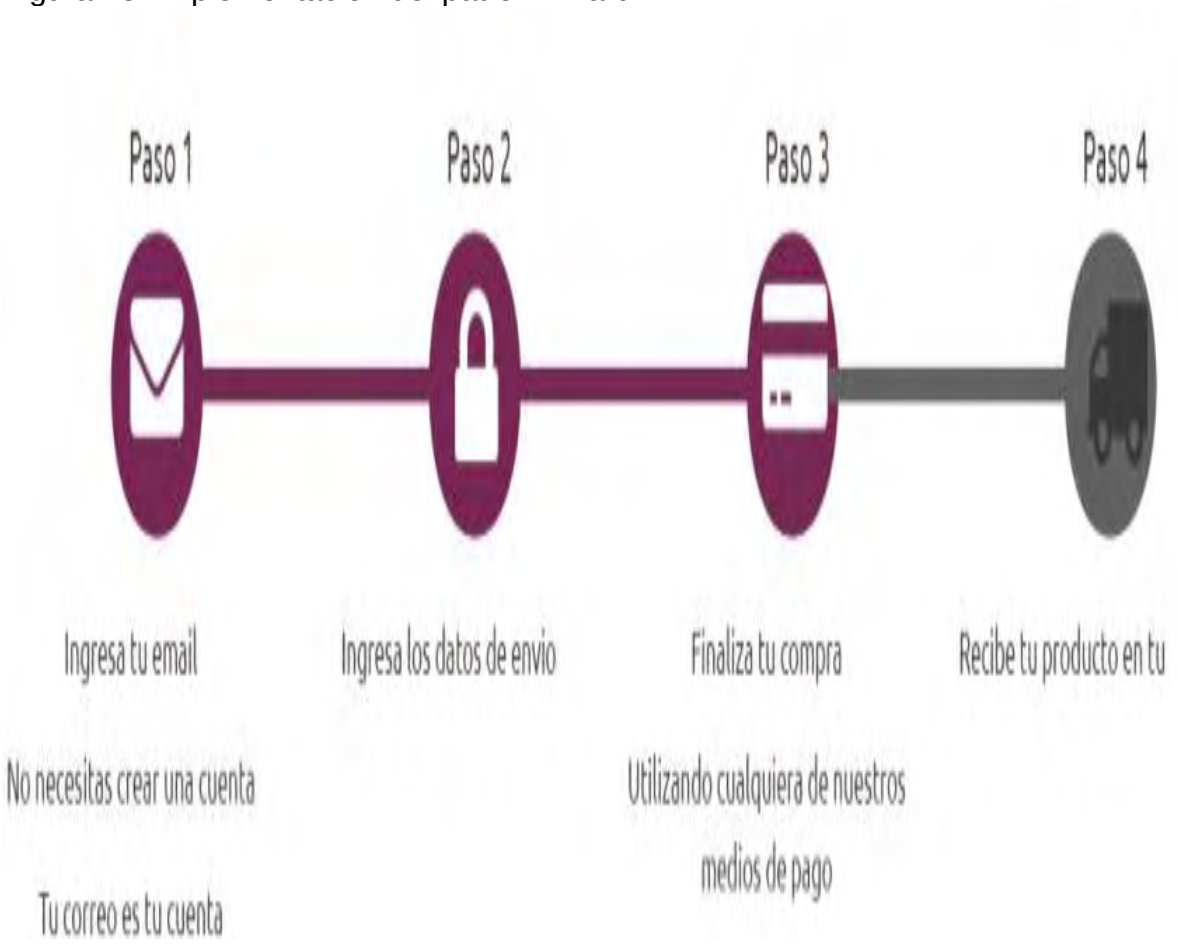
En la figura 41, se muestra la implementación del patrón flujo de noticias, que consiste en mostrar los elementos en una lista cronológica inversa, es decir las últimas noticias aparecen en la parte superior, este patrón es generalmente, usado en sitios o aplicaciones de redes sociales, noticias, correos electrónicos, blogs, entre otros. Mediante el patrón flujo de noticias se le facilita al usuario, mantenerse informado de acontecimientos recientes, ya que puede seguirles el ritmo a las noticias.

Figura 42. Implementación del patrón administrador pictórico



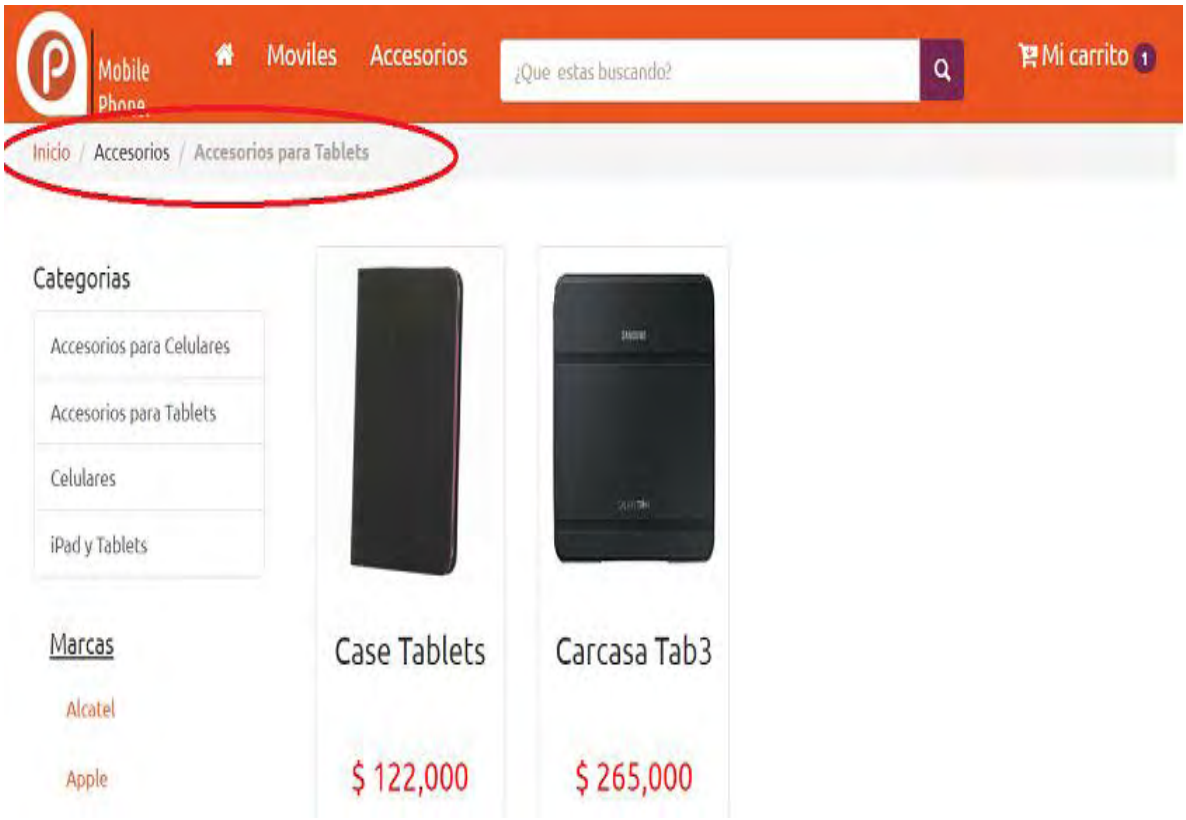
En la figura 42, se muestra la implementación del patrón administrador pictórico, en la que se utilizó una interfaz para la gestión de las imágenes, mediante miniaturas, vista de elementos y una interfaz de navegación. Que brindan facilidad en la organización, navegación y vista de la información.
En la figura 43, se puede observar la implementación del patrón wizard.

Figura 43. Implementación del patrón wizard



En la figura 43, se muestra la implementación del patrón wizard, que consiste en guiar al usuario a través de la interfaz, mostrar paso a paso cada una de las tareas, donde en cada una se puede escoger o cambiar valores preestablecidos, en un orden prescrito. Al dividir las tareas por partes, hace que el usuario entienda mejor lo que está ejecutando y a su vez terminar el o los procesos de forma satisfactoria.

Figura 44. Implementación del patrón migas de pan

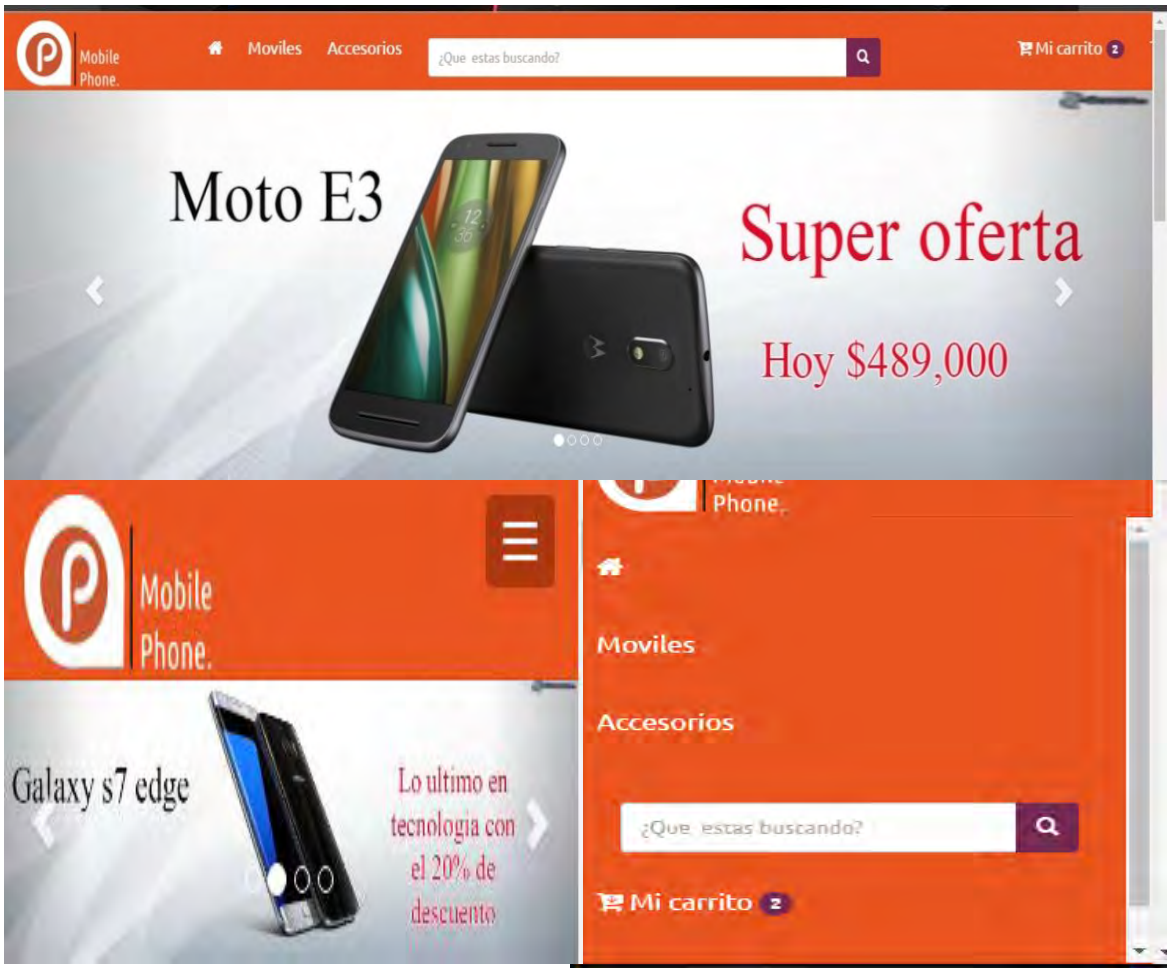


En la figura 44, se muestra la implementación del patrón migas de pan, donde se puede observar cada nivel de jerarquía con respecto a la página actual, desde lo más alto de la aplicación hasta los más abajo. En buen sentido, muestra una parte lineal del mapa general de la aplicación, este patrón ayuda al usuario a saber en dónde se encuentra ubicado. Esto es muy útil si el usuario ha realizado un salto abrupto hacia algún lugar profundo en el árbol de jerarquía, como resultado de una búsqueda o una herramienta de navegación. Por último, usualmente las migas de pan son enlaces o botones sobre los que se puede hacer clic, lo que los convierte en dispositivos de navegación independientes.

Figura 45. Implementación del patrón pie de página

En la figura 45, se muestra la implementación del patrón pie de página, que consiste en una representación de la estructura de un sitio utilizado para la navegación sobre el mismo. Esto permite tener una vista general de los contenidos del sitio de un solo vistazo. Al utilizar un patrón pie de página, los visitantes pueden saltar directamente a cualquier página listada. Este patrón hace que los sitios complejos sean más fáciles de explorar.

Figura 46. Implementación del patrón disposición liquida

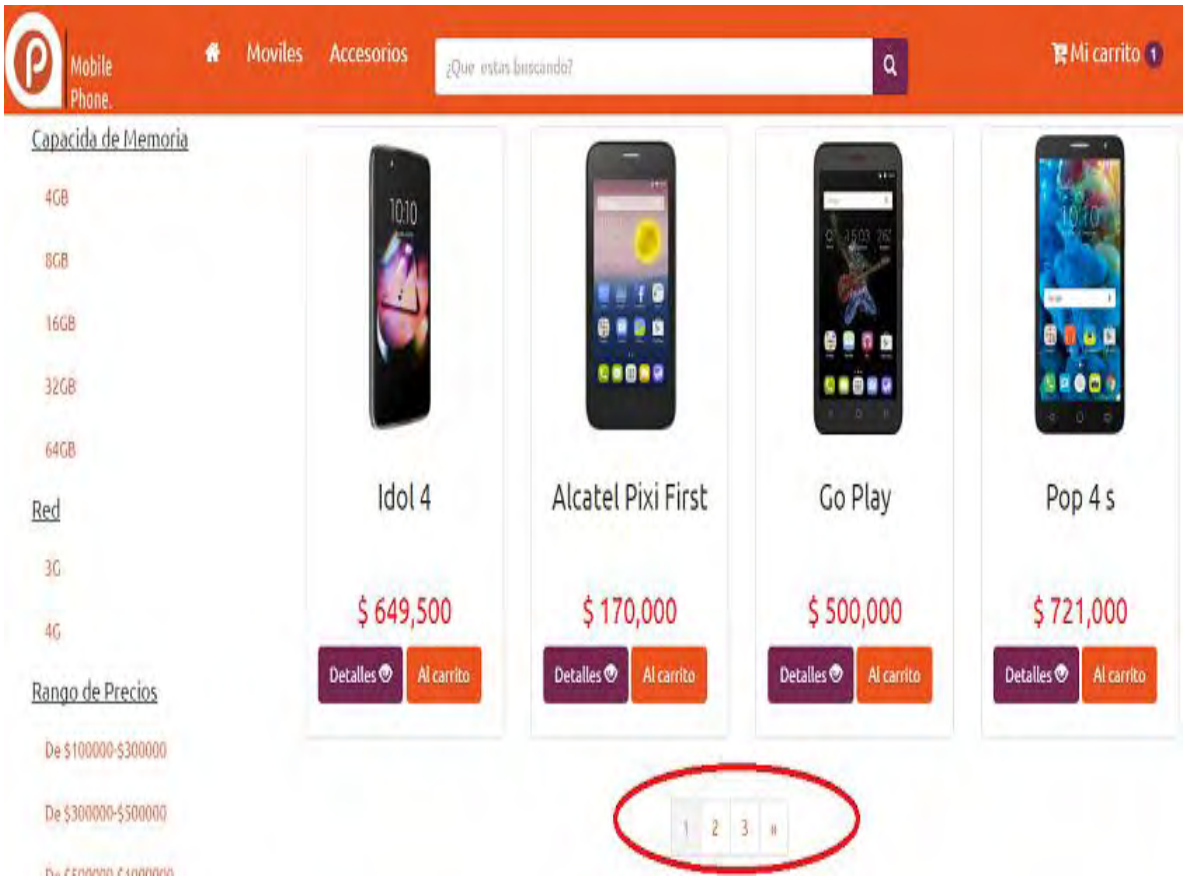


En la figura 46, se muestra la implementación del patrón, disposición líquida, este patrón surge ante la necesidad ver el contenido de un tamaño adaptable en el dispositivo que se esté usando, al tamaño de la pantalla del usuario lo que se conoce como “Responsive design”. Además, mejora la experiencia de navegación del usuario al ofrecer una página web adaptable, flexible y optimizada para el dispositivo que esté utilizando.

Figura 47. Implementación del patrón carrusel

En la figura 47, se muestra la implementación del patrón carrusel, donde se observa una moderada cantidad de elementos mediante una interfaz circular que se extiende de manera horizontal, desde la cual el usuario puede visualizar varias imágenes o contenido los cuales rotan en intervalos o moverse según gusto del usuario. Este patrón es usado muy comúnmente en tiendas virtuales y la mayor parte de sitios web por su atractivo visual. Además, genera un impacto visual agradable al usuario, el cual reconoce fácilmente el contenido de los elementos.

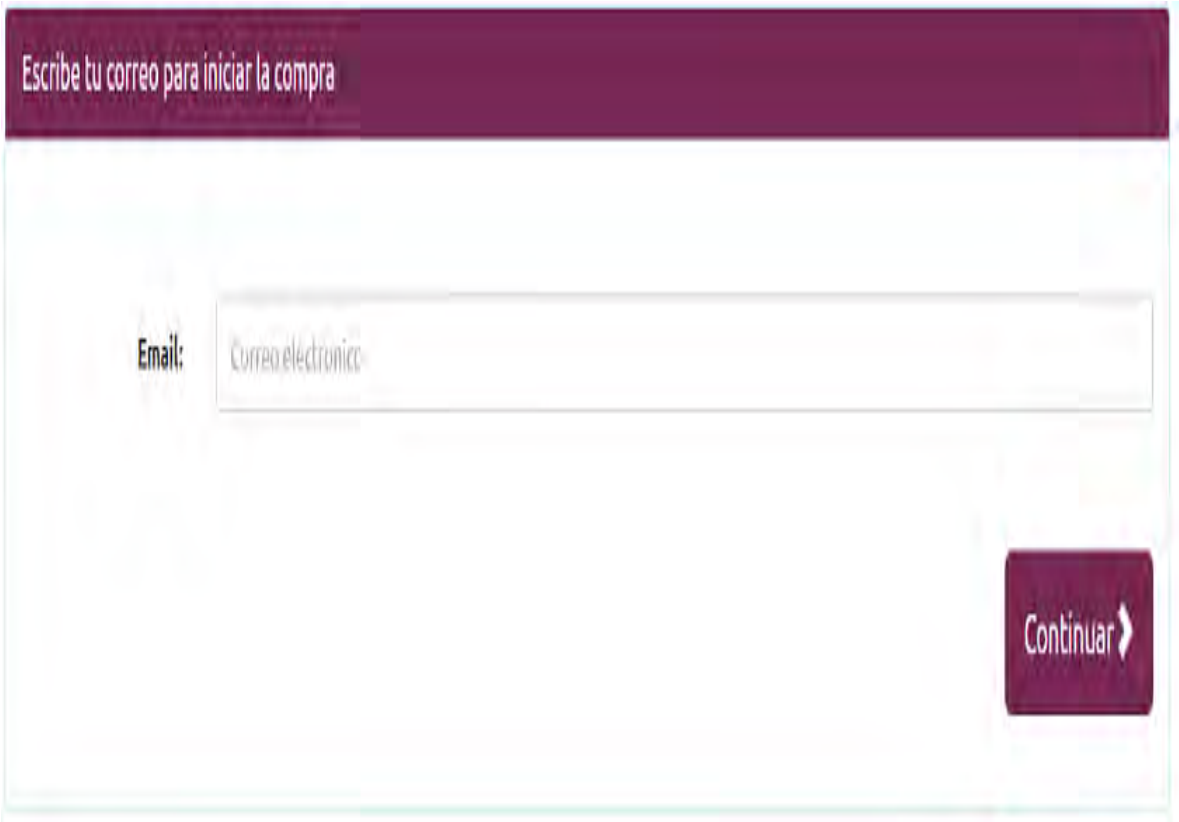
Figura 48. Implementación del patrón paginación



En la figura 48, se muestra la implementación del patrón paginación, que consiste en mantener el tamaño de cada página dentro de lo manejable en cuanto a navegación e interfaz. Al mismo tiempo, permite reducir el tamaño de la página la cantidad de información a transferir. La implementación de una correcta interfaz de paginación garantiza en gran medida una buena experiencia del usuario que busca algo en un mar de opciones

.

Figura 49. Implementación del patrón botón hecho

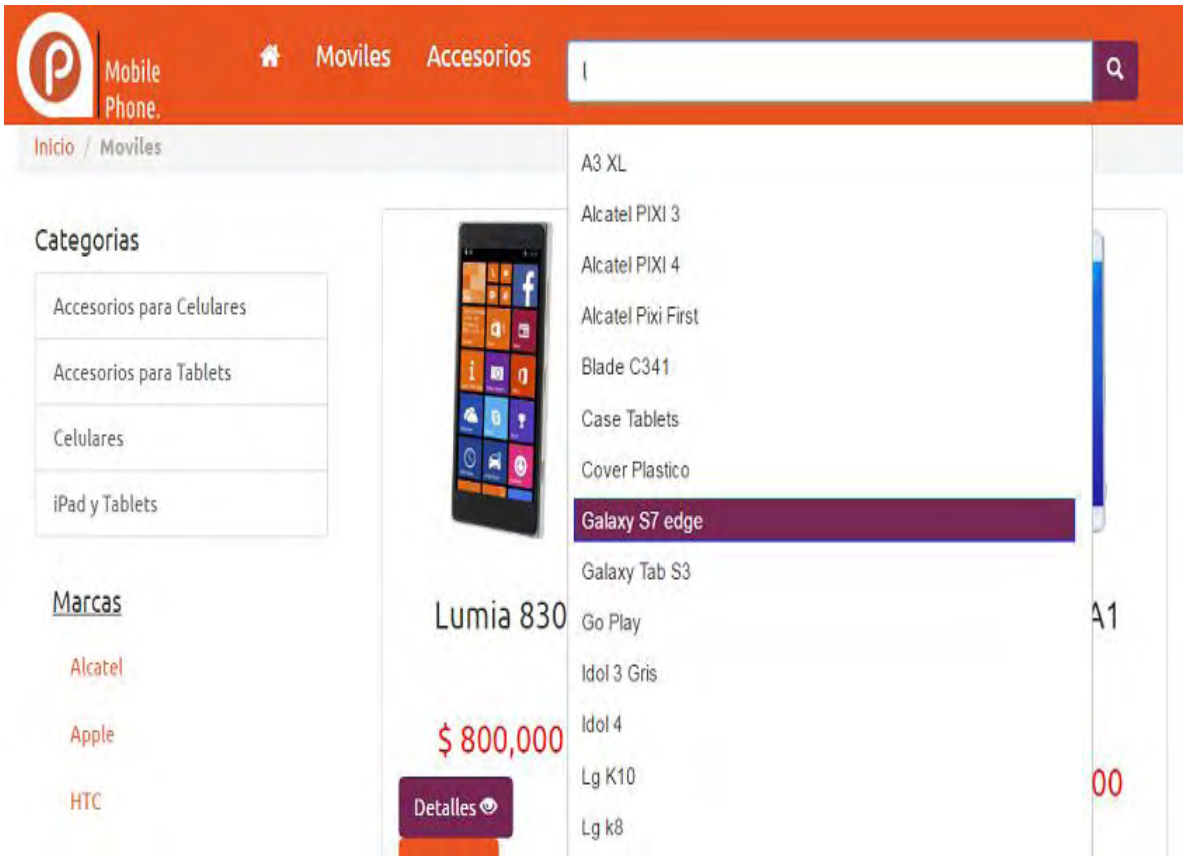


En la figura 49, se muestra la implementación del patrón botón hecho, donde se ubica un botón que finaliza una transacción al final del flujo visual; este debe ser grande y bien etiquetado. El tener un último paso en la transacción da el sentido de terminación. El usuario entiende que la transacción se realizará cuando se oprima el botón. Además, ofrece al usuario la posibilidad de realizar sus procesos de forma fluida y continúa evitando detenciones no deseadas para buscar el botón para finalizar la acción.

Figura 50. Implementación del patrón mensajes de error

En la figura 50, se muestra la implementación del patrón mensajes de error, que consiste en mostrar mensajes de error en la misma página de un formulario, con indicadores a la par de donde se origina el error. Se utiliza en formularios donde el usuario pueda ingresar datos que no son aceptados, cuando se pueda olvidar de llenar un campo, direcciones de correo inválidas, entre otros. Además, brinda una idea al usuario de qué tipo de error sucedió, debido a los datos ingresados en un campo en específico y se evita que se ingresen datos incorrectos que puedan afectar el funcionamiento de la aplicación.

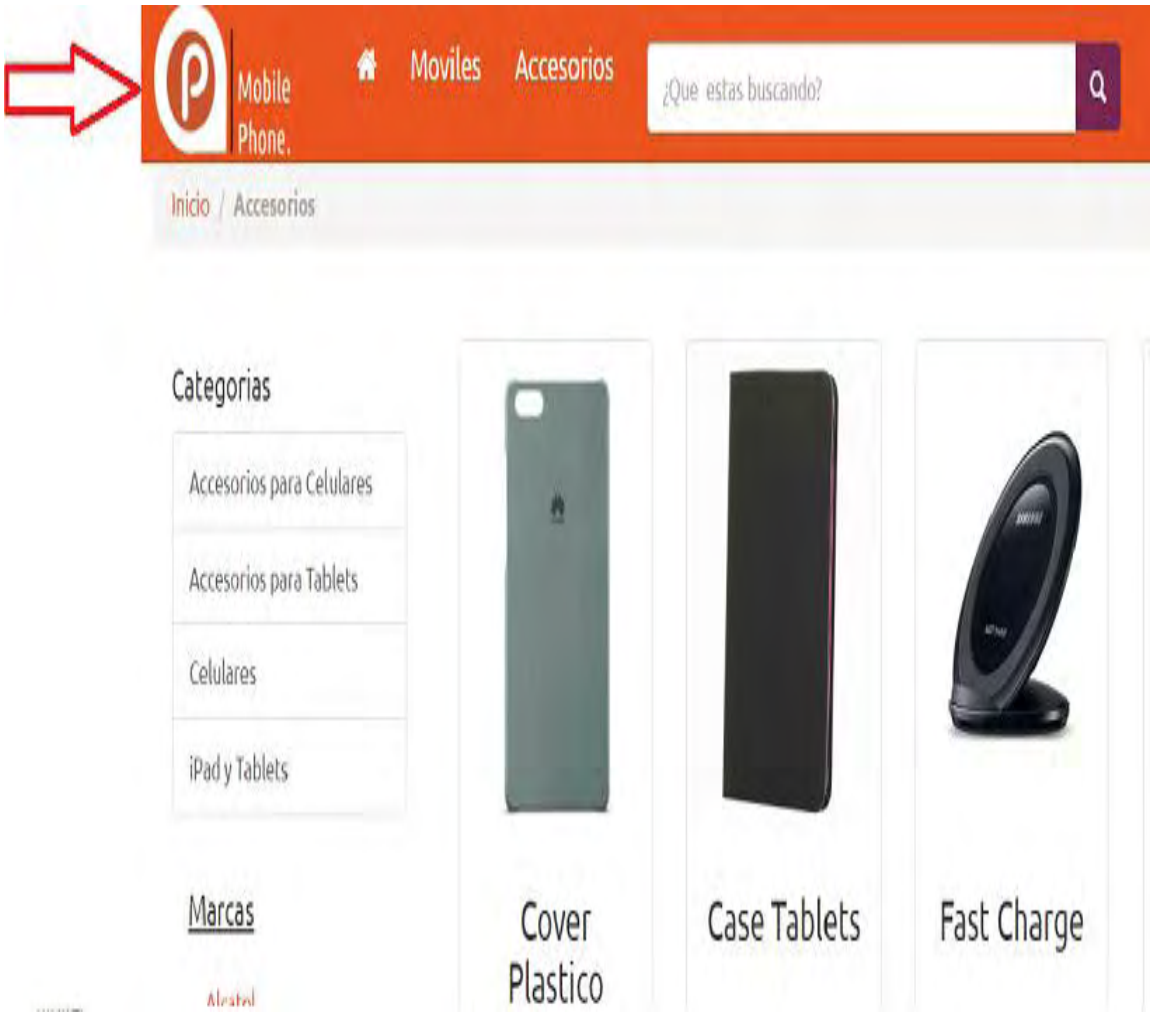
Figura 51. Implementación del patrón autocompletado



En la figura 51, se muestra la implementación del patrón autocompletado, que consiste en autocompletar un campo de texto mientras el usuario escribe en este, anticipando posibles respuestas y mostrándolas en una lista. El uso de este patrón ahorra tiempo al usuario, así este no tiene que escribir más de lo necesario cuando la lista muestra la respuesta que éste busca, especialmente con palabras muy largas o direcciones de correo que pueden ser olvidadas, se previenen también errores, una hilera de texto larga tiene más probabilidades de tener un error, que con entradas autocompletadas no suceden.

.

Figura 52. Implementación de la directriz ubicación del logotipo



En la figura 52, se muestra la implementación de la directriz ubicación del logotipo, donde se ubicó el logotipo de la aplicación en la parte superior izquierda, siendo esta la mejor ubicación para colocar el logo en la página web. El logotipo de los sitios más visitados del mundo como son: Facebook, Twitter, Yahoo! y Google. En todos los sitios anteriormente mencionados, el logotipo se encuentra en la parte superior izquierda y tienen un hipervínculo con la página de inicio

Figura 53. Implementación de la directriz campos obligatorios

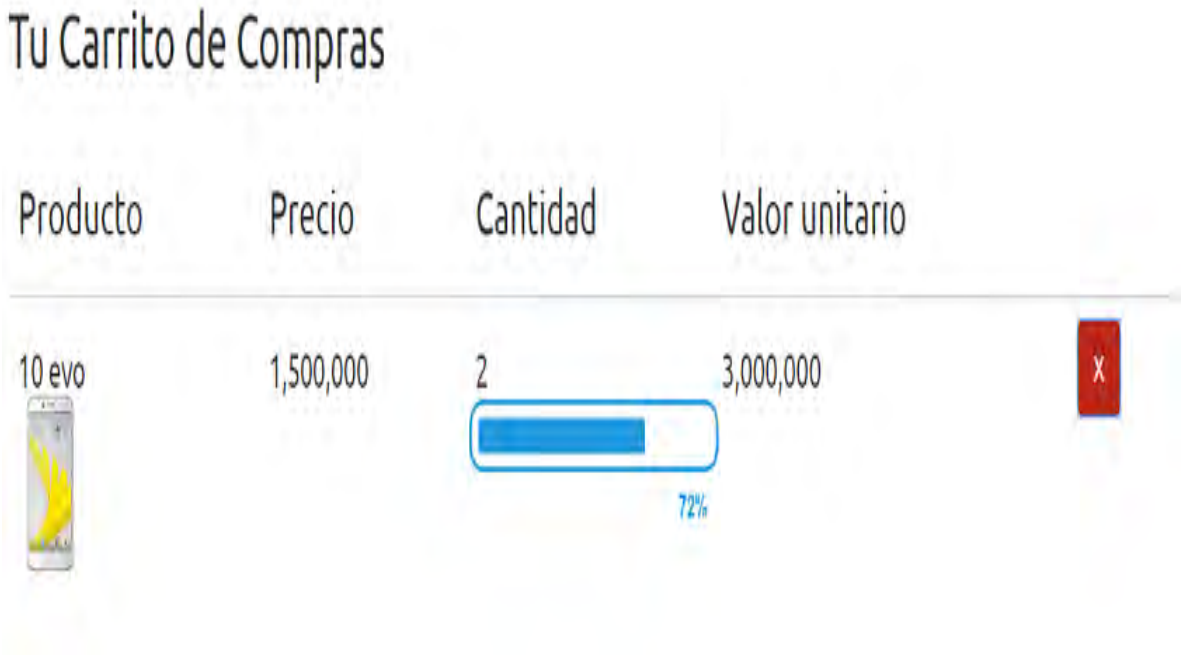
En la figura 53, se muestra la implementación de la directriz campos obligatorios, donde se distingue claramente los campos obligatorios del formulario indicado con el texto “Este campo es obligatorio”. No llenar un campo obligatorio desencadena en una página errores por ausencia de datos.

Figura 54. Implementación de la directriz asociación de etiquetas y campos

En la figura 54, se muestra la implementación de la directriz asociación de etiquetas y campos, esta directriz hace que se tenga una interacción fluida con los formularios para que se conozca los datos que el sistema espera que se ingresen en cada campo asociando claramente las etiquetas con los campos del formulario. Un adecuado rotulado de los campos de formulario, permite una lectura rápida y un ingreso ágil de la información

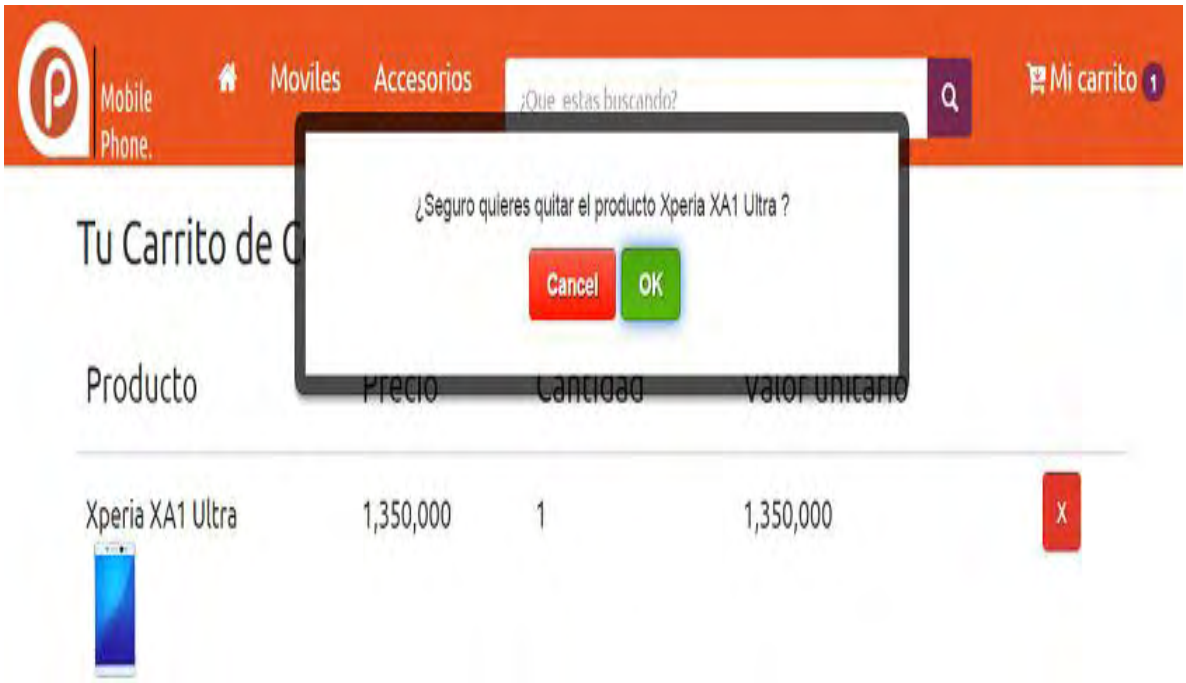
3.3.4.2 Implementación de las tácticas en tiempo de ejecución. Para la construcción del producto software se implementó, tácticas de iniciativa del sistema, la cual fue la que obtuvo mayor valoración resultado del modelo matemático, donde se llevó a la práctica mecanismos de interacción y retroalimentación de usabilidad.

Figura 55. Implementación del mecanismo progress feedback



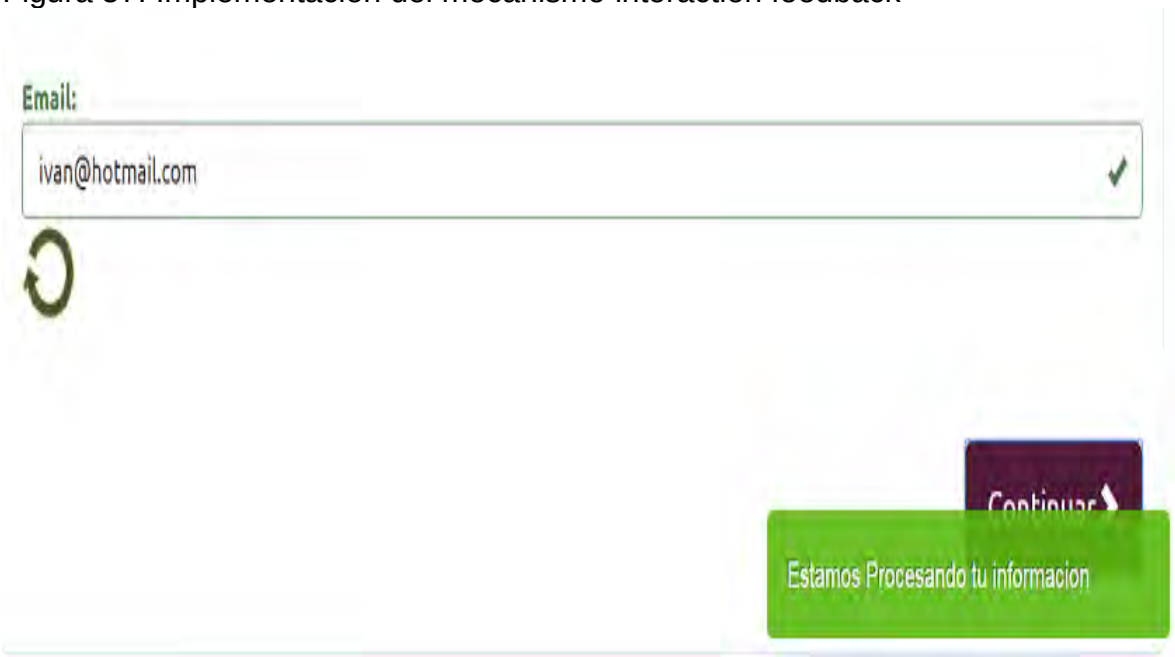
En la figura 55, se muestra la implementación del mecanismo de retroalimentación progress feedback, el cual consiste en mostrar el tiempo restante para la finalización de un servicio que es ejecutado por él usuario o el sistema. A este mecanismo se le asigna la siguiente forma de uso la de informar al usuario del tiempo restante para la finalización de la ejecución de un servicio.

Figura 56. Implementación del mecanismo warning



En la figura 56, se muestra la implementación del mecanismo de retroalimentación warning, el cual solicita al usuario confirmación en caso que se cometa algún error y las consecuencias puedan ser irreversibles, de esta forma se evita que el usuario cometa errores. A este mecanismo se le asigna la siguiente forma de uso la cual es pedir confirmación para la ejecución de servicios con consecuencias irreversibles.

Figura 57. Implementación del mecanismo interaction feedback



En la figura 57, se muestra la implementación del mecanismo de retroalimentación interaction feedback, el cual consiste en informar al usuario de que sus peticiones de interacción han sido recibidas se entiende que actúa cuando se realiza un evento, como por ejemplo, cada vez que el usuario hace una interacción con el sistema, tales como un movimiento del ratón o pulsar una tecla, se le tiene que brindar información por parte del sistema que esta petición ha sido recibida, a este mecanismo se le puede dar la siguiente forma de uso, informar al usuario de que el sistema está procesando la petición de ejecución de un servicio.

3.3.5 Síntesis del desarrollo de la aplicación. En el desarrollo de este fin se construyó un prototipo de una aplicación web sobre una tienda online con el objetivo, de transferir el conocimiento teórico en práctico, donde se llevó a la práctica la estrategia arquitectónica de usabilidad web, en la que se implementó la táctica en tiempo de diseño y la táctica en tiempo de ejecución. Siguiendo la arquitectura del patrón modelo vista controlador - MVC.

En la táctica en tiempo de diseño, se aplicó la táctica diseño de interfaces de usuario, la cual fue la que obtuvo mayor valoración resultado de la aplicación del modelo matemático, en la cual se implementó los patrones propuestos por Tidwell y las directrices de usabilidad en donde se llevó a la práctica los patrones y directrices de acuerdo con el contexto de la aplicación web que se desarrolló. Los patrones que se aplicaron fueron los siguientes: producto, búsqueda y navegación,

flujo de noticias, administrador pictórico, wizard, migas de pan, pie de página, disposición líquida, carrusel, paginación, botón hecho, mensajes de error y autocompletado. Las directrices que se implementaron fueron las siguientes: ubicación del logotipo, campos obligatorios y asociación de etiquetas y campos.

En la táctica en tiempo de ejecución, se aplicó la táctica mantener un modelo del sistema, la cual fue la que obtuvo mayor valoración resultado de la aplicación del modelo matemático, en donde se implementó las buenas prácticas de mecanismos de interacción de feedback, los mecanismos de interacción que se aplicaron fueron los siguientes: progress feedback, warning e interaction feedback.

4. CONCLUSIONES

De la metodología aplicada se puede concluir que las estrategias arquitectónicas de usabilidad web permiten identificar tácticas de usabilidad y decisiones de diseño, categorizadas en tácticas en tiempo de, diseño y ejecución.

Se concluye que el análisis de ventajas y desventajas permite proponer una forma cuantitativa para identificar la favorabilidad de las tácticas de usabilidad web, la cual, se puede adaptar para realizar análisis de otras tácticas relacionadas con atributos de calidad.

De las tácticas en tiempo de diseño se puede concluir que las tácticas “diseño de interfaces de usuario” presentan un mayor porcentaje de favorabilidad (85%) en la valoración total, porque, tiene una documentación completa, hace uso de patrones de diseño y directrices.

De las tácticas en tiempo de ejecución se puede concluir que las tácticas “iniciática del sistema” presentan un mayor porcentaje de favorabilidad (80%) en la valoración total, porque tiene modelos, mecanismos y patrones de fácil implementación.

Se concluye que la construcción de un producto software orientado a la web permitió poner en práctica el conocimiento teórico de las estrategias arquitectónicas de usabilidad web, implementado las tácticas en tiempo de, diseño y ejecución; e incorporando el patrón de arquitectura Modelo Vista Controlador - MVC. En la táctica en tiempo de diseño, se aplicó el diseño de interfaces de usuario, y se implementó los patrones propuestos por Tidwell. En la táctica en tiempo de ejecución, se aplicó, mantener un modelo del sistema, en donde se implementó las buenas prácticas de interacción y retroalimentación.

5. RECOMENDACIONES

Incluir dentro del proceso de formación de los estudiantes del programa de Ingeniería de Sistemas materias u ofertar un curso electivo, con aspectos relacionados con estrategias y tácticas de usabilidad en la construcción de software.

Implementar tácticas de usabilidad tanto en tiempo de ejecución como de diseño haciendo uso de los diferentes patrones y reglas en la construcción de software.

Realizar un estudio donde se combine al atributo de calidad de usabilidad web con otros atributos de calidad, como eficiencia y portabilidad. Además, se recomienda definir una forma cuantitativa para seleccionar las tácticas que posteriormente se analizaran.

Realizar trabajos donde se implementen otras tácticas de usabilidad web de menor porcentaje de favorabilidad incluidas en este estudio.

BIBLIOGRAFÍA

ABASCAL, Julio; AEDO,Ignacio; CAÑAS, José; GEA, Miguel; GIL, Ana; LORE, Jesus; MARTINEZ; Ana, ORTEGA, Manuel; VALERO, Pedro y VÉLEZ, Manuel. La interacción persona ordenador. Primera edición. Lleida: LORES, Jesus . 2001. 20 p.

ALVAREZ, Miguel. ¿Qué es MVC? .2016. [en línea]. < https://desarrolloweb.com/articulos/que-es-mvc.html>.[citado el 01 de diciembre de 2016].

Annett, J y Duncan, K. Task analysis and training design. Journal of Occupational Psychology.1967 41, p 211-221.

ARCINIEGAS, José L; FERNÁNDEZ, Verónica; HORMIGA, Amparo; TULANDE, ALEYDA; URBANO, Fernando A; COLLAZOS, César A. Proceso de requerimiento y análisis para la definición de la arquitectura desde la perspectiva de usabilidad para el desarrollo de aplicación.

BARBACCI, Mario; MARK, Klein; THOMAS A; LONGSTAFF, CHARLES, B; WEINSTOCK, Technical report, CMU/SEI-95-TR-021, ESC-TR-95-021, Quality Attributes, diciembre 1995.

BASS, L., CLEMENTS, P., y KAZMAN, R. Software Architecture in practice. 2003.Addison-Wesley.

BASS, Len; CLEMENTS y KAZMAN, Software architecture in practice, Third edition, 2013. Capítulo 4: Understanding quality attributes.

BASS, Len; CLEMENTS, Paul; KAZMAN, Rick. Software Architecture in Practice. Second edition. WESLEY, Addison. 2003. ISBN: 0-321-15495-9.

BUSCHMANN, Frank; MEUNIER, Regine; ROHNERT, Hans; SOMMERLAD, Peter Y STAL, Michael. Pattern-oriented software architecture: A System of Patterns. Vol 1: WILEY, John Y Sons.1996. p. 145-168. ISBN 0-471-95869-7.

CALLEJAS CUERVO, Mauro; CASTILLO ESTUPIÑAN, Luz Yadira; FERNÁNDEZ Álvarez, RUBY, Mónica. HELER: UNA HERRAMIENTA PARA LA INGENIERÍA DE REQUISITOS AUTOMATIZADA. Entramado, vol. 6, núm. 2, julio-23 diciembre, 2010, p. 184-200.

CARD, Stuart; THOMAS, P; MORAN y NeWELL, Allen. The keystroke-level model for user performance time with interactive systems. Communications of the ACM, 23(7),1980. p 396-410.

CARVAJAL, Mario; SAAB, Juan. Documento de análisis de prácticas y recomendaciones mundiales en Usabilidad. 23 de agosto de 2010. Disponible en:<http://programa.gobiernoenlinea.gov.co/apc-aa-files/5854534aee4eee4102f0bd5ca29479>.

Casallas, Rubby. ¿Aún en crisis? Algunos mitos y desafíos de la Ingeniería de Software. 2007, Sistemas, p. 8-17.

CATALINI, E. Arquitectura Modelo/Vista/Controlador. [en línea]. <https://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador/> [citado el 04 de febrero de 2017].

COLCIENCIAS. Generación de estrategias para el desarrollo tecnológica del sector software y servicios de ti mediante la aplicación de vigilancia. Bogotá: Colciencias, 2011. p.7.

CROSS, Nigel. Ingeniería del diseño. 1999. Capítulo 2.

DÁVILA, Abraham; MELENDEZ, Karin; FLORES, Luis. Determinación de los requerimientos de calidad del producto software basados en normas internacionales. IEEE Latin América Transactions, 2006, vol. 4, no 2, p. 100-106.

Definista. Definición de Documentación.2016[Entrada de blog]. Disponible en:< http://conceptodefinicion.de/documentacion/> [citado el 14 de enero de 2017].

Design Patterns. Elements of Reusable Object-Oriented Software - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - Addison Wesley (GoF- Gang of Four.
FERRE, Javier. Principios Básicos de Usabilidad para Ingenieros Software. Investigación. Madrid: Universidad Politécnica de Madrid. Facultad de Informática, 2005. 8 p. Disponible en Internet: <http://is.ls.fi.upm.es/xavier/papers/usabilidad.pdf>.

FISCHER, Gerhard. User modeling in human-computer interaction. En: User Modeling and User-Adapted Interaction, 11 . 2001. p 65-86.

GALINDO, M., CAMPS, R. Diseño e implementación de un marco de trabajo (framework) de presentación para aplicaciones JEE. 2008.

GRANT, R. Dirección estratégica. Conceptos, técnicas y aplicaciones.2004 Ed. THOMSON – CIVITAS.

HASSAN MONTERO, Yusef.Introducción a la usabilidad. 01 de noviembre de 2002. [en
linea].<http://www.nosolousabilidad.com/articulos/introduccion_usabilidad.htm>[citado el 15 de diciebre de 2016].
HASSAN MONTERO, Yusef; MARTÍN FERNÁNDEZ, Francisco J. La experiencia del usuario. En: No solo usabilidad, nº 4, 2005. ISSN 1886-8592.

HASSAN, Y. Introducción a la usabilidad. En: No Solo Usabilidad, noviembre 1, 2002. [citado el 06 de junio de 2016] <nosolousabilidad.com>.
.
HASSAN, Y. Introducción a la usabilidad. En: No Solo Usabilidad, noviembre 1, 2002.Disponible
en<http://www.nosolousabilidad.com/articulos/introduccion_usabilidad.htm>.
[citado el 06 de junio de 2016].

HERNÁNDEZ Sampieri, Roberto, FERNÁNDEZ, Carlos y BAPTISTA, Pilar. Fundamentos de metodología de la investigación. S.l.: MCGRAW-HILL, 2007.

HERNANDEZ, Helena; ALVAREZ, Guillermo, ARTEAGA, Muñoz. Patrones de interacción para el diseño de Interfaces wEB usables. Puebla México. 2003. Instituto Nacional de Astrofísica Óptica y Electrónica.

HESHUSIUS RODRÍGUEZ, Karen. Colombia: Desafíos de una Industria en Formación. En: BASTOS TIGRE, Paulo y SILVEIRA MARQUES, Felipe. Desafíos y oportunidades de la Industria de Software en América Latina. Bogotá, Colombia: Mayol ediciones S.A., 2009. p. 143.

HOFFMAN, Donna ; NOVAK,Thomas ; PERALT, Marcos. Building consumer trust in Online Environments. En: The case for information privacy. 1999. p. 80-85.
ISO, 9126. Information Technology-software Product Evaluation-Quality Characteristics and Guidelines for their use. ISO, ISO, 1991. Disponible en: <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16883>.

ISO, 9241. Ergonomics requeriments foroffice work with visual display terminals.ISO, 1998. Disponible en: <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16883>. [citado el 06 de junio 2016].

JURISTO, N; MORENO, A; SÁNCHEZ-SEGURA, M; Davis, A. Gathering Usability Information through Elicitation Patterns. 2005.

LANDEAU, Rebeca. Elaboración de trabajos de investigación. S.l.: ALFA, 2007.

LOVERA, Eduardo. Web taller. En: Diseño y usabilidad. [en línea]. <http://www.webtaller.com/maletin/articulos/como-hacer-copias-seguridad.php/diseno_y_usabilidad.php> [citado el 03 de mayo de 2016].

MERLINO, H; DIESTE, P; PESADO; GARCÍA-MARTÍNEZ, R. Modelo de Inclusión de la Funcionalidad UNDO/REDO. Universidad Nacional de La Plata. Programa de Doctorado en Ciencias Informáticas. Facultad de Informática.

MORENO, A.M.; SÁNCHEZ-SEGURA, M. Patrones de usabilidad: Mejora de la usabilidad del software desde el momento de arquitectónico. Alicante, noviembre 2003. VIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 03).

MORENO, Ana. patrones de usabilidad: Mejora de la usabilidad del software desde el momento arquitectónico. VIII Jornadas de ingeniería del software y bases de datos (JISBD 03). Alicante, noviembre 2013.

MURRAY, Tom. Authoring Tools for Advanced Technology Learning Environments, Dordrecht: Kluwer Academic Publishers.2003.

NIELSEN, Jakob, [en línea] <http://www.useit.com/alertbox/20030825.htm> [citado el 20 de abril de 2016].

NIELSEN, Jakob, Usability engineering. AP Professional, Boston, MA, 1993.

NIELSEN, Jakob. (1999). [en línea] <http://www.useit.com/alertbox/20030825.html. > [citado el 20 de abril de 2016].

NIELSEN, Jakob. 10 Usability heuristics for user interface design. En: Nielsen norman group. 1995. Disponible en: <https://www.nngroup.com/articles/ten-usability-heuristics/ >. [citado el 17 de diciembre de 2016].

NIELSEN, Jakob. 10 Heuristics for User Interface Design.2011.

PANACH, José Ignacio. Incorporación de mecanismos de usabilidad en un entorno de Producción de Software dirigido por modelos. Trabajo de grado. Valencia: Universidad Politécnica de Valencia. Departamento de Sistemas Informáticos y Computación, 2010. 570 p.

PANACH, Jose. Incorporación de Mecanismos de Usabilidad en un Entorno de Producción de Software Dirigido por Modelos. Tesis doctoral). Universidad Politécnica de Valencia. 40 p.

PATERNÒ F. Model–based design and evaluation of interactive application. 2000 Springer–Verlag.

PAVÓN, Juan. Estructura de las aplicaciones orientadas a objetos: El patrón modelo-vista-controlador. Universidad Complutense Madrid. 2008 Disponible en: <https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.> [citado el 31 de agosto de 2016].
PEINADO, F. Patrón Modelo-Vista-Controlador. Universidad Complutense Madrid. Disponible en <https://www.fdi.ucm.es/profesor/fpeinado/courses/oop/LPS-14ModeloVistaControlador.pdf >. [citado el 31 de agosto de 2016].

Perzel, K., Kane, D. Usability Patterns for Applications on the World Wide Web. 1999. PloP 99 Conference.

PROEXPORT. Guía Web Proexport 1.0. 2008. Disponible en: <http://www.brreg.no/elmer/elmer2-english.pdf>. [citado el 14 de octubre de 2016].

RODRÍGUEZ, Francy. Obtención y uso de patrones para la implementación de funcionalidades de usabilidad en aplicaciones web. Tesis de doctorado. Universidad Politécnica de Madrid.

Seffah y Taleb. Entorno de diseño asistido por patrón de usabilidad (UPDATE) es lenguaje Web Y el ambiente de diseño desarrollado por la Universidad Concordia. Grupo de ingeniería de software centrado en el ser humano.2008.

SEFFAH, Ahmed; TALEB, Mohamed. Tracing the evolution of HCI patterns as an interaction design tool.London.2 de diciembre de 2011. p 1-18.

SEVENMARKETING. ¿Porque es importante el atributo ALT en etiqueta de imágenes? 2014. Disponoble en <http://www.sevenmarketing.pe/por-que-e>. [citado 31 de junio de 2016].

Sommerville, Ian. Ingeniería de Software. s.l.: Pearson Education, 2011. p. 83.
Stephen, Albin. The Art of Software Architecture: Design Methods and Techniques. 2007. p 224-226. ISBN 0471228869.

TIDWELl, Jennifer.Patrones de diseño de Interacción. 2010. [en línea]. <http://www.mit.edu/~jtidwell/interaction_patterns.html> [citado el 3 de octubre de 2016].

TIDWELL, Jennifer. Designing Interfaces. 2010. O'Reilly Media.
WELIE, Martin Van. A Pattern Library for Interaction Design. 2008.Disponible en: < http://www.welie.com/>.

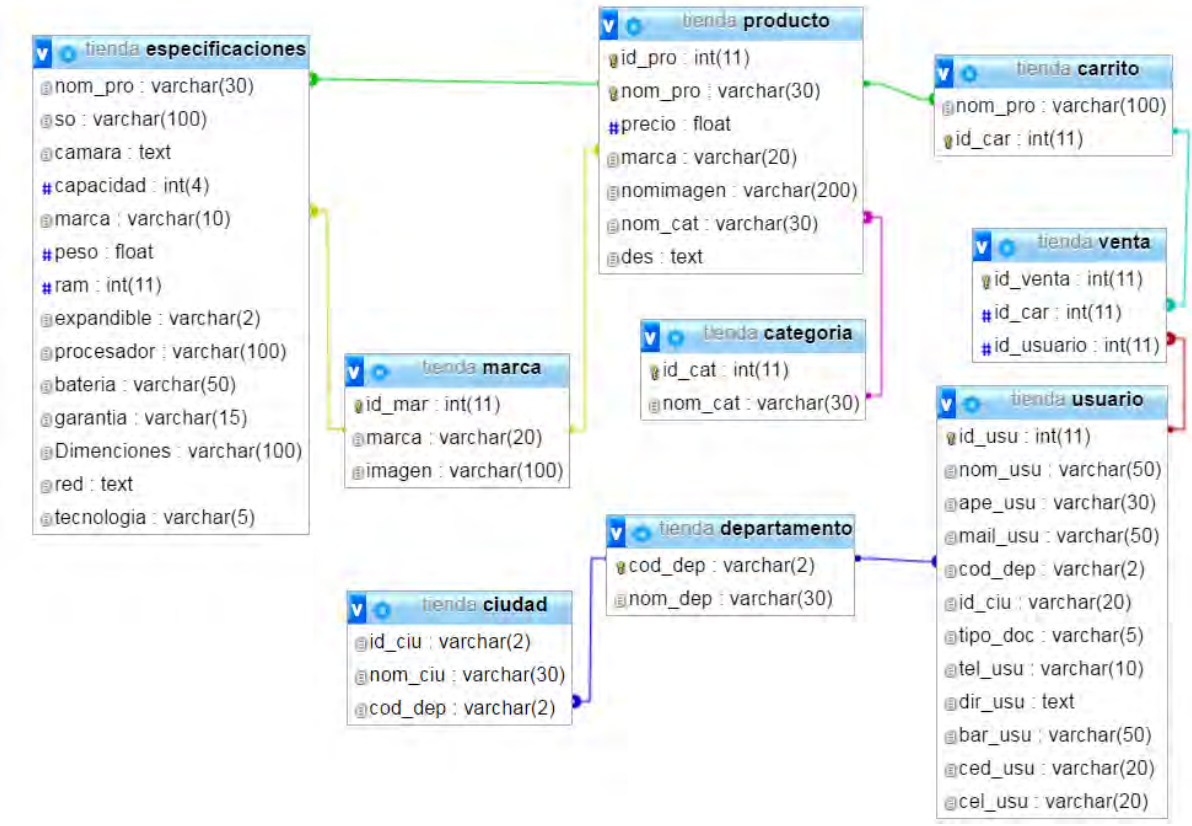
WELIE, M. A Pattern Library for Interaction Design. 2008. [en linea]. <http://www.welie.com/>[citado el 30 de agosto de 2016].

ANEXOS

Anexo A: Carpeta Software

- TiendaOnline.rar

Anexo B: Modelo relacional de la base de datos



Anexo C. Herramientas y tecnologías utilizadas para el desarrollo de la aplicación web.

En la implementación del aplicativo web, se utilizó las siguientes tecnologías y herramientas

XAMPP. Es una herramienta de desarrollo de software libre. Consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. Este paquete ha sido diseñado para ser increíblemente fácil de instalar y usar.

MYSQL. En la actualidad, el software de base de datos ha experimentado un crecimiento extraordinario, debido a la gran cantidad de información que manejan la totalidad de organizaciones de hoy en día. Esta es la razón, que existan muchos gestores de base de datos, que permiten manejar la información de forma más sencilla. En este sentido se encuentran a MySQL™, PostgreSQL, Oracle®, Microsoft® SQL Server® entre otros. Algunos de estos son comerciales y otros son software libre. Para el desarrollo del trabajo se eligió MYSQL, este es un sistema gestor de bases de datos muy conocido y ampliamente usado por su simplicidad y notable rendimiento, es una opción atractiva tanto para aplicaciones comerciales, como de entretenimiento

NETBEANS. Permite de manera rápida y fácilmente desarrollar aplicaciones de escritorio Java, móviles y aplicaciones web, así como aplicaciones HTML5 con HTML, JavaScript y CSS. El IDE también proporciona un gran conjunto de herramientas para desarrolladores de PHP y C / C ++. Es gratuito y de código abierto y tiene una gran comunidad de usuarios y desarrolladores de todo el mundo. Además, se apoya en bibliotecas de terceros como jQuery.

HTML. Son siglas asignadas para “Hyper Text Markup Language”, que traducido al español significa “Lenguaje de Marcas de Hipertexto”. HTML es un lenguaje que se basa en etiquetas, es utilizado en la informática para modelar la forma como la información es presentada a los usuarios, cuyo fin es el desarrollo de páginas web, indicando cuales son los elementos que la compondrán, orientando hacia cuál será su estructura y también su contenido, básicamente es su definición; por medio del HTML se indica tanto el texto como imágenes pertenecientes a cada página de internet.

CSS. Son hojas de estilo en cascada, este lenguaje fue creado con el fin de controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

BOOTSTRAP. Es un framework de desarrollo liberado por Twitter que tiene como objetivo facilitar el diseño web. Bootstrap permite crear de forma sencilla webs con interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas. Es de código abierto, por lo que se puede usar de forma gratuita y sin restricciones.

LENGUAJE JAVASCRIPT. Abreviado comúnmente como JS es un lenguaje de programación interpretado, o para ser más claro: Es para el que la mayoría de sus implementaciones ejecuta instrucciones directamente, sin una previa compilación del programa a instrucciones en lenguaje máquina. Además orientado a objetos, basado en prototipos, imperativo, y dinámico.

Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web, permite mejoras en la interfaz de usuario y páginas web dinámicas aunque existe una forma de JavaScript del lado del servidor como (Server-side JavaScript).

Todos los navegadores modernos interpretan el código JavaScript integrado en páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con ayuda de otras tecnologías como AJAX, que también se utilizó para el desarrollo de la aplicación. Permite a las páginas hacer una pequeña petición de datos al servidor y recibirla sin necesidad de cargarla página entera. El incremento de las actualizaciones “on the fly” elimina el tener que refrescar el navegador, algo bastante apreciado a la hora de operar en una aplicación web.

PHP. Es un lenguaje de programación interpretado, de código abierto, adecuado para el desarrollo web y que puede ser incrustado en HTML. Es usado principalmente en interpretación del lado del servidor, pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluye aplicaciones con interfaz gráfica usando bibliotecas Qt o GTK+. Además de su posibilidad de acceso a muchos tipos de bases de datos, también es

importante destacar su capacidad de crear páginas dinámicas, así como la posibilidad de separar el diseño del contenido de una web.