AAS9391

UCAB Universidad Católica ANDRÉS BELLO

UNIVERSIDAD CATOLICA ANDRES BELLO

VICERRECTORADO ACADEMICO

ESTUDIOS DE POSTGRADO

ÁREA DE INGENIERÍA

POSTGRADO EN SISTEMAS DE INFORMACIÓN

TRABAJO DE GRADO DE MAESTRIA

MODELO PARA LA OPTIMIZACIÓN DEL DISEÑO DE ARQUITECTURAS DE SOFTWARE EN APLICACIONES EMPRESARIALES CON BASE EN LA USABILIDAD EN LAS PYMES CONSULTORAS VENEZOLANAS

Presentado Por

Uriola Raymond, Emily L.

Para Optar al Título de

Magister en Sistemas de Información

Tutor

Bonillo, Pedro

Caracas, Julio 2014

Índice General

Lista de ilustraciones....iv
Lista de tablas....vi
Dedicatoria....viii
Agradecimiento....ix
Carta de aprobación....x
Aprobación de la empresa....xi
Introducción....1
Capitulo I: El Problema....3
1.1 Planteamiento del Problema....3
1.2 Objetivos de la Investigación....7
1.2.1 Objetivo general....7
1.2.2 Objetivos específicos....7
1.3 Justificación e Importancia de la Investigación....8
1.4 Alcance y Limitaciones....9
Capítulo II: Marco Teórico....11
2.1 Antecedentes de la Investigación....11
2.2 Fundamentos Teóricos....14
2.2.1 Modelo....14
2.2.2 La Experiencia del Usuario....16
2.2.3 Usabilidad....19
2.2.4 Componentes de Software....21
2.2.5 Arquitectura de Software....22
2.2.6 Patrones Arquitecturales para Aplicaciones Empresariales....27
Capítulo III: Marco Metodológico....29
3.1 Consideraciones generales....29
3.2 Tipo de Investigación....29
3.3 Diseño de la Investigación....31
3.3 Población y Muestra....31

3.5 Técnicas e Instrumentos para la *Recolección de Información* 35
3.6 Validez de los Instrumentos de Recolección de Información 39
3.7 Análisis e Interpretación de la Información 39
3.8 Metodología a utilizar en función de los objetivos específicos 39
3.8.1 Determinación la Naturaleza del sistema 41
3.8.2 Tipo de evaluación. 42
3.8.3 Identificación los individuos (Stakeholder) 42
3.8.4 Estudio y Análisis 42
3.8.5 Retroalimentación 43
3.9 Viabilidad de la Investigación 43
3.9.1 Factibilidad Económica 44
3.9.2 Factibilidad Técnica 45
3.9.3 Factibilidad Operativa 45
Capítulo IV: Formulación del modelo 48
4.1 Clasificación de Métodos de Evaluación de la Usabilidad 48
4.2 Naturaleza Del sistema 52
4.3 Tipo de Evaluación 53
4.4 Usuarios Afectados *(Stakeholders)* 54
4.5 Preguntas Claves 54
4.6 Diseño del modelo 60
Capítulo V: Caso de Estudio 68
Capítulo VI: Conclusiones y Recomendaciones 78
Referencias Bibliográficas 83
Anexo 1: Patrones Arquitecturales para Aplicaciones Empresariales 88
Anexo 2: Modelo del cuestionario aplicado a los usuarios 103
Anexo 3: Validación de Instrumento de investigación 106
Anexo 4 Documento de Arquitectura de Software del Caso de estudio 109
Anexo 5: Escenarios detectados en el Caso de estudio 124
Anexo 6: Resultados Tabulados de los cuestionarios aplicados a los usuarios 129
Anexo 7: Clasificación de los Métodos de Evaluación de la Usabilidad 132

Lista de Ilustraciones

Figura pág.

1. Iceberg de la Usabilidad. 5
2. Mapa mental del marco teórico 12
3. Diseño de la Investigación. 33
4. Metodología SESL (System Evaluation for Stakeholder Learning) 41
5. Relación entre Proyecto factible y la metodología SESL 42
6. Clasificación de métodos de evaluación de usabilidad según Whitefield, Wilson y Dowell 50
7. Relación entre el enfoque gestión y el enfoque técnico del proceso de diseño del software 53
8. Resultados en la categoría eficiencia (preguntas nro. 1, 2 y 3) 55
9. Resultados en la categoría eficiencia (pregunta nro. 4) 55
10. Resultados en la categoría eficiencia (preguntas nro. 5 y 6) 56
11. Resultados en la categoría fiabilidad (preguntas nro. 7, 8 y10) 56
12. Resultados en la categoría fiabilidad (preguntas nro. 9) 57
13. Resultados en la categoría satisfacción (preguntas nro. 13, 14 y 15) 57
14. Resultados en la categoría satisfacción (preguntas nro. 11 y 12) 58
15. Resultados en la categoría satisfacción (preguntas nro. 15 a la 20) 58
16. Resultados en la categoría satisfacción (preguntas nro. 21) 59
17. Resultados en la categoría satisfacción (preguntas nro. 22) 59
18. Diseño para la usabilidad 61
19. Relaciones entre atributos y propiedades de la usabilidad 61
20. Plantilla para la realización de escenarios 67
21. Proceso de Análisis y Diseño de Parodi Solutions 68
22. Proceso de Desarrollo de Parodi Solutions 69
23. Proceso de Análisis y diseño incluyendo la fase sugerida por el modelo de evaluación 69
24. Visualización Patrón Modelo-Vista-Controlador en sitios Web 71

25. Flujo de Trabajo del framework 71
26. Diagrama de arquitectura de grupos del Caso de estudio 72

Lista de Tablas

Tabla pág.

1. Usuarios afectados significativos 34
2. Operacionalización de Variables 37
3. Cronograma de actividades 47
4. Escenarios Clasificados 74
5. Matriz de Escenarios 75
6. Resultados de la ponderación de escenarios 75

UNIVERSIDAD CATOLICA ANDRES BELLO
ESTUDIOS DE POSTGRADO
ÁREA DE INGENIERÍA
POSTGRADO EN SISTEMAS DE INFORMACIÓN

MODELO PARA LA OPTIMIZACIÓN DEL DISEÑO DE ARQUITECTURAS DE SOFTWARE EN APLICACIONES EMPRESARIALES CON BASE EN LA USABILIDAD EN LAS PEQUEÑAS Y MEDIANAS EMPRESAS CONSULTORAS VENEZOLANAS

Autor: Emily L. Uriola R.

Tutor: Pedro Nolasco Bonillo Ramos

Resumen

En los proyectos de tecnología de la información se ha concebido la facilidad de uso como una propiedad conectada en su totalidad a la presentación de la información y se cree que basta con separarla del resto del proyecto para cubrir los requisitos de las pruebas de usabilidad. Pero esta propiedad se encuentra ligada a diversos factores, como la facilidad de aprendizaje, no sólo a la apariencia de la aplicación. El no tomar en cuenta a esta propiedad en fases tempranas del proyecto, terminan afectando profundamente el área funcional, los cuales se traducen en un incremento del tiempo y costo. El objetivo de este trabajo es la creación de un modelo para permitir el incremento de la usabilidad en el sistema, al generar una herramienta para analizar el comportamiento de la usabilidad desde una fase temprana, guiando de forma asertiva la elección de la arquitectura, la correcta implementación de los diseños, promoviendo cambios en la visión del proceso de creación de soluciones tecnológicas y el establecimiento de métricas para evaluar la usabilidad en fases tempranas para resolver este conflicto.

Palabras Claves: modelo, usabilidad, arquitectura de software, métrica, optimización.

Línea de investigación: Ingeniería del software.

Dedicatoria

A mi madre,

A mi tía,

Y a mi abuela,

Quienes me han apoyado siempre

y me han incentivado en la búsqueda de la verdad y el conocimiento.

Gracias siempre.

Mily

Agradecimiento

A Dios Todopoderoso, porque con él todo y sin él nada.

A mi familia, por creer en mí y apoyarme una vez más en esta difícil etapa.

A mis amigos por su alegría y por formar parte de mis mejores recuerdos.

A mi tutor, el profesor Pedro Bonillo, por su profesionalismo, aporte académico, paciencia y apoyo.

A la UCAB, por ser una vez más el escenario de los acontecimientos importantes en mi vida.

UCAB Universidad Católica ANDRÉS BELLO

Caracas, 30 de Mayo de 2014

UNIVERSIDAD CATÓLICA ANDRES BELLO

Postgrado en Sistemas de Información

Dirección del programa

Presente.-

CARTA DE APROBACIÓN

Por la presente me permito comunicar que he sido el tutor del Trabajo de Grado de Maestría del estudiante Emily L. Uriola Raymond cédula de identidad nro. 16857137, quien opta por el título de Magister en Sistemas de Información, intitulado "Modelo Para La Optimización Del Diseño De Arquitecturas De Software En Aplicaciones Empresariales Con Base En La Usabilidad en las PYMES Venezolanas".

Asimismo, hago constar que como tutor estoy conforme con el contenido presentado, por lo que cuenta con mi aprobación para ser inscrito como Trabajo de Grado de Maestría.

Sin otro particular al cual hacer referencia, se despide cordialmente,

Bonillo Ramos Pedro

Pedro Nolasco Bonillo Ramos

C.I. 10.868.538

APROBACIÓN DE LA EMPRESA

Caracas, a los 03 días del mes de Mayo de 2014

Sres.

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Postgrado en Sistemas de Información

Caracas

Nos dirigimos a ustedes para informarles que hemos autorizado al Ing. Emily L. Uriola Raymond C.I.: 16857137, quién labora en esta organización, a hacer uso de la información proveniente de Parodi Solutions C.A, para documentar y soportar los elementos de los distintos análisis estrictamente académicos que conllevarán a la realización del Trabajo Especial de Grado titulado "Modelo Para La Optimización Del Diseño De Arquitecturas De Software En Aplicaciones Empresariales Con Base En La Usabilidad" como requisito para optar al título de Especialista/Magíster en Sistemas de Información, exigidos por la Dirección General de los Estudios de Postgrado de la Universidad Católica Andrés Bello.

Sin más a que hacer referencia, atentamente,

Dr. Luis Eduardo Parodi

CI: 7.758.646

Presidente

RIF.: J-31737894-6

Av. 64 Edificio Luis Parejo local Pb entre Av. Bella Vista y Av. Santa Rita

Maracaibo -Zulia

Introducción

El aspecto y la sensación describen la interacción del usuario con un sistema, pero aquello que este puede pedir al sistema que haga, está definido por el modelo que compone los aspectos relacionados con lo que se está tratando de lograr, o en otras palabras, los objetivos de la tarea del usuario. Por lo tanto al decir que un sistema es "usable", no se puede tomar en cuenta sólo lo referente al aspecto de éste, ya que esta cualidad también dependerá de aquello que se pueda realizar y de qué forma, puntos relacionados directamente con la funcionalidad de sistema.

En base a esto, se pueden considerar dos tipos de atributos: los cuantificables de forma objetiva y cuantificables de forma subjetiva. Los objetivos como la eficacia (o número de errores cometidos por el usuario durante la realización de una tarea), y la eficiencia (o tiempo empleado por el usuario para la consecución de una tarea), y los subjetivos, como la satisfacción de uso, que debe ser entendida siempre en relación con la forma y condiciones de uso por parte de sus usuarios, así como con las características y necesidades propias de estos.

El presente trabajo pretende crear un modelo que disminuya las fallas generadas al no contemplar las condiciones anteriormente descritas en el nivel arquitectural de los sistemas, así como formas de detección, corrección temprana y prevención de las mismas.

Dentro de las áreas temáticas de sistemas de información, esta investigación se sitúa dentro de las ciencias de la computación y consta de seis capítulos, organizados de la siguiente forma:

En el capítulo I, se presenta la conceptualización y descripción de la investigación, detallando el objetivo general y los objetivos específicos, así como la justificación y el alcance del mismo.

En el Capítulo II, se expondrán a través de los estudios de mayor relevancia en el tema, la influencia de la usabilidad en el proceso de desarrollo, los escenarios identificados donde la usabilidad es sensible a la arquitectura utilizada y así como qué características de las arquitecturas de software mejoran la usabilidad y su relación con la investigación, también los

fundamentos teóricos de la misma, los cuales representan la base para mantener la relación entre la investigación y la teoría.

En el Capítulo III, se muestra el marco metodológico compuesto por el tipo y el diseño elegido para llevar a cabo la investigación, las técnicas y herramientas utilizadas para el desarrollo de la misma, así como la definición y operacionalización de las variables.

En el Capítulo IV, se exponen los resultados de la investigación, en forma de gráficos y tablas, realizando un análisis de los mismos, y posteriormente la proposición un modelo para realizar evaluaciones con base en la usabilidad.

En el Capítulo V, se presenta el caso de estudio donde se realizó la aplicación del modelo propuesto.

En el Capítulo VI, se observan los resultados obtenidos a través de las referencias realizadas en el marco teórico, para evaluar el cumplimiento de los objetivos de la investigación y formular recomendaciones para futuros trabajos sobre el tema.

Y por último, se presentan las referencias bibliográficas utilizadas para la investigación, así como los anexos realizados.

Capítulo I

El Problema

Para iniciar este capítulo se describe a continuación la formulación del problema, la descripción de sus manifestaciones y evidencias, posteriormente la formulación el objetivo general y los objetivos específicos, describiendo además la justificación de la investigación y sus limitaciones.

1.1 Planteamiento del Problema

Desde una perspectiva general, en los proyectos de tecnología de información se ha concebido la usabilidad como una propiedad conectada en su totalidad a la presentación de la información, encapsulando la presentación y separándola del resto del proyecto. Lo cual hace que no cubra los requisitos de las pruebas de usabilidad y en el peor de los casos, los problemas generados no se resolverán modificando sólo la presentación, sino también la funcionalidad.

La usabilidad, es un anglicismo que significa "facilidad de uso", como lo indican Bevan, Kirakowski, y Maissel (1991), y parece tener su origen en la expresión "user friendly". Pero la definición más extendida es la dada por ISO 9241-11 (1998): "Usabilidad es la efectividad, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico". Esta definición propone que la usabilidad se compone de dos tipos de atributos: los cuantificables de forma objetiva como son la eficacia o número de errores cometidos por el usuario durante la realización de una tarea, y la eficiencia o tiempo empleado por el usuario para la consecución de una tarea, y por otro lado los atributos cuantificables de forma subjetiva, como es la satisfacción de uso, medible a través de la interrogación al usuario.

De acuerdo con Berry (2000), los desarrolladores suelen mostrarse sorprendidos cuando explica que no considera el "look and feel" como uno de los principales contribuyentes en la capacidad de uso general de una aplicación o sitio Web, sino al conjunto de los conceptos

conocidos por el usuario, que permiten aprender y comprender como utilizar un producto o sitio Web de forma eficaz.

Dado que la facilidad de uso se encuentra íntimamente ligada a facilidad de aprendizaje, la cual no depende exclusivamente de los aspectos de la apariencia, sino también con la variedad de posibilidades con las que el usuario y el sistema pueden intercambiar información y el nivel de apoyo dado al usuario para facilitar el cumplimiento de sus objetivos, estando estos dos últimos aspectos completamente ligados al área funcional y no estética, se puede decir que esta variable se encuentra profundamente arraigada en cada paso del proceso de desarrollo, donde cualquier falla relacionada con ésta puede afectar de igual manera el desarrollo del proyecto como lo harían otros factores.

Al no tomar en cuenta esto, los modelos de desarrollo han fallado incontables veces en el pasado, porque a menudo estas debilidades detectadas a un nivel superficial no parecen afectar en gran medida el desarrollo del proyecto, pero al ser evaluadas detalladamente se evidencia que afectan profundamente al área funcional y en muchos casos han tenido que realizarse profundos cambios después de haber detectado un desperfecto en la facilidad de uso.

Berry (2000), considera que, en primer lugar, es necesario que los usuarios posean representación desde las etapas más tempranas del diseño, para comprender quiénes son en términos de sus habilidades y motivaciones, y las tareas que realizan. En segundo lugar, se debe aprovechar la ventaja que representa el conocimiento del usuario en tareas y métodos de ingeniería de software para modelar los conceptos, así como el "look and feel". El "user model", es la base del diseño, no sólo como marco a través del cual se puede evaluar y validar los diseños, sino también como una fuente para la documentación crucial necesaria para coordinar y dirigir un equipo de desarrollo. Y en última instancia, el "user model" puede ayudar a mejorar en gran medida sus probabilidades de ofrecer diseños que los usuarios encuentren intuitivo.

Por lo cual, Berry propone que la usabilidad puede analizarse mediante la identificación de tres conjuntos de aspectos: el aspecto (look), la sensación (feel) y el modelo de usuario, como se muestra en la Figura 1. El "look" abarca aspectos visuales, la retroalimentación, y la estética. La sensación (feel) incluyen las asignaciones de teclado y de los botones del ratón,

las estructuras, los menús, accesos directos y las convenciones de la interfaz de navegación. En conjunto, el aspecto (look) y la sensación (feel) describen como un usuario puede interactuar con un sistema pero no lo que el usuario puede pedir al sistema que haga, por otro lado el modelo de usuario se compone de aspectos relacionados con lo que el usuario está tratando de lograr, o en otras palabras, los objetivos de la tarea del usuario.

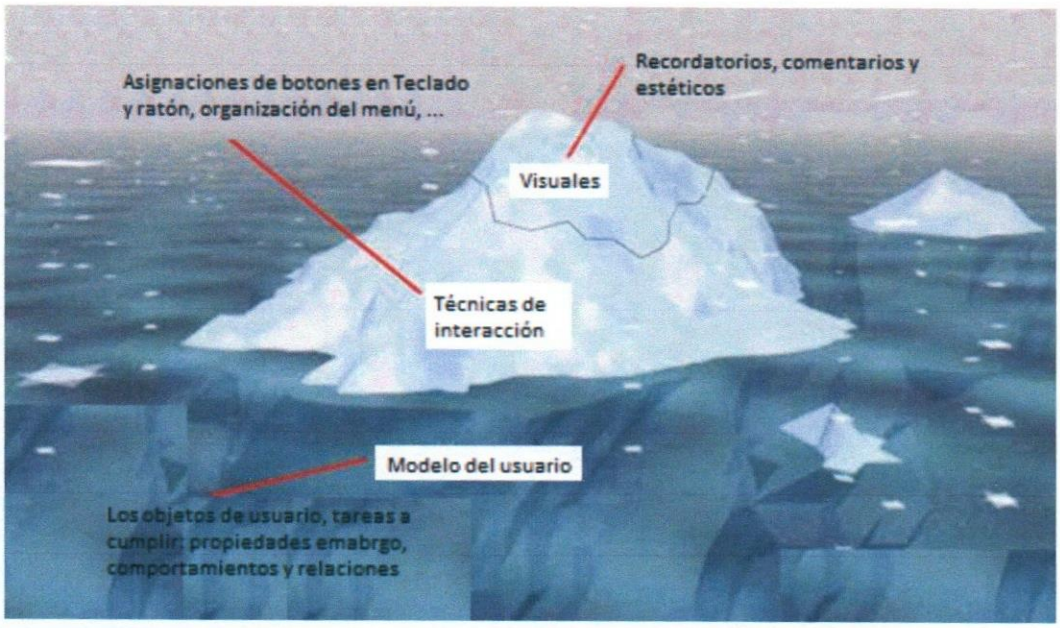


Figura 1. Iceberg de la Usabilidad.
Fuente: Tomado de "La experiencia del usuario - La analogía del iceberg de la usabilidad", Berry, 2000.
Traducción propia

Visto lo anterior, se evidencia que no basta con tener en cuenta la usabilidad como un elemento de la presentación, tanto en sistemas simples como en los más complejos, como entornos distribuidos, transaccionales, o aquellos donde existe un gran volumen de concurrencia, esta debería ser un factor presente desde el inicio del proyecto, es decir desde que se diseña la arquitectura del software.

Cuando se diseña la arquitectura del software se determina la estructura de funcionamiento del proyecto y la división del mismo, según la analogía del Iceberg de Berry la arquitectura está directamente relacionada al modelo de usuario por ser parte del conjunto de factores que al finalizar el proyecto darán al usuario la posibilidad de realizar todas las actividades por medio

del sistema, realizar una mala elección con respecto al patrón utilizado en esta, acarrea como consecuencia el retardo en la detección de problemas básicos y de funcionalidades no previstas, y si posteriormente se analizan los escenarios de interacción, se observara que la causa es no haber tenido en cuenta al usuario al inicio del diseño del sistema, es decir cuando se realizó la arquitectura.

Esta relación puede ser fácilmente explicada con un ejemplo práctico, supóngase un caso donde un arquitecto quiere regalar a su esposa una casa, este realiza los planos, compra un terrero en una zona maravillosa de la ciudad y lleva a cabo la construcción en secreto como una sorpresa para su aniversario, ejecuta perfectamente la construcción bajo un diseño moderno, con cuartos espaciosos en ambos pisos, grandes ventanas y pisos de madera.

Cuando finalmente presenta la casa a su esposa, esta se muestra asombrada, y al terminar el recorrido por la casa aunque muy agradecida con su marido decide sincerarse y le explica que aunque sus intenciones fueron buenas ha fallado en gravemente en muchos detalles, por ejemplo el piso de madera es más difícil de limpiar lo cual acarrearía de su parte un esfuerzo extra diariamente, restándole tiempo para otras actividades, las ventanas concebidas así por el estilo dado por el arquitecto a la casa, son bastante engorrosas para su limpieza y mantenimiento, además la zona en la que compró el terreno queda mucho más lejos del sitio de trabajo de ambos, lo cual hará que deban despertarse más temprano para llegar a tiempo a este. Finalmente, ambos acuerdan cambiar el piso, aunque acarree un gasto extra, pero algunos otros detalles como la distancia entre la casa y el sitio de trabajo o las ventanas no pueden ser reparados.

En esta situación se puede apreciar, al equipo de desarrollo de un proyecto , porque aunque se desee realizar todo el proceso con la mayor eficiencia posible utilizando las mejores e innovadoras técnicas, si no se toma en cuenta al usuario final probablemente el software no cumpla todos los objetivos, tal vez en mayor o menor medida sea "usable", pero no será la mejor solución al problema. En este caso el piso no representaba un problema en sí mismo, pero para una persona ocupada como la esposa del arquitecto este era un inconveniente, también existen otro tipo de problemas como la distancia entre la casa y el trabajo o las ventanas, errores como estos son los peores dentro de un proyecto, porque están relacionados intrínsecamente con la arquitectura y en muchos casos no son reparables por condiciones

específicas del ambiente y la modalidad con la cual se trabaja, representando una gran pérdida de tiempo y dinero para las empresas.

En la medida que se logren, determinar una serie de momentos de interacción o tareas del usuario que para ser soportadas por el sistema deban estar definidas en la arquitectura del software, se podrán tomar decisiones de forma más asertiva al momento de la elección de una arquitectura. También es necesario que los usuarios estén representados en las primeras etapas del diseño, para comprender así lo que son, en términos de sus habilidades, motivaciones y tareas que realizan.

Por último, dada la situación presentada anteriormente, se toma como interrogante de investigación: ¿Cuál será el modelo que permita la optimización del diseño de arquitecturas de software para aplicaciones empresariales con base en la usabilidad?

1.2 Objetivos de la Investigación

A continuación se presenta el objetivo general y los objetivos específicos de esta investigación.

1.2.1 Objetivo general.

Desarrollar un modelo para la optimización del diseño de arquitecturas de software en aplicaciones empresariales con base en la usabilidad en las pequeñas y medianas empresas consultoras venezolanas.

1.2.2 Objetivos específicos.

- Analizar las técnicas y métodos existentes para evaluar la usabilidad de un sistema.
- Determinar la factibilidad de diseñar un modelo para la optimización del diseño arquitecturas de software en aplicaciones empresariales con base en la usabilidad.
- Elaborar una técnica de recopilación de información, que permita obtener la relación entre las características de la arquitectura de software de un proyecto y la usabilidad.
- Establecer técnicas y procedimientos que permitan la optimización de la usabilidad dentro de las arquitecturas de software para aplicaciones empresariales acorde a los resultados obtenidos.

- Proponer un modelo para realizar evaluaciones enfocadas en la usabilidad, con base en las técnicas de evaluación de atributos de calidad del software existentes.
- Diseñar un modelo para la optimización del diseño de arquitecturas de software en aplicaciones empresariales con base en la Usabilidad en las pequeñas y medianas empresas consultoras venezolanas.
- Aplicar el modelo a la empresa venezolana de consultoría Parodi Solutions C.A, a fin de verificar su validez a un caso de estudio representativo.

1.3 Justificación e Importancia de la Investigación

Se presentó un modelo basado en la investigación, que desarrolle técnicas y procedimientos que permitirán el incremento de la usabilidad en el sistema, generara de manera directa una mejora en la calidad de la calidad de los procesos de creación de software en el ámbito empresarial, buscando impulsar así cambios en la visión del proceso de creación de soluciones, lo cual puede considerarse como un proyecto de innovación.

Tomando en consideración que dentro del campo de la maestría en sistemas de información se encuentra la formación para contribuir y liderar proyecto de innovación tecnológica, esta propuesta representa una contribución a la forma en que se lleva a cabo el proceso de diseño y desarrollo de proyectos en el área de tecnología de la información.

Cuando las empresas en Venezuela toman decisiones referentes a la tecnología, se utilizan enfoques que tienden a subestimar o ignorar aspectos importantes de la selección y uso de ésta, la toma de decisiones de forma más asertiva en la elección de una arquitectura puede significar el éxito o fracaso de un proyecto en gran medida, por lo tanto el contar con métodos y herramientas eficaces que apoyen la toma de decisiones, junto a la correcta implementación de los diseños y los esqueletos que sostienen a la interfaz, disminuirá el tiempo perdido dentro del proyecto por errores estructurales y de arquitectura, lo cual repercutirá directamente sobre el tiempo de cumplimiento del mismo, y de manera sensible sobre el rendimiento económico de las empresas y sus posibilidades de desarrollo futuro.

El producto de este trabajo podrá ser utilizado en otras investigaciones como base para la realización de evaluaciones sobre otros proyectos reales, y de ser profundizada, llegar incluso a crear paradigmas en la determinación de patrones arquitecturales para resolver de manera

más acertada una situación dada, según el tipo de sistema que desee elaborarse, apoyado en parámetros y estándares establecidos, lo cual disminuiría la probabilidad de fallas y contratiempos que afecten a la usabilidad en gran medida dentro de las empresas.

Como las investigaciones realizadas en el campo de la usabilidad son recientes y aun no han alcanzado su punto máximo de madurez, cualquier investigación que profundice en el área, podría ser utilizada como un complemento teórico de las ya existentes, mas aun cuando esta explora un área específica como lo es el subdominio empresarial, del cual no existe un extenso material documentado. Esta investigación aportara mejoras a las condiciones en la elaboración de proyectos de tecnología, impulsando la generación de mejores resultados como profesionales de la tecnología y disminuyendo el porcentaje de errores que de manera inconsciente o consiente que puedan entorpecer el correcto termino de un proyecto. Beneficiará también a los profesionales del área, facilitando el correcto cumplimiento del trabajo en el momento de la toma de decisiones sobre diseño y arquitectura, resaltándola como parte principal del proceso, mejorando así la calidad del producto.

1.4 Alcance y Limitaciones

Esta investigación planteó un modelo de evaluación y optimización, que buscó establecer las relaciones y condiciones necesarias entre la arquitectura de un sistema y la usabilidad como un atributo de la calidad del software, que mejoren la calidad del sistema, dentro de un modelo que permita obtener una buena estimación, ofreciendo a las consultoras venezolanas, un conjunto de criterios para evaluar exitosamente el uso de las arquitecturas de software en el marco de los intereses del negocio. Los resultados obtenidos en esta investigación fueron:

- Descripción de las técnicas y métodos existentes para la evaluación de la usabilidad en un sistema.
- Estudio de factibilidad de diseñar un modelo para la optimización en el diseño de arquitecturas de software en aplicaciones empresariales con base en la usabilidad.
- Modelo para la optimización del diseño de arquitecturas de software en aplicaciones empresariales con base en la usabilidad en las pequeñas y medianas industrias venezolanas.
- Aplicación del modelo propuesto en un caso práctico.

Esta investigación se limitó al estudio de la población seleccionada, no contemplaba la aplicación masiva del modelo en el mercado consultor venezolano debido al factor tiempo y recursos, pero se deja abierta la posibilidad para que este estudio sea tomado como base en futuras investigaciones del área.

Las diferentes causas que obstaculizaron que la investigación se realizara de forma apropiada, fueron las siguientes:

- La falta de tiempo y recursos humanos, que acarrea un retraso en el proceso de recabar la información, necesaria para la investigación.
- La dificultad del investigador para conseguir estudios y documentación en español sobre el área, lo cual hizo necesario llevar a cabo la traducción de documentos en distintos idiomas (inglés y portugués).
- La disponibilidad de los sujetos seleccionados en la muestra para contestar el cuestionario.

Capítulo II

Marco Teórico

En este capítulo se hace referencia a los aspectos teóricos que respaldaron la investigación propuesta en forma de un modelo, esta se situó dentro de los conceptos, bases y referencias teóricas anunciadas a continuación. En estos conocimientos se basó el análisis expuesto en capítulos posteriores para la búsqueda de soluciones al problema planteado.

Según Tamayo (2003), en los antecedentes se trata de hacer una síntesis conceptual de las investigaciones o trabajos realizados sobre el problema formulado, con el fin de determinar el enfoque metodológico de la misma investigación. Los antecedentes son todas aquellas investigaciones realizadas sobre el tema que servirán juzgar e interpretar la información.

A continuación, en la Figura 2, se presenta un mapa mental, que representa los tópicos profundizados en el marco teórico de esta investigación.

2.1 Antecedentes de la Investigación

Berry (2000) a través de su artículo titulado *The user experience*, plantea de qué forma influencia la facilidad de uso a cada una de las capas de una aplicación (visuals, interaction techniques y user model) mediante la analogía de un iceberg donde la punta representa el elemento visual (visuals), porque según lo planteado, durante el desarrollo de aplicaciones se pretende que factores como la usabilidad estén sólo ligados a esta capa.

Este material fue uno de los principales puntos de partida para esta investigación, ya que explica gráficamente la relación entre el "user model", el "look and feel" y la usabilidad, y como ésta puede a llegar a afectar gravemente cualquiera de las capas. Plantea como conclusión que es necesario que los usuarios estén representados en las primeras etapas de diseño, y comprender lo que son en términos de sus habilidades y motivaciones, y las tareas que realizan.

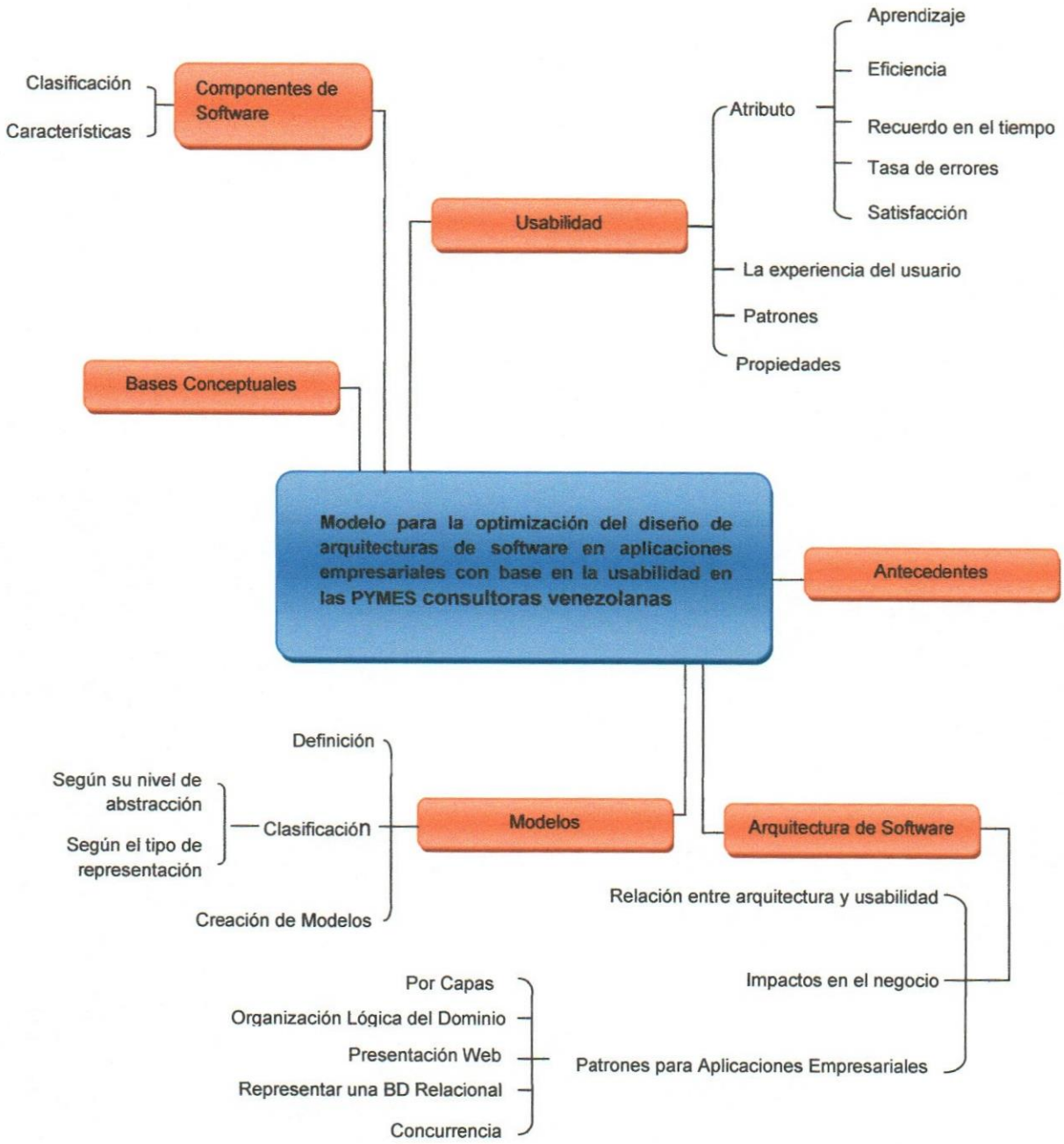


Figura 2. Mapa mental del marco teórico

Además se puede tomar ventaja del conocimiento de los usuarios de tareas y métodos de ingeniería de software para modelar los conceptos, así como el "look and feel" de la aplicación, los modelos de los usuarios puede ayudar a mejorar en gran medida sus probabilidades de ofrecer que los usuarios acciones intuitivas.

Otra investigación de gran importancia, es la de Bass & Kates (2001), en la cual han inventariado una serie de momentos de interacción, relacionadas con el usuario, que para ser soportadas por el sistema, deben ser definidas en la arquitectura del software. En el mismo trabajo se propone una solución en forma de patrón software para cada escenario, lo cual será de gran apoyo y guía al momento de la creación de instrumentos para la evaluación realizada en esta investigación. Esta no es la única investigación de Bass sobre el tema, ya que tanto el cómo su equipo, realizan periódicas investigaciones y análisis sobre el área desde distintas perspectivas.

También existe el proyecto Software Architecture that supports Usability (STATUS), financiado por la Comisión Europea en su Programa Sociedad de la Información Tecnologías, la línea de acción IV.3. El objetivo del proyecto STATUS es estudiar y determinar las conexiones entre la arquitectura de software y la usabilidad del sistema resultante, este explica las características de las arquitecturas de software que mejoran la usabilidad. Esta investigación brinda un punto de vista interesante sobre la aplicación de artefactos y mecanismos para evaluar la forma en que la arquitectura afecta a la usabilidad a lo largo del proceso de desarrollo, lo cual servirá de apoyo en la creación y aplicación de artefactos en esta investigación.

En su tesis doctoral Obeso (2005) , llevo a cabo el desarrollo de una metodología de evaluación de usabilidad para sitios web educativos, con un enfoque de evaluación centrado en la perspectiva del usuario y del experto.

Otra documentación importante, es la tesis doctoral de Torrente M. (2011), titulada *SIRIUS: Sistema de Evaluación de la Usabilidad Web Orientado al Usuario y basado en la Determinación de Tareas Críticas,* que tiene como objetivo principal desarrollar un sistema de evaluación para la medición de la usabilidad de cualquier tipo de sitio web mediante la revisión de heurísticas, aplicable en cualquier fase del ciclo de vida, que proporcione un valor

porcentual del nivel de usabilidad del sitio evaluado, cuyos resultados dependan del tipo de sitio en evaluación y que permita inferir los elementos a subsanar en este atendiendo a su prioridad en el arreglo.

2.2 Fundamentos Teóricos

2.2.1 Modelo

Un modelo es una aproximación a lo deseado, concepción de lo perfecto o del deber ser. Se puede describir también como una descripción sistemática simplificada de un fragmento de la realidad, expresado mediante símbolos, signos, graficas o palabras.

Por otra parte, Palella & Martins (2006) lo definen como la representación idealizada de una clase de objetos reales. Entonces, se puede decir que los modelos facilitan la compresión de situaciones o procesos, de distintas áreas del conocimiento humano, ya que son una abstracción de la misma.

- En el proceso de creación, existen las siguientes fases:
- Definición de la situación o ámbito situacional
- Formulación del modelo
- Selección de variables a ser incluidas
- Selección y nivel apropiado de agregación, clasificación y definición de los elementos epistemológicos (conocimiento), ontológicos (ser) y axiológicos (valores).
- Decisiones sobre el tratamiento del tiempo.

Según las especificaciones de los fines u objetivos innovadores para los que se va a utilizar el modelo:

- Simulación y / o validación del modelo.
- Evaluación del modelo.

La genuina producción teórica del modelo viene dada por su cualidad innovadora, asociada directamente a la idea de producir una idea que genere mejoras y acarree cambios en algún sector. Tomando como base todo lo anterior, se puede definir un modelo como la representación racional, innovadora, lógica, comprobable y original que permite construir principios epistemológicos (conocimiento), axiológicos (valores) y ontológicos (ser), creando

una nueva manera de hacer, pensar o aprender algo. Entre las ventajas señaladas en relación a los modelos por Hurtado (2010), se tiene que:

- Los modelos muestran los aspectos más relevantes de aquello que representan y facilitan su comprensión.
- Permiten establecer nexos entre diversas disciplinas.
- Permiten conectar los aspectos teóricos de una situación con sus consecuencias en la realidad.
- Tienen valor heurístico y permiten el surgimiento de nuevas investigaciones y aplicaciones prácticas del conocimiento.

Según Hurtado (2010), existen diversos tipos de modelos que se pueden clasificar en:

Según su objetivo:

- Modelo representativo: es una representación simplificada de una situación u objeto en base a sus características fundamentales, facilitando así la visualización del objeto en su conjunto.
- Modelo explicativo o teórico: permite ilustrar la explicación de un proceso o fenómeno, integrando los diferentes eventos que forman parte del mismo y la manera como se relacionan e interactúan.
- Modelo normativo: proporciona un conjunto de criterios y pautas para la acción, como lineamientos que permiten llevar a cabo cada una de las etapas de un proceso, a modo de plan de acción.
- Modelo operativo: da cuenta del funcionamiento de ciertos objetos, artefactos o de algún fenómeno en general, visualizando sus proceso de funcionamiento internos.

Según su nivel de abstracción:

- Modelo formal: está compuesto por un conjunto de conceptos vinculados mediante ciertas relaciones de tipo lógica o analógica, se pueden derivar de las relaciones entre un conjunto de datos de forma inductiva o mediante proceso inductivos. El modelo formal puede estar basado en metáforas o analogías.

- Modelo material: relaciona características concretas en lugar de conceptos o analogías, se basa en referentes empíricos, representan a un objeto o constructo, por lo general son representaciones a escala de un objeto físico real.

Según el tipo de representación utilizada:

- Verbales: es aquel en el cual se utilizan las descripciones y explicaciones verbales como lenguaje para expresar el fenómeno que se desea representar.
- Grafico o icónicos: este tipo de modelos recurre a los gráficos tales como dibujos, imágenes y diagramas, como medio para expresar sus representaciones.
- Matemáticos: son aquellos que utilizan símbolos numéricos y sus relaciones matemáticas como medio de representación, por lo general toman forma de ecuaciones.
- Modelo físico: son los que utilizan materiales físicos para crear la representación.

Los modelos, por ser representaciones de innovación deben adaptarse al constante cambio de las organizaciones, por lo cual se hace necesario que los mismos sean dinámicos para permitir tu adaptación a la realidad existente, mas aun tratándose de un área del conocimiento en continuo cambio como los sistemas de información.

Según las diversas clasificaciones anteriormente mostradas, el modelo que se utilizó en esta investigación según sus objetivos es de tipo explicativo o teórico ya que ilustró el proceso mostrando cómo se relacionaban e interactuaban los eventos, evidenciando sus mejoras. Según su nivel de abstracción es un modelo formal, ya que está compuesto por conceptos vinculados entre sí, relacionados de manera lógica, y según su representación fue de tipo tanto verbal como grafico.

2.2.2 La Experiencia del Usuario

Según D'Hertefelt (2000) la Experiencia del Usuario representa un cambio emergente del propio concepto de usabilidad, donde el objetivo no se limita a mejorar el rendimiento del usuario en la interacción (eficacia, eficiencia y facilidad de aprendizaje), sino que se intenta resolver el problema estratégico de la utilidad del producto y el problema psicológico del placer y diversión de su uso.

2.2.2.1 Origen

Según Kankainen (2002), el origen del concepto de Experiencia del Usuario se encuentra en el campo del Marketing, vinculado directamente con el concepto de Experiencia de Marca (pretensión de establecer una relación familiar y consistente entre consumidor y marca). En el contexto del Marketing, un enfoque centrado en la Experiencia del Usuario conllevaría no sólo analizar los factores que influyen en la adquisición o elección de un determinado producto, sino también analizar cómo los consumidores usan en producto y la experiencia resultante de su uso.

2.2.2.2 Definición

Encontrar una definición consensuada de un concepto de tan reciente aplicación en el campo del diseño es una tarea difícil. Por lo tanto se tomaran distintos modelos y conceptos propuestos, con la intención de clarificar y componer una aproximación a la Experiencia del Usuario. Arhippainen y Tähti (2003) definen la Experiencia del Usuario sencillamente como la experiencia que obtiene el usuario cuando interactúa con un producto en condiciones particulares. En otro trabajo Arhippainen (2003) la define como las emociones y expectativas del usuario y su relación con otras personas y el contexto de uso.

Knapp (2003) es más específico al definirla como un conjunto de ideas, sensaciones y evaluaciones del usuario resultado de la interacción con un producto; es resultado de los objetivos del usuario, las variables culturales y el diseño del interfaz", especificando no sólo de qué fenómeno es resultante, sino también qué elementos la componen y qué factores intervienen en la interacción. En el contexto de la Web, DNX (2005) definen la buena experiencia del usuario como un objetivo, lo que se persigue es generar sensaciones y valoraciones de los usuarios hacia nuestro sitio web lo más agradables, positivas y satisfactorias posibles, además de reseñar la "fidelidad del usuario" como consecuencia de alcanzar este objetivo.

Nielsen Norman Group (2003) la define como termino integrador de todos los aspectos de la interacción entre el usuario final y la compañía, sus servicios y productos. Aunque esta definición resulta bastante abstracta, es destacable el análisis de la experiencia de interacción más allá que como un fenómeno interactivo entre usuario y producto, sino también entre

usuario y proveedor. Por razones similares algunos autores (Bou Bouza; 2003) defienden el estudio de los sitios web desde su visión como servicios, y no como productos.

Por otro lado, Dillon (2001) propone un sencillo modelo que define la Experiencia del Usuario como la suma de tres niveles: Acción, qué hace el usuario; Resultado, qué obtiene el usuario; y Emoción, qué siente el usuario. La diferencia respecto a las anteriores definiciones es que el autor descompone el fenómeno causante (interacción) en dos niveles, Acción y Resultado; y enfatiza el aspecto emocional de la experiencia resultante. En resumen, se puede definir la Experiencia del Usuario como la sensación, sentimiento, respuesta emocional, valoración y satisfacción del usuario respecto a un producto, resultado del fenómeno de interacción con el producto y la interacción con su proveedor.

2.2.2.3 Factores que componen la Experiencia del Usuario.

Uno de los modelos más completo y exhaustivo de descomposición de la Experiencia del Usuario, es el propuesto en los trabajos de Arhippainen y Tähti (2003). Las autoras clasifican los diferentes factores en cinco grupos diferenciados: factores propios del usuario, factores sociales, culturales, del contexto de uso y propios del producto.

Para Kankainen (2002) la Experiencia del Usuario es resultado de una acción motivada en un contexto determinado, haciendo especial énfasis en la importancia condicionante de las expectativas del usuario y las experiencias previas, y por tanto en la capacidad de influencia de la actual experiencia en sus expectativas y futuras experiencias.

En el contexto de la Web, Morville (2004) propone el análisis de la experiencia del usuario es base a siete facetas o propiedades que debe cumplir un sitio web: útil, usable, deseable, encontrable, accesible, creíble y valioso. en este mismo contexto Mahlke (2002) identificaba cuatro propiedades del producto web (utilidad, facilidad de uso, cualidad de hedónico y atractivo visual) como factores percibidos por el usuario y que influirán con distinto peso en la intención de uso del producto.

2.2.3 Usabilidad

La definición de arquitectura de software, enuncia que esta es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto y los principios que orientan su diseño y evolución (IEEE std 1471-2000). El objetivo fundamental de la arquitectura de software es aportar elementos que faciliten la toma de decisiones y al mismo tiempo proporcionar conceptos y un lenguaje común para establecer la comunicación entre los equipos participantes en el proyecto, para lo cual se construyen abstracciones materializadas en forma de diagramas comentados.

Una aplicación normalmente se diseña con la intención de satisfacer las necesidades de una audiencia concreta y determinada, por lo que será más usable cuanto más adaptado esté su diseño a esta audiencia específica, y por tanto menos lo esté para el resto de personas (Hassan, Martín Fernández, Iazza, 2004). Por lo tanto pretender que una aplicación sea usable independientemente de quién y cómo la use se corresponde más con una visión o enfoque universalista de la usabilidad, que con una visión realista y práctica.

El concepto de usabilidad no sólo puede ser definido como atributo de calidad de un producto, sino consecuentemente, como metodología de diseño y evaluación. en este sentido se suele hablar de ingeniería de la usabilidad (UE, Usability Engineering), un conjunto de fundamentos teóricos y metodológicos que aseguren el cumplimiento de los niveles de usabilidad requeridos para la aplicación y diseño centrado en el usuario (UCD, User Centered Deisgn) conjunto de procesos y metodologías que aseguren empíricamente el cumplimiento de los niveles de usabilidad requeridos para el producto (Hassan, Martín Fernández, Iazza, 2004). Este conjunto de métodos y técnicas puede ser clasificado en cuatro grandes categorías: métodos de indagación, de prototipado y categorización, de inspección, y de test (Hom, 1998).

La ingeniería de usabilidad se puede definir como una aproximación al desarrollo de sistemas en la que se especifican niveles cuantitativos de usabilidad a priori, y el sistema se construye para alcanzar dichos niveles, que se conocen como métricas (Preece, Rogers, Sharp, Benyon, Holland, Carey; 1994), La ingeniería de usabilidad proporciona un modo práctico de asegurar que el software desarrollado alcanza un cierto nivel de usabilidad y está basada en la evaluación mediante test de usabilidad con usuarios.

2.2.3.1 Atributos de Usabilidad

La usabilidad es una cualidad demasiado abstracta como para ser medida directamente. Para poder estudiarla se descompone habitualmente en los siguientes cinco atributos básicos (Nielsen, 1993):

- Facilidad de aprendizaje: Cuán fácil es aprender la funcionalidad básica del sistema, como para ser capaz de realizar correctamente la tarea que desea realizar el usuario. Se mide normalmente por el tiempo empleado con el sistema hasta ser capaz de realizar ciertas tareas en menos de un tiempo dado (el tiempo empleado habitualmente por los usuarios expertos). Este atributo es muy importante para usuarios noveles.
- Eficiencia: El número de transacciones por unidad de tiempo que el usuario puede realizar usando el sistema. Lo que se busca es la máxima velocidad de realización de tareas del usuario. Cuanto mayor es la usabilidad de un sistema, más rápido es el usuario al utilizarlo, y el trabajo se realiza con mayor rapidez. Nótese que eficiencia del software en cuanto su velocidad de proceso no implica necesariamente eficiencia del usuario en el sentido en el que aquí se ha descrito.
- Recuerdo en el tiempo: Para usuarios intermitentes (que no utilizan el sistema regularmente) es vital ser capaces de usar el sistema sin tener que aprender cómo funciona partiendo de cero cada vez. Este atributo refleja el recuerdo acerca de cómo funciona el sistema que mantiene el usuario, cuando vuelve a utilizarlo tras un periodo de no utilización.
- Tasa de errores: Este atributo contribuye de forma negativa a la usabilidad de un sistema. Se refiere al número de errores cometidos por el usuario mientras realiza una determinada tarea. Un buen nivel de usabilidad implica una tasa de errores baja. Los errores reducen la eficiencia y satisfacción del usuario, y pueden verse como un fracaso en la transmisión al usuario del modo de hacer las cosas con el sistema.
- Satisfacción: Éste es el atributo más subjetivo. Muestra la impresión subjetiva que el usuario obtiene del sistema.

Algunos de estos atributos no contribuyen a la usabilidad del sistema en la misma dirección, pudiendo ocurrir que el aumento de uno de ellos tenga como efecto la disminución de otro. Por ejemplo, esto puede ocurrir con la facilidad de aprendizaje y la eficiencia. Es preciso

realizar el diseño del sistema cuidadosamente si se desea tanto una alta facilidad de aprendizaje como una alta eficiencia; siendo el uso de aceleradores (combinaciones de teclas que ejecutan operaciones de uso habitual) la solución más común para conjugar ambos atributos de usabilidad.

La usabilidad del sistema no es una simple adición del valor de estos atributos, sino que se define para cada sistema como un nivel a alcanzar para algunos de ellos. Estos cinco atributos pueden descomponerse a su vez para conseguir una mayor precisión en los aspectos de usabilidad en los que se quiere poner mayor énfasis. Por ejemplo, rendimiento en uso normal y uso de opciones avanzadas son ambos subatributos de eficiencia, mientras que primera impresión es un subatributo de satisfacción.

2.2.4 Componentes de Software

Según Szyperski (2002), un componente de software es una unidad de composición con interfaces contractualmente especificadas y explícitas sólo con dependencias dentro de un contexto. Un componente de software puede ser desplegado independientemente y es sujeto a la composición de terceros.

Sin embargo, Colmenares (2003) indica que más allá de su definición existen algunas características claves para que un elemento pueda ser catalogado como componente:

- Identificable: Debe tener una identificación que permita acceder fácilmente a sus servicios y que permita su clasificación.
- Auto contenido: Un componente no debe requerir de la utilización de otros para finiquitar la función para la cual fue diseñado.
- Puede ser remplazado por otro componente: Se puede remplazar por nuevas versiones u otro componente que lo remplace y mejore.
- Con acceso solamente a través de su interfaz: Debe asegurar que estas no cambiaran a lo largo de su implementación.
- Sus servicios no varían: Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación sí.
- Bien Documentado: Un componente debe estar correctamente documentado para facilitar su búsqueda si se quiere actualizar, integrar con otros, adaptarlo, etc.

- Es genérico: Sus servicios debe servir para varias aplicaciones.
- Reutilizado dinámicamente: Puede ser cargado en tiempo de ejecución en una aplicación.
- Independiente de la plataforma: Hardware, Software, S.O.

La clasificación de componentes es un proceso extenso, ya que un componente involucra en sí mismo varios atributos relevantes. En este segmento se expone una serie de variables que podrían considerarse criterios de clasificación de componentes:

Tamaño: El tamaño de los componentes puede ser medido por medio de las métricas utilizadas en diseño orientado a objetos. Esto significa que la medición del tamaño de un componente puede ser medido a través de: Líneas de Código (LDC) y Orientadas a Función.

Complejidad: En algunas ocasiones, son utilizadas métricas de tamaño para evaluar la complejidad, pero es recomendable hacer uso de otro tipo de métricas. "Si un componente es demasiado trivial no podrá sacársele mayor provecho en su reutilización, y si el componente es demasiado complejo será difícil asegurar su calidad", (Cai Xia, 2001, p. 374-375).

Frecuencia de re- uso: El número de veces que ha sido utilizado un componente dentro de distintas aplicaciones, es sin lugar a dudas el mejor indicador de frecuencia de re– uso. Cabe anotar que este atributo puede ser sólo medido en componentes que ya han sido expuestos al mercado.

Confiabilidad: Es la probabilidad de falló en el funcionamiento del componente dentro de cierto escenario operacional.

2.2.5 Arquitectura de Software

Aunque para la mayoría de los usuarios la interfaz es la aplicación, puesto que es la parte que ven y a través de la cual interactúan según Hartson, (1998), la usabilidad de la aplicación depende no sólo del diseño del interfaz, sino también de su arquitectura (estructura y organización), en otras palabras, del componente no visible del diseño. Folmer y Bosch (2003) estudian este hecho en aplicaciones software concluyendo que el diseño a nivel de arquitectura tiene una gran influencia en la usabilidad del sistema.

Existen muchas definiciones de arquitectura del software y no parece que ninguna de ellas haya sido totalmente aceptada. en un sentido amplio se puede decir que la arquitectura del software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de:

- Definir los módulos principales.
- Definir las responsabilidades que tendrá cada uno de estos módulos.
- Definir la interacción que existirá entre dichos módulos.
- Control y flujo de datos.
- Secuenciación de la información.
- Protocolos de interacción y comunicación.
- Ubicación en el hardware.

La arquitectura del software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño. La definición oficial la arquitectura del software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución (IEEE Std 1471-2000).

El objetivo principal de la arquitectura del software es aportar elementos que ayuden a la toma de decisiones y, al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto. Para conseguirlo, la arquitectura del software construye abstracciones, materializándolas en forma de diagramas (blueprints) comentados. No hay estándares en cuanto a la forma y lenguaje a utilizar en estos blueprints. De todas formas, existe consenso en cuanto a la necesidad de organizar dichas abstracciones en vistas, tal y como se hace al diseñar un edificio. La cantidad y tipos de vistas difieren en función de cada tendencia arquitectónica.

2.2.5.1 Arquitectura como decisiones tempranas de diseño.

El diseño de la arquitectura describe la forma en que el sistema está compuesto. Esto hace que se creen una seria de restricciones a la implementación como la forma de comunicación, y como se van a asignar los recursos. La arquitectura también afecta la organización, esto se debe a que a partir de la misma arquitectura se pueden deducir grupos de trabajo, ya sea para

la parte de desarrollo, pruebas, mantenimiento, etc. Lo que implica entrenamientos, herramientas específicas y conocimiento.

La descripción de la arquitectura también establece ciertos parámetros en la adquisición de componentes comerciales, basándose en su facilidad de integración.

2.2.5.2 Impactos en el negocio de las decisiones arquitectónicas.

Los beneficios de un sistema de software sólo se pueden verificar comparando con las metas que deba cumplir el sistema para el negocio. Estos beneficios resultan de las interacciones entre la funcionalidad del sistema y sus atributos de calidad. En la mayoría de casos, los atributos de calidad son propuestos por las decisiones del diseño arquitectónico. Es por eso que el diseño arquitectónico es crucial para hacer un análisis del costo beneficio.

2.2.5.3 Relación entre Arquitectura del Software y Usabilidad

La usabilidad de una interfaz puede ser analizada mediante la identificación de tres aspectos: look, feel y user model, como se muestra en la Figura 1. El look abarca aspectos tales como las señales visuales, la retroalimentación y la estética. Transmitir las señales visuales que den la capacidad del usuario para realizar una acción. Por ejemplo, el aspecto tridimensional de los botones en las interfaces tradicionales es una señal que dice "push me". Feedback ofrece reconocimiento a los usuarios de que la solicitud se está realizando, y la estética incluye todos los aspectos subjetivos, tales como el uso del color, estilos de línea, y la tipografía. El feel incluye las asignaciones de teclado, las asignaciones de botón izquierdo del ratón, estructura de menús, atajos y las convenciones de interfaz de navegación. En conjunto, la apariencia de una interfaz es similar a la sintaxis de un lenguaje. Ellos describen como un usuario puede interactuar con un sistema pero no lo que el usuario puede pedir al sistema que haga.

El modelo de usuario se compone de aspectos relacionados con lo que el usuario está tratando de lograr, o en otras palabras, los objetivos de la tarea del usuario. Continuando con la analogía del lenguaje, los aspectos relacionados con la meta-representan la semántica. Son portadores de significado en la conversación entre el usuario y el sistema. El user model proporciona un marco comprensible y coherente de los conceptos que los usuarios pueden relacionarse, y que permite a los usuarios realizar sus tareas.

Un modelo de usuario suele ser descrito en términos de los objetos de usuario, el comportamiento y las propiedades de esos objetos, y sus interrelaciones. Aquellos que estén familiarizados con la programación orientada a objetos (POO) pueden pensar que esto suena igual, y tienen razón, hasta cierto punto. La diferencia es que la POO se refiere principalmente a aspectos de la aplicación, mientras que los modelos de usuario tratan sólo con los aspectos que los usuarios se espera que la experiencia mediante el aprendizaje y uso, tales como los objetos, o cosas, los usuarios utilizan para realizar su trabajo.

En el uso de un sistema, una aplicación o un sitio Web para realizar una tarea, el primer obstáculo que cualquier usuario tiene que superar es la de mapear sus metas a las capacidades disponibles. Estas funciones se entienden mejor en términos de las "cosas" que se proporcionan a los usuarios al trabajar, y todas las características de esas cosas, sus propiedades (valores), conductas (lo que pueden hacer), y sus relaciones (cómo trabajan en conjunto y cómo puede el usuario combinarlas).

En un user model se describen todos estos aspectos de una manera coherente, e idealmente de una manera que los usuarios encuentren intuitiva. Las metáforas se emplean a veces en los diseños para ayudar a saltar a los usuarios de los modelos familiares a los nuevos. Los usuarios a desarrollar un modelo mental de cualquier sistema con el que se relacionan. Aunque los modelos mentales pueden ser subconscientes, los modelos dirigen sus interacciones e influyen en su éxito o fracaso. Un modelo mental del usuario se refiere a veces como modelo conceptual del usuario. Idealmente, el modelo mental del usuario coincide con el modelo de usuario que el diseñador crea.

Los aspectos de apariencia tradicional de interfaces gráficas de usuario suelen ser dictadas por la plataforma en la que se encuentra la aplicación. La combinación de la caja de herramientas de una plataforma de interfaz de usuario y guía de estilo suelen ser más que aspectos de la apariencia. Para las aplicaciones de interfaz gráfica de usuario, el modelo de usuario es prácticamente determinado por el sistema operativo y diseño de aplicaciones. Para aplicaciones basadas en Web, el modelo de usuario del sistema operativo se ha vuelto menos relevante, si es que existe. En ambos ambientes la principal fuente de los conceptos de modelo de usuario es el diseño de la aplicación en sí.

Por ejemplo, el kit de herramientas de Windows proporciona los campos de entrada, botones, casillas de verificación, una variedad de cuadros de mensaje, y cuadros de diálogo estándar, junto con una guía de estilo documentada con las buenas prácticas de uso. El modelo de usuario común se compone de datos y directorios, representados como archivos y carpetas en un escritorio, un portapapeles con cortar / copiar / pegar las acciones y programas, accesos directos y representados por medio de asociaciones de tipo de datos. Aplicaciones contribuyen conceptos tales como los párrafos, secciones, tablas, encabezados y pies de página, diapositivas y presentaciones de diapositivas, hoja de cálculo de filas y columnas, la fórmula, y así sucesivamente.

En primer lugar, es necesario que los usuarios estén representados en las primeras etapas del diseño. Es necesario comprender lo que son, en términos de sus habilidades y motivaciones, y las tareas que realizan. En segundo lugar, se puede tomar ventaja del conocimiento de los usuarios de tareas y métodos de ingeniería de software para modelar los conceptos, así como la imagen y apariencia. Los modelos de usuario son la base de diseño. Que no sólo sirven como marcos a través de la cual podemos evaluar y validar los diseños, sino también como una fuente para la documentación crucial que se necesita para coordinar y dirigir un equipo de desarrollo, incluyendo programadores, diseñadores gráficos, escritores, y los probadores.

De esta manera se entiende que el user model constituye los cimientos de un proyecto, y si bien es indispensable, no todos hacen uso de él, se pueden construir aplicaciones de mil maneras pero si no se tienen en cuenta estos tres elementos, sus diferencias y la importancia de cada uno, se pueden tener múltiples problemas a lo largo del proceso de desarrollo.

Sin embargo estos tres puntos están conectados, la suma de los tres da un sitio exitoso. La usabilidad no sólo tiene que ver con las interfaces de las aplicaciones sino también con las estructuras, los diseños y los esqueletos que sostienen a esa interfaz, es todo ese trabajo que resulta invisible a simple vista del usuario, pero que en un plano mental, le dirá que la aplicación funciona. Es un trabajo arduo que ni los usuarios ni los clientes pueden ver, son como las redes hidráulicas de una ciudad, indispensables pero invisibles.

Se conocen una serie de escenarios de usabilidad sensibles a la arquitectura del software. El trabajo de Bass y de este equipo se basa en inventariar una serie de escenarios de usabilidad

sensibles a la arquitectura del software. Esto significa determinar una serie de momentos de interacción o tareas del usuario que, para que el sistema los pueda soportar, tienen que estar definidos en la arquitectura del software.

Hasta el momento, este inventario está compuesto de 26 escenarios que se pueden consultar en el documento CMU/SEI-2001-TR-005 de Bass y otros. La lista completa es demasiado extensa, pero a modo de ejemplo, se enumeran a continuación los cinco primeros escenarios:

- Agregación de datos: Poder aplicar simultáneamente una acción a un conjunto de datos u objetos.
- Agregación de comandos: Agrupar acciones que se puedan ejecutar de una sola vez.
- Comandos de cancelación.
- Uso de aplicaciones de forma concurrente.
- Validación automática de la entrada de datos.

Bass continúa con una propuesta de patrón arquitectural para cada escenario (sobre el tema de patrones, con la cual el objetivo es doble:

- Facilitar a los expertos en usabilidad una checklist con escenarios que muestre aspectos de usabilidad importantes a tener en cuenta en tiempo de requerimientos.
- Facilitar a los arquitectos patrones que los guíen en el momento de dar soporte a los escenarios.

2.2.6 Patrones Arquitecturales para Aplicaciones Empresariales

Para Fowler (2002), el objetivo del patrón es una solución particular, que es a la vez común y eficaz en el tratamiento de uno o más problemas recurrentes. Una parte clave de los patrones es que están arraigados en la práctica. Se pueden encontrar patrones que observan lo que la gente lo hace, o de las cosas que funcionan, y luego buscan el "núcleo de la solución."

Para estructurar aplicaciones empresariales sofisticadas, especialmente aquellas que operan a través de varias capas de procesamiento. La arquitectura, las partes más importantes de una aplicación empresarial y cómo se conectan esas partes. Los patrones arquitectónicos representan decisiones importantes sobre esas partes, los patrones de diseño ayudan a materializar esa arquitectura. El patrón arquitectónico dominante es el de capas, cómo se

descompone una aplicación empresarial en capas y cómo estas capas trabajan juntas. Para más detalles este tema ver Anexo 1.

Describiendo de esta forma el marco teórico que integra este trabajo, a continuación se presenta la metodología a través de la cual se propondrá el modelo, así también la definición de la muestra, y como se llevaron a los objetivos específicos de la investigación.

Capítulo III

Marco Metodológico

En este capítulo se describen las bases metodológicas utilizadas para la realización de la investigación, así como la definición de la población y muestra de la misma, los métodos de recolección de información y las variantes involucradas en el proceso.

3.1 Consideraciones generales

Investigar es identificar la composición de un todo, componentes e interacción entre ellos. Evidentemente, el análisis de la usabilidad no puede limitarse a la mera identificación o cuantificación de componentes, elementos o principios, este debe concebirse como una fundada descripción de los contenidos, y una determinación lo más exacta posible de sus características, sus principios y relaciones.

Lo último es de vital importancia, ya que la sola identificación de los componentes no puede develar su interacción, si bien es cierto que sus características dan una idea de las funciones y relaciones que podrían cumplir en la estructura, la interacción de los componentes devela la interacción de sus partes, es decir el dinamismo de una estructura.

3.2 Tipo de Investigación

La investigación planteada, identificó la necesidad de proponer dentro del marco de las ciencias de la computación, específicamente en el área de la ingeniería del software un modelo para la optimización en el diseño de arquitecturas de software en aplicaciones empresariales con base en la usabilidad en las PYMES consultoras venezolanas.

Planteándose esta como la investigación, elaboración y desarrollo de un modelo operativo viable para solucionar los requerimientos o necesidades (como la detección de situaciones de sensibilidad de la usabilidad dentro del proceso de desarrollo en los patrones y su posterior corrección), se definió como un proyecto factible, el cual consistió según la definición dada por la Universidad Experimental Simón Rodríguez (1999) en: "una proposición sustentada en

un modelo operativo factible, orientada a resolver un problema planteado o a satisfacer necesidades de una institución o campo de interés nacional" (p. 79).

Del mismo modo, Arias (2006) señala: "Que se trata de una propuesta de acción para resolver un problema práctico o satisfacer una necesidad. Es indispensable que dicha propuesta se acompañe de una investigación, que demuestre su factibilidad o posibilidad de realización" (p. 134).

Para llevar a cabo el proyecto factible, primero debió realizarse un diagnóstico de la situación planteada; en segundo lugar, plantear y fundamentar con basamentos teóricos la propuesta a elaborar, tanto los procedimientos metodológicos como las actividades y los recursos necesarios para realizar la investigación, y posteriormente la ejecución de la propuesta con su respectiva evaluación.

Al ser este trabajo el resultado del estudio de una situación que se encuentra bajo la observación directa sobre las características del problema, se llevo a cabo una exploración practica a través de un caso de estudio revisando el proceso de desarrollo de las PYMES consultoras venezolanas, según Yin (1994), el caso de estudio se define como:

> Una investigación empírica que estudia un fenómeno contemporáneo dentro de su contexto de la vida real, especialmente cuando los límites entre el fenómeno y su contexto no son claramente evidentes. (...) Una investigación de estudio de caso trata exitosamente con una situación técnicamente distintiva en la cual hay muchas más variables de interés que datos observacionales; y, como resultado, se basa en múltiples fuentes de evidencia, con datos que deben converger en un estilo de triangulación; y, también como resultado, se beneficia del desarrollo previo de proposiciones teóricas que guían la recolección y el análisis de datos (p. 13).

Ya que se debió producir un conocimiento de la realidad, y dada la complejidad de la tecnología, se requirió de métodos de análisis que contemplaran nuevos paradigmas en materia de investigación del ámbito.

3.3 Diseño de la Investigación

Según Hurtado (2010) el diseño de la investigación "se refiere a dónde y cuándo se recopila la información, así como la amplitud a recopilar, de manera que dé respuesta a la pregunta de la investigación de la forma más idónea posible." (p. 119).

La forma adoptada fue aquella que sustento las fases del proyecto factible, por lo tanto, según la dimensión táctica se tomó un diseño de fuente mixta (documental y de campo) y transeccional (cuando se desea obtener información de un evento actual en un único momento del tiempo), que según lo indicado por Zorrilla (1993) "La investigación mixta es aquella que participa de la naturaleza de la investigación documental y de la investigación de campo." (p. 43).

Al ser este trabajo el resultado del estudio de una situación determinada, que se encuentra basada en el análisis de la forma en que se manifiesta un fenómeno y sus componentes, permitiendo detallar el fenómeno estudiado a través de la medición de uno o más de sus atributos en ambientes reales (para más detalles sobre el contexto ver Figura 3), se optó por un diseño con carácter de campo, el cual consistió en un estudio del problema para ampliar y profundizar el conocimiento de su naturaleza, basada en el análisis de datos, seleccionando una serie de aspectos y midiéndolos independientemente.

Pero dado que muchas de las fuentes principales de investigación fueron documentos elaborados por expertos en el área, se realizó una exhaustiva revisión de fuentes bibliográficas, desarrollando un análisis que apoye el cumpliendo con los objetivos planteados, lo cual le da a este estudio también la naturaleza documental, convirtiéndolo en una investigación mixta.

3.3 Población y Muestra

La población de esta investigación se corresponde con las pequeñas y medianas empresas consultoras del mercado venezolano (PYMES), siendo este dominio muy amplio, fue necesario seleccionar una muestra representativa, como una referencia que permitiera la verificación de los elementos que sustentarían la investigación, por la cual se separan la población y la muestra. Según señala Tamayo (2003), la población es la totalidad del

fenómeno a estudiar donde las unidades de población poseen una característica común, la cual se estudia y da origen a los datos de la investigación.

Y la muestra, según Sabino (2000), no es más que eso, una parte del todo que llamamos universo y que sirve para representarlo. Por la extensa población objeto del estudio, se hizo necesario tomar una muestra del total para facilitar el proceso de recolección de datos, siendo esta el subgrupo elegido de la población, representativa de la misma y que reproduce las características de la misma de la forma más exacta posible. Cabe acotar que para esta investigación, se utilizó además una muestra no probabilística, definida por Hernández (1998) como:

> En las muestras no probabilísticas, la elección de los elementos no depende de la probabilidad, sino de causas relacionadas con las características del investigador o del que hacer de la muestra. Aquí el procedimiento no es mecánico, ni con base en fórmulas de probabilidad, sino depende del proceso de toma de decisiones de una persona o grupo de personas. (p. 208).

En la investigación se tomó una muestra no probabilística, pues se considera una selección informal y derivada de un proceso arbitrario. Siendo este un tipo de muestra además muy útil para determinados diseños de estudio, donde más que una gran representatividad en los elementos de la población, se requiere una cuidadosa y controlada selección de sujetos con ciertas características especificadas en la investigación, y dado que la investigación pretendió representar un modelo, la muestra debería ser definida de manera que permitiera abstraer características que proporcionen la aplicación y estudio del mismo.

Para Hernández, Fernández y Baptista (1991), existen varias clases de muestras no probabilísticas, y se hallan clasificadas de la siguiente forma: muestras de sujetos voluntarios, muestras de expertos, sujetos tipos o usuarios afectados y muestras por cuota.

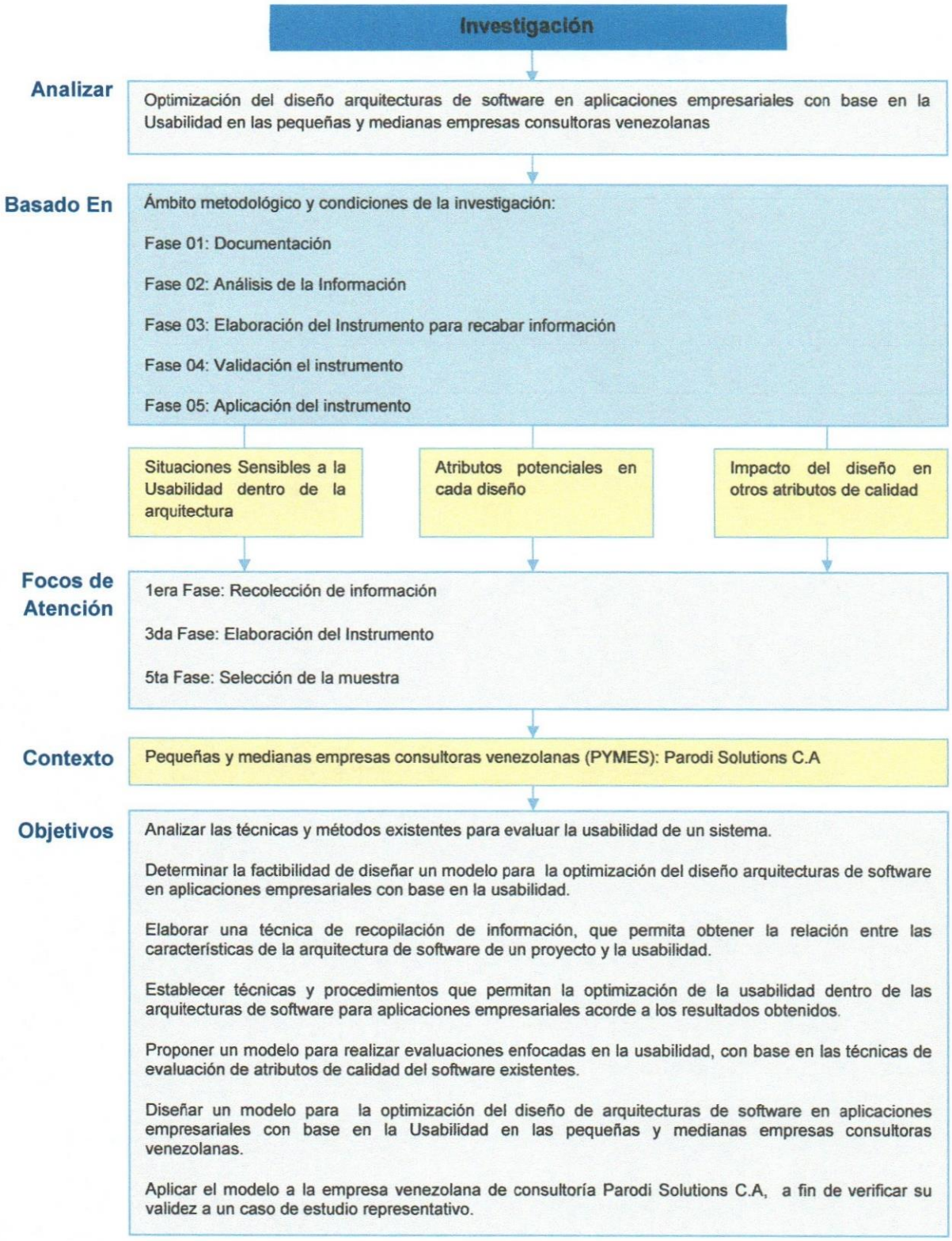


Figura 3. Diseño de la Investigación.

Para lograr la identificación de todos los factores, el estudio tomó como población y como muestra específicamente, una muestra no probabilística de la clase de sujeto tipo o usuarios afectados, de la empresa venezolana de consultoría de software Parodi Solutions C.A, siendo la unidad de análisis el sector relacionado con el diseño y desarrollo de proyectos de tecnología de la información en dicha empresa, el cual se representa en la muestra y en el caso de estudio de la investigación, buscando obtener un acercamiento con la realidad y hallar una perspectiva de evaluación, generalizar la estrategia, y obtener una buena estimación de los criterios para aplicarlos al resto de las empresas venezolanas en el marco de los intereses del negocio. La definición de "stakeholders" tiene una acepción muy amplia; siendo la que más se apega a la utilizada en esta investigación, la dada por Freeman (1983) "El término Stakeholders es utilizado para referirse a un grupo de individuos identificables quienes pueden ser afectados o son afectados por un sistema" (p. 88)

Para la selección de los usuarios afectados o stakeholders, se tomaron en consideración a todos los actores de la población que influencian o son influenciados positiva o negativamente en el proceso diseño y desarrollo, los cuales se consideren que aporten riqueza, profundidad y calidad a la información. Con la finalidad de fijar adecuadamente el alcance de la investigación a continuación se describen los usuarios afectados significativos en la Tabla 1.

Cargo	Cantidad
Director	1
Gerente de Administración y Finanzas	1
Gerente de Tecnología de Información	1
Líderes de Proyecto	3
Usuarios	20
Analistas	5
Líder del área de diseño	1
Vendedores	2
Total	34 usuarios afectados

Tabla 1. Usuarios afectados significativos

3.4 Operacionalización de las Variables

Tamayo (2003), afirma que una variable "se utiliza para designar cualquier característica de la realidad que pueda ser determinada por observación y que pueda mostrar diferentes valores de una unidad de observación a otras." (p. 163).

La usabilidad es una característica que está relacionada con la medida de la calidad de los sistemas interactivos utilizados por usuarios específicos en un contexto de uso, para conseguir objetivos específicos con efectividad, eficiencia y satisfacción, en términos de utilidad, facilidad de uso, facilidad de aprendizaje y apreciación.

- Efectividad: la precisión y completitud con la que el usuario alcanza sus objetivos, relacionados con este concepto está la facilidad de aprendizaje, la tasa de errores cometidos y la facilidad de recordar sus funcionalidades y procedimientos.
- Eficiencia: los recursos empleados relacionados con la precisión y completitud con que el usuario alcanza sus objetivos. Relacionados con este concepto está la facilidad de aprendizaje, la tasa de errores cometidos y la facilidad de la aplicación para ser recordada.
- Satisfacción: la ausencia de incomodidad y una actitud positiva hacia el uso de la aplicación. La aplicación debe ser agradable de usar de forma que el usuario esté satisfecho al utilizarla.

Ya que el ámbito de estos atributos es extenso y puede resultar ambiguo, para el presente estudio fueron desglosados en otras propiedades específicas para precisar las métricas y mecanismos, determinando las variables de estudio de la siguiente manera (ver Tabla 2).

3.5 Técnicas e Instrumentos para la Recolección de Información

La recopilación de datos que oportunamente analizados dieron respuesta a los objetivos de esta investigación procedieron de dos fuentes: la realidad empírica propia de la investigación de campo y las fuentes teóricas dada la naturaleza documental de la investigación. Las dos vertientes estuvieron representadas por las siguientes técnicas:

- Observación Descriptiva: Mediante esta se toman en cuenta todos los elementos que guardan relación con la investigación y se utiliza sin necesidad de ningún instrumento de registro lo cual facilita su aplicación al lugar de los hechos. La observación descriptiva es la más común entre las técnicas de investigación; la observación sugiere y motiva los problemas y conduce a la necesidad de la sistematización de los datos (Tamayo, 2003, p. 55).
- Entrevista: mediante la cual se obtienen datos de manera verbal por parte del informante.
- Cuestionarios: Los datos se obtienen a partir de realizar un conjunto de preguntas normalizadas dirigidas a una muestra representativa o al conjunto total de la población estadística en estudio, con el fin de conocer opiniones o hechos específicos, seleccionando las preguntas más convenientes, de acuerdo con la naturaleza de la investigación. Además de las preguntas y categorías de respuestas, un cuestionario está formado por instrucciones que indican al encuestado cómo contestar. Las instrucciones son tan importantes como las preguntas y es necesario que sean claras para las personas a las que van dirigidas. Y una instrucción muy importante es agradecer al sujeto por haberse tomado el tiempo de contestar el cuestionario (ver Anexo 2).
- Revisión bibliográfica: constara de libros, documentos y artículos, que brindaran al investigador el soporte del marco teórico, pero también será utilizada como base complementaria con el fin de recopilar todos aquellos documentos que permitan confrontar los aspectos teóricos con una situación real dentro del modelo evaluado.

Tomando la observación, encuestas al personal y entrevistas, como técnicas elegidas, se obtuvo una perspectiva diferente a la teórica brindada por consulta de documentos, así como información significativa, para clasificar, comprender y analizar el problema. Además para facilitar la recopilación y clasificación de la información, se utilizaron otros métodos como refuerzo, como las fichas de trabajo, para permitir una mejor organización del contenido extraído de las fuentes documentales.

Objetivo Específico	Variable	Dimensión Conceptual	Dimensiones (factores)	Indicadores (definición operacional)	Medida derivada
• Analizar las técnicas y métodos existentes para evaluar la usabilidad de un sistema.	Técnicas y métodos	Es el conjunto de conocimientos técnicos utilizados para la evaluación de usabilidad u otras propiedades de la calidad del software	Tecnologías utilizadas mundialmente para la evaluación de usabilidad	Madurez	Nivel de madurez
				Campo de aplicación	Posee aplicación en el campo de la arquitectura de software (si / no)
				Dominio	Es reconocida internacionalmente (si / no)
• Determinar la factibilidad de diseñar un modelo para la optimización del diseño arquitecturas de software en aplicaciones empresariales con base en la usabilidad.	Factibilidad	Disponibilidad de los recursos necesarios para llevar a cabo los objetivos o metas señaladas	Técnica, operativa y económica	Disponibilidad de recursos	Es factible (si /no)
• Elaborar una técnica de recopilación de información, que permita obtener la relación entre las características de la arquitectura de software de un proyecto y la usabilidad.	Eficiencia	Resultados de la interacción en términos de la celeridad y errores.	Rendimiento del usuario	Comprensión de los elementos funcionales	Porcentaje de elementos funcionales utilizados sin error
				Eficiencia temporal	Eficiencia = efectividad/tiempo de tarea
				Eficiencia relativa del usuario	eficiencia del usuario = (eficiencia relativa/eficiencia del experto)* 100
				Efectividad de las tareas	% de efectividad = (cantidad * calidad)/100
	Fiabilidad	Medida de conformidad con una especificación de comportamiento. Recuperación tras error, o la indulgencia se incluyen en todos los principios referidos.	Detección de errores	Advertencia ante comandos peligrosos	Si existe o no la emisión de un mensaje de advertencia en el elemento funcional
				Información sobre las acciones irreversibles	Si existe o no la emisión de un mensaje de alerta en el elemento funcional
			Manejo de errores	Efectividad en la detección errores	Porcentaje de errores detectados en un elemento funcional
				Corrección de errores (posibilidad para que los usuarios corrijan su error)	Si un elemento funcional posee esa posibilidad o no

Tabla 2. Operacionalización de Variables.

Objetivo Especifico	Variable	Dimensión Conceptual	Dimensiones (factores)	Indicadores (definición operacional)	Medida derivada
• Elaborar una técnica de recopilación de información, que permita obtener la relación entre las características de la arquitectura de software de un proyecto y la usabilidad.	Satisfacción	Niveles aceptables de costes humanos en términos del cansancio, la molestia, la frustración y el esfuerzo personal	Facilidad de uso	Tiempo para usar	Tiempo medio para utilizar correctamente un elemento funcional
				Tiempo de maestría	Tiempo medio para ser un experto en el elemento funcional
			Costos	Costo en Adquisición	Porcentaje entre el valor deseado y el valor real invertido en la adquisición del producto
				Costo en mantenimiento	Porcentaje entre el valor deseado y el valor real invertido en el mantenimiento del producto
				Costo en soporte	Porcentaje entre el valor deseado y el valor real invertido en el soporte del producto
	Aprendizaje	Relación del funcionamiento al entrenamiento y a la frecuencia del uso, es decir el tiempo del aprender por usuario principiante con entrenamiento especificado, y la retención de parte del usuario acostumbrado	Legibilidad del diseño	Correspondencia entre el sistema y el mundo real	Numero de términos técnicos utilizados en el elemento funcional
				Nombres significativos	Porcentaje de elementos funcionales con nombres relacionados a su funcionalidad
				Estándares en la documentación	Existen o no estándares en los documentos.
			Contenido de los manuales	Cobertura de los manuales	Porcentaje de interfaces y métodos descritos en los manuales
				Efectividad de los manuales	Porcentaje de elementos funcionales utilizados correctamente luego de leer el manual
			Carga de memoria del usuario	Reconocimiento del usuario	Porcentaje de elementos familiares en los elementos funcionales
				Versatilidad en la navegación	Es versátil o no la navegación entre pantallas

Tabla 3. Operacionalización de Variables.

3.6 Validez de los Instrumentos de Recolección de Información

Con respecto a la confiabilidad del instrumento, este fué evaluado y validado por parte de expertos, seleccionados tomando en cuenta los siguientes factores: conocimientos, experiencia en el área y experiencia laboral; los cuales consideraron que el contenido a su juicio permitía obtener información que facilitaría el cumplimiento de los objetivos del trabajo, además de si el vocabulario de este hacia entendible su contenido (ver Anexo 3).

Esta actividad que se realizó en todas las fases de la investigación, a fin de someter el modelo a la consideración y juicio de expertos en la materia facilitando el montaje metodológico del instrumento en forma y fondo, con el fin de hacer las correcciones que tuvieran lugar, y así garantizar la calidad y certidumbre del modelo.

3.7 Análisis e Interpretación de la Información

Para llevar a cabo el análisis de la información obtenida de los datos aportados por los instrumentos diseñados, fue necesario procesarlos, analizarlos y representarlos de manera que la cuantificación y tratamiento estadístico permitieran sustentar el resultado (ver Anexo 6). Se realizó la codificación de los datos y se tabuló para determinar la frecuencia de los casos que se presentan, y en el caso de preguntas de opinión se tomó en cuenta el criterio de repetición del contenido, luego se realizaron cuadros y gráficos representativos, para analizarlos por separados en forma cuantitativa y cualitativa.

Del análisis progresivo de la información estudiada surgieron las conclusiones y recomendaciones, que fueron evaluadas y perfeccionadas a través de un proceso de síntesis, lo cual se entendió como la composición de las partes o elementos de un todo, para integrarlas en una forma coherente que condujera a conclusiones y recomendaciones finales, racionalmente fundamentadas.

3.8 Metodología a utilizar en función de los objetivos específicos de la investigación

El procedimiento metodológico elegido para llevar a cabo el cumplimiento de los objetivos específicos de esta investigación estuvo orientado hacia la rama evaluativa, porque siendo la finalidad de este estudio la generación de un modelo basado en mejorar las condiciones

actuales existentes en la población, la evaluación es el factor clave ya que los resultados de este estudio de esta nutrirán el modelo generado, y de ellos dependió en gran medida las soluciones a proponer.

Razón por la cual fue llevado a cabo mediante la aplicación de una instanciación de la metodología propuesta por Magnus Ramage en 1997, para los sistemas de trabajo cooperativo *(Cooperative Work),* llamada Systemic Evaluation for Stakeholder Learning (SESL): "La combinación de la tecnología, personas, y la organización que facilita la comunicación y coordinación necesaria para que un grupo trabaje efectivamente con la finalidad de alcanzar un objetivo común". La cual describe el proceso evaluativo de la siguiente forma:

> "La evaluación es una tarea compleja, metodológicamente difícil (debido a que los métodos comúnmente utilizados para la evaluación son imprecisos al ser generalizados en múltiples problemas), prácticamente confusa (los efectos del cambio pueden verse sólo después de un largo período de tiempo y en diferentes sitios), psicológicamente compleja (los diseñadores y evaluadores del sistema deben adoptar un estilo Copérnico, cambiando desde la tecnología en uso hasta la organización) y políticamente frustrante (al iniciar la evaluación la organización empieza a ver la poca prioridad que tiene y se inician los conflictos de interés)". (Magnus, 1997, p. 3)

Dicho esto, se evidencia que la evaluación es una tarea compleja, sujeta a muchos factores y por lo cual, debe precisarse primero que se entiende por el término evaluar dentro del contexto de las ciencias de la computación donde se conoce como "estudiar un sistema con la finalidad de saber si cumple con un conjunto de requerimientos y criterios" y las acciones a mejorar, que en este estudio se encuentran relacionadas con las mejoras de uso, el logro de las especificaciones y requerimientos.

Una metodología que propone cómo sortear estas deficiencias es la SESL, para los sistemas de trabajo cooperativo, entre los cuales se enmarca este trabajo de investigación. Esta metodología propone los siguientes pasos dentro de un flujo de estados, como se muestra en la Figura 4. Los pasos propuestos por esta metodología representaran las tres fases que rigen un Proyecto Factible, según muestra la Figura 5.

3.8.1 Determinación la Naturaleza del sistema

Es importante decidir cuál es el sistema a evaluar, a fin de utilizar una herramienta acorde al dominio en el que se encuentre y dado que SESL es una metodología para evaluar sistemas cooperativos (CSCW).

En esta investigación se estudio el contexto dentro del cual se encuentra la tecnología, enfocándose en el estudio inicial social o técnico del sistema, ignorando el hecho de que la tecnología cambia la estructura organizacional, razón por la que se representó el sistema del cual se buscaba inferir el modelo utilizando la teoría general de sistemas.

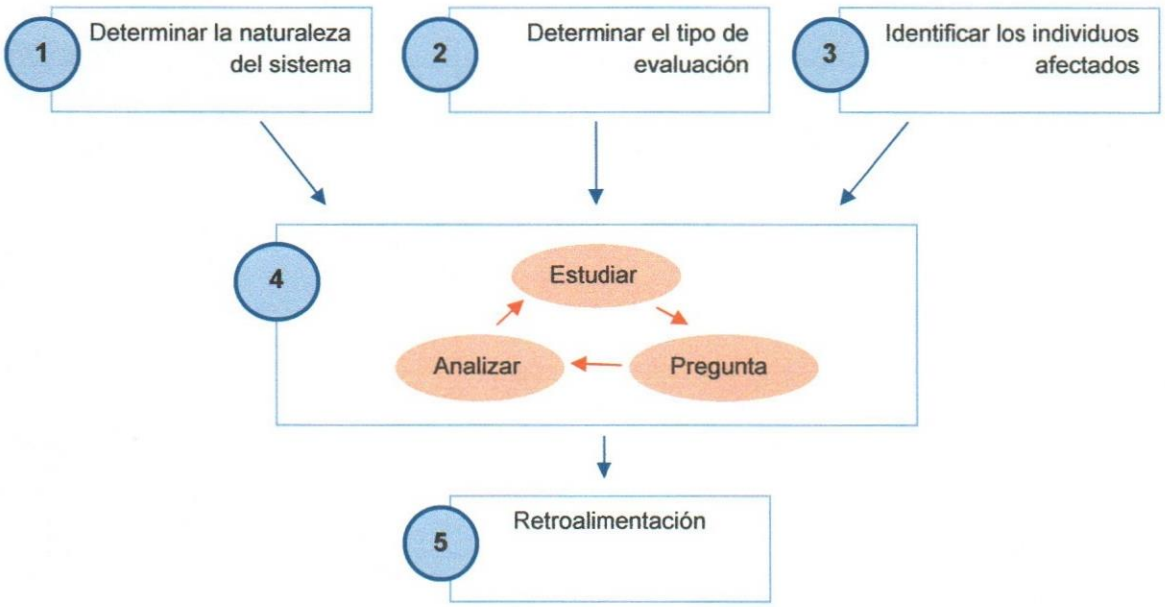


Figura 4. Metodología SESL (System Evaluation for Stakeholder Learning)

Fuente: adaptado de "El desarrollo de una metodología para la evaluación de sistemas cooperativos", Manus, R, 1997

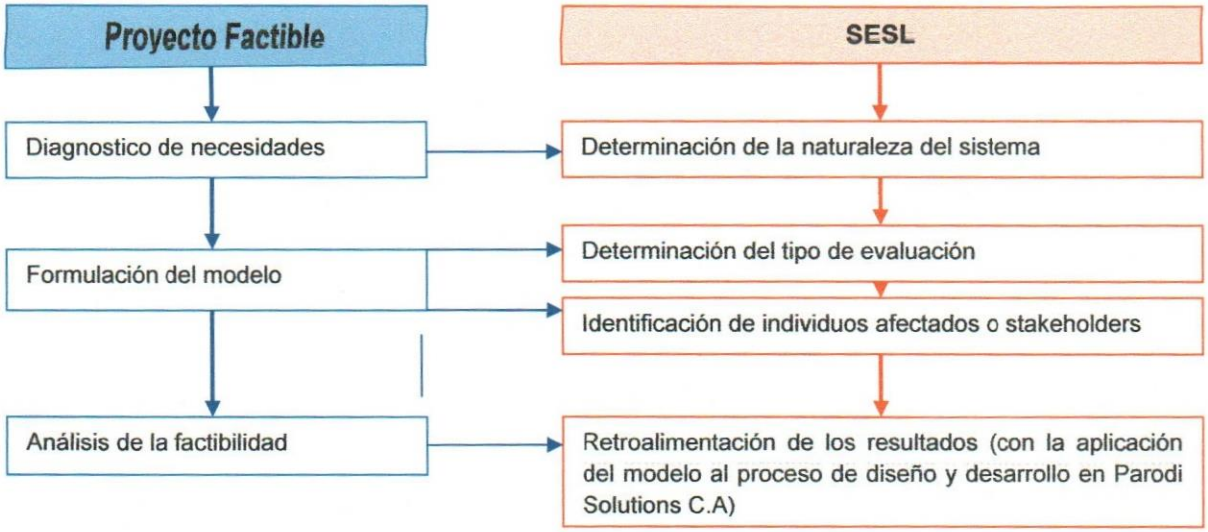


Figura 5. Relación entre Proyecto factible y la metodología SESL para el cumplimento de los objetivos específicos de la investigación.

3.8.2 Tipo de evaluación.

Es de mucha ayuda considerar que tipo de evaluación ,el no tener esto claro puede invalidar los criterios del evaluador. Manus (1997) propone 5 tipos de evaluación:

- Evaluación de los efectos del sistema.
- Evaluación formativa.
- Evaluación de conceptos (en proyectos de investigación).
- Evaluación exclusivamente de los efectos organizacionales y partes psicológicas del sistema.
- Evaluación para la compra de nuevos productos.

3.8.3 Identificación los individuos (Stakeholder)

Se determinó, quienes son los individuos, y cuáles son sus intereses. Se buscaron aquellos a quienes afectaba, la forma en que influencian y son influenciados por el sistema, haciendo que este grupo fuese lo más representativo posible. "El término Stakeholders es utilizado para referirse a un grupo de individuos identificables quienes pueden ser afectados o son afectados por un sistema" (Freeman, 1983, p. 106)

3.8.4 Estudio y Análisis

Para esto se requiere el planteamiento de preguntas claves para guiar el proceso. Una vez establecidas y analizadas las preguntas claves es necesario obtener la información asociada a

estas preguntas en la Organización, dado que la metodología no sugiere una técnica en especial, se decidió que las herramientas utilizadas serán los cuestionarios y entrevistas.

3.8.5 Retroalimentación

En esta parte del proceso se realizó una propuesta, basada en el análisis de la información y lo que se ha determinado a través de la misma. En la siguiente parte se realizó el proceso de construir la decisión, con la finalidad de utilizar una herramienta, implementar un método o proponer una metodología.

Este paso involucró la elaboración de un conjunto de preguntas que sirvieron como instrumento de medición de acuerdo a los insumos y tomando como referencia los stakeholders identificados, cumpliendo además con las restricciones impuestas en los pasos en la identificación del sistema. Y finalmente se realizó la ejecución del modelo a un caso del mundo real utilizando como ambiente la organización venezolana Parodi Solutions C.A, a través del instrumento de evaluación y la ponderación respectiva.

3.9 Viabilidad de la Investigación

La palabra investigación, consiste en un método riguroso, exhaustivo y lo suficientemente preciso como para obtener conclusiones irrevocables y reales, y para la obtención de estas conclusiones la investigación es viable o es factible desarrollar, para obtener un fin, que en este caso es mejor conocimiento científico. Para este estudio, fue necesario considerar las fases del diseño de un proyecto factible, que según Cerda (1995), refiere: diagnóstico, factibilidad, diseño y validación. Así mismo, el autor afirma que la factibilidad de un proyecto tiene como finalidad permitir la selección entre las variantes (si ésta no se ha cumplido en la fase anterior), determinar las características técnicas de la operación, fijar los medios a implementar, establecer los costos de operación y evaluar los recursos disponibles, reales y potenciales.

Así mismo, los resultados del estudio de factibilidad, influyen en las decisiones tomadas por las personas responsables del proyecto. A tal efecto, se puede abandonar el proyecto si el estudio contradice los enfoques anteriores o se pueden continuar los trabajos para superar o resolver los problemas o limitaciones identificadas. En todo caso el tener antecedentes analizados progresivamente en las diferentes etapas del trabajo caracterizan su viabilidad

técnica, económica, financiera, administrativa, social e institucional. Los aspectos señalados anteriormente, conducen al conocimiento del sistema donde se inserta el proyecto, la capacidad de producción, proceso técnico, obras físicas, organización, cronograma del proyecto, presupuestos, financiamiento y la operación del mismo, Gómez (1995). La factibilidad de un proyecto se determina con los siguientes indicativos:

- Factibilidad Económica.
- Factibilidad Técnica.
- Factibilidad Operativa.

3.9.1 Factibilidad Económica

Se refiere a los recursos económicos y financieros necesarios para desarrollar o llevar a cabo las actividades, dado que la mano de obra que es uno recurso más importante y de los más costosos fueron llevadas a cabo por el investigador, se consideró cero (0) como el costo en recursos humanos. Los recursos consumibles que fueron utilizados para la investigación se detallan a continuación, estimado en base a proyectos realizados anteriormente.

Gastos en consumibles	**total**
Libros varios	4600Bs
Cartuchos de tinta para impresora Canon PX	3600Bs
4 Resmas de Papel Bond Blanco Tamaño Carta	2800Bs
Gastos varios	1400Bs
Fotocopias	1220Bs
Pago de nueve (9) créditos académicos por concepto de trabajo especial de maestría (900Bs por UC)	8100Bs
Subtotal	21720Bs

Los recursos de operación, se presentan en la siguiente tabla:

Gastos en recursos de operación	Tiempo estimado	Costo mensual	total
Conexión internet	6 meses	600Bs	3600Bs
Línea de teléfono móvil	6 meses	70Bs	420Bs
Servicio de luz eléctrica	6 meses	40Bs	240Bs
		Subtotal	4260Bs

Generalmente la factibilidad económica es el elemento más importante ya que a través de él se solventan las demás carencias de otros recursos, es lo más difícil de conseguir y requiere de actividades adicionales cuando no se posee. En el caso de esta investigación los gastos resultaron relativamente bajos y por lo tanto accesible para el investigador.

3.9.2 Factibilidad Técnica

Se refiere a los recursos necesarios como herramientas, conocimientos, habilidades, experiencia, etc., que son necesarios para efectuar las actividades o procesos que requiere el proyecto. Generalmente elementos tangibles o medibles. El proyecto debe considerar si los recursos técnicos actuales son suficientes o deben complementarse. Siendo este un proyecto de investigación se baso en conocimientos generados con anterioridad en los que se apoya el estudio, propuestos en el marco teórico, y se buscó la creación de nuevos enfoques y artefactos en el área basándose en las habilidades y experiencia del investigador, al desarrollar esta propuesta se ofrece una respuesta adecuada a las empresas consultoras venezolanas que buscan realizar un estudio en etapas tempranas del desarrollo de software, través de la creación de un modelo basado en la información recabada y la investigación realizada adaptado a la perspectiva de las empresas consultoras en el entorno nacional.

3.9.3 Factibilidad Operativa.

Contempla todos aquellos recursos donde interviene algún tipo de actividad, que depende de los recursos humanos que participaron durante la operación del proyecto. Durante esta etapa se identificaron todas aquellas actividades que son necesarias para lograr el objetivo y se evalúa y determino todo lo necesario para llevarlo a cabo. En el caso de esta investigación las

actividades estuvieron ligadas a los objetivos específicos establecidos en función del cumplimiento del objetivo principal ejecutadas por el investigador. para lo cual se planificaron las actividades mostradas en la Tabla 3.

Por lo tanto luego de analizar cada uno de los enfoques de la factibilidad no se encontró ninguna razón a juicio del investigador por lo cual este proyecto no pudiera ser llevado a cabo dentro del tiempo previsto con los recursos ya mencionados.

	2013																				2014							
	Agosto				Septiembre				Octubre				Noviembre				Enero				Febrero				Marzo			
Descripción de las actividades	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	26	27	28	29
Revisión y análisis de la información obtenida sobre métodos para la evaluación de la usabilidad en patrones arquitecturales																												
Determinar cuáles son los métodos de evaluación de uso más frecuente.																												
Elaboración de un resumen de las Investigaciones.																												
Complementación y/o Verificaciones de las Investigaciones.																												
Evaluación y correlación de la información conseguida sobre los patrones arquitecturales y escenarios en estos donde la usabilidad se vea afectada por el diseño.																												
Elaboración de un instrumento para recabar información de la muestra (encuestas).																												
Validación del Instrumento propuesto.																												
Aplicación de los instrumentos.																												
Analizar la información recolectada con los instrumentos.																												
Diseñar artefactos con la finalidad de documentar los escenarios sensibles a la usabilidad.																												
Deducir métricas para el análisis y mejora de la usabilidad en arquitecturas de software acorde a estos artefactos.																												
Diseñar un modelo basado en la investigación realizada, que desarrolle técnicas y procedimientos que permitan el incremento de la usabilidad en el sistema.																												
Aplicar el modelo al caso de estudio.																												

Tabla 4. Cronograma de actividades.

Capítulo IV

Formulación del modelo para la optimización del diseño de arquitecturas de software en aplicaciones empresariales con base en la usabilidad en las PYMES consultoras venezolanas

En este capítulo se presenta la formulación de un modelo para la optimización del diseño de arquitecturas de software en aplicaciones empresariales con base en la usabilidad en las PYMES consultoras venezolanas, empleando los pasos propuestos por la metodología SESL, detallada en el capítulo anterior.

4.1 Clasificación de Métodos de Evaluación de la Usabilidad

Para la creación del modelo propuesto por este estudio, se realizó una investigación de los diferentes métodos de evaluación de usabilidad existentes discriminados por autor, los estudios encontrados permiten asegurar que aún no existe un acuerdo para la clasificación de los métodos de evaluación de usabilidad y que los diferentes autores del campo, han creado clasificaciones propias de métodos para la evaluación de la usabilidad, aunque en algunas categorías existe coincidencia y solapamiento entre otras, además de que ninguna está orientada a la realización de evaluaciones en etapas tempranas del desarrollo del software.

Medlock, M.C., Wixon, D., McGee, M., & Welsh, D. (2005): la metodología RITE (Rapid Iterative Test Evaluation) la cual tiene su origen en Microsoft Game Studio (2002) con Michael Medlock y Daniel Wixon. Se utilizó para identificar errores y soluciones eficaces para el juego de manera rápida y sin que implicara un alto coste económico. El RITE testing es un test de usabilidad en laboratorio pero enfocado en agilizar la toma de decisiones respecto a los problemas encontrados, con especial foco en la definición y validación de soluciones. Metodológicamente es muy parecido a un test de usabilidad en laboratorio: sesiones con un número reducido de usuarios que permita detectar los principales problemas. A partir de aquí sin embargo el proceso cambia. El RITE testing está enfocado en encontrar y validar soluciones muy rápidamente, así que parte del proceso es

sustituir los informes por rápidas reuniones, tras cada X usuarios testeados (un mínimo que le de relevancia y contexto a los problemas), con todos los actores implicados en el desarrollo del producto. En estas rápidas reuniones, una vez identificado el problema(s), se busca identificar una solución y se realizan cambios rápidamente que puedan ser puestos a testar con otro grupo de usuarios, y así de manera iterativa. Cada iteración permite validar nuevas soluciones además de seguir encontrando problemas, que pueden tener o no relación con la solución propuesta. Dado que la agilidad es la premisa principal de esta metodología, este tipo de reuniones, decisiones y cambios deben producirse de manera paralela al desarrollo del test.

Whitefield, Wilson y Dowell (1991): proponen una clasificación de métodos de evaluación de usabilidad, subdividiendo los recursos que están disponibles durante la evaluación en cuatro categorías. Usuarios reales, sistemas reales, usuarios representativos y sistemas representativos, y ubican estas categorías en una matriz de dos por dos, para poder clasificarlos como: Observacional, reporte de especialistas, reportes de usuarios y métodos analíticos, de tal manera que satisfagan la matriz, tal como se muestra en la Figura 6.

- Método observacional: puede ser usado sólo con usuarios y computadores reales. En este caso los recursos computadora y usuarios están realmente disponibles, así que un número de métodos de evaluación pueden y deben ser usados.
- Reporte de usuarios: puede ser obtenido sólo cuando están disponibles usuarios reales y computadoras representativas.
- Reporte de especialistas: obtenido de usuarios representativos, quienes por definición son "descripciones o modelos de usuarios".
- Método analítico: preparado sólo cuando algunas de las técnicas no encajan fácilmente en una clasificación debido a que su uso puede ser considerablemente variable.

Obeso (2005) que aplico la investigación empírica basado en la investigación de Nielsen y Molich (1990): dividen los métodos de evaluación en cuatro categorías:

- Evaluación automática: procedimientos computarizados para la evaluación.

- Evaluación empírica: experimentación con usuarios. La mayoría de situaciones prácticas conducen a evaluaciones empíricas por falta de tiempo, especialización, inclinación, o simplemente tradición para hacerlo.
- Evaluación heurística: dado un conjunto de reglas o heurísticas, un experto en usabilidad valora la interfaz y genera un informe de acuerdo a su propia opinión.
- Evaluación formal: análisis de la interfaz desde criterios técnicos.

Figura 6. Clasificación de métodos de evaluación de usabilidad según Whitefield, Wilson y Dowell (1991).

Preece (1993): considera cuatro métodos para la evaluación de usabilidad:

- Evaluación de expertos o evaluación heurística: por lo general es llevada a cabo por personas experimentadas en diseño de interfaces y/o en la investigación de factores humanos, a los cuales se solicita describir los problemas potenciales que ellos consideran para usuarios menos experimentados, proponiendo soluciones a los problemas identificados. Estos expertos no deberán estar implicados con versiones previas del prototipo bajo evaluación y su rol estará definido previamente para asegurar que ellos adopten la apropiada perspectiva cuando usamos el prototipo.
- Evaluación observacional: tolera la colección de datos que proveen información acerca de aquello que los usuarios están haciendo cuando interactúan con el software. Pueden ser usadas algunas colecciones de datos técnicos. De acuerdo a Preece pueden obtenerse dos categorías de datos: cómo capturan los usuarios las

tareas dadas (dónde están las mayores dificultades y que puede hacerse) y medidas de desempeño (tales como frecuencia de completar la tarea correcta, cronometrado de la tarea, frecuencia de errores de los participantes, etc.).

- Evaluación por investigación: se utiliza para conocer la opinión de los usuarios o para deducir sus preferencias sobre un producto potencial o existente por medio de cuestionarios y entrevistas.
- Evaluación experimental: en este caso un evaluador puede manipular un número de factores asociados con la interfaz de usuario y estudiar sus efectos en el desempeño del mismo. Es necesario planear muy cuidadosamente el nivel de experiencia requerido del usuario, hipótesis a ser probada.

Hix y Hartson (1993): describen dos clases de evaluación:

- Evaluación analítica: basada en análisis de las características de un diseño, a través de la revisión de la representación de un diseño, prototipo, o implementación.
- Evaluación empírica: basada en observación del desempeño del usuario del diseño en uso.

Wixon y Wilson (1997): Ofrecen una visión desde la cual el usuario en el centro del proceso de desarrollo. Proponen la siguiente clasificación:

- Evaluación Formativa vs. Sumativa: los métodos de evaluación formativa son usados para generar nuevas ideas durante el desarrollo, en tanto que los métodos de evaluación sumativos son usados para evaluar sistemas existentes.
- Método de evaluación de descubrimiento vs Método de decisión: los métodos de descubrimiento o también llamados métodos cualitativos son usados para descubrir cómo trabajan, se comportan o piensan los usuarios y que problemas tienen. Por otro lado los métodos de decisión también llamados cuantitativos, son usados en la selección de un diseño determinado entre algunas alternativas o para escoger elementos de diseño de interfaz.
- Evaluación formalizada vs. Evaluación informal: los primeros utilizan análisis técnico mientras los segundos son más bien de juicio.

- Evaluación con usuarios comprometidos vs. Evaluación con usuarios no comprometidos: Estos métodos se diferencian en el grado de compromiso del usuario en la evaluación, análisis y diseño
- Evaluación completa vs. Evaluación de un componente: los primeros llamados así por que cubren todos los pasos necesarios para completar los esfuerzos de diseño de usabilidad, mientras que los segundos representan sólo una parte de un proceso completo de usabilidad.

Además, la clasificación de los métodos de evaluación de usabilidad puede obedecer a diferentes criterios: el grado de implicación del usuario, escenarios de tarea, el empleo de reglas o por el objetivo de la evaluación, entre otros. En el Anexo 7 se dispone de una clasificación más detallada de algunos métodos de evaluación de la usabilidad propuestos por diferentes autores. Igualmente existe una serie de técnicas auxiliares que pueden ayudar a tomar mejores decisiones proveyendo herramientas y conceptos adaptados al contexto particular de uso; para que partir del estudio de estas experiencias y conocimientos, se puedan generar variantes de los métodos existentes o crear uno completamente nuevo que se adapten a ciertas condiciones particulares.

4.2 Naturaleza Del sistema

Como se explicó en el capítulo III, la metodología SESL indica que es de vital importancia decidir cuál es el sistema a evaluar, a fin de formular una metodología, la presente investigación planteó un método para la optimización de la usabilidad dentro del contexto de las empresas consultoras venezolanas.

Dado que el método se especificó en el dominio particular del proceso de diseño de software, que es donde comienza con la construcción del modelo de la arquitectura software a partir de un conjunto de requisitos funcionales, se describió la naturaleza del dominio donde se ejecuta este proceso, y diseño preliminar obtenido evaluado con respecto a ciertos atributos de calidad, concretamente la usabilidad mediante sus atributos.

El diseño del software es el proceso mediante el cual se traducen los requisitos en una representación del software, que se acerca mucho al código fuente. Desde el punto de vista

de la gestión del proyecto, el diseño del software se realiza en dos etapas: el diseño preliminar y el diseño detallado, el diseño preliminar se centra en la transformación de los requisitos en datos y arquitectura del software y el diseño detallado se ocupa del refinamiento y de la representación arquitectónica que lleva a una estructura de datos refinada y a las representaciones algorítmicas del software.

4.3 Tipo de Evaluación

En la metodología SESL, se define la existencia de cinco tipos de evaluación: evaluación formativa, evaluación de los efectos del sistema, evaluación de los efectos organizacionales y partes psicológicas del sistema, evaluación de conceptos y evaluación para la compra de nuevos productos.

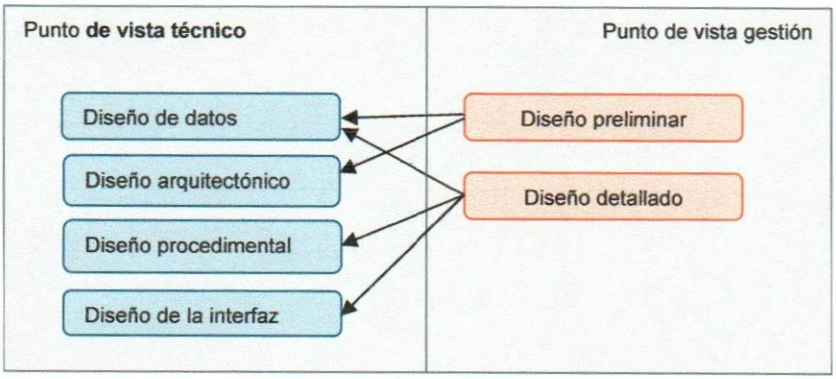


Figura 7. Relación entre el enfoque gestión y el enfoque técnico del proceso de diseño del software.

Al sistema objeto de este estudio, se le realizó la evaluación de los conceptos presentes en la propuesta de un modelo para la optimización del diseño de arquitecturas de software en aplicaciones empresariales con base en la usabilidad en las PYMES consultoras venezolanas. Esta investigación busca validar y relacionar conceptos de las ciencias de la computación (como lo son la arquitectura del software y la usabilidad como un atributo de calidad), específicamente en el área de la ingeniería de software a fin mejorar las técnicas de evaluación existentes en fases tempranas, permitiendo la reutilización de mejores prácticas, y promoviendo la calidad del proceso y el producto de software a través de la revisión del uso y la relación de los diversos patrones de arquitectura.

4.4 Usuarios Afectados *(Stakeholders)*

Luego de definir el tipo de evaluación, se debe averiguar quiénes son los stakeholders y en qué están interesados. Esta tarea se realizó de una manera muy simple, buscando quiénes afecta, dependen de o pueden influenciar al sistema; y en qué forma son afectados o influenciados por el sistema.

En el sistema objeto de estudio, los stakeholders son definidos a través del rol que desempeñan a lo largo del proceso de creación del sistema: Usuarios finales, analistas, Gerencia de TI, Alta Gerencia. Los intereses a considerar por cada uno de estos stakeholders son los siguientes:

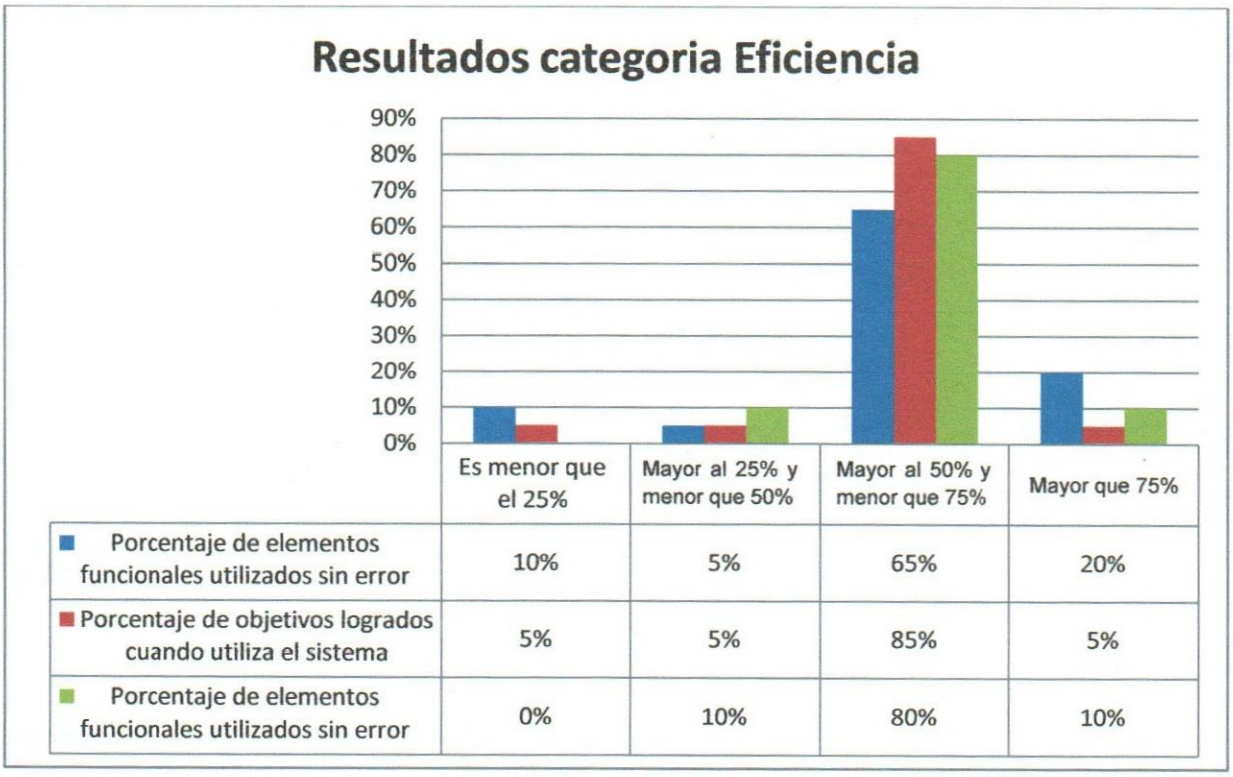
- *Usuarios:* maximizar la eficiencia en el uso del sistema y el tiempo de aprendizaje, facilitar las funciones diarias.
- *Analistas:* minimizar el tiempo de desarrollo, mejorar la calidad del producto, minimizar el número de errores en el sistema, maximizar el nivel de experticia del usuario a través de un enfoque basado en aprendizaje.
- *Gerencia de TI:* minimizar los costos de soporte, maximizar las capacidades de interacción personal (trabajo en equipo), minimizar la cantidad de fallas en el diseño y desarrollo, minimizar los tiempos de desarrollo de sistemas, mejorar la calidad del proceso y del producto.
- *Alta Gerencia:* maximizar el retorno de inversión en las aplicaciones, minimizar los costos de planeación, adquisición, implantación, operación de los sistemas creados.

4.5 Preguntas Claves

Al estudiar los intereses establecidos para los stakeholders, se hizo necesario establecer las preguntas claves en el estudio del sistema a evaluar, mediante la determinación de las áreas principales de estudio y los principales objetivos de cada uno de los afectados.

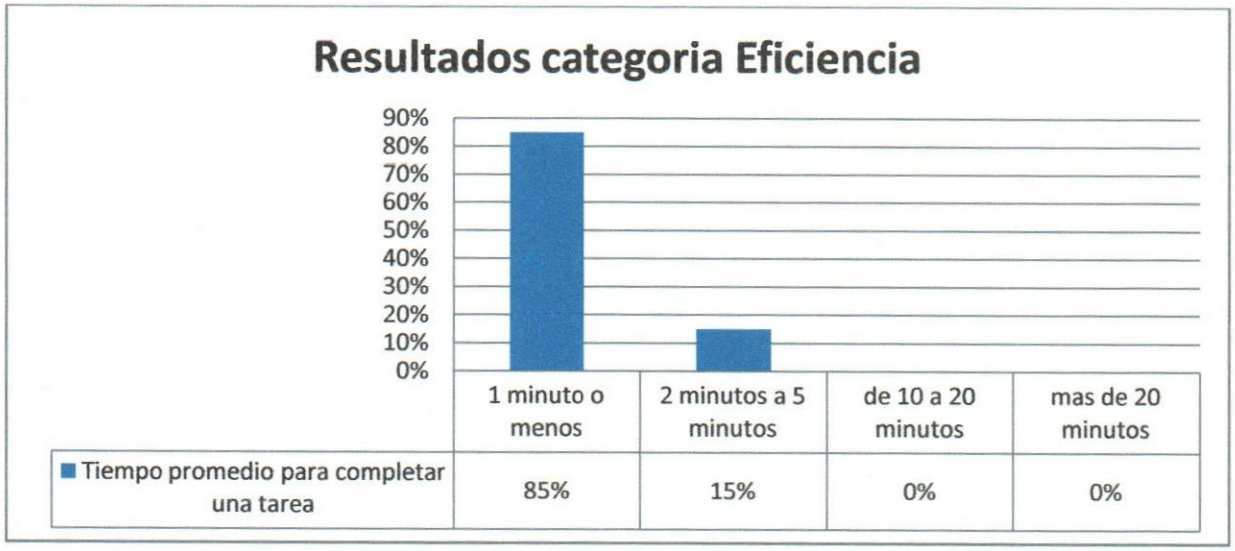
Posteriormente a la construcción de las preguntas claves, se realizó un cuestionario por stakeholder, entrevistas y observación sobre cada uno de ellos, obteniendo así la información necesaria de acuerdo a las técnicas de recolección de datos descritas en el marco metodológico de esta investigación, para mas detalles esta información puede ser

solicitada al investigador. A continuación se presentan los resultados obtenidos en los cuestionarios en las Figuras de la 8 a la 17:



	Es menor que el 25%	Mayor al 25% y menor que 50%	Mayor al 50% y menor que 75%	Mayor que 75%
Porcentaje de elementos funcionales utilizados sin error	10%	5%	65%	20%
Porcentaje de objetivos logrados cuando utiliza el sistema	5%	5%	85%	5%
Porcentaje de elementos funcionales utilizados sin error	0%	10%	80%	10%

Figura 8. Resultados en la categoría eficiencia de los cuestionarios aplicados (preguntas nro. 1, 2 y 3).



	1 minuto o menos	2 minutos a 5 minutos	de 10 a 20 minutos	mas de 20 minutos
Tiempo promedio para completar una tarea	85%	15%	0%	0%

Figura 9. Resultados en la categoría eficiencia de los cuestionarios aplicados (pregunta nro. 4).

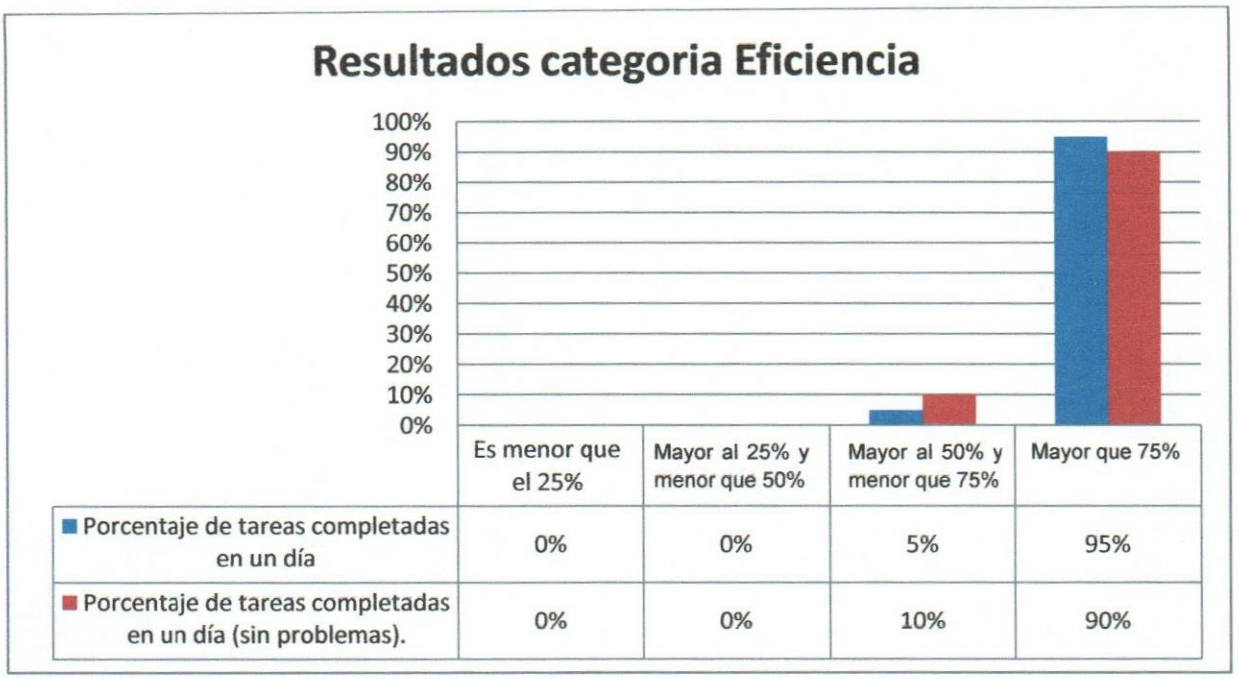


Figura 10. Resultados en la categoría eficiencia de los cuestionarios aplicados (preguntas nro. 5 y 6).

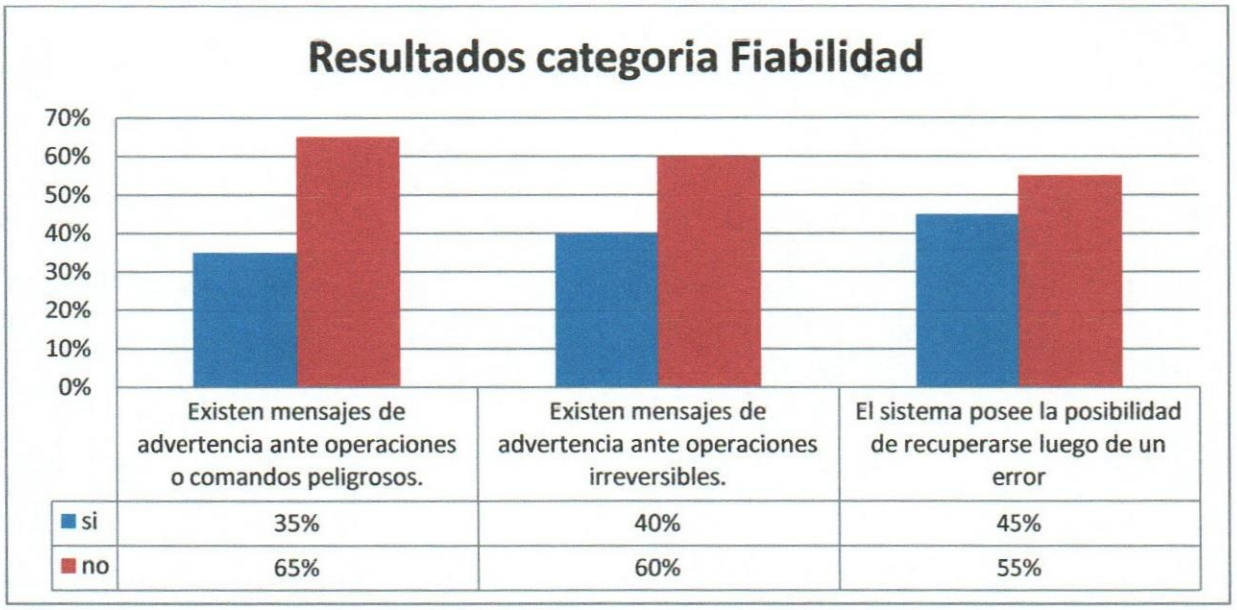


Figura 11. Resultados en la categoría fiabilidad de los cuestionarios aplicados (preguntas nro. 7, 8 y10).

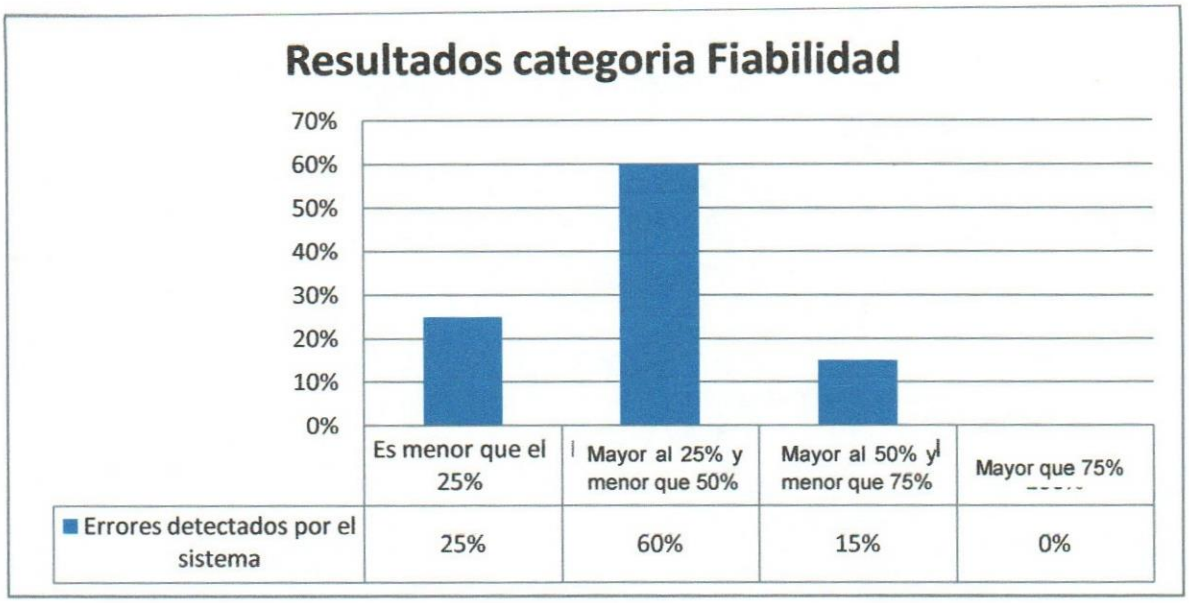


Figura 12. Resultados en la categoría fiabilidad de los cuestionarios aplicados (preguntas nro. 9).

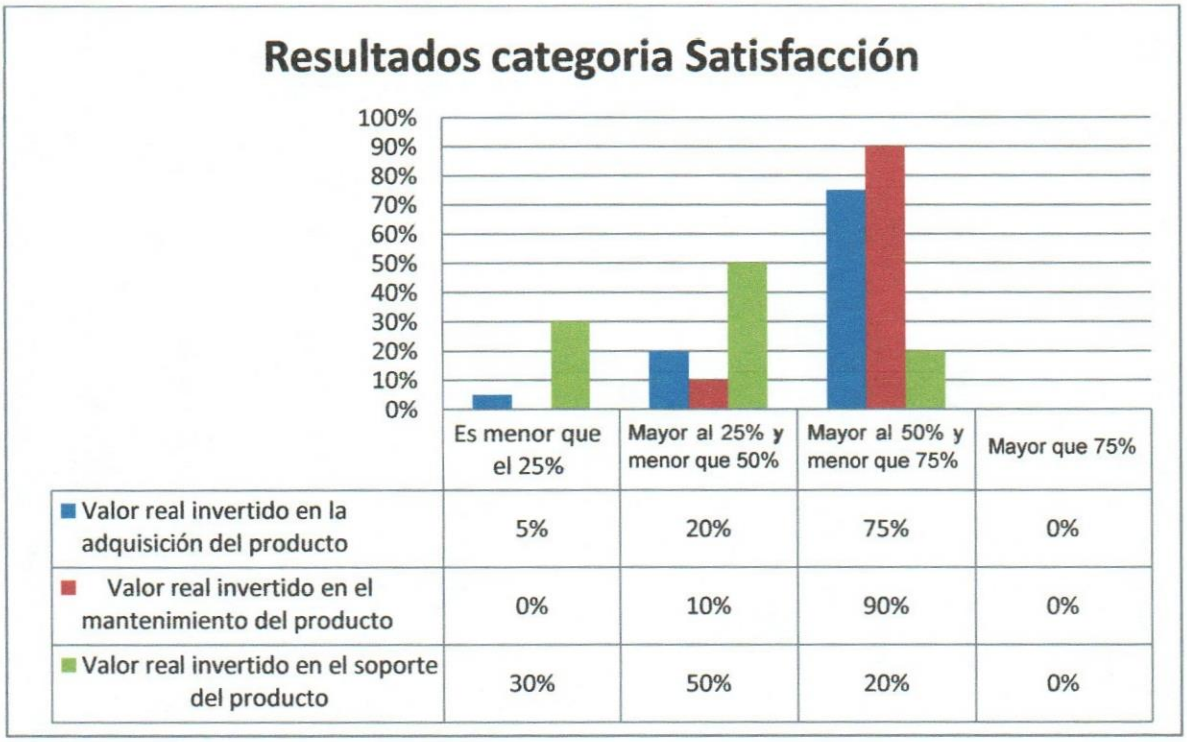
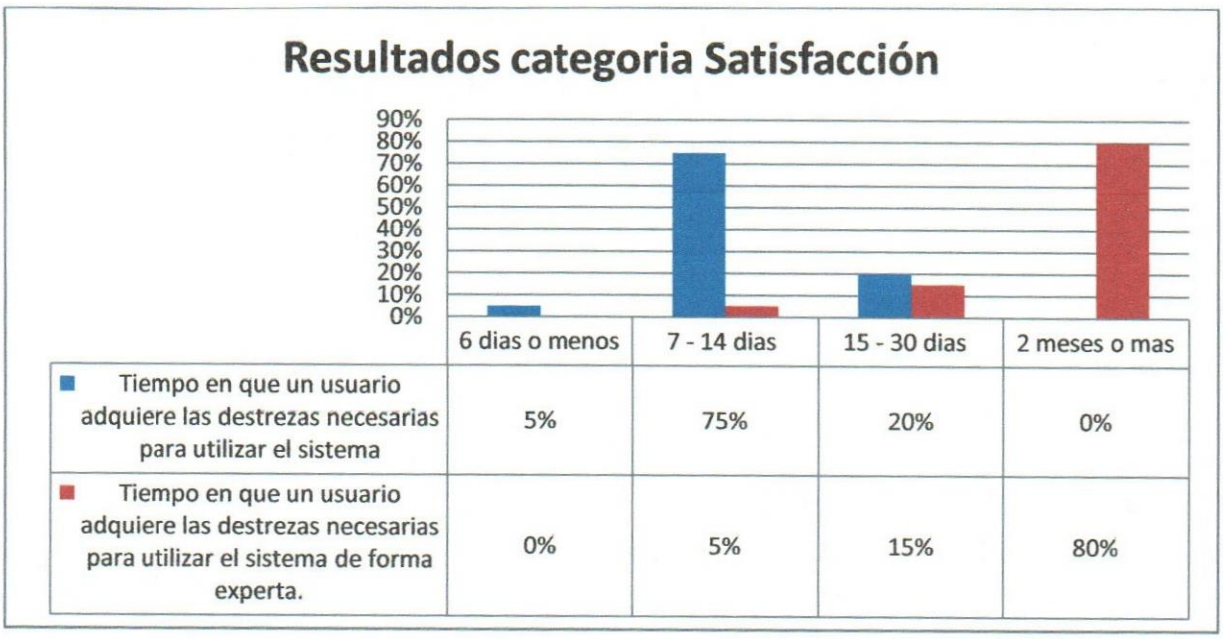
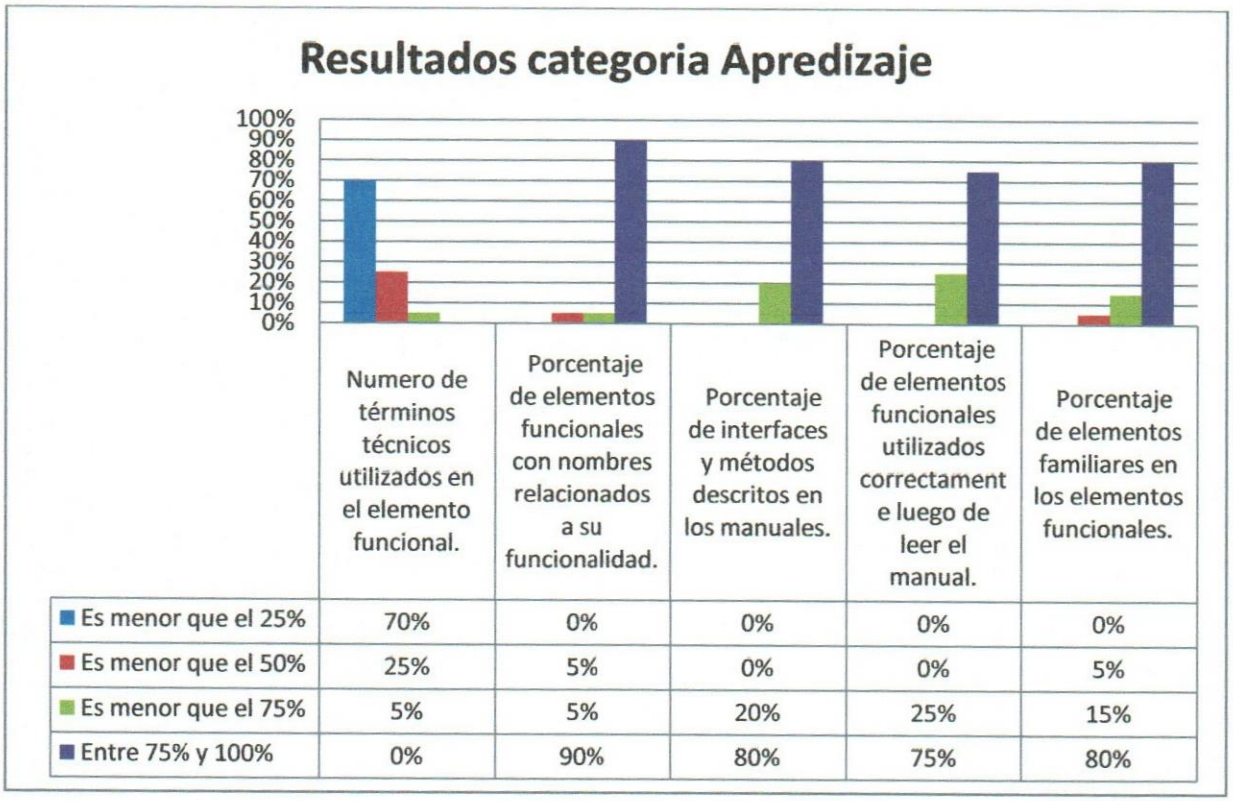


Figura 13. Resultados en la categoría satisfacción de los cuestionarios aplicados (preguntas nro. 13, 14 y 15).



	6 dias o menos	7 - 14 dias	15 - 30 dias	2 meses o mas
Tiempo en que un usuario adquiere las destrezas necesarias para utilizar el sistema	5%	75%	20%	0%
Tiempo en que un usuario adquiere las destrezas necesarias para utilizar el sistema de forma experta.	0%	5%	15%	80%

Figura 14. Resultados en la categoría satisfacción de los cuestionarios aplicados (preguntas nro. 11 y 12).



	Numero de términos técnicos utilizados en el elemento funcional.	Porcentaje de elementos funcionales con nombres relacionados a su funcionalidad.	Porcentaje de interfaces y métodos descritos en los manuales.	Porcentaje de elementos funcionales utilizados correctamente luego de leer el manual.	Porcentaje de elementos familiares en los elementos funcionales.
Es menor que el 25%	70%	0%	0%	0%	0%
Es menor que el 50%	25%	5%	0%	0%	5%
Es menor que el 75%	5%	5%	20%	25%	15%
Entre 75% y 100%	0%	90%	80%	75%	80%

Figura 15. Resultados en la categoría satisfacción de los cuestionarios aplicados (preguntas nro. 15 a la 20).

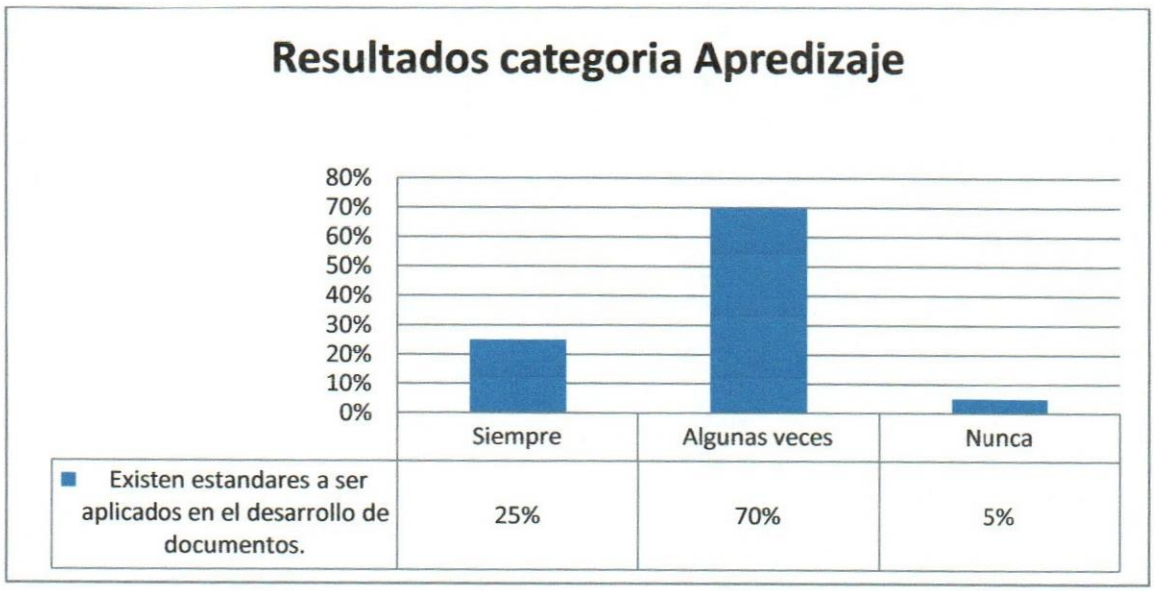


Figura 16. Resultados en la categoría satisfacción de los cuestionarios aplicados (preguntas nro. 21).

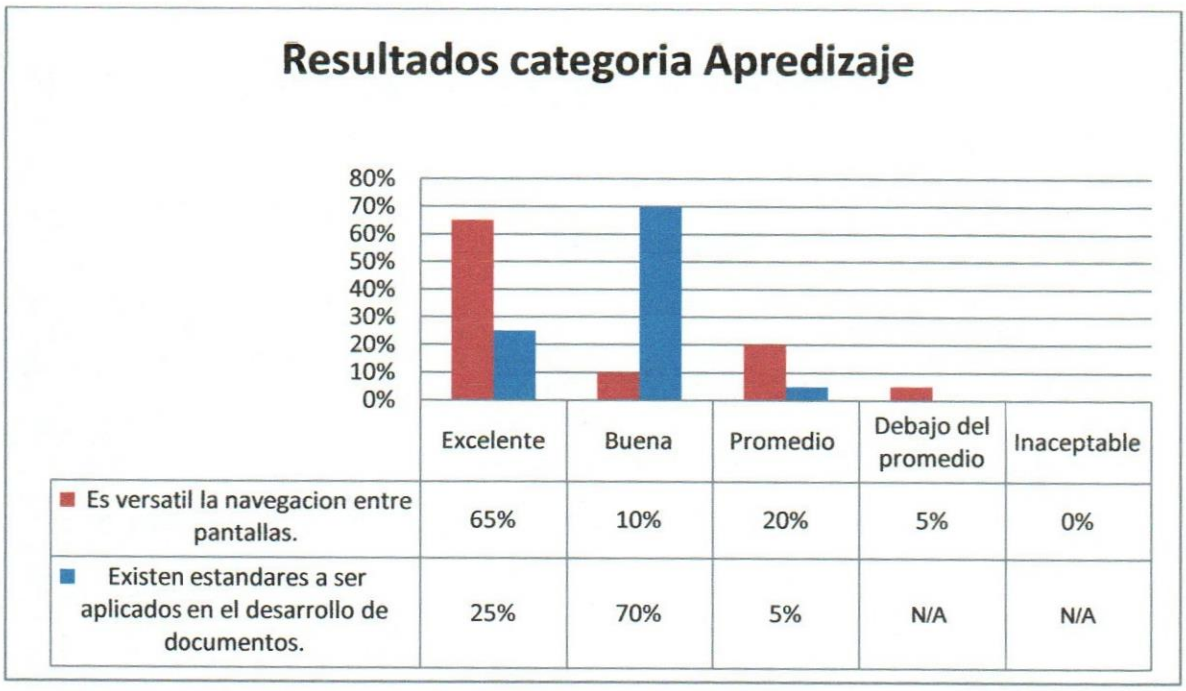


Figura 17. Resultados en la categoría satisfacción de los cuestionarios aplicados (preguntas nro. 22).

A través de los resultados se dedujo que las áreas que presentan más problemas a los usuarios son la satisfacción (en lo relativo al precio) y la fiabilidad (la detección y recuperación de errores), razón por la cual en base a esto se fijo la ponderación dando mayor relevancia a estos puntos, cuando se estableció el peso total de los escenarios en la visión global del sistema, y aconsejando tomar mayores previsiones en los escenarios que presenten sensibilidad relacionada a estos puntos.

4.6 Diseño del modelo para la optimización del diseño de arquitecturas de software en aplicaciones empresariales con base en la usabilidad

Una vez establecidas las preguntas asociadas a los intereses de los stakeholders y obtenidos los valores a través de una herramienta de recopilación de información (cuestionarios y entrevistas), se procedió al diseño del modelo.

El planteamiento principal de este modelo consistió en introducir una nueva fase dentro del proceso de diseño de software concretamente luego del modelado, donde se comienza con la creación de un modelo de la arquitectura software a partir de un conjunto de requisitos funcionales, luego el diseño preliminar obtenido es evaluado con respecto a ciertos atributos de calidad, concretamente la usabilidad. Si el valor de usabilidad obtenido es adecuado con respecto a lo solicitado en los requisitos, se finaliza el proceso y se pasaría a la etapa desarrollo. Si la usabilidad necesita mejorarse, se pasaría a un ciclo de mejora en la arquitectura software y nuevas evaluaciones, hasta que el nivel de usabilidad sea el adecuado.

La usabilidad es evaluada en los sistemas finalizados analizando los siguientes atributos: aprendizaje, eficiencia, fiabilidad y satisfacción. Sin embargo el nivel de abstracción de estos es demasiado alto como para poder estudiar las técnicas que deben aplicarse a la arquitectura para mejorarlos, por lo que se descompusieron cada uno de estos atributos en propiedades especificas, que determinaron que características debían ser mejoradas en el sistema, para identificar la relación entre los atributos y las propiedades se utilizó un procedimiento descrito por Bosh (2002) para el proyecto STATUS.

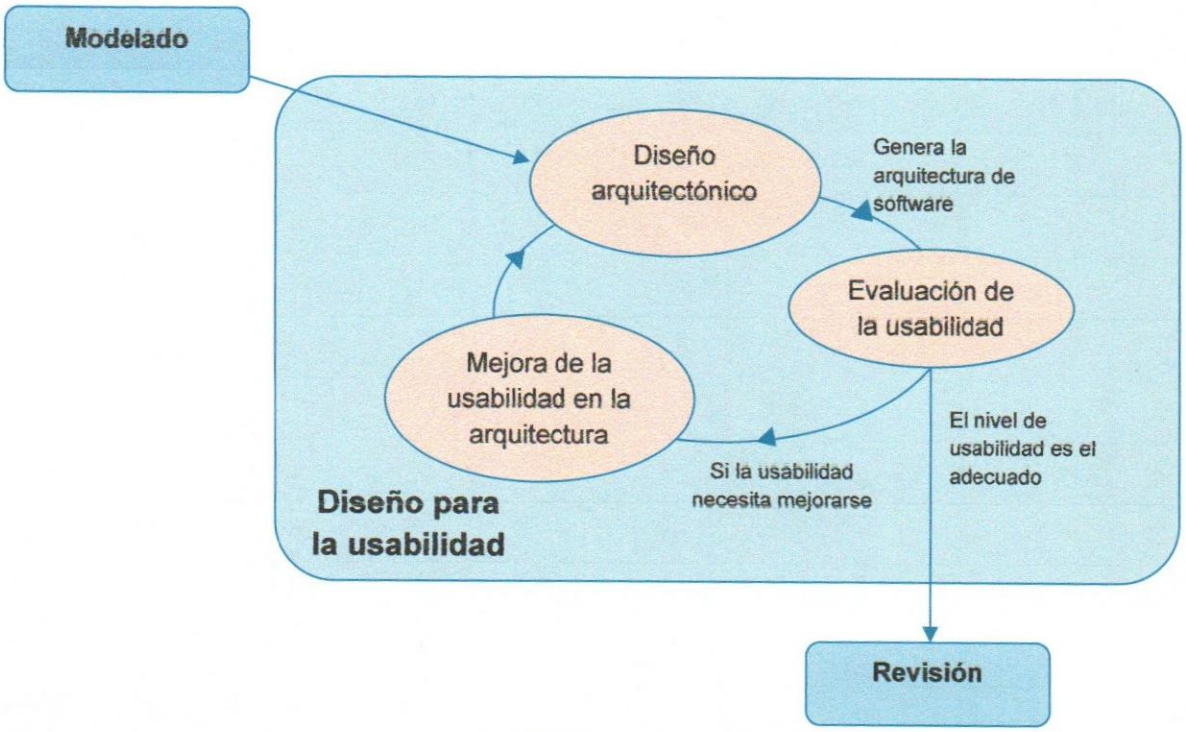


Figura 18. Diseño para la usabilidad.

Una vez descompuestos los atributos, se obtendrá un nivel de visualización más cercano la realidad y se identificaran mecanismos concretos que podrían incorporarse a la arquitectura de software para mejorar la usabilidad del sistema final, que abordan algunas necesidades indicadas por una propiedad de usabilidad (ver Figura 19).

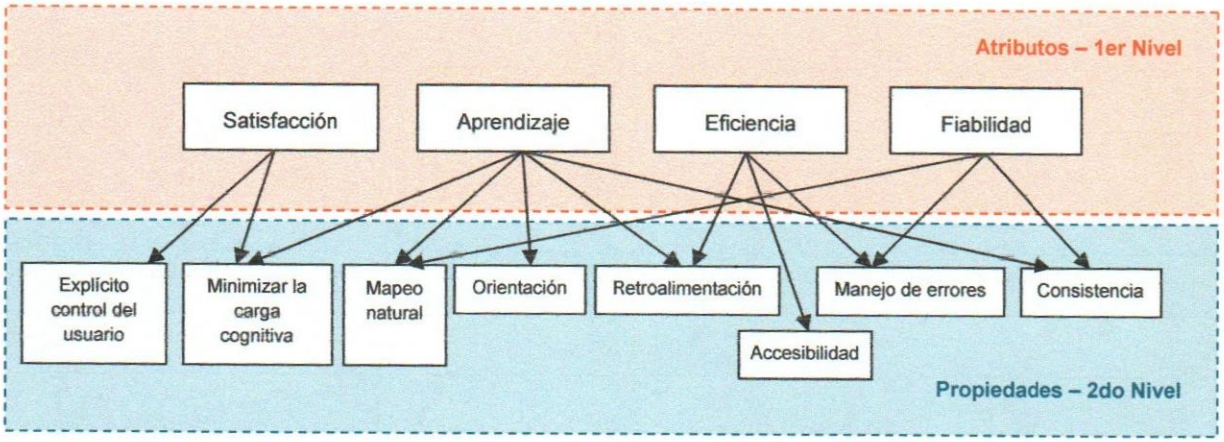


Figura 19. Relaciones entre atributos y propiedades de la usabilidad.

Tras una minuciosa revisión de la literatura relacionada con dicha descomposición a continuación se describen de forma sintetizada:

- *Explícito control del usuario:* El usuario puede controlar en todo momento la situación de sus tareas.
- *Minimizar la carga cognitiva:* El usuario no debe tener que recordar ningún dato previo y con la mínima información que aparece en la página debe ser capaz de manera eficiente/ eficaz de realizar su tarea.
- *Mapeo natural:* Los contenidos han de estar dispuestos generalmente en forma de secuencia de arriba abajo a lo largo de la pantalla.
- *Orientación:* El principal problema de interfaz es el sentido de la orientación dentro de la organización local de la información, el segundo factor es la certidumbre en la navegación, que el usuario sepa dónde puede ir. En resumen, que el usuario sepa en todo momento donde está y a donde puede ir.
- *Retroalimentación:* El usuario debe conocer en todo momento la situación de sus tareas.
- *Accesibilidad:* variedad de dispositivos desde donde se puede acceder a los contenidos. La navegación debe estar adaptada a los diferentes tipos de usuarios.
- *Manejo de errores:* Los mensajes de error han de ser adecuados al usuario.
- *Consistencia:* se considera que un sistema es consistente si todos los mecanismos que se utilizan son siempre usados de la misma manera, siempre que se utilicen y sea cual sea el momento en el que se haga.

El patrón arquitectónico representa el último eslabón en la cadena atributo, propiedad y patrón, conectando la usabilidad del sistema con la arquitectura de éste. Un patrón muestra la relación entre los elementos que trabajan conjuntamente para resolver un problema concreto y estos muestran una solución a un problema. Dado que se buscaba definir un mecanismo para incorporarlo en el diseño de una arquitectura software a fin de abordar una propiedad de usabilidad concreta, el patrón arquitectónico determinó cómo se incorporó y qué efecto tuvo en los componentes del sistema. El procedimiento que se ha seguido para identificar los mecanismos que representan soluciones de diseño para los patrones, está compuesto de la siguiente forma:

1. *Aplicar un proceso de inducción para abstraer los patrones arquitectónicos a partir de los diseños particulares del proyecto (Se construyen el modelo de diseño del sistema):* se describirá la arquitectura del sistema a evaluar por medio de artefactos y diagramas, deben incluirse los componentes de datos y conexiones relevantes, así como la descripción del comportamiento general del sistema.
2. *Se analizan las situaciones sensibles presentadas en el patrón con respecto a la usabilidad, y se representan los escenarios:* Determinan una serie de momentos de interacción o tareas del usuario que, para que el sistema los pueda soportar, tienen que estar definidos en la arquitectura del software (se tomara como base el trabajo de Bass y de su equipo, los cuales inventariaron una serie de escenarios de usabilidad sensibles a la arquitectura del software). Este inventario está compuesto de 26 escenarios que se pueden consultar en el documento CMU/SEI-2001-TR-005 de Bass y otros. Se identificaran cuales de los escenarios existentes en la lista están presentes en el sistema a evaluar y procederán a representarlos según el formato diseñado por el investigador (ver Tabla 3 ejemplificada en el caso de estudio expuesto en el capitulo V).
3. *Clasificación de los escenarios:* en este paso se determinara cuales de los escenarios sensibles pueden satisfacer las necesidades de usabilidad sin modificación cuáles no.
4. *Evaluación Individual de los escenarios:* sólo valido para aquellos escenarios que en el paso anterior hayan demostrado no poder satisfacer las necesidades de usabilidad del sistema sin que se realice alguna modificación. Para cada uno de estos se procederá a rellenar la plantilla correspondiente a la Figura 20, el primer renglón contiene los datos de identificación del caso donde se encuentra el campo "identificador" o id del escenario según la nomenclatura tomada (pueden usarse las iníciales del proyecto seguidas del numero del caso), "escenario" corresponde al nombre del escenario y una breve explicación del mismo, "identificación según Bass" clasifica cuál de los 26 tipos escenarios inventariados por Bass corresponde al caso estudiado (debido a que los escenarios identificados por Bass abarcan una gran variedad de situaciones comúnmente vistas en el área, lo que hace mas practico localizar la clasificación correspondiente en su catalogo, pero de no encontrarse

ninguna semejante al escenario estudiado entre estas entonces sólo deberá dársele un nombre y colocar una breve explicación de funcionalidad del mismo haciéndose la acotación de que este no es un caso inventariado, aunque sólo se recomienda utilizar esta opción como última instancia), "contexto" explica el problema u oportunidad presentado dentro de la funcionalidad que realiza el caso, "propiedades afectadas" enumera las propiedades de la usabilidad afectadas por este escenario dentro del dominio de las mostradas en el segundo nivel de la Figura 19, el segundo renglón corresponde a las "decisiones arquitectónicas", esta es sección de la plantilla donde se plantearan las posibles soluciones al problema ideadas por el equipo de diseño y desarrollo, junto a la columna "decisiones" se encuentran cuatro (4) columnas mas con las letras S (sensibilidad), R (riesgo), M (en caso que aporte alguna mejora), las cuales sólo deben rellenarse si la decisión planteada en esa fila afecta a alguna de las propiedades de los demás escenarios en alguna de estas tres formas, en cuyo caso debe colocarse en esa casilla el identificador del caso afectado seguido de un guion (-) y la abreviación de la propiedad afectada (las abreviaciones se encuentran en la parte inferior derecha de la plantilla), este punto en particular ayudara en la realización de la valoración cualitativa. Luego le siguen el renglón correspondiente a "problemas" y el "peso global o ponderación global del escenario" correspondientes a la valoración cuantitativa. Como se menciono anteriormente, la evaluación de cada factor en estos escenarios implica una valoración cuantitativa y una cualitativa, explicadas a continuación:

4.1. *Valoración cuantitativa:* corresponde a la identificación y descripción de problemas encontrados (correspondiente al renglón de problemas en la Figura 19), asignándole además un valor dependiendo de su frecuencia e impacto, para indicar si es un problema de forma o si es un problema de usabilidad (ponderado como menor o mayor). La integración de los resultados se realiza analizando los valores tanto del resultado de la ponderación o peso global del escenario (la ponderación será dada por el promedio de las puntuaciones que el equipo le asigne al caso, representada por un numero del 1 al 10, tomando en consideración sumarle 2 puntos en el caso de que el escenario involucre un problema de fiabilidad y contemplando que la puntuación dada a al renglón de

problemas se encontrara representada al lado de la ponderación entre paréntesis con un signo más "+" antepuesto).

4.2. *Valoración cualitativa:* se realiza a través de una matriz o tabla de valoración N x N, siendo N el numero de escenarios disponibles representados por medio de su identificador, colocados en la parte superior y a la derecha de la matriz como títulos, la relación entre estos estará dada por la intersección de las coordenadas indicadas, por ejemplo el contenido de M[4,5] corresponderá las decisiones arquitectónicas sugeridas (correspondiente al renglón llamado "decisiones arquitectónicas" de la plantilla en la Figura 20) tomadas en el caso 4 que afectan al caso 5, también se tomara de esta plantilla el tipo de relación: S para sensibilidad, R para riesgo, M en caso que aporte alguna mejora, se denotara colocándolo entre paréntesis junto a la propiedad afectada. De esta forma al terminar el análisis se podrá generar una matriz que represente la relación entre los casos, dando una visión global.

5. *Se obtiene una evaluación global del sistema:* acorde a los resultados arrojados por cada escenario evaluado por separado, debe asignársele un peso a cada escenario, en términos de su importancia relativa al éxito del sistema, para generar una evaluación global de la arquitectura.
6. *Se generan alternativas que prevengan, disminuyan o eliminen estas situaciones en el patrón presentado*: se generan consideraciones, advertencias o modificaciones en el modelo arquitectónico, basada en el análisis de la evaluación global.
7. *Se modifica el diseño para incluir la funcionalidad correspondiente a los ajustes considerados:* se introducen los cambios y consideraciones al modelo arquitectónico.

Dado que el objetivo de esta evaluación es saber si pueden habilitar los requerimientos, atributos de usabilidad y restricciones para asegurar que el sistema a ser construido cumple con las necesidades de los stakeholders estos ocho pasos pueden ser repetidos iterativamente hasta que se considere que el nivel de usabilidad es adecuado y cumple con las expectativas necesarias para el éxito del sistema como se muestra en la Figura 18.

Como recomendación general, para sistemas de mayor tamaño se sugiere, a fin de hacer mas legible la matriz resultante, implementar una de las siguientes opciones: la primera contemplar en la matriz sólo aquellos casos que se relacionen o afecten a otros o crear una matriz general donde sólo se especifique cuales casos se afectan entre sí marcándolo con una exis (X) y luego exista un documento de detalle donde se explique a que atributos afecta y en qué forma. la segunda opción mantendría en cierto porcentaje el poder obtener una visión global de la relación de los escenarios, pero a un nivel básico. Esta recomendación podría ser de utilidad para sistemas que requieran el estudio de más de veinte (20) escenarios, donde una matriz que contiene texto se hace menos legible.

Identificación		Referencia	
Escenario			
Identificación según Bass			
Contexto			
Propiedades Afectadas			

Decisiones Arquitectónicas				
Decisiones	S	R	M	N/A

Problemas	Ponderación

(valor entre el 1-10)

Peso o Ponderación Global del escenario	

(valor entre el 1-10)

		Leyenda
Explícito control del usuario = EU	Retroalimentación = RE	Sensibilidad = S
Minimizar la carga cognitiva = MC	Accesibilidad = AC	Riesgo = R
Mapeo Natural = MN	Manejo de errores = ME	Mejora = M
Orientación = OR	Consistencia = CO	No Afecta = N/A

Figura 20. Plantilla para la realización de escenarios.

Capítulo V

Caso de Estudio

En este capítulo se presenta la aplicación del método a un caso de estudio real, utilizando como dominio el proyecto Resort Around The World de la compañía Parodi Solutions C.A, un sistema orientado a la gestión de alojamientos turísticos (alquiler, compra y venta). Se decidió tomar como ejemplo un sistema de pequeña escala por factores de tiempo. Ya que el modelo requiere un análisis elaborado de los escenarios. A continuación se procederá a realizar la aplicación de los pasos enumerados, que conforman el modelo mostrado en el capitulo anterior.

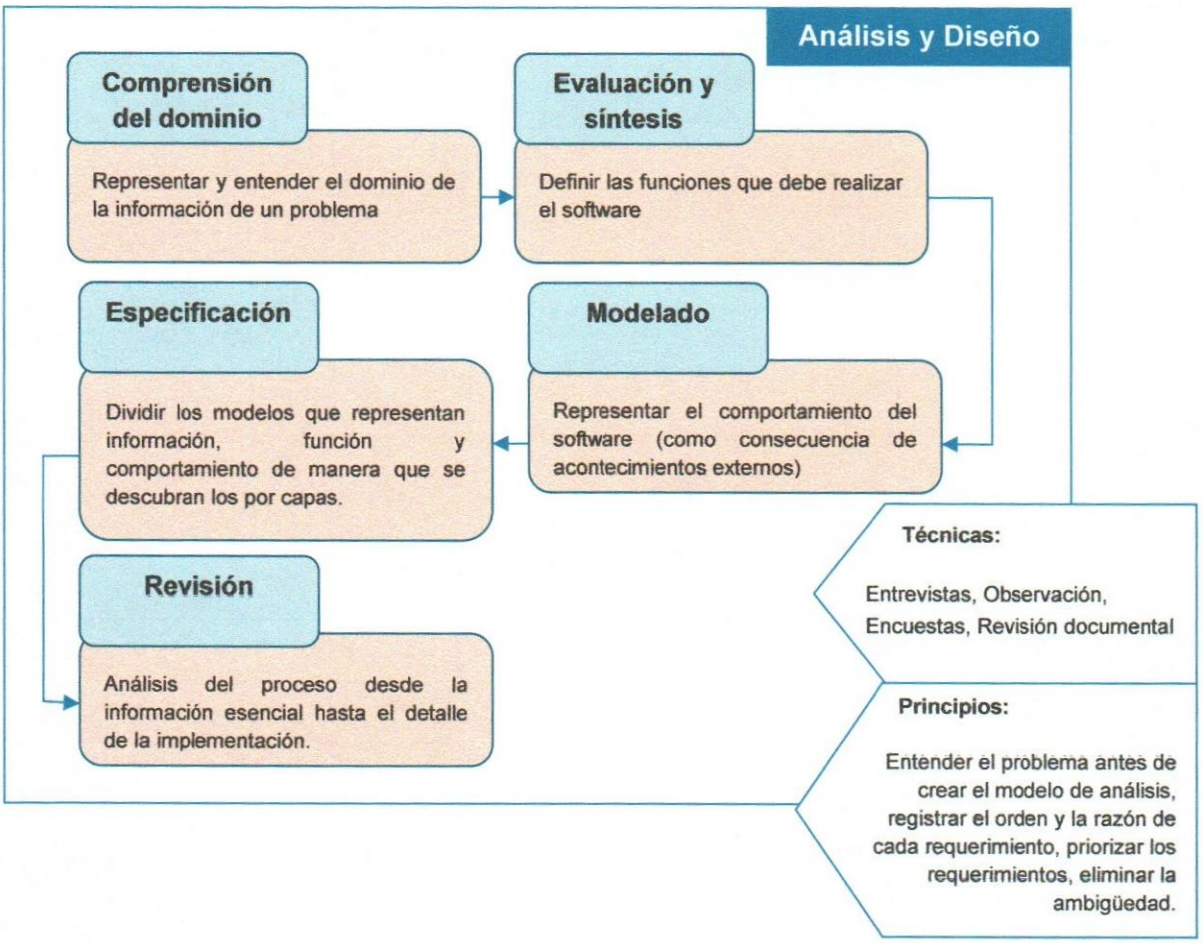


Figura 21. Proceso de Análisis y Diseño de Parodi Solutions.

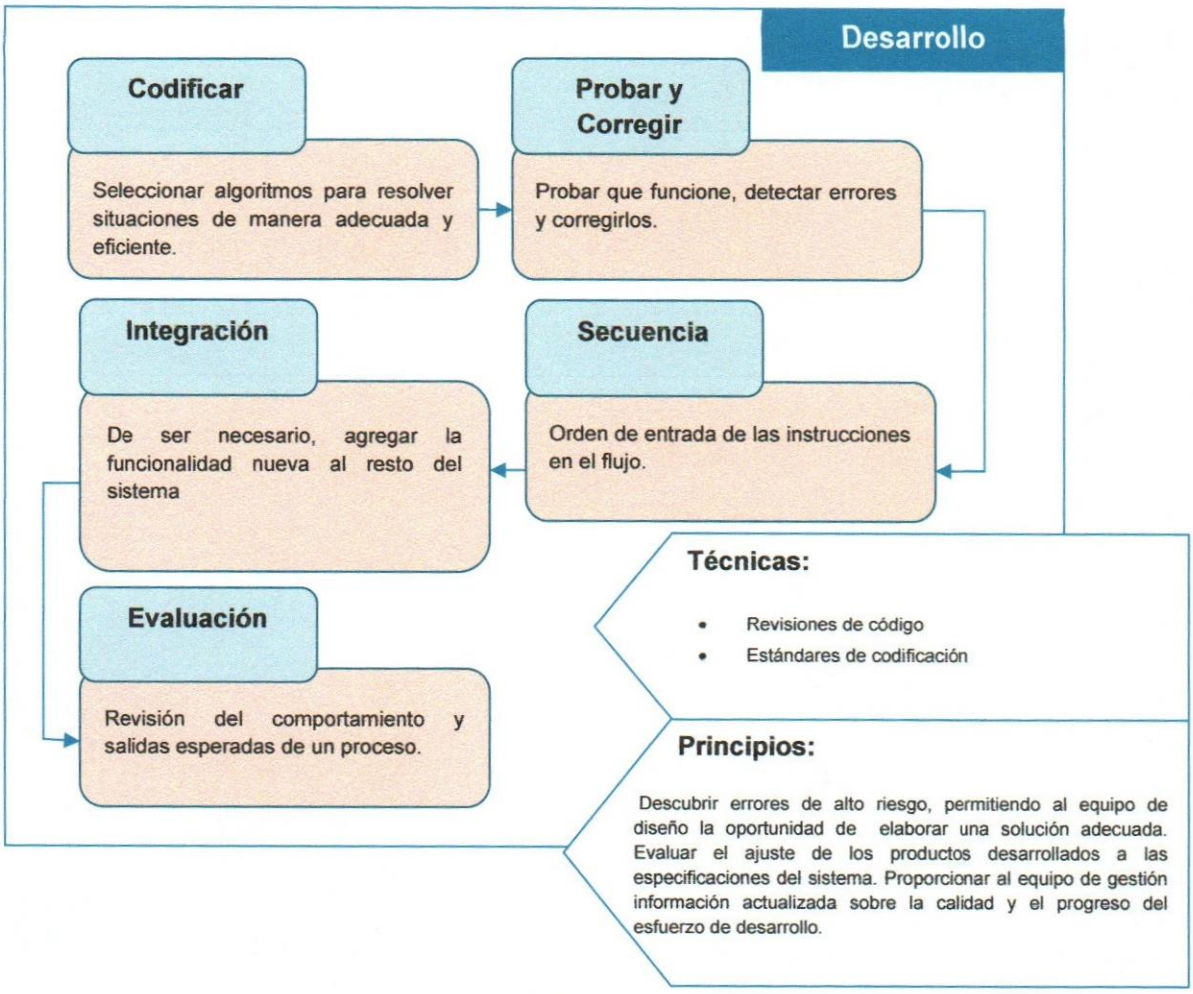


Figura 22. Proceso de Desarrollo de Parodi Solutions.

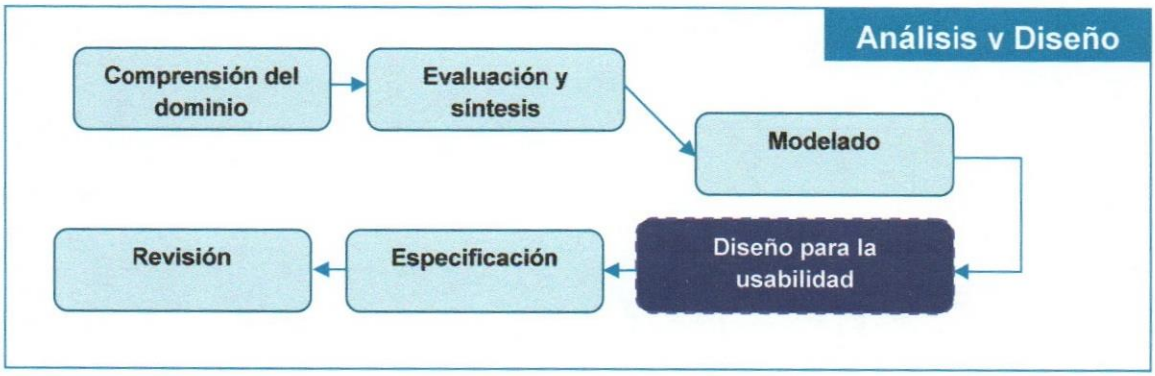


Figura 23. Proceso de Análisis y diseño incluyendo la fase sugerida por el modelo de evaluación.

Dado que la metodología se centró en un caso de estudio representado por los procesos de análisis y desarrollo de software dentro de la empresa Parodi Solutions, a continuación se describirá la naturaleza del dominio donde se ejecutan: el proceso de análisis y diseño de requerimientos (ver Figura 21), que permite especificar y evaluar las características operacionales del software (funciones, datos y rendimientos), indica la interfaz del software, así como otros elementos del sistema y establece las restricciones que debe cumplirse y el proceso de desarrollo (ver Figura 22) que permite generar soluciones acorde a los requerimientos presentados.

5.1 Aplicar un proceso de inducción para abstraer los patrones arquitectónicos a partir de los diseños particulares del proyecto (Se construyen el modelo de diseño del sistema).

El modelo utilizado por la compañía para llevar a cabo este proyecto, es el patrón Modelo, Vista, Controlador o MVC el cual busca agrupar los componentes de la aplicación en tres niveles lógicos (ver Figura 24), y para la ejecución del mismo se utilizó el framework Symfony, diseñado para optimizar el desarrollo de las aplicaciones web basado en este mismo patrón. Este framework separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web, además proporciona varias herramientas y clases que ayudan reducir el tiempo de desarrollo y promueven estándares de codificación. También, automatiza las tareas más comunes permitiendo disminuir su tiempo de implementación.

El sistema se encuentra integrado por los siguientes componentes:

- La capa del Modelo
    - Abstracción de la base de datos
    - Acceso a los datos
- La capa de la Vista
    - Vista
    - Plantilla
    - Layout
- La capa del Controlador
    - Controlador frontal
    - Acción

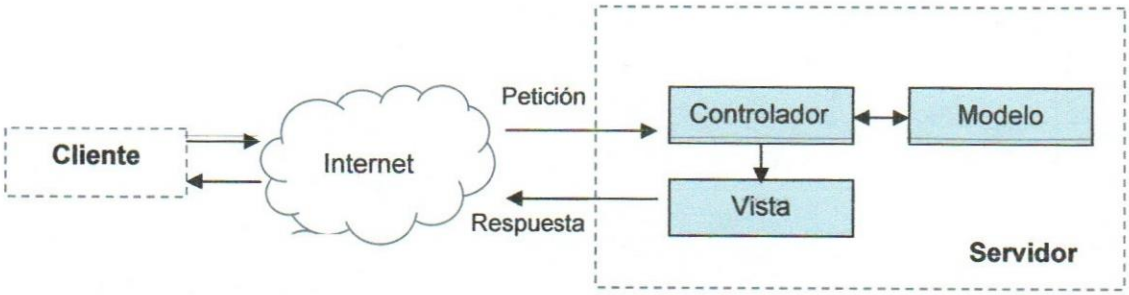


Figura 24. Visualización Patrón Modelo-Vista-Controlador en sitios Web.

En primer lugar, el controlador frontal y el layout son comunes para todas las acciones de la aplicación. Se pueden tener varios controladores y varios layouts, pero solamente es obligatorio tener uno de cada. El controlador frontal es un componente que sólo tiene código relativo al MVC, por lo que no es necesario crear uno, ya que Symfony lo genera de forma automática. Las clases de la capa del modelo son generadas automáticamente por el framework, en función de la estructura de datos de la aplicación.

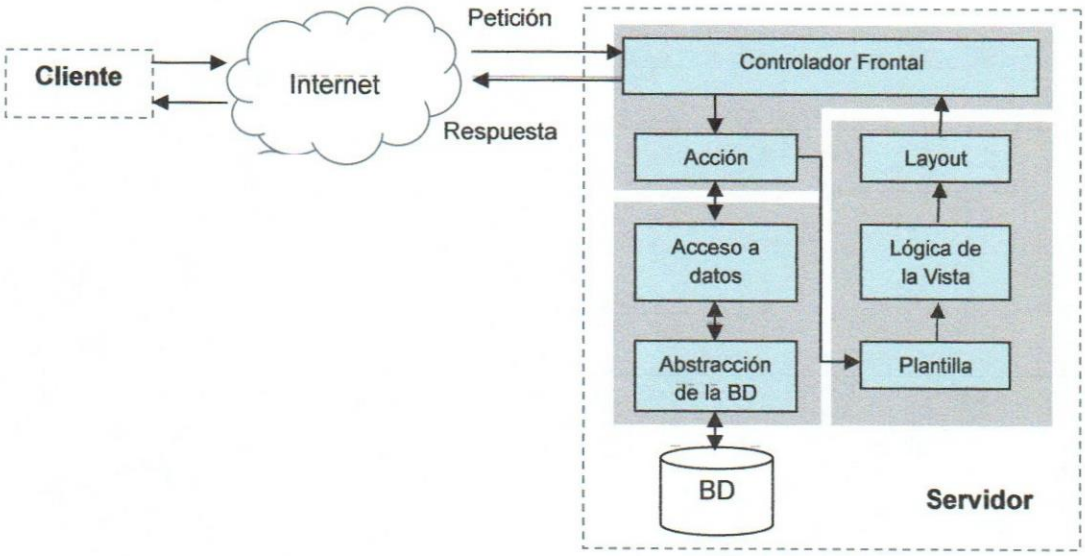


Figura 25. Flujo de Trabajo del framework.

Para la descripción del modelo de diseño, también se dispuso de la siguiente información: diagrama de arquitectura de grupos (ver Figura 26), el Documento de Arquitectura de Software (DAS) (ver Anexo 4), suministrada por el equipo de desarrollo de la compañía en cuestión.

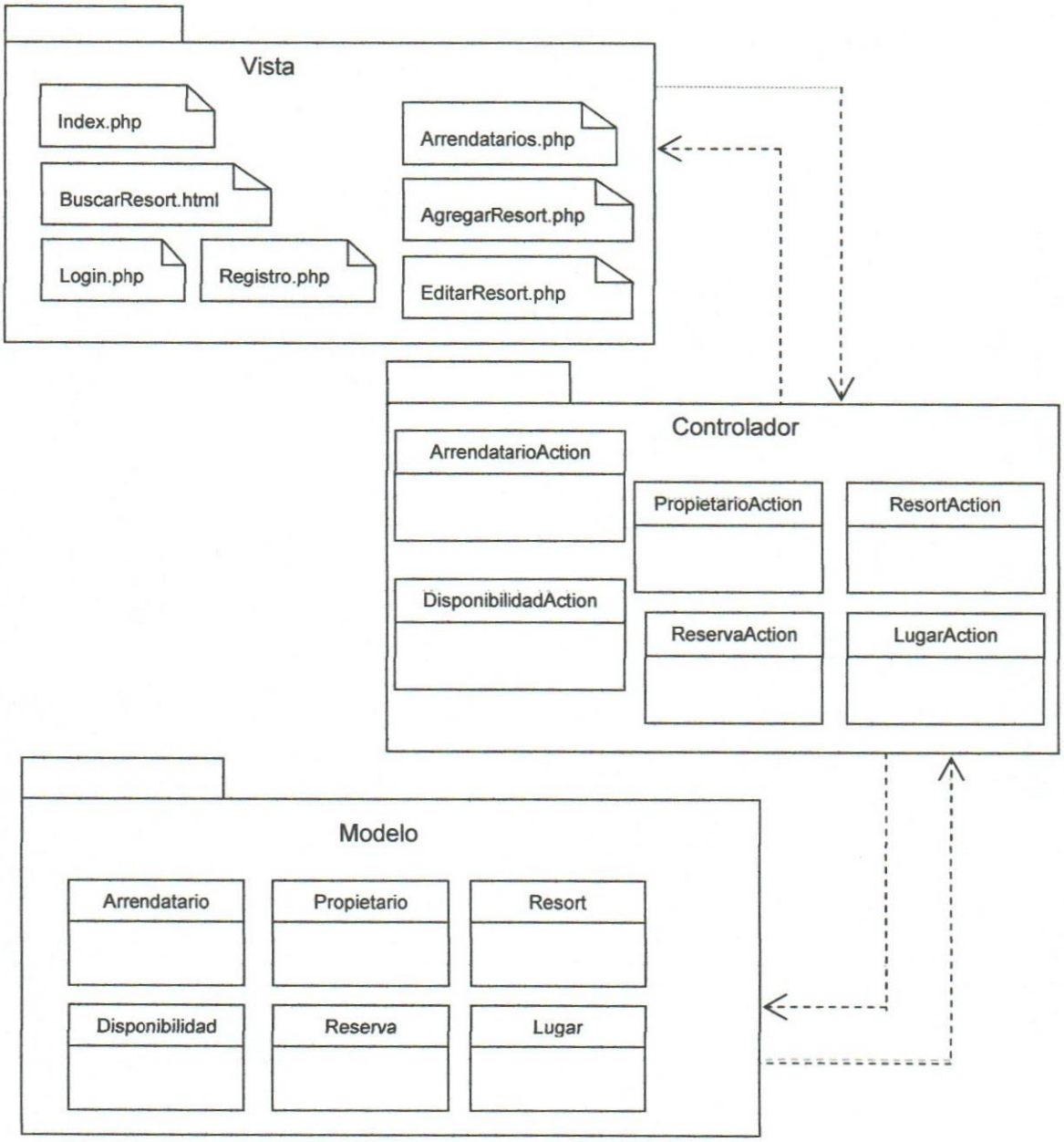


Figura 26. Diagrama de arquitectura de grupos del Caso de estudio Resort Around The World

Fuente: creación propia

5.2 Se analizan las situaciones sensibles presentadas en el patrón con respecto a la usabilidad, y se representan los escenarios.

Mediante el análisis de la información presentada en el paso anterior y con base en la serie de escenarios de usabilidad sensibles a la arquitectura de software descritos por Bass, se identificaron las situaciones sensibles halladas en el diseño del proyecto, las cuales fueron descritas, discutidas y estudiadas, para ser posteriormente presentadas en el siguiente paso, debidamente clasificadas.

5.3 Clasificación de los escenarios.

Los escenarios identificados durante la investigación se presentan a continuación (ver Tabla 4), clasificados según cuales necesitan modificación y cuáles pueden cubrir las necesidades sin serlo.

5.4 Evaluación Individual de los escenarios.

Y los escenarios resultantes se evaluaron y documentaron a través de las plantillas indicadas por el método (ver Anexo 5).

5.5 Se obtiene una evaluación global del sistema.

Acorde a los resultados arrojados por cada escenario evaluado por separado, el total del puntaje del sistema fue 28 (+14), obtenido a través de la suma de la ponderación individual de cada escenario y de los problemas por separado (ver Tabla 6), luego se generó una matriz para la visualización de las repercusiones de las decisiones arquitecturales de cada uno de los escenarios tendrán sobre los demás (ver Tabla 5), como se puede observar los casos RATWCE05 y RATWCE06, son tomados en cuenta sólo como posibles afectados, ya que al no requerir modificación no afectaran a los demás casos de sensibilidad, esta matriz sólo muestra la relación entre los casos estudiados, no si el efecto es positivo o negativo, para profundizar en este aspecto ver Anexo 5.

Identificador	Escenario	Identificación según Bass	Contexto	Requiere modificación
RATWCE01	Agregar Disponibilidad (consiste en agregar una nueva fecha de posible estadía válida para un resort por parte de su propietario)	Agregación de datos (Aggregating Data)	El usuario desea agregar una nueva fecha de posible estadía en uno de los resort de su propiedad una vez que ha agregado este al sistema, esta semana de estadía servirá para ser reservada por clientes potenciales y aprobada por el administrador. Dado que el propietario puede querer agregar varias semanas, es decir realizar una o más acciones en más de un objeto (resort), o también cuantas se encuentran asociadas a este y su estatus (reservada o libre).	si
RATWCE02	Agregar Nuevo Resort	Comprobación de la corrección (Checking for Correctness)	Al agregar un nuevo objeto resort se solicitan algunos datos que quizás puedan encontrarse almacenados previamente en la bd, ya que un resort puede repetirse si dos usuarios poseen acciones en el mismo y lo registran por separado.	si
RATWCE03	Autentificación	Recuperación de contraseñas olvidadas (Retrieving Forgotten Passwords)	Al entrar al sistema el usuario debe llevar a cabo la autentificación en caso de estar registrado, pero si no recuerda alguno de los datos (usuario o login) no podrá entrar al sistema. En este momento no existe un mecanismo para la recuperación de datos olvidados.	si
RATWCE04	Registro en el sistema	Proporcionar una buena ayuda (Providing Good Help)	Al existir dos posibles perfiles de usuarios con registro en el sistema (arrendatario y propietario) .El usuario debe encontrar que los procedimientos de ayuda del sistema se adaptan al contexto.	si
RATWCE05	Aprobar Solicitudes de Reserva	Agregación de datos (Aggregating Data)	Cuando el administrador desea aprobar las reservas existentes, deberá realizar esta tarea individualmente para cada objeto lo cual puede resultar tedioso	no
RATWCE06	Aprobar Solicitudes de Reserva	Información Reutilizada (Reusing Information)	Cuando el propietario de un resort es informado, que fue aprobada una nueva reserva para uno de sus sitios debe llenar una planilla otorgando el permiso de utilización para lo cual debe repetir datos básicos existentes en su registro como (nombre, apellido, dirección, etc.)	no
RATWCE07	Buscar Resorts	Soportar Búsquedas exhaustivas (Supporting Comprehensive Searching)	Cuando se realiza la búsqueda del resort quizás el usuarios sólo posea uno de los datos (parte de la ubicación, el nombre, el número de habitaciones), es importante permitir que el usuario pueda realizar una búsqueda poseyendo escasos datos y que brinde resultados acertados. El sistema debe permitir a los usuarios buscar datos de una manera integral y coherente por criterios pertinentes.	si

Tabla 5. Escenarios Clasificados.

		Afectados						
		RATWCE01	RATWCE02	RATWCE03	RATWCE04	RATWCE05	RATWCE06	RATWCE07
afectan	RATWCE01	N/A	N/A	N/A	N/A	Manejo de errores - S	N/A	N/A
	RATWCE02	N/A	N/A	N/A	N/A	N/A	N/A	Consistencia - M Orientación - M Minimizar la carga cognitiva - R
	RATWCE03	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	RATWCE04	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	RATWCE05							
	RATWCE06							
	RATWCE07	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Tabla 6. Matriz de Escenarios

Escenario	Puntaje	Puntaje Problemas
RATWCE01	6	1
RATWCE02	7	6
RATWCE03	4	0
RATWCE04	3	0
RATWCE07	8	7
Totales	28	14
Ponderación Final	**28 (+14)**	

Tabla 7. Resultados de la ponderación de escenarios

Mediante el análisis de esta matriz y las plantillas de los escenarios detectados, se puede determinar que el caso RATWCE01 (agregar disponibilidad) afecta al caso RATWCE05 (aprobar solicitudes de reserva) convirtiéndolo en un caso sensible, afectando el manejo de errores, ya que al ser una aplicación web, debe asegurarse la coherencia entre los datos en todo momento y la solución arquitectónica propuesta: la agregación múltiple de disponibilidades en una misma pantalla, podría ocasionar que en ciertos momentos la información mostrada para el caso RATWCE05 no esté actualizada y ciertos errores no sean detectados, ni advertidos, razón por la que debe contemplarse algún mecanismo que

permita que la información en ambas pantallas, sea la misma (a través mecanismos de actualización o refrescamiento de pantalla dinámico, etc.).

Por otro lado el caso la solución propuesta para el caso RATWCE02 (agregar nuevo resort), mejora dos propiedades en el caso RATWCE07, ya que al proporcionar un mecanismo de autocompletación de datos para los resorts, facilita el manejo de las búsquedas dándole consistencia con lo mostrado en el formulario de registro, lo cual instruirá al usuario en los parámetros pueden utilizarse posteriormente para realizar búsquedas avanzadas, de igual forma compromete el atributo de minimizar la carga cognitiva al aumentar el nivel de complejidad en las tareas del usuario, por lo cual se debe analizar más a fondo la implementación de este caso.

5.6 Se generan alternativas que prevengan, disminuyan o eliminen estas situaciones en el patrón presentado.

Para este caso los cambios sugeridos serian los siguientes:

- Agregar Disponibilidad (RATWCE01): se generara un mecanismo para la agregación de múltiples disponibilidades simultáneamente en una misma pantalla, en el cual no se permitirá la edición o eliminación de una reservación de dicha pantalla que posea el status "reservado". Para este caso debe asegurarse un método que mantenga actualizadas las demás áreas del sistema cuando ocurra un cambio en esta pantalla, asegurando la consistencia de la información mostrada en todas las áreas del sistema en un momento dado.
- Agregar Nuevo Resort (RATWCE02): se generara una estructura de datos que contenga los datos comunes o repetidos en el registro de todos los resorts y con esta realizar búsquedas al momento de la agregación de un nuevo resort ofreciendo a los usuarios autocompletar los datos con las coincidencias halladas.
- Autentificación (RATWCE03): se generara un método y funcionalidad asociada, que permita el restablecimiento de una nueva contraseña, mediante el envío de un email con un una dirección especifica donde el usuario podrá elegir una nueva clave.

- Registro en el sistema (RATWCE04): Implementación de dos mecanismos de ayuda distintos, customizados, que serán mostrados de acuerdo al tipo de usuario que desee realizar el registro.
- Buscar Resorts (RATWCE07): Implementar un mecanismo de búsqueda que permita al usuario seleccionar que parámetro desea utilizar para la realización de la búsqueda, insertando la palabra clave correspondiente, de no seleccionar ninguno el sistema realizara una búsqueda avanzada de esta palabra en cada uno de los parámetros existentes.

Recomendaciones: para el caso RATWCE02, se recomienda analizar el incremento del nivel de complejidad en las tareas del usuario, y en general tener en cuenta aquellos casos que pudieran afectar el tiempo de respuesta del sistema elevándolo, como búsquedas avanzadas entre otros, ya que este problema fue mencionado en varios escenarios detectados.

5.7 Se modifica el diseño anterior para incluir la funcionalidad correspondiente al patrón considerado.

Se introducen los cambios y consideraciones al modelo arquitectónico.

Capítulo VI

Conclusiones y Recomendaciones

Al incluir los atributos y propiedades de la usabilidad en un sistema software, se requiere tomar en cuenta la diversidad de situaciones que en muchas oportunidades ni siquiera los usuarios afectados (stakeholders) son capaces de plantear.

Por lo tanto, se necesita una gran cantidad de información producto de extensas discusiones entre los usuarios y los desarrolladores del sistema, para así poder especificar adecuadamente las distintas características de usabilidad.

El método elaborado por esta investigación, ayuda a detectar y prevenir errores a un nivel temprano del proceso de desarrollo, y definir las características diferenciadas en los escenarios contemplados, así como orientar de una manera precisa las discusiones que se deben llevar a cabo para mejorar las condiciones a lo largo del proceso de desarrollo del sistema a construir, en cuanto a usabilidad se refiere.

Con base en todo lo planteado hasta este momento en la investigación, es importante recalcar que en la actualidad no existe un tratamiento lo suficientemente consolidado en el tema de la usabilidad, que tome en cuenta esta propiedad desde etapas tempranas del proceso de creación de software. Mediante esta investigación se aborda este problema, obteniendo los siguientes resultados:

1. Recopilatorio de métodos y técnicas que permitan la evaluación de la usabilidad dentro del proceso de desarrollo, basadas en la documentación existente.
2. Estudio de viabilidad de la investigación.
3. Cuestionarios realizados para de recopilación de información acerca características de la arquitectura de software de un proyecto y la usabilidad.
4. Técnicas y procedimientos que permitan la optimización de la usabilidad dentro de las arquitecturas de software para aplicaciones empresariales acorde a los resultados obtenidos.

5. Un modelo para realizar evaluaciones enfocadas en la usabilidad, con base en las técnicas de evaluación de atributos de calidad del software existentes: un estudio de las alternativas existentes para la selección, aplicación y verificación de componentes de software desde el punto de vista de la usabilidad (mediante el análisis de los atributos: satisfacción, aprendizaje, eficiencia y fiabilidad). Resultante de establecer técnicas y procedimientos que permitan la optimización de esta y que puede ser aplicado sobre entidades de muy diferentes y en etapas diversas del ciclo de vida del software. Este método tiene además las siguientes características especiales:
    a. Puede ser utilizado con facilidad por una organización de consultoría y desarrollo de software.
    b. Fomenta la colección y utilización del conocimiento que habitualmente se pierde, durante el proceso de diseño de software.
    c. No está atado a ningún patrón de arquitectura, por lo cual puede ser usado en cualquier organización sin importar cual sea el utilizado por esta, con el fin de evaluar las decisiones arquitecturales tomadas.
6. Documentación sobre el caso de estudio donde se llevó a cabo la aplicación del modelo en la empresa venezolana de consultoría Parodi Solutions C.A, para la verificación de su validez en un caso de estudio representativo.

Cuando se realiza el proceso de diseño de software, es frecuente el verificar si la combinación de patrones y componentes encajan correctamente, pero existen defectos que no tienen relación alguna con las signaturas de los componentes, y que surgen solamente de su combinación. Y estos defectos no suelen detectarse de forma automatizada; desde el enfoque de la usabilidad se está proponiendo un método para detectar posibles situaciones, en etapas tempranas de ese proceso, teniendo en cuenta las restricciones de funcionamiento de los componentes y poner en relación esas restricciones, sean del tipo que sean (funcionales, de rendimiento o de propósito).

Muchos de estos defectos podrían detectarse en etapas tempranas, sólo se necesita la identificación de las situaciones sensibles, y determinar en qué grado afectan la totalidad del sistema. Solo se necesita un método que permita adjuntar a cada situación a una

descripción de las restricciones conocidas respecto a su funcionamiento, y utilizar esas restricciones en la toma de decisiones. Tal como se propone en el método expuesto en este trabajo.

En la verificación bibliográfica y demás consultas hechas a las diferentes fuentes documentales y en el tiempo de recolección de toda la data necesaria para la misma, no se hicieron presente situaciones emergentes no solucionables que impidiesen la consecución de los objetivos planteados.

A nivel metodológico ciertos puntos de la investigación se hicieron bastante engorrosos, al comienzo crear el nivel de abstracción necesario que permitiera generar mecanismos para la evaluación de la usabilidad basándose en un modelo metodológico sistémico, no utilizado anteriormente por el investigador genero múltiples dudas sobre el rumbo de la investigación, al tener que plantear el proceso como el análisis un todo pero a la vez conseguir mecanismos específicos que atacaran eficientemente el problema desde diferentes ángulos.

Particularmente se presentaron inconvenientes con el personal en la etapa del caso de estudio, al incluir una nueva fase dentro del proceso de análisis y diseño de la empresa donde se realizo el caso de estudio, surgió un rechazo natural al cambio en la forma de realizar las tareas, ya que a vista se pedía un mayor nivel de análisis que los hacía tardar más tiempo en sus labores. Pero luego de entender el aporte que esto daba a la calidad de su trabajo y como disminuía el tiempo perdido en errores a largo plazo esta resistencia comenzó a ceder.

Desde el punto de vista del investigador, este trabajo genero múltiples aportes a nivel profesional, cambiando su visión sobre ciertos tópicos, entendiendo que aunque el avance de las tecnologías brinde a los expertos en el área más y mejores herramientas para la resolución de situaciones, estas jamás brindaran un aporte adecuado al cliente, si no se orientan adecuadamente y si durante algunas etapas en la creación del producto se excluye a este basándose en su desconocimiento en el área tecnológica, o en el nivel de abstracción que requiere el proceso de desarrollo. Se debe tomar en cuenta que aunque esto sea cierto, el cliente también posee conocimiento valioso que puede nutrir el proceso, como lo es la

lógica de negocio, la frecuencia de uso del sistema y el contacto de primera mano con la realidad que se busca mejorar a través de la tecnología, ideas que no solo deben tomarse en cuenta al momento del levantamiento de información, sino a lo largo de todo el ciclo de vida del software.

Se puede concluir la investigación, determinando que no existe un tratamiento lo suficientemente exhaustivo y consolidado para detectar problemas de usabilidad en fases tempranas del ciclo de vida del software, a pesar de su gran importancia.

Sin el establecimiento de métricas para evaluar la usabilidad en fases tempranas (ya que las utilizadas para evaluar un sistema terminado no pueden ser medidas en la etapa de diseño, por carecer de la información necesaria en ese momento). Para resolver este conflicto, una posible solución a largo plazo puede basarse en dos ideas principales. En primer lugar, es preciso definir y consensuar modelos de usabilidad para productos en etapas tempranas. Y en segundo lugar, la evaluación de la usabilidad en el diseño basado en su arquitectura debiera hacerse por entidades, al menos hasta que los desarrolladores y arquitectos de software adquieran madurez en el tema de usabilidad.

Con respecto al aporte de esta investigación al área de Sistemas de Información, este punto se encuentra íntimamente ligado a lo dicho anteriormente con respecto al aporte profesional del investigador, este trabajo fue a partes iguales una búsqueda de conocimiento necesario para la evaluación y mejora en la forma en que los sistemas actuales interactúan con el usuario y una forma de acercar el conocimiento del usuario a los expertos en tecnología y cambiar la su interacción, obligándolos a entender desde fases tempranas del ciclo como sus decisiones afectan las funcionalidades que puede llevar a cabo el usuario con el producto final, pero preservando el aspecto técnico.

Como una recomendación importante resalta la aplicación de estas técnicas a entornos y arquitecturas con marcadas diferencias permitiendo comprobar la flexibilidad y amplitud del método propuesto. Además, se deberán elegir casos de estudio variados, no relacionados entre sí, representativos de diversos aspectos del desarrollo del software, y sobre todo se deberá tratar de casos de estudio no tomados en cuenta a la hora de formular el modelo o las herramientas.

Se aconseja la creación de una herramienta automatizada para el seguimiento del método de evaluación, lo cual posiblemente aporte facilidad al cumplimento del proceso.

Sería importante realizar una descripción detallada aquellos escenarios sensibles frecuentemente hallados en los proyectos analizados, construyendo una solución desde la experiencia en los distintos casos donde se presento, con las alternativas de solución encontradas, y posteriormente crear una comunidad digital que se nutra de estos contenidos. Así se podrá generar una biblioteca de conocimiento con respecto a los escenarios más comunes y las mejores prácticas para su solución, brindando formas de resolución basadas en conocimiento previo y actualización continua de la información.

Referencias Bibliográficas

Arhippainen, L. (2003). Capturing user experience for product design. IRIS26, the 26thInformation Systems Research Seminar in Scandinavia. Porvoo, Finland, 9-12 August 2003. Recuperado de: http://www.vtt.fi/virtual/adamos/material/arhippa2.pdf

Arhippainen, L., Tähti, M. (2003). Empirical Evaluation of User Experience in Two Adaptative Mobile Application Prototypes. Proceedings of the 2nd International Conference on Mobile and Ubiquitous Multimedia, 10–12 December, 2003, Norrköping, Sweden. Recuperado de: http://www.ep.liu.se/ecp/011/007/ecp01 1007.pdf

Alva, M.(2005). Metodología de Medición y Evaluación de la Usabilidad en Sitios Web Educativos, Tesis doctoral, Universidad de Oviedo.

Arias, F. (2006). El proyecto de Investigación. Introducción a la metodología científica. Caracas, Editorial Episteme.

Bass L., Clements P., Kazman R. (2004). Software Architecture in practice. Addison-Wesley. SEI Series in Software Engineering.

Bass, L., John, B., Kates, J. (2001). Achieving Usability Through Software Architecture (CMU/SEI-2001-TR-005).Software Engineering Institute, Carnegie Mellon University. Recuperado el 22 de junio de 2012 de: http://www.sei.cmu.edu/library/-abstracts/reports/01tr005.cfm

Berry, D. (2000). *The user experience: The iceberg analogy of usability.* Technical library of the IBM Ease of Use Team. Recuperado el 22 de junio de 2012 de: http://www.ibm.com/developerworks/library/w-berry/

Bevan, N., Kirakowski, J., Maissel, J. (1991). What is Usability?. Proceedings of the 4th International Conference on HCI, Stuttgart, September 1991. Elsevier.

Bou Bauzá, G. (2003). El guión multimedia. España: Anaya Multimedia. 2003.

Cerda, H. (2003). Cómo elaborar proyectos : Diseño, ejecución y evaluación de proyectos sociales y educativos.: Cooperativa Editorial Magisterio, Bogotá.

Cai Xia, Lyu Michael R., Wong, K., Ko R. (2001). Component-Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes. The Chinese University of Hong Kong. . Recuperado el 01 de diciembre de 2011 de: http://www.cse.cuhk.edu.hk/~lyu/paper_pdf/apsec.pdf.

IEEE-SA Standards Board (2000). *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. Doi: 10.1002/sys.20008

Dillon, A. (2001). Beyond Usability: Process, Outcome and Affect in human computer interactions. Lazerow Lecture 2001, at the Faculty of Information Studies, University of Toronto, March 2001. Recuperado de: http://www.ischool.utexas.edu/~ad illon/publications/beyond_usability.html

D'Hertefelt, S. (2000). Emerging and future usability challenges: designing user experiences and user communities. Recuperado el 2 Febrero de 2011 de: http://www.interactionarchitect.com/future/vision200 00202shd.htm

DNX (2005). Usabilidad y Experiencia de Usuario. Microsoft España: Guía Práctica de Usabilidad Web. Recuperado el 2 Febrero de 2011 de: http://www.microsoft.com /spain/empresas/guias/usabilidad/experiencia_usuario.mspx

Fernández, F., Montero, Y. (2005). La Experiencia del Usuario. *No sólo usabilidad, revista multidisciplinaria sobre diseño, personas y tecnología [ISSN 1886-8592].* Recuperado de: http://www.nosolousabilidad.com/articulos/experiencia_del_ usuario.htm

Ferré, X. (2010). Principios Básicos de Usabilidad para Ingenieros Software. Recuperado de: http://is.ls.fi.upm.es/xavier/papers/usabilidad.pdf

Folwer, M. (2002). Patterns of Enterprise Application Architecture. Boston: Addison-Wesley Professional.

Folmer, E., Bosch, J. (2003). Architecting for usability: a survey. En: Journal of Systems and Software. Febrero 2004, v. 70, n. 1-2. pp. 61-78.

Freeman, E. (1983). Stakeholder Theory of the Modern Corporation. . Recuperado el 01 de diciembre de 2011 de: http://academic.udayton.edu/lawrenceulrich/Stakeholder%20Theory.pdf

Knapp Bjerén, A. (2003). La Experiencia del Usuario. En: Knapp Bjerén, A. (coord.). La Experiencia del Usuario. Madrid: Anaya Multimedia, 2003, ISBN 84-415-1044-X.

Kankainen, A. (2002). Thinking model and tools for understanding user experience related to information appliance product concept. Tesis Doctoral, Helsinky University of Technology, Recuperado el 9 de Diciembre de 2002, de: http://lib.tkk.fi/Diss/2002/isbn9512263076/

Hassan, Y., Martín Fernández, F., Iazza, G. (2004). Diseño Web Centrado en el Usuario: Usabilidad y Arquitectura de la Información. Hipertext.net, núm. 2, 2004. Recuperado de: http://www.hipertext.net/web/pag206.htm.

Hartson, H. (1998). Human-computer interaction: Interdisciplinary roots and trends. Journal of Systems and Software, Noviembre 1998, v. 43, n. 2, pp. 103-118.

Hernández, R., Fernández, C., Baptista, P. (2010) Metodología de la Investigación. 5ta Edición. México , México D.F.: Editorial McGraw Hill.

Hernández, R..(1998). Metodología de la Investigación. Editorial McGraw Hill.

Hix, D., Hartson, H. (1993). Developing user interface: Ensuring usability through product & process. New York, John Wiley and Sons.

Hom, J. (1998). The Usability Methods Toolbox. Recuperado el 10 de Junio de 1998, de: http://jthom.best.vwh.net/usability/

Hurtado, J. (2010), Metodología de la Investigación. Caracas: Quiron Ediciones. Cuarta Edición.

Mahlke, S. (2002). Factors influencing the experience of website usage. Recuperado el 2 Febrero de 2011 de: http://www.nosolousabilidad.com/articulos/experiencia_del_usuario.htm#sthash.T9e5XZtO.dpuf

Magnus, R. (1996) . CSCW Evaluation in Five Types. Computing Department Lancaster University.

Manus, R. (1997). Developing a Methodology for the Evaluation Cooperative Systems. Computing Department Lancaster University.

Moreno, A., Sánchez, M. (2012). Patrones de Usabilidad: Mejora de la Usabilidad del Software desde el momento de Arquitectónico. Recuperado el 02 de Enero de 2012 de http://www.willydev.net/descargas/prev/PatronesUsa.pdf

Nielsen, J. (1994). Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier. Recuperado el 02 de Enero de 2012 de http://www.useit.com/papers/guerrilla_hci.html

Nielsen, J. (1993). Usability Engineering. AP Professional.

Norman, D. (2002). Emotion and Design: Attractive things work better. Interactions Magazine, ix (4), pp. 36-42. Recuperado de: http://www.jnd.org/dn.mss/emotion_design_at.html

Nielsen & Norman Group. (2003). User Experience - Our Definition. Nielsen Norman Group. Recuperado de: http://www.jnd.org/dn.mss/emotion_design_at.html#sthash.PJsQJMc1.dpuf.

Nielsen, J., Molich, R.. (1990). Heuristic evaluation of user interfaces, Proceedings of the CHI'90. New York, pp.349-256.

Morville, P. (2004). User Experience Design. Semantic Studios, Recuperado el 21 de Junio de 2004, de: http://semanticstudios.com/publications/semantics/000029.php

Obeso, M. (2005). Obeso (2005) Metodología de Medición y Evaluación de la Usabilidad en Sitios Web Educativos. Recuperado de: http://di002.edv.uniovi.es/ ~cueva/investigación/tesis/Elena.pdf

Palella, S., Martins, F. (2006). Metodología de la investigación cuantitativa. Segunda Edición. Caracas: Fondo Editorial de la Universidad Pedagógica Libertador.

Preece, J. , Rogers, Y., Sharp, H., Benyon, D. , Holland, S. y Carey. T. (1994). Human-Computer Interaction. Addison Wesley.

Sabino, C. (2000). El proceso de la investigación. Caracas: Panapo.

Szyperski, C. (2002). Component Software Component Software Beyond Object – Oriented Programming. Second Edition. Addison-Wesley / ACM Press.

Tamayo, M. T. (2003). El proceso de la investigación científica. México DF: Limusa.

Torrente M. (2011). SIRIUS: Sistema de Evaluación de la Usabilidad Web Orientado al Usuario y basado en la Determinación de Tareas Críticas. Recuperado de: http://di002.edv.uniovi.es/~cueva/investigacion/tesis/Sirius.pdf

Yin, Robert K. (1994). Case Study Research. Design and Methods. London: SAGE, 1994.

Whitefield, A., Wilson, F., Dowell, J. (1991). A framework for human factors evaluation.

Wixon, D., Wilson, C. (1997). The usability-engineering framework for product design and evaluation. Handbook of HCI, 2nd edition, Elsevier Science, p653-688.

Anexos

Anexo 1: Patrones Arquitecturales para Aplicaciones Empresariales

Basado en Patterns of Enterprise Application Architecture

Martin Fowler y David Rice

Las aplicaciones empresariales normalmente involucran persistencia de datos. Normalmente hay muchos datos y se acceden a ellos desde múltiples sitios al mismo tiempo, concurrentemente. Generalmente, deben integrarse con otras aplicaciones empresariales. Incluso dispone de una única tecnología para la integración de aplicaciones, existen diferencias en los proceso de negocios y modelos conceptuales que dificultan dicha integración.

Cuando se necesita un patrón, debe encontrarse la manera de aplicarlo a las circunstancias. Un aspecto clave acerca de los patrones es que no puedes aplicar la solución a ciegas, por lo que las herramientas del patrón han fracasado tan miserablemente. Cada patrón es relativamente independiente, pero los patrones no están aislados unos de otros. En la estructura de un patrón el primer elemento es el nombre del mismo, los nombres de patrones son cruciales, ya que parte del propósito de los patrones es la creación de un vocabulario que permita a los diseñadores una comunicación más efectivamente. Dos elementos que van juntos: la intención y el boceto. La intención, resume el patrón en una frase o dos, y el boceto es una representación visual del patrón, a menudo, pero no siempre es un diagrama UML. La idea es crear un breve recordatorio del patrón que haga más fácil recordarlo. Otro punto importante es la motivación del patrón, ya que esto puede no ser el único problema que soluciona el patrón, pero es lo que mejor lo motiva.

1 Arquitectura por Capas

Una de las técnicas más comunes que para descomponer un problema informático complejo. La capa superior utilizan los servicios definidos por la inferior, pero la capa inferior no es consciente de la capa superior. Además cada capa normalmente esconde los niveles más bajos de las capas superiores.

Ventajas:

- Sólo es necesario conocer la capa superior y es posible despreocuparse de las otras capas por debajo.
- Se pueden reemplazar capas por implementaciones alternativas de los mismos servicios básicos.
- Se minimizan las dependencias entre capas.
- Una capa puede ser reutilizada por múltiples servicios de niveles superiores.

Desventajas:

- Posibilidad de cambios en cascada (p.e. nuevo campo en la interfaz de usuario supone cambio en la base de datos y en todas las capas intermedias).
- Las capas extra pueden reducir el rendimiento.
- La parte más complicada es decidir que capas y que funcionalidad poner en cada capa.

1.1 Evolución de las capas en las aplicaciones empresariales

El concepto de capas se popularizó en los 90 con la aparición de los sistemas cliente-servidor. Estos son sistemas de dos capas: el cliente con la interfaz de usuario y código de aplicación y el servidor, generalmente una base de datos. El problema surgió con la lógica de dominio: reglas de negocio, validaciones, cálculos, no encajaba ni en la capa cliente ni en el servidor. La respuesta es la arquitectura en tres capas: una capa presentación para la interfaz de usuario, una capa de dominio con la lógica de dominio y una capa de datos. El éxito del modelo Web ha acabado de consolidarla arquitectura de tres capas.

1.2 La Arquitectura de tres capas

Esta arquitectura, la más común en la actualidad, está compuesta por las siguientes capas:

- *Presentación:* comprende la lógica para manejar la interacción entre el usuario y la aplicación. Posibles opciones son la línea de comandos, sistemas de menús basados en texto, el cliente rico o la interfaz de usuario HTML.

- *Fuentes de Datos:* tiene que ver con la comunicación con otros sistemas que llevan a cabo tareas en nombre de la aplicación. Monitores de transacciones, otras aplicaciones, sistemas de mensajería, pero generalmente base de datos.
- *Lógica de Dominio:* la lógica de dominio o empresarial es la funcionalidad específica que la aplicación debe hacer para el dominio de trabajo. Los cálculos sobre la base de las entradas y los datos almacenados, la validación de los datos de entrada o determinar la fuente de datos implicada son ejemplos de funcionalidades en la capa de dominio. Un aspecto difuso de esta arquitectura es lo que sucede cuando no hay una persona utilizando el software, por ejemplo un servicio web o un proceso por lotes (batch). En este caso se difumina la diferencia entre presentación y datos, ya que las dos se refieren a conexiones con el exterior de la aplicación.

La distinción a hacer es entre una interfaz que la aplicación proporciona como un servicio a otros, y el uso que se haga de los servicios proporcionados por otros como fuentes de datos.

1.3 Ejecución de las capas

La separación entre capas es útil incluso si se ejecutan en el mismo ordenador. De todas formas, para la mayoría de las aplicaciones la decisión es si ejecutar una parte del procesamiento en el cliente o en el servidor. El caso más simple es ejecutar todo en el servidor, por ejemplo utilizando una interfaz HTML, lo que facilita la actualización y mantenimiento. Los argumentos en favor de ejecutar parte en el cliente se justifican básicamente por conseguir una mayor capacidad de respuesta o el funcionamiento sin estar conectado (offline).

2 Organización de la Lógica de Dominio

El enfoque más sencillo para gestionar la lógica de dominio es el Transaction Script. Es esencialmente un procedimiento que recibe los parámetros de entrada de la presentación, los procesa, almacena información en la base de datos, invoca operaciones de otros sistemas y responde con más datos a la presentación. La organización fundamental es un procedimiento único para cada acción del usuario, por ejemplo un script. De todas formas no tiene porque ser un procedimiento de código auto contenido, podría haber subrutinas compartidas entre diferentes Transaction Scripts.

Una tienda online tendría Transaction Scripts para añadir productos al carro de la compra, pasar por caja, mostrar el estado del pedido, etc.

Ventajas: Modelo procedural simple, funciona bien con una capa de datos simple. hace obvios los límites de la transacción: se abre al principio del procedimiento y se cierra al acabar.

Desventajas: Aparecen muchas a medida que la complejidad de la lógica de dominio aumenta. A menudo habrá código duplicado, difícil de eliminar. Una lógica compleja requiere el uso de objetos para definir un Domain Model. Aunque esta es la mejor elección, cuanto más rico sea el modelo de dominio más complejo será el mapeo de éste a la base de datos relacional.

La tercera opción para estructurar la lógica de domino es el Table Module, diseñado para funcionar con Record Set, el cual se obtiene como resultado de realizar consultas a la base de datos. El Table Module está a medio camino entre Transaction Script y Domain Model. Organizar la lógica de dominio en base a las tablas permite una mayor estructura que con los procedimientos de Transaction Script, pero carece de herencia, estrategias y otros patrones de orientación a objetos. La mayor virtud de Table Module es que encaja perfectamente en arquitecturas como .NET.

2.1 La interfaz a la lógica de dominio

Las principales opciones son la interfaz basada en HTTP o la orientada a objetos. El caso más común de interfaz HTTP es el navegador web pero puede utilizarse esta interfaz para más que simplemente presentaciones web utilizando Servicios Web, la comunicación entre sistemas sobre el protocolo HTTP, normalmente utilizando XML para los mensajes intercambiados.

El hecho de que XML es un formato común con parsers disponibles en muchas plataformas permite la comunicación entre aplicaciones basadas en plataformas diferentes, como lo hace el hecho de que HTTP es prácticamente universal. HTTP también facilita los temas de seguridad ya que suele ser un puerto abierto en los cortafuegos, mientras que razones de seguridad hacen difícil abrir otros puertos muy a menudo.

Sin embargo, una interfaz HTTP es poco explícita, no queda claro a primera vista que es lo que hay, mientras una interfaz orientada a objetos tiene objetos con métodos y parámetros marcados más explícitamente. Si no es necesario atravesar ningún cortafuegos, puede resultar más conveniente una interfaz orientada a objetos, que también facilita mecanismos tales como los de control de acceso o las transacciones.

3 Presentación Web

Uno de los mayores cambios en las aplicaciones empresariales en los últimos años es la aparición de interfaces de usuario basadas en navegadores web. Las principales ventajas: no hay software de cliente que instalar, una aproximación de interfaz de usuario común y el acceso universal. El trabajo del servidor web es interpretar las URL de las solicitudes que recibe y pasar el control a una aplicación del servidor web. Principalmente, hay dos formas de estructurar una aplicación en un servidor web: como un script o como una página de servidor. El script es un programa, normalmente con funciones o métodos para procesar la llamada HTTP, ejemplos son los scripts CGI o los servlets de Java.

La salida es un cadena de texto HTML, la respuesta, que el script genera utilizando las operaciones de escritura común en muchos lenguajes de programación. Las páginas de servidor surgieron para hacer este proceso más fácil. La página se compone de porciones de HTML y de scripts mezclados, de forma que es más intuitivo como se construye el HTML de salida. Esta aproximación es la más apropiada cuando hay poco procesamiento de la respuesta. Ejemplos de páginas de servidor son PHP, ASP y JSP. Los scripts funcionan mejor para interpretar la solicitud y las páginas de servidor para formatearla respuesta.

La presentación web se ajusta muy bien al patrón Model View Controler. La solicitud HTTP le llega a un controlador de entrada. Se remite a la lógica de negocio a un objeto apropiado del modelo. El objeto del modelo se comunica con la fuente de datos y hace todo lo indicado por la solicitud así como recopilar información para la respuesta. Cuando se ha hecho todo esto, retorna al controlador de entrada. Este último decide que vista es la más apropiada y le pasa el control remitiéndole los datos necesarios a través de algún tipo de objeto de sesión HTTP compartido.

3.1 Patrones Vista

Respecto a la vista hay tres patrones: Transform View, Template View y Two Step View. Los patrones básicos para Transform View y Template View suponen una única etapa. Two Step View es una variación que puede aplicarse tanto a Transform View como a Template View. Template View permite escribir la presentación en la estructura de la página e incrustar marcadores en la página para indicar dónde incluir el contenido dinámico. Es el patrón habitual para las páginas de servidor. Facilita el desarrollo pero provoca un código desordenado que se hace difícil de mantener.

Transform View utiliza algún tipo de lenguaje de transformación, normalmente XSLT que es muy efectivo para datos XML. El controlador de entrada selecciona la transformación XSLT apropiada y la aplica al XML generado a partir del modelo. Proporciona mucha flexibilidad, siendo fácil de generar vista p ara diferentes presentaciones o dispositivos, aunque representa un mayor grado de complejidad. La segunda decisión es si se utiliza un patrón basado en una única etapa o en dos. Los patrones de una sola etapa normalmente tienen un componente de vista para cada pantalla de la aplicación. La vista recibe datos basados en el dominio y los muestra como HTML.

Una patrón vista en dos fases, Two Step View, subdivide el proceso en dos pasos, el primero produce una pantalla lógica a partir de los datos del dominio y luego, el segundo, la presentación.

La ventaja de Two Step View es que se pone la decisión de que HTML generar en un sólo lugar, esto hace que los cambios globales al HTML sean más fáciles. Es apropiada para sitios donde diferentes pantallas comparten el mismo diseño básico.

3.2 Patrones Controlador de Entrada

Hay dos patrones para el controlador de entrada. El caso más común es tener un objeto controlador de entrada para cada página del sitio web, Page Contraller. Normalmente habrá un Page Controller para cada acción: un botón o enlace. Cada controlador de entrada tiene que hacer frente a dos responsabilidades: atender las solicitudes HTTP y decidir qué hacer con ellas. La aproximación del otro patrón controlador, Front Controller, es separar estas dos responsabilidades. Con el Front Controller hay un único objeto que recibe todas las

peticiones y crea un objeto separado para procesar la solicitud. Esto permite centralizar toda la gestión de las solicitudes HTTP en un único objeto.

4 Representación de una Base de Datos Relacional

Cuando se utilizan objetos el problema es como interactuar con bases de datos relacionales, dos estructuras de datos bastante diferentes. Además, los objetos están en la memoria y es necesario coordinarlos con lo que está sucediendo en el disco, la base de datos.

Con una base de datos orientada a objetos no hay que preocuparse de mapeos. Esto mejora la productividad, ya que el mapeo supone alrededor de un tercio del esfuerzo de programación y mantenimiento. Sin embargo, la mayoría de los proyectos no utilizan bases de datos OO debido al riesgo de esta tecnología todavía no suficientemente madura en comparación a las BBDD relacionales.

4.1 Patrones Arquitectónicos

Los patrones arquitectónicos para el mapeo a base de datos relacional son principalmente Active Record, Row Data Gateway, Table Data Gateway y Data Mapper. Row Data Gateway y Table Data Gateway se basan ambos en el patrón básico Gateway, una clase en memoria que mapea a una tabla en la base de datos y tiene un campo por columna de la base de datos. Los patrones basados en Gateway contienen todo el código de mapeo a la base de datos pero no código de la lógica de dominio.

La elección entre Row Data Gateway y Table Data Gateway depende de la lógica de dominio y también del soporte para el patrón básico Record Set que proporciona la plataforma utilizada para el desarrollo de la aplicación. Las plataformas de Microsoft, por ejemplo, utilizan Record Set ampliamente. Gran parte del funcionamiento de la interfaz de usuario se basa en manipular

Record Sets. Por lo tanto, en este tipo de plataformas la opción natural es Table Data Gateway. Por el contrario, Row Data Gateway es la mejor elección cuando los Result Sets son más difíciles de manipular porque no tienen tan buen soporte en la plataforma. En este caso lo que se obtiene es un objeto por cada registro en la fuente de datos.

Si la aplicación tiene un modelo de dominio que se parece mucho al modelo de la base de datos, entonces es mejor considerar Active Record, que combina el Gateway y el objeto de dominio en una sola clase que incluye tanto código de acceso a la BB.DD. como la lógica empresarial. Pese a todo, es recomendable mantener estas responsabilidades separadas.

La tercera opción es Data Mapper. Este es el patrón más complejo, pero también el más flexible. Se necesita una inversión de desarrollo pero permite que los objetos del dominio sean totalmente independientes de la base de datos. En cualquier caso, a medida que el mapeo se complica se hace recomendable no desarrollar el mapeo nosotros mismos y reutilizar herramientas existentes de mapeo Objecto-Relacional.

4.2 Concurrencia

Se trata del problema de cómo conseguir que los diversos objetos utilizados se carguen y salven en la base de datos. Por ejemplo, utilizando Data Mapper, los objetos del dominio no pueden hacer llamadas a la capa de mapeo ya que es totalmente transparente para ellos. Además, hay que seguirla pista de cada objeto modificado y asegurarse de escribir todos los cambios en la BB.DD.

Además, hay el problema de la concurrencia, asegurar que ningún otro proceso cambia cualquiera de los objetos que se han leído mientras se trabaja con ellos. Un patrón para resolver estos problemas es Unit of Work. Este patrón mantiene un registro de todos los objetos leídos desde la base de datos, junto con todos los modificados. También maneja cómo las actualizaciones se escriben en la base de datos. Sin Unit of Work, la capa de dominio es que debe decidir cuándo leer y escribir en la base de datos. Unit of Work resulta de extraer el comportamiento de control del mapeo de datos a un objeto específico para esta funcionalidad.

4.3 Lectura de Datos

Es conveniente encapsular el SQL en las clases de mapeo objeto-relacional correspondientes. Los métodos de búsqueda encapsulan comandos SQL "SELECT" con una API, por ejemplo "buscar(id)" o "buscarCliente(idCliente)" encapsularían las consultas SQL correspondientes para recuperar un objeto con identificador id o un cliente con idCliente.

Es recomendable utilizar un patrón Identity Map para seguir la pista de lo que se ha leído para evitar hacerlo dos veces. Si se está utilizando Unit of Work, el lugar obvio para mantener el Identity Map es el objeto correspondiente al Unit of Work.

4.4 Patrones de Representación de Estructuras de Datos

Estos patrones describen cómo los datos en una base de datos relacional se mapean a datos en objetos. El problema surge de la diferente manera en que los objetos y las tablas manejan los enlaces. Los objetos manejan los enlaces almacenando referencias y las tablas mediante una clave apuntando a otra tabla. El segundo tema es que los objetos pueden hacer fácilmente uso de colecciones para manejar múltiples referencias contenidas en un sólo campo, mientras que la normalización fuerza que todos los campos de base de datos tengan un único valor. La solución es mantener la identidad relacional de cada objeto como un campo identidad, Identity Field, en el objeto, y utilizarla para mapear entre las referencias a objetos y las claves relacionales.

Cuando se leen objetos, cada vez que se encuentra una clave foránea, se mira en el Identity Map para establecer las referencias entre objetos. Si la clave no está en el Identity Map, entonces se va a la base de datos para cargarlo. Al guardar un objeto, se salva en la fila con la clave apropiada. Cualquier referencia entre objetos se sustituye por Identity Field del objeto referenciado. Si un objeto tiene una colección, hay que lanzar otra consulta para encontrar todas las filas que enlazan con el identificador del objeto que tiene la colección. Salvar dicho objeto implica guardar cada objeto en la colección y asegurarse de que tiene una clave foránea al objeto de la colección.

4.5 Herencia

Las jerarquías de clases en OO vienen definidas por la relación de herencia. En las bases de datos relacionales no hay una forma estándar para representar la herencia, por lo que hay que realizar u n mapeo al almacenarla en la base de datos.

4.6 Representación de Metadatos

El enfoque más sofisticado de mapeo es utilizando Metadata Mapping, que se basa en escribirlos detalles del mapeo en un archivo de metadatos, datos sobre los datos. Los detalles del mapeo son cómo las columnas de la base de datos se corresponden con los

campos en los objetos. Los metadatos permiten tener que escribir código repetitivo utilizando técnicas de generación de código a partir de los metadatos o mediante programación reflexiva (el código puede modificarse a sí mismo).

5 Concurrencia

Siempre que tengamos múltiples procesos o hilos manipulando los mismos datos, existe el riesgo de problemas de concurrencia. Las transacciones proporcionan un marco que ayuda a evitar muchos de los aspectos más complicados de concurrencia. Siempre y cuando se haga toda la manipulación de datos dentro de una transacción. Desafortunadamente, hay muchas interacciones con un sistema que no se pueden mantener dentro de una sola transacción de base de datos. En estos casos hay que aplicar concurrencia offline, es decir control de la concurrencia para datos manipulados durante múltiples transacciones de base de datos. Además, hay la concurrencia del servidor de aplicaciones: dar soporte a múltiples hilos de ejecución en un servidor de aplicaciones.

5.1 Problemas de Concurrencia

Los principales problemas que los sistemas de control de la concurrencia intentar evitar son:

- Actualizaciones perdidas: el usuario A edita un archivo para hacer algunos cambios. Mientras, el usuario B realiza un pequeño cambio en el mismo archivo de manera que termina antes que A. Desafortunadamente, cuando A leyó el archivo no incluía los cambios de B, así que cuando A acaba y guarda sus cambios, sobrescribe la versión de B de forma que su cambio se pierde.
- Lecturas inconsistentes: ocurren cuando se leen dos datos que eran correctos por separado, pero no eran correctos en el mismo momento. El usuario A quiere saber cuántos archivos hay en una carpeta que contiene dos subcarpetas. A mira en la primera subcarpeta y ve 7 archivos. Antes de mirar en la otra subcarpeta, el usuario B añade 2 ficheros en la primera subcarpeta y 3 a los 5 que había en la segunda. A mira ahora en la segunda subcarpeta y cuenta 8 archivos. Para él el total de archivos de la carpeta es 15.

Una tienda online tendría Transaction Scripts para añadir productos al carro de la compra, pasar por caja, mostrar el estado del pedido, etc.

Ventajas: Modelo procedural simple, funciona bien con una capa de datos simple. hace obvios los límites de la transacción: se abre al principio del procedimiento y se cierra al acabar.

Desventajas: Aparecen muchas a medida que la complejidad de la lógica de dominio aumenta. A menudo habrá código duplicado, difícil de eliminar. Una lógica compleja requiere el uso de objetos para definir un Domain Model. Aunque esta es la mejor elección, cuanto más rico sea el modelo de dominio más complejo será el mapeo de éste a la base de datos relacional.

La tercera opción para estructurar la lógica de domino es el Table Module, diseñado para funcionar con Record Set, el cual se obtiene como resultado de realizar consultas a la base de datos. El Table Module está a medio camino entre Transaction Script y Domain Model. Organizar la lógica de dominio en base a las tablas permite una mayor estructura que con los procedimientos de Transaction Script, pero carece de herencia, estrategias y otros patrones de orientación a objetos. La mayor virtud de Table Module es que encaja perfectamente en arquitecturas como .NET.

2.1 La interfaz a la lógica de dominio

Las principales opciones son la interfaz basada en HTTP o la orientada a objetos. El caso más común de interfaz HTTP es el navegador web pero puede utilizarse esta interfaz para más que simplemente presentaciones web utilizando Servicios Web, la comunicación entre sistemas sobre el protocolo HTTP, normalmente utilizando XML para los mensajes intercambiados.

El hecho de que XML es un formato común con parsers disponibles en muchas plataformas permite la comunicación entre aplicaciones basadas en plataformas diferentes, como lo hace el hecho de que HTTP es prácticamente universal. HTTP también facilita los temas de seguridad ya que suele ser un puerto abierto en los cortafuegos, mientras que razones de seguridad hacen difícil abrir otros puertos muy a menudo.

Sin embargo, una interfaz HTTP es poco explícita, no queda claro a primera vista que es lo que hay, mientras una interfaz orientada a objetos tiene objetos con métodos y parámetros marcados más explícitamente. Si no es necesario atravesar ningún cortafuegos, puede resultar más conveniente una interfaz orientada a objetos, que también facilita mecanismos tales como los de control de acceso o las transacciones.

3 Presentación Web

Uno de los mayores cambios en las aplicaciones empresariales en los últimos años es la aparición de interfaces de usuario basadas en navegadores web. Las principales ventajas: no hay software de cliente que instalar, una aproximación de interfaz de usuario común y el acceso universal. El trabajo del servidor web es interpretar las URL de las solicitudes que recibe y pasar el control a una aplicación del servidor web. Principalmente, hay dos formas de estructurar una aplicación en un servidor web: como un script o como una página de servidor. El script es un programa, normalmente con funciones o métodos para procesar la llamada HTTP, ejemplos son los scripts CGI o los servlets de Java.

La salida es un cadena de texto HTML, la respuesta, que el script genera utilizando las operaciones de escritura común en muchos lenguajes de programación. Las páginas de servidor surgieron para hacer este proceso más fácil. La página se compone de porciones de HTML y de scripts mezclados, de forma que es más intuitivo como se construye el HTML de salida. Esta aproximación es la más apropiada cuando hay poco procesamiento de la respuesta. Ejemplos de páginas de servidor son PHP, ASP y JSP. Los scripts funcionan mejor para interpretar la solicitud y las páginas de servidor para formatearla respuesta.

La presentación web se ajusta muy bien al patrón Model View Controler. La solicitud HTTP le llega a un controlador de entrada. Se remite a la lógica de negocio a un objeto apropiado del modelo. El objeto del modelo se comunica con la fuente de datos y hace todo lo indicado por la solicitud así como recopilar información para la respuesta. Cuando se ha hecho todo esto, retorna al controlador de entrada. Este último decide que vista es la más apropiada y le pasa el control remitiéndole los datos necesarios a través de algún tipo de objeto de sesión HTTP compartido.

3.1 Patrones Vista

Respecto a la vista hay tres patrones: Transform View, Template View y Two Step View. Los patrones básicos para Transform View y Template View suponen una única etapa. Two Step View es una variación que puede aplicarse tanto a Transform View como a Template View. Template View permite escribir la presentación en la estructura de la página e incrustar marcadores en la página para indicar dónde incluir el contenido dinámico. Es el patrón habitual para las páginas de servidor. Facilita el desarrollo pero provoca un código desordenado que se hace difícil de mantener.

Transform View utiliza algún tipo de lenguaje de transformación, normalmente XSLT que es muy efectivo para datos XML. El controlador de entrada selecciona la transformación XSLT apropiada y la aplica al XML generado a partir del modelo. Proporciona mucha flexibilidad, siendo fácil de generar vista p ara diferentes presentaciones o dispositivos, aunque representa un mayor grado de complejidad. La segunda decisión es si se utiliza un patrón basado en una única etapa o en dos. Los patrones de una sola etapa normalmente tienen un componente de vista para cada pantalla de la aplicación. La vista recibe datos basados en el dominio y los muestra como HTML.

Una patrón vista en dos fases, Two Step View, subdivide el proceso en dos pasos, el primero produce una pantalla lógica a partir de los datos del dominio y luego, el segundo, la presentación.

La ventaja de Two Step View es que se pone la decisión de que HTML generar en un sólo lugar, esto hace que los cambios globales al HTML sean más fáciles. Es apropiada para sitios donde diferentes pantallas comparten el mismo diseño básico.

3.2 Patrones Controlador de Entrada

Hay dos patrones para el controlador de entrada. El caso más común es tener un objeto controlador de entrada para cada página del sitio web, Page Contraller. Normalmente habrá un Page Controller para cada acción: un botón o enlace. Cada controlador de entrada tiene que hacer frente a dos responsabilidades: atender las solicitudes HTTP y decidir qué hacer con ellas. La aproximación del otro patrón controlador, Front Controller, es separar estas dos responsabilidades. Con el Front Controller hay un único objeto que recibe todas las

peticiones y crea un objeto separado para procesar la solicitud. Esto permite centralizar toda la gestión de las solicitudes HTTP en un único objeto.

4 Representación de una Base de Datos Relacional

Cuando se utilizan objetos el problema es como interactuar con bases de datos relacionales, dos estructuras de datos bastante diferentes. Además, los objetos están en la memoria y es necesario coordinarlos con lo que está sucediendo en el disco, la base de datos.

Con una base de datos orientada a objetos no hay que preocuparse de mapeos. Esto mejora la productividad, ya que el mapeo supone alrededor de un tercio del esfuerzo de programación y mantenimiento. Sin embargo, la mayoría de los proyectos no utilizan bases de datos OO debido al riesgo de esta tecnología todavía no suficientemente madura en comparación a las BBDD relacionales.

4.1 Patrones Arquitectónicos

Los patrones arquitectónicos para el mapeo a base de datos relacional son principalmente Active Record, Row Data Gateway, Table Data Gateway y Data Mapper. Row Data Gateway y Table Data Gateway se basan ambos en el patrón básico Gateway, una clase en memoria que mapea a una tabla en la base de datos y tiene un campo por columna de la base de datos. Los patrones basados en Gateway contienen todo el código de mapeo a la base de datos pero no código de la lógica de dominio.

La elección entre Row Data Gateway y Table Data Gateway depende de la lógica de dominio y también del soporte para el patrón básico Record Set que proporciona la plataforma utilizada para el desarrollo de la aplicación. Las plataformas de Microsoft, por ejemplo, utilizan Record Set ampliamente. Gran parte del funcionamiento de la interfaz de usuario se basa en manipular

Record Sets. Por lo tanto, en este tipo de plataformas la opción natural es Table Data Gateway. Por el contrario, Row Data Gateway es la mejor elección cuando los Result Sets son más difíciles de manipular porque no tienen tan buen soporte en la plataforma. En este caso lo que se obtiene es un objeto por cada registro en la fuente de datos.

Si la aplicación tiene un modelo de dominio que se parece mucho al modelo de la base de datos, entonces es mejor considerar Active Record, que combina el Gateway y el objeto de dominio en una sola clase que incluye tanto código de acceso a la BB.DD. como la lógica empresarial. Pese a todo, es recomendable mantener estas responsabilidades separadas.

La tercera opción es Data Mapper. Este es el patrón más complejo, pero también el más flexible. Se necesita una inversión de desarrollo pero permite que los objetos del dominio sean totalmente independientes de la base de datos. En cualquier caso, a medida que el mapeo se complica se hace recomendable no desarrollar el mapeo nosotros mismos y reutilizar herramientas existentes de mapeo Objecto-Relacional.

4.2 Concurrencia

Se trata del problema de cómo conseguir que los diversos objetos utilizados se carguen y salven en la base de datos. Por ejemplo, utilizando Data Mapper, los objetos del dominio no pueden hacer llamadas a la capa de mapeo ya que es totalmente transparente para ellos. Además, hay que seguirla pista de cada objeto modificado y asegurarse de escribir todos los cambios en la BB.DD.

Además, hay el problema de la concurrencia, asegurar que ningún otro proceso cambia cualquiera de los objetos que se han leído mientras se trabaja con ellos. Un patrón para resolver estos problemas es Unit of Work. Este patrón mantiene un registro de todos los objetos leídos desde la base de datos, junto con todos los modificados. También maneja cómo las actualizaciones se escriben en la base de datos. Sin Unit of Work, la capa de dominio es que debe decidir cuándo leer y escribir en la base de datos. Unit of Work resulta de extraer el comportamiento de control del mapeo de datos a un objeto específico para esta funcionalidad.

4.3 Lectura de Datos

Es conveniente encapsular el SQL en las clases de mapeo objeto-relacional correspondientes. Los métodos de búsqueda encapsulan comandos SQL "SELECT" con una API, por ejemplo "buscar(id)" o "buscarCliente(idCliente)" encapsularían las consultas SQL correspondientes para recuperar un objeto con identificador id o un cliente con idCliente.

Es recomendable utilizar un patrón Identity Map para seguir la pista de lo que se ha leído para evitar hacerlo dos veces. Si se está utilizando Unit of Work, el lugar obvio para mantener el Identity Map es el objeto correspondiente al Unit of Work.

4.4 Patrones de Representación de Estructuras de Datos

Estos patrones describen cómo los datos en una base de datos relacional se mapean a datos en objetos. El problema surge de la diferente manera en que los objetos y las tablas manejan los enlaces. Los objetos manejan los enlaces almacenando referencias y las tablas mediante una clave apuntando a otra tabla. El segundo tema es que los objetos pueden hacer fácilmente uso de colecciones para manejar múltiples referencias contenidas en un sólo campo, mientras que la normalización fuerza que todos los campos de base de datos tengan un único valor. La solución es mantener la identidad relacional de cada objeto como un campo identidad, Identity Field, en el objeto, y utilizarla para mapear entre las referencias a objetos y las claves relacionales.

Cuando se leen objetos, cada vez que se encuentra una clave foránea, se mira en el Identity Map para establecer las referencias entre objetos. Si la clave no está en el Identity Map, entonces se va a la base de datos para cargarlo. Al guardar un objeto, se salva en la fila con la clave apropiada. Cualquier referencia entre objetos se sustituye por Identity Field del objeto referenciado. Si un objeto tiene una colección, hay que lanzar otra consulta para encontrar todas las filas que enlazan con el identificador del objeto que tiene la colección. Salvar dicho objeto implica guardar cada objeto en la colección y asegurarse de que tiene una clave foránea al objeto de la colección.

4.5 Herencia

Las jerarquías de clases en OO vienen definidas por la relación de herencia. En las bases de datos relacionales no hay una forma estándar para representar la herencia, por lo que hay que realizar u n mapeo al almacenarla en la base de datos.

4.6 Representación de Metadatos

El enfoque más sofisticado de mapeo es utilizando Metadata Mapping, que se basa en escribirlos detalles del mapeo en un archivo de metadatos, datos sobre los datos. Los detalles del mapeo son cómo las columnas de la base de datos se corresponden con los

campos en los objetos. Los metadatos permiten tener que escribir código repetitivo utilizando técnicas de generación de código a partir de los metadatos o mediante programación reflexiva (el código puede modificarse a sí mismo).

5 Concurrencia

Siempre que tengamos múltiples procesos o hilos manipulando los mismos datos, existe el riesgo de problemas de concurrencia. Las transacciones proporcionan un marco que ayuda a evitar muchos de los aspectos más complicados de concurrencia. Siempre y cuando se haga toda la manipulación de datos dentro de una transacción. Desafortunadamente, hay muchas interacciones con un sistema que no se pueden mantener dentro de una sola transacción de base de datos. En estos casos hay que aplicar concurrencia offline, es decir control de la concurrencia para datos manipulados durante múltiples transacciones de base de datos. Además, hay la concurrencia del servidor de aplicaciones: dar soporte a múltiples hilos de ejecución en un servidor de aplicaciones.

5.1 Problemas de Concurrencia

Los principales problemas que los sistemas de control de la concurrencia intentar evitar son:

- Actualizaciones perdidas: el usuario A edita un archivo para hacer algunos cambios. Mientras, el usuario B realiza un pequeño cambio en el mismo archivo de manera que termina antes que A. Desafortunadamente, cuando A leyó el archivo no incluía los cambios de B, así que cuando A acaba y guarda sus cambios, sobrescribe la versión de B de forma que su cambio se pierde.
- Lecturas inconsistentes: ocurren cuando se leen dos datos que eran correctos por separado, pero no eran correctos en el mismo momento. El usuario A quiere saber cuántos archivos hay en una carpeta que contiene dos subcarpetas. A mira en la primera subcarpeta y ve 7 archivos. Antes de mirar en la otra subcarpeta, el usuario B añade 2 ficheros en la primera subcarpeta y 3 a los 5 que había en la segunda. A mira ahora en la segunda subcarpeta y cuenta 8 archivos. Para él el total de archivos de la carpeta es 15.

Lamentablemente 15 no ha sido en ningún momento la respuesta correcta. El recuento correcto era 12 antes de los cambios del usuario B y 17 después. Cualquiera de las dos respuestas hubiese sido correcta, incluso si no era el valor en ese momento. Para evitar estos problemas se utilizan mecanismos de control de la concurrencia.

5.2 Contextos de Ejecución

Desde el punto de vista de la interacción de la aplicación con el exterior, los contextos de ejecución más importantes son la solicitud (request) y la sesión (session). La solicitud corresponde a una llamada desde el exterior que se procesa en la aplicación, que opcionalmente envía de vuelta una respuesta. Durante una solicitud el procesamiento se produce en gran medida en los servidores y se supone que el cliente espera la respuesta. Una sesión es una interacción entre un cliente y un servidor a lo largo de un periodo de tiempo. Una sesión puede constar de una sola solicitud, pero comúnmente consta de una serie de peticiones, una serie que el usuario considera como una secuencia lógica coherente. Normalmente comenzará con la autenticación del usuario (log in) y terminará cuando el usuario sale de su cuenta (log out).

Otro contexto de ejecución, éste ligado al procesamiento y no a la interacción, es el proceso. Un proceso es un contexto de ejecución denominado pesado ya que proporciona aislamiento para los datos que se procesan internamente. Por el contrario, los hilos (threads) son contextos de ejecución más ligeros que los procesos, de hecho un proceso puede dar soporte a múltiples hilos. Los hilos permiten soportar múltiples flujos de ejecución desde un único proceso, pero los hilos no proporcionan aislamiento para los datos que procesan y comparten el mismo espacio de memoria, lo que puede provocar problemas de concurrencia. Finalmente, otro concepto relacionado con los contextos de ejecución, cuando se trata con bases de datos, son las transacciones, que permiten agrupar varias solicitudes del cliente y tratarlas como si todo el procesamiento se hiciese en una único solicitud, evitando así muchos problemas de concurrencia.

5.3 Aislamiento

Los problemas de concurrencia tienen su origen en que más de un agente activo, proceso o hilo, tiene acceso al mismo dato. El aislamiento es una forma de evitar estos problemas: repartir el acceso a los datos de manera que cualquier dato sólo pueda ser accedido por un

agente activo. Así es como los procesos trabajan en la memoria del sistema operativo. De forma similar existen los bloqueos de archivos que evitan que más de un usuario abra simultáneamente un mismo archivo para modificarlo.

5.4 Control de la Concurrencia Optimista y Pesimista

Cuando tenemos datos que pueden ser modificador pero no es posible aislarlos, ya que son compartidos, es necesario aplicar medidas de control de la concurrencia. Existen dos opciones: las medidas optimistas y las pesimistas. Partamos del escenario en que dos usuarios quieren editar el mismo archivo al mismo tiempo. Con un bloqueo optimista, ambos pueden hacer una copia del archivo y editarlo libremente. El control de concurrencia entra en juego cuando el segundo usuario intenta guardar sus cambios. Se detecta el conflicto entre ambos cambios, se le avisa al segundo usuario que no es posible guardar sus cambios y es entonces responsabilidad del segundo usuario ver cómo puede evitar perder sus cambios. Con una aproximación pesimista, quien primero bloquea el archivo para poder modificarlo impide que nadie más pueda editarlo.

El bloqueo optimista trata de detectar el conflicto, mientras que el pesimista bloquea los recursos para prevenirlo. Si los conflictos son suficientemente raros, o las consecuencias no son críticas, se recomienda utilizar el optimista ya que permite un mayor grado de concurrencia. Es decir, es posible que más usuarios puedan trabajar en paralelo sin tener que esperar a que los recursos se vayan liberando, consiguiéndose por lo tanto un mayor rendimiento.

5.5 Prevención de Lecturas Inconsistentes

Se pueden evitar las lecturas inconsistentes mediante bloqueos pesimistas. Para leer un dato se necesita un bloqueo de lectura (compartido) y para escribirlo se necesita un bloqueo de escritura (exclusivo). Muchos usuarios pueden tener el bloqueo de lectura a la vez ya que es compartido, pero si alguien tiene el de lectura nadie puede escribir. Por otro lado, una vez que alguien tiene el bloqueo de escritura, entonces nadie puede leer ni escribir.

5.6 Punto Muerto

Un problema de las técnicas pesimistas es llegar a un punto muerto que impida seguir el procesamiento. Si dos usuarios están editando diferentes archivos, por lo que están

bloqueados, y para completar sus tareas necesitan ambos editar el archivo que está siendo bloqueado por el otro usuario, entonces han llegado a un punto muerto desde el que no pueden seguir trabajando. Varias técnicas intentan evitar los puntos muertos. Una es tener un software que detecte los puntos muertos y seleccione a una víctima que deberá entonces liberar todos los recursos que tiene bloqueados. Una aproximación similar es dar una caducidad a cada bloqueo, aunque lamentablemente entra en juego incluso cuando no se ha llegado a ningún punto muerto todavía. Otra forma de evitar puntos muertos es que los usuarios adquieran todos sus bloqueos a la vez al comienzo de su trabajo, y una vez que los tienen impedir que puedan bloquear más recursos. Los bloqueos se consiguen siguiendo un orden común para todos los usuarios, por ejemplo para bloquear archivos se sigue un orden alfabético según sus nombres.

5.7 Transacciones

La principal herramienta para el control de la concurrencia en aplicaciones empresariales es la transacción. Una transacción es una secuencia de acciones delimitada de manera clara por un punto inicial y uno final. Todos los recursos implicados están en un estado consistente tanto cuando la transacción comienza y como cuando termina. Además, cada transacción debe completarse totalmente, todas las acciones realizadas, o ninguna acción debe tener efecto. Es decir, o todo o nada. Las propiedades de una transacción son:

- Atomicidad: cada paso en la secuencia de acciones de la transacción debe completarse satisfactoriamente o todo lo realizado hasta el momento debe deshacerse.
- Consistencia: los recursos del sistema deben estar en un estado consistente tanto al principio como al final de la transacción.
- Aislamiento: el resultado de una transacción no deber ser visible para el resto de transacciones hasta que no se haya completado con éxito.
- Persistencia: el resultado de una transacción exitosa debe ser permanente. Los recursos gobernados por las transacciones suelen ser bases de datos, el caso más común, aunque también pueden ser colas de mensajes, impresoras, etc.

Para conseguir el mayor rendimiento, las transacciones deben ser tan cortas como sea posible. Se recomienda que las transacciones no abarquen varias solicitudes. Normalmente se inicia la transacción al comienzo de la solicitud y se finaliza al final de la solicitud.

5.8 Transacciones de Negocio y de Sistema

Las transacciones de sistema están soportadas por bases de datos y monitores de transacciones. Se componen de un grupo de comandos SQL delimitados por instrucciones para abrir y cerrar la transacción. Para el usuario de un sistema de banca en línea una transacción se compone de los pasos: seleccionar una cuenta, establecer algunos pagos de facturas y finalmente hacer click en el botón "Aceptar"' para realizar los cargos en las cuentas. Esto es lo que llamamos una transacción de negocio.

La manera obvia para hacer que una transacción de negocio tenga las mismas propiedades que una de sistema (atomicidad, consistencia, aislamiento y persistencia) es ejecutarla toda dentro de una transacción de sistema. El problema es que las transacciones de negocio suelen abarcar varias solicitudes. Así que, si se quiere tener una aplicación con capacidad de respuesta, es necesario romperla transacción de negocio en una serie de transacciones de sistema cortas. Si no se hiciese así, como puede pasar tiempo entre solicitud y solicitud, la aplicación podría llegar a quedar bloqueada debido a la existencia de muchos recursos bloqueados por diferentes transacciones de negocio.

Atomicidad y persistencia son las propiedades más fáciles de conseguir en las transacciones de negocio. Ambas se basan en aplicar los cambios de la transacción, cuando el usuario pulsa "Aceptar", dentro de una transacción de sistema. El problema en este caso es mantener un registro detallado de los cambios realizados durante la transacción de negocio. Si la aplicación utiliza un Domain Model se puede utilizar Unit of Work para monitorizarlos cambios.

El problema es conseguir el aislamiento entre transacciones de negocio. Los fallos en el aislamiento conducen a problemas de consistencia. Dentro de una transacción la responsabilidad de la aplicación para mantener la consistencia es obligar el cumplimiento de todas las reglas de negocio. Cuando entran en juego múltiples transacciones, la

responsabilidad es asegurar que una sesión no sobrescriba los cambios realizados por otras sesiones.

5.9 Patrones para el Control de la Concurrencia Offline

Estos patrones son algunas técnicas que pueden resultar útiles para tratar con la concurrencia que abarca más que una transacción de sistema. Son sólo necesarias si no se pueden enmarcar todas las transacciones de negocio dentro de una sola solicitud, por lo tanto tampoco dentro de una transacción de sistema. Optimistic Offline Lock es un patrón que usa control de concurrencia optimista en las transacciones de negocio. Es fácil de programa y proporciona una buena capacidad de respuesta.

Su limitación es que sólo se descubre que una transacción de negocio va a fallar cuando se intentan acometer la transacción así que todo el trabajo realizado puede perderse. La alternativa es Pessimistic Offline Lock, ya que no se llega a la situación anterior aunque es más difícil de programar y reduce la capacidad de respuesta de la aplicación. Con cualquiera de las aproximaciones previas, se puede reducir enormemente la complejidad de desarrollo si no se intenta gestionarlos bloqueos a nivel de cada objeto individual. Es mejor operar a un nivel de detalle menor y gestionarla concurrencia para grupos de objetos.

5.10 Concurrencia a Nivel de Servidor de Aplicaciones

Hasta ahora se ha hablado de concurrencia de múltiples sesiones operando sobre una fuente de datos compartida. Otra forma de concurrencia es a nivel del servidor de aplicaciones. Cómo maneja dicho servidor múltiples solicitudes simultáneas. La mayor diferencia es que la concurrencia en el servidor de aplicaciones no involucra transacciones.

La forma más sencilla de manejar este caso es usar un proceso por sesión, cada sesión se ejecuta en su propio proceso. La gran ventaja es que el estado de cada proceso está totalmente aislado de los otros procesos. El problema es que se utiliza muchos recursos de servidor, cada proceso supone un gran consumo de memoria. Para ser más eficiente se puede utilizar un fondo común de procesos (process pool). Se trata de un conjunto de procesos fijo que se van turnando para ir atendiendo las solicitudes que van llegando al servidor de aplicaciones. De esta manera cada proceso se ocupa de una solicitud cada vez, pero puede manejar múltiples solicitudes de sesiones diferentes en secuencia.

Anexo 2: Modelo del cuestionario aplicado a los usuarios

Estimado Usuario:

El presente instrumento tiene como finalidad determinar los factores que influyen en la facilidad con que las personas pueden utilizar un sistema para alcanzar un objetivo concreto, en este caso el sistema con el que Ud. trabaja diariamente. La información recabada tendrá relevancia dentro del desarrollo de una Tesis de Maestría en Sistemas de Información. Todos los cuestionarios tendrán un carácter anónimo y sus resultados serán presentados de una forma global y consolidada, garantizando así, la confidencialidad. Agradeciendo su colaboración.

El Investigador

Instrucciones: A continuación se presentan una serie de preguntas que son presentadas de tal manera que sólo admiten las respuestas presentadas. Marque con una "X" una sola respuesta, aquella que más se aproxime a su opinión sobre cada enunciado, en este sentido, lo más importante es la honestidad, concreción y objetividad al contestar cada uno de los enunciados planteados.

Muchas gracias por su atención

1.- Porcentaje de elementos funcionales utilizados sin error

☐ Es menor que el 25% ☐ Mayor al 25% y menor que 50% ☐ Mayor al 50% y menor que 75% ☐ Mayor que 75%

2.- Porcentaje de objetivos logrados cuando utiliza el sistema

☐ Es menor que el 25% ☐ Mayor al 25% y menor que 50% ☐ Mayor al 50% y menor que 75% ☐ Mayor que 75%

3.- Promedio de precisión de tareas completadas con el sistema

☐ Es menor que el 25% ☐ Mayor al 25% y menor que 50% ☐ Mayor al 50% y menor que 75% ☐ Mayor que 75%

4.- Tiempo promedio para completar una tarea

☐ 1 minuto o menos ☐ 2 minutos a 5 minutos ☐ de 10 a 20 minutos ☐ más de 20 minutos

5.- Porcentaje de tareas completadas en un día

☐ Es menor que el 25% ☐ Mayor al 25% y menor que 50% ☐ Mayor al 50% y menor que 75% ☐ Mayor que 75%

6.- Porcentaje de tareas completadas en un día (sin problemas).

☐ Es menor que el 25% ☐ Mayor al 25% y menor que 50% ☐ Mayor al 50% y menor que 75% ☐ Mayor que 75%

7.- *Existen mensajes de advertencia ante operaciones o comandos peligrosos.*

☐ si ☐ no

8.- Existen mensajes de advertencia ante operaciones irreversibles.

☐ si ☐ no

9.- Errores detectados por el sistema

☐ Es menor que el 25% ☐ Mayor al 25% y menor que 50% ☐ Mayor al 50% y menor que 75% ☐ Mayor que 75%

10.- El sistema posee la posibilidad de recuperarse luego de un error

☐ Si ☐ No

11.- Tiempo en que un usuario adquiere las destrezas necesarias para utilizar el sistema

☐ 6 días o menos ☐ 7 - 14 días ☐ 15 - 30 días ☐ 2 meses o mas

12.- Tiempo en que un usuario adquiere las destrezas necesarias para utilizar el sistema de forma experta

☐ 6 días o menos ☐ 7 - 14 días ☐ 15 - 30 días ☐ 2 meses o mas

13.- Valor real invertido en la adquisición del producto

- [ ] Es menor que el 25% del valor deseado
- [ ] Es mayor que 25% y menor que el 50% del valor deseado
- [ ] Mayor al 50% y menor que 75% del valor deseado
- [ ] Igual al valor deseado
- [ ] Mayor al valor deseado

14.- Valor real invertido en el mantenimiento del producto

- [ ] Es menor que el 25% del valor deseado
- [ ] Es mayor que 25% y menor que el 50% del valor deseado
- [ ] Mayor al 50% y menor que 75% del valor deseado
- [ ] Igual al valor deseado
- [ ] Mayor al valor deseado

15.- Valor real invertido en el soporte del producto

- [ ] Es menor que el 25% del valor deseado
- [ ] Es mayor que 25% y menor que el 50% del valor deseado
- [ ] Mayor al 50% y menor que 75% del valor deseado
- [ ] Igual al valor deseado
- [ ] Mayor al valor deseado

16.- Numero de términos técnicos utilizados en el elemento funcional.

- [ ] Es menor que el 25%
- [ ] Mayor al 25% y menor que 50%
- [ ] Mayor al 50% y menor que 75%
- [ ] Es mayor que el 75%

17.- Porcentaje de elementos funcionales con nombres relacionados a su funcionalidad.

- [ ] Es menor que el 25%
- [ ] Mayor al 25% y menor que 50%
- [ ] Mayor al 50% y menor que 75%
- [ ] Es mayor que el 75%

18.- Porcentaje de interfaces y métodos descritos en los manuales.

- [ ] Es menor que el 25%
- [ ] Mayor al 25% y menor que 50%
- [ ] Mayor al 50% y menor que 75%
- [ ] Es mayor que el 75%

19.- Porcentaje de elementos funcionales utilizados correctamente luego de leer el manual.

- [ ] Es menor que el 25%
- [ ] Mayor al 25% y menor que 50%
- [ ] Mayor al 50% y menor que 75%
- [ ] Es mayor que el 75%

20.- Porcentaje de elementos familiares en los elementos funcionales.

- [ ] Es menor que el 25%
- [ ] Mayor al 25% y menor que 50%
- [ ] Mayor al 50% y menor que 75%
- [ ] Es mayor que el 75%

21.- Existen estándares en los documentos.

- [ ] siempre
- [ ] algunas veces
- [ ] nunca

22.- Es versátil la navegación entre pantallas.

- [ ] Excelente
- [ ] Buena
- [ ] Promedio
- [ ] Debajo del promedio
- [ ] Inaceptable

Anexo 3: Validación de Instrumento de investigación

Caracas, 2013

Presente.-

Constancia de Validación del Instrumento de investigación.

Yo, **Pedro Bonillo Ramos**, titular de la Cedula de Identidad Nro. **10.868.538**, de profesión **PHD Computer Science Software Engineering**, ejerciendo actualmente como **Tecnical and finance Project Management Control**, en la Institución: **CANTV**.

Por medio de la presente hago constar que he revisado y validado a los fines de su aplicación el instrumento (cuestionario), para el trabajo de grado que lleva por título: **Modelo Para La Optimización Del Diseño De Arquitecturas De Software En Aplicaciones Empresariales Con Base En La Usabilidad En Las PYMES Consultoras Venezolanas,** realizado para optar al título de Magister en Sistemas de Información, en la Universidad Católica Andrés Bello.

En Caracas, a los 30 días del mes de Octubre de 2013

Bonillo Ramos Pedro

Firma

Caracas, 2013

Presente.-

Constancia de Validación del Instrumento de investigación.

Yo, **Michael Fung Ng**, titular de la Cedula de Identidad Nro. **16.218.809**, de profesión **Ingeniero Informático**, ejerciendo actualmente como **Analista de Tendencias de Pérdidas**, en la Institución: **Banesco Banco Universal.**

Por medio de la presente hago constar que he revisado y validado a los fines de su aplicación el instrumento (cuestionario), para el trabajo de grado que lleva por título: **Modelo Para La Optimización Del Diseño De Arquitecturas De Software En Aplicaciones Empresariales Con Base En La Usabilidad En Las PYMES Consultoras Venezolanas,** realizado para optar al título de Magister en Sistemas de Información, en la Universidad Católica Andrés Bello.

En Caracas, a los 28 días del mes de Octubre de 2013

Firma

Caracas, 2013

Presente.-

Constancia de Validación del Instrumento de investigación.

Yo, **Luis David Farias Pino**, titular de la Cedula de Identidad Nro. **16.662.061**, de profesión **Ingeniero Informático**, ejerciendo actualmente **Analista Desarrollador Web**, en la Institución: **Parodi Solutions C.A.**

Por medio de la presente hago constar que he revisado y validado a los fines de su aplicación el instrumento (cuestionario), para el trabajo de grado que lleva por título: **Modelo Para La Optimización Del Diseño De Arquitecturas De Software En Aplicaciones Empresariales Con Base En La Usabilidad En Las PYMES Consultoras Venezolanas,** realizado para optar al título de Magister en Sistemas de Información, en la Universidad Católica Andrés Bello.

En Caracas, a los 28 días del mes de Octubre de 2013

Firma

Anexo 4 Documento de Arquitectura de Software del Caso de estudio "Resort Around The World"

PROYECTO:

"Resort Around The World"

Documento de Arquitectura de Software

Historial de Revisiones

Versión	Fecha	Autor	Descripción
1.0.0	*dd-mm-aaaa*	*Nombre y apellido*	*Especificaciones*
1.0.1	09-02-2013	Emily Uriola.	Documento de Arquitectura de Software
1.0.2	19-02-2013	Luis Farias	Documento de Arquitectura de Software

Índice de Contenido

Selección de puntos de vista de la arquitectura 116
Visión general 116
Vista de Lógica 116
Vista de Escenarios 117
Vista de Procesos 118
Diagramas de Secuencias 119
Vista Física 121
Diagrama Despliegue 122

Objetivo del Documento

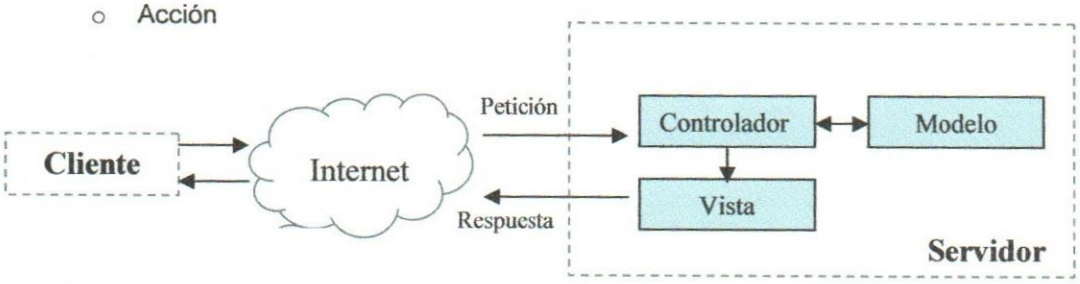
El documento de la arquitectura del Software proporciona una descripción arquitectónica comprensible para el proyecto Resort Around The World, usando vistas arquitectónicas para representar diversos aspectos del sistema. Este sirve como medio de comunicación entre los miembros del equipo del proyecto con relación a las decisiones arquitectónicas significativas que se han tomado con respecto al diseño del proyecto

Definiciones de Arquitectura

El modelo utilizado por la compañía para llevar a cabo este proyecto, es el patrón Modelo, Vista, Controlador o MVC el cual busca agrupar los componentes de la aplicación en tres niveles lógicos (ver Figura 1), y para la ejecución del mismo se utilizó el framework Symfony, diseñado para optimizar el desarrollo de las aplicaciones web basado en este mismo patrón. Este framework separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web, además proporciona varias herramientas y clases que ayudan reducir el tiempo de desarrollo y promueven estándares de codificación. También, automatiza las tareas más comunes permitiendo disminuir su tiempo de implementación.

El sistema se encontrara integrado por los siguientes componentes:

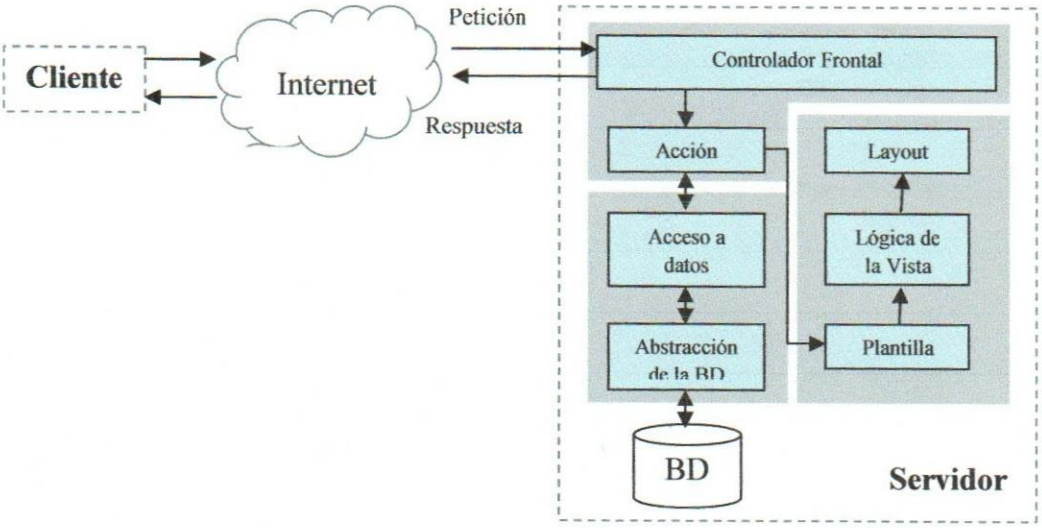
- La capa del Modelo
  - Abstracción de la base de datos
  - Acceso a los datos
- La capa de la Vista
  - Vista
  - Plantilla
  - Layout
- La capa del Controlador
  - Controlador frontal
  - Acción



Visualización Patrón Modelo-Vista-Controlador en sitios Web

En primer lugar, el controlador frontal y el layout son comunes para todas las acciones de la aplicación. Se pueden tener varios controladores y varios layouts, pero solamente es obligatorio tener uno de cada. El controlador frontal es un componente que sólo tiene código relativo al MVC, por lo que no es necesario crear uno, ya que Symfony lo genera de forma

automática. Las clases de la capa del modelo son generadas automáticamente por el framework, en función de la estructura de datos de la aplicación.



Flujo de Trabajo

Identificación de Stakeholders y sus Responsabilidades

Stakeholder	Descripción	Escenario
Administrador	Es el usuario dueño del negocio y encargado de administrar está encargado de la gestión directiva de las reservas y sitios.	• Escenario de negocios • Escenario de diseño
Arrendador	Es la persona que desea ofrecer en alquiler o venta sus propiedades turísticas.	• Escenario de negocios • Escenario de diseño
Arrendatario	Es la persona que interactúa con el sistema y hace las compras/reservaciones de sitios turísticos según su preferencia.	• Escenario de negocios • Escenario de diseño

Vistas

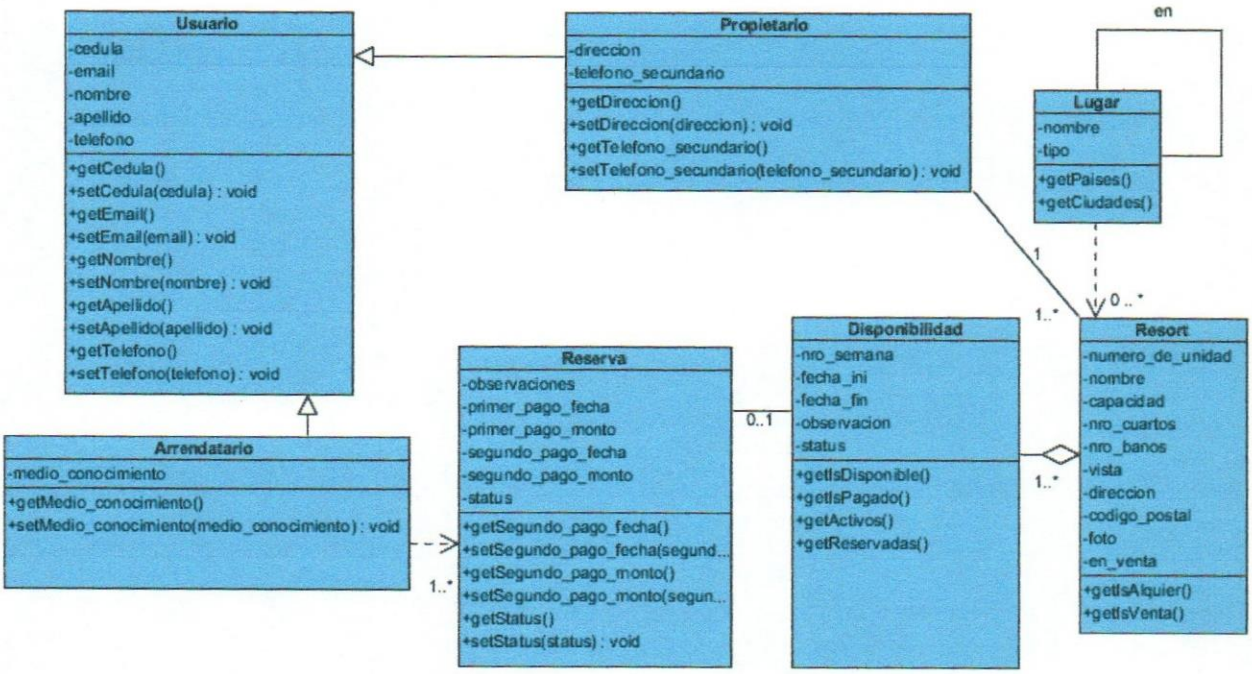
Selección de puntos de vista de la arquitectura

Vistas	UML
Escenarios	Casos de uso
Lógica	Clases
Física	Despliegue
Procesos	Secuencia

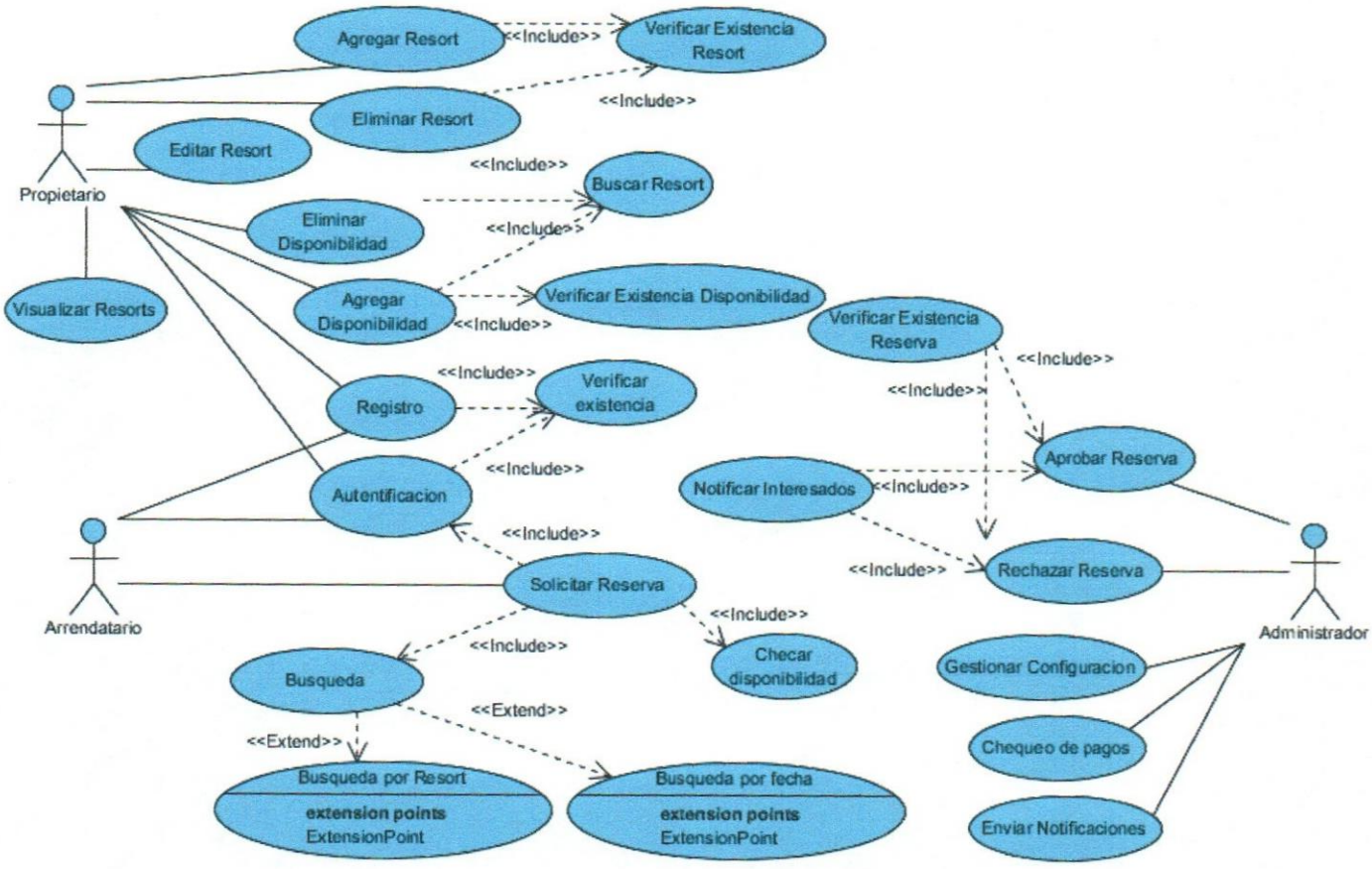
Visión general

Los principales módulos y sub-módulos de la aplicación son los siguientes:

Vista de Lógica

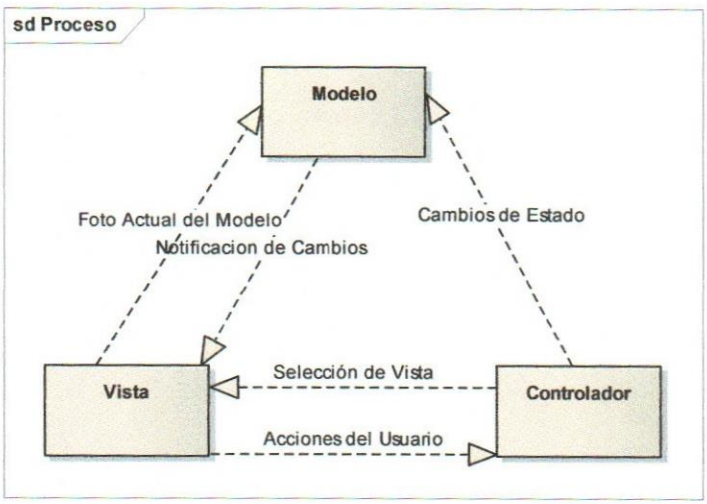


Vista de Escenarios



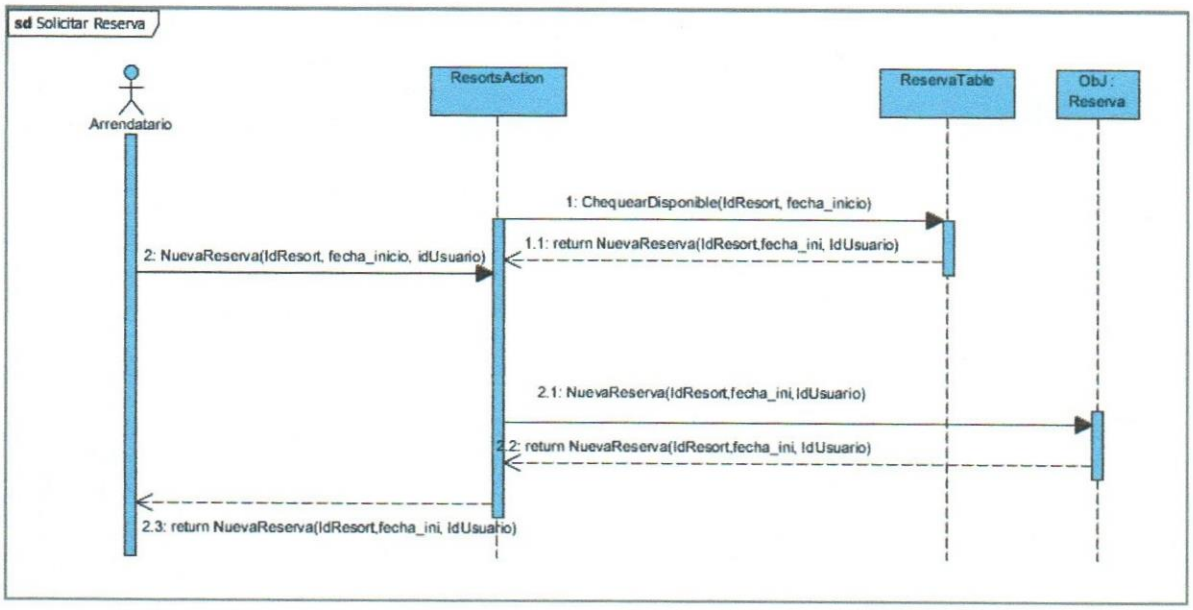
Vista de Procesos

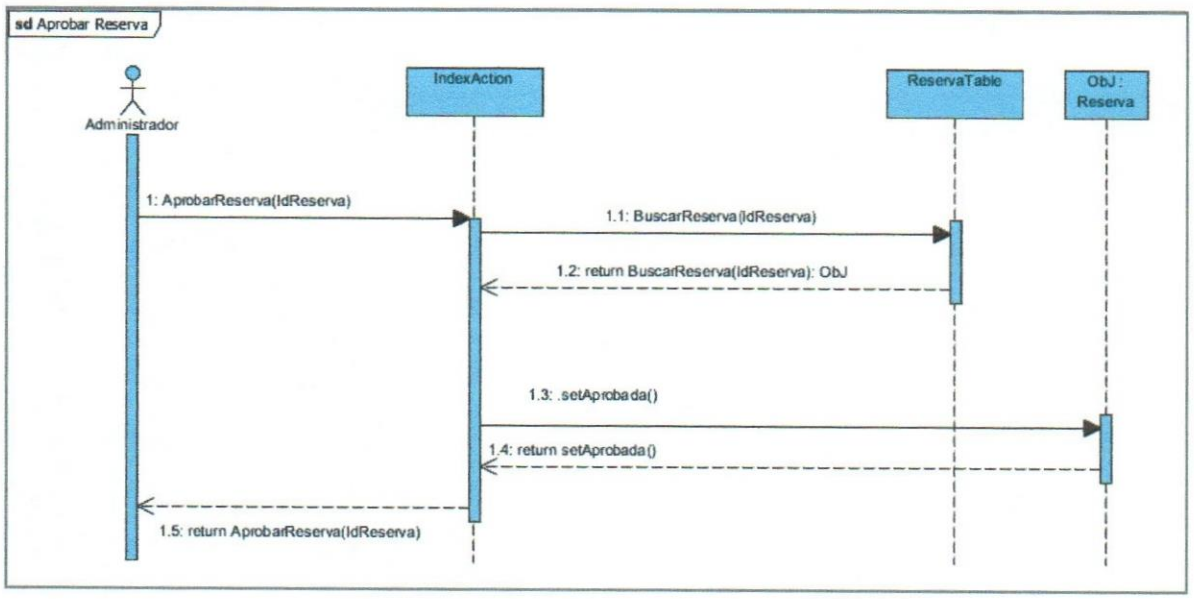
El modelo utilizado por el desarrollo involucra tres tipos de procesos, aunque no se trate de procesos realmente sino de clases que se comunican por tratarse de un desarrollo orientado a objetos. Aquellos destinados a la manipulación de los datos se los denomina "Modelos". A los que se encargan de mostrar la información al usuario se los denomina "Vistas" y finalmente, existe un nexo entre los anteriores que se denomina "Controlador".

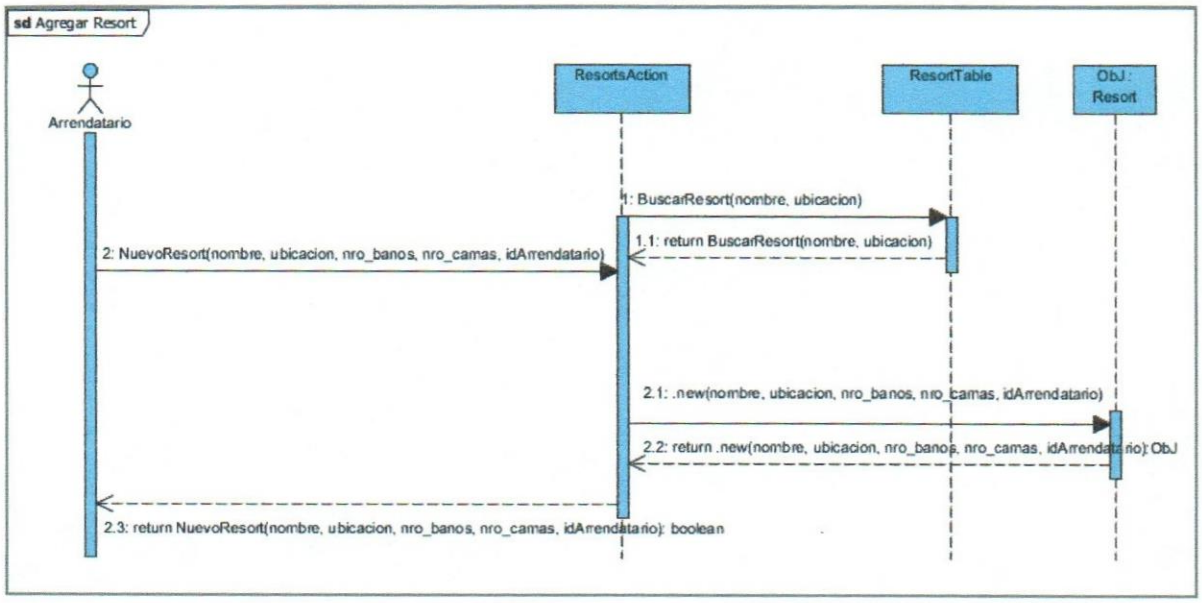


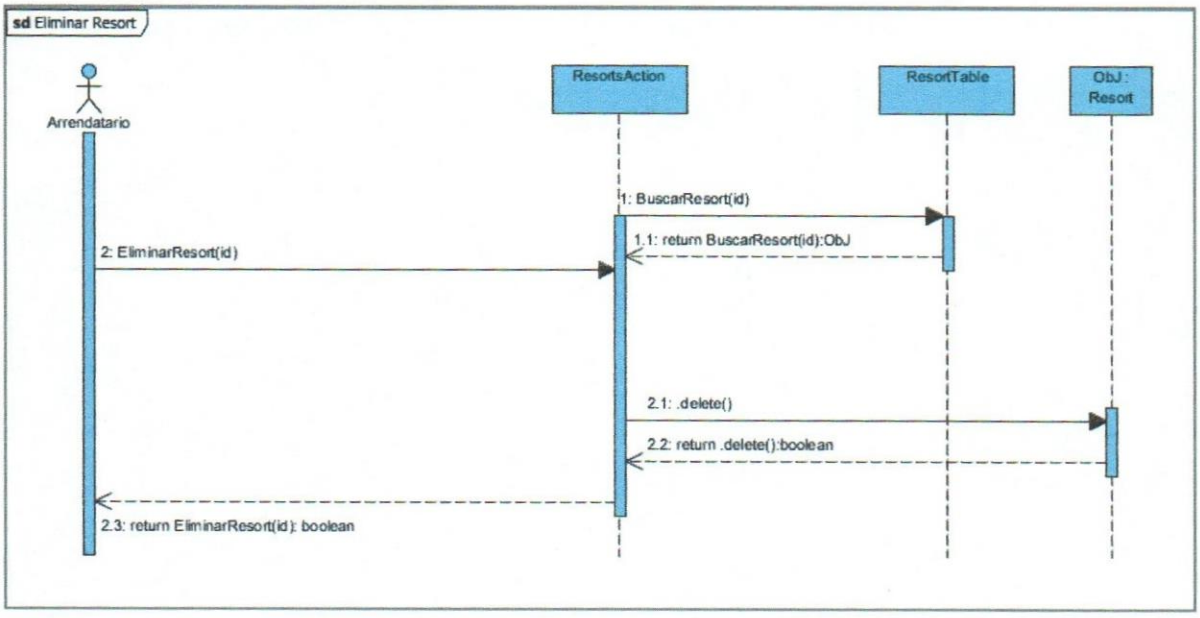
De cada proceso existen varias instancias en función de la entidad que representan, manteniendo siempre la misma lógica de comunicación entre ellos.

Diagramas de Secuencias






sd Agregar Resort
Arrendatario
ResortsAction
ResortTable
ObJ : Resort
1: BuscarResort(nombre, ubicacion)
1.1: return BuscarResort(nombre, ubicacion)
2: NuevoResort(nombre, ubicacion, nro_banos, nro_camas, idArrendatario)
2.1: .new(nombre, ubicacion, nro_banos, nro_camas, idArrendatario)
2.2: return .new(nombre, ubicacion, nro_banos, nro_camas, idArrendatario):ObJ
2.3: return NuevoResort(nombre, ubicacion, nro_banos, nro_camas, idArrendatario): boolean


sd Eliminar Resort
Arrendatario
ResortsAction
ResortTable
ObJ : Resort
1: BuscarResort(id)
1.1: return BuscarResort(id):ObJ
2: EliminarResort(id)
2.1: .delete()
2.2: return .delete():boolean
2.3: return EliminarResort(id): boolean

Vista Física

La vista de despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Aunque UML no es un lenguaje de especificación hardware de propósito general, se ha diseñado para modelar muchos de los aspectos hardware de un sistema a un nivel suficiente para que un ingeniero software pueda especificar la plataforma sobre la que se ejecuta el software del sistema.

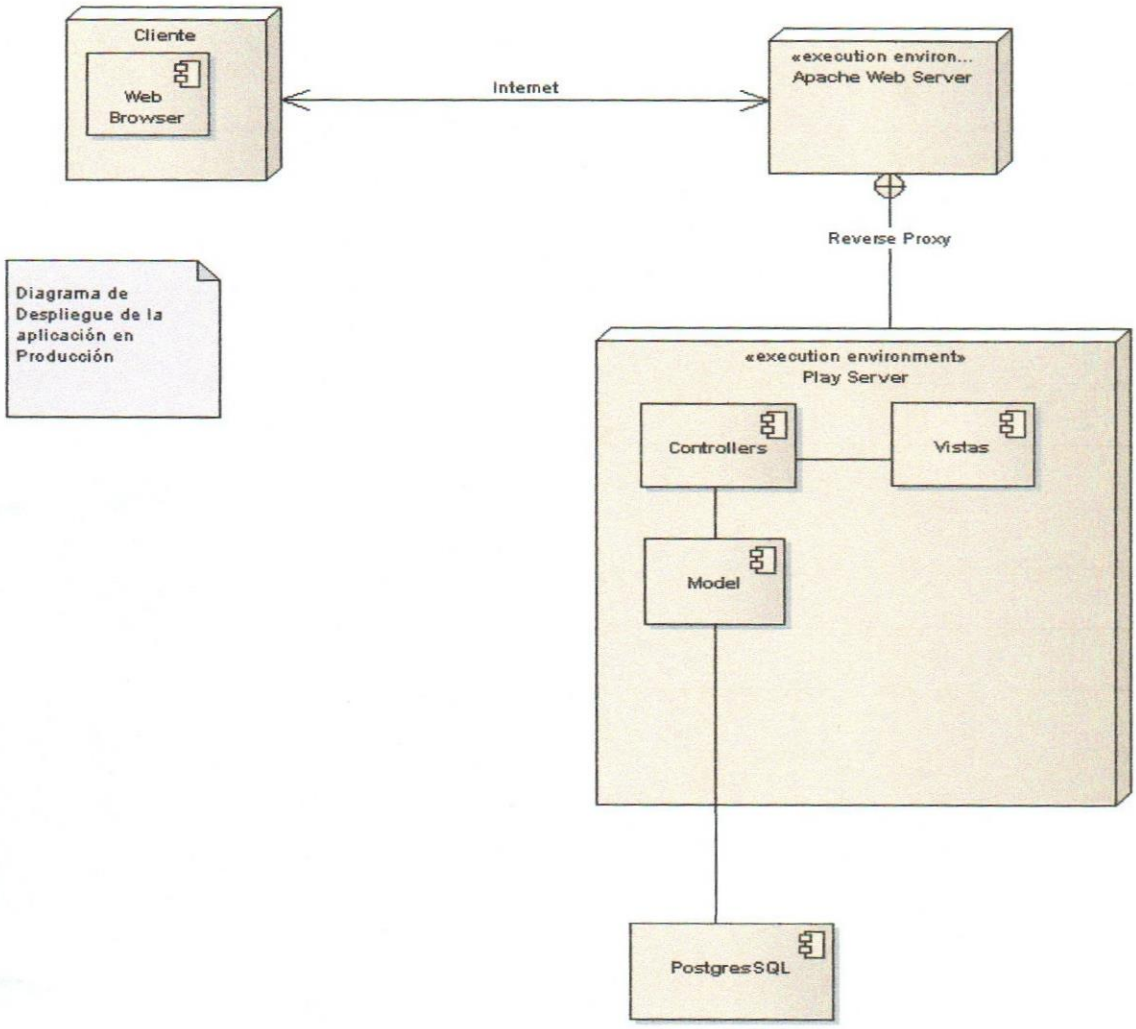
Plataforma 1 – Configuración de servidores

Descripción	Servidor con descripción de la redundancia a nivel disco y proceso: podemos hablar de un proxy que distribuye carga entre dos servers por la posibilidad que se caiga alguno.
Sistema Operativo	Ubuntu Linux
Bases de Datos	Postgre SQL 8.4.4
AppServer / WebServer	Apache 2.2.15
Lenguajes utilizados	PHP, Java, javascript, HTML, XHTML, CSS.

Plataforma 2 – Configuración de clientes

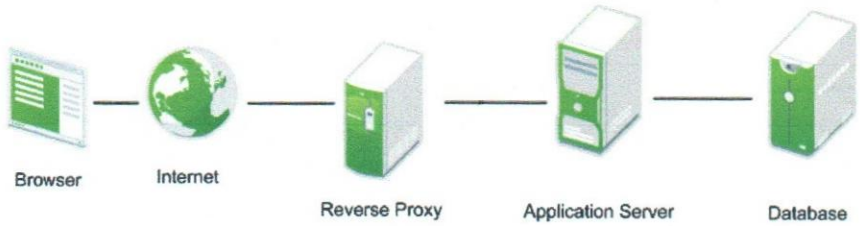
Descripción	Cliente
Sistema Operativo	Windows – Linux - MAC OS Cualquiera con una conexión a internet y el navegador Google Chrome
Bases de Datos	-
Browser	Google Chrome 26.0.1410.64 +
Lenguajes utilizados	JavaScript, HTML, XHTML, CSS

Diagrama Despliegue



Inicialmente, la arquitectura física del sistema estará compuesta por los siguientes componentes, como se puede apreciar en el diagrama:

- Application Server
- Base de Datos
- Reverse Proxy o Proxy Reverso



La petición llegará Proxy Reverso que hace las veces de cache de contenido estático. Este dejará pasar los pedidos que necesiten contenido dinámico hacia el Application Server, el cual a su vez buscará los datos en la Base de Datos en caso de ser necesario. De esta forma podemos liberar al Application Server de la carga de trabajo necesaria para mostrar contenido estático. A su vez, esta arquitectura puede ser ampliada a una más completa que puede ofrecer o redundancia o balanceo de carga mediante la multiplicación de los Application Servers y los servidores de contenido estático en caso de ser necesario debido al volumen de datos.

Anexo 5: Escenarios detectados en el Caso de estudio "Resort Around The World"

Identificación	RATWCE01	Referencia	
Escenario	Agregar Disponibilidad (consiste en agregar una nueva fecha de posible estadía válida para un resort por parte de su propietario)		
Identificación según Bass	Agregación de datos (Aggregating Data)		
Contexto	El usuario desea agregar una nueva fecha de posible estadía en uno de los resort de su propiedad una vez que ha agregado este al sistema, esta semana de estadía servirá para ser reservada por clientes potenciales y aprobada por el administrador. Dado que el propietario puede querer agregar varias semanas, es decir realizar una o más acciones en más de un objeto (resort), o también cuantas se encuentran asociadas a este y su estatus (reservada o libre).		
Propiedades Afectadas	Explicito control del usuario, Orientación		

Decisiones Arquitectónicas				
Decisiones	**S**	**R**	**M**	**N/A**
Generar un mecanismo para la agregación de múltiples disponibilidades simultáneamente en una misma pantalla.	RATWCE05-ME			

Problemas	Ponderación
La edición o eliminación de una reservación de dicha pantalla que posea el status "reservado" puede generar incongruencias	1

(valor entre el 1-10)

Peso o Ponderación Global del escenario	6 (+1)

(valor entre el 1-10)

		Leyenda
Explícito control del usuario = EU	Retroalimentación = RE	Sensibilidad = S
Minimizar la carga cognitiva = MC	*Accesibilidad = AC*	*Riesgo = R*
Mapeo Natural = MN	Manejo de errores = ME	Mejora = M
Orientación = OR	Consistencia = CO	No Afecta = N/A

Identificación	RATWCE02	Referencia	
Escenario	Agregar Nuevo Resort		
Identificación según Bass	Comprobación de la corrección (Checking for Correctness)		
Contexto	Al agregar un nuevo objeto resort se solicitan algunos datos que quizás puedan encontrarse almacenados previamente en la bd, ya que un resort puede repetirse si dos usuarios poseen acciones en el mismo y lo registran por separado.		
Propiedades Afectadas	Consistencia, Explicito control del usuario		

Decisiones Arquitectónicas				
Decisiones	**S**	**R**	**M**	**N/A**
Generar una estructura de datos que contenga los datos comunes o repetidos en el registro de todos los resorts y con esta realizar búsquedas al momento de la agregación de un nuevo resort ofreciendo a los usuarios autocompletar los datos con las coincidencias halladas.		RATWCE07 –MC	RATWCE07 – CO RATWCE07 - OR	

Problemas	Ponderación
El tiempo de respuesta del sistema	2
El consumo de recursos del sistema en caso de elevada concurrencia	4

(valor entre el 1-10)

Peso o Ponderación Global del escenario	7 (+6)

(valor entre el 1-10)

		Leyenda
Explícito control del usuario = EU	Retroalimentación = RE	Sensibilidad = S
Minimizar la carga cognitiva = MC	*Accesibilidad* = AC	*Riesgo* = R
Mapeo Natural = MN	Manejo de errores = ME	Mejora = M
Orientación = OR	Consistencia = CO	No Afecta = N/A

Identificación	RATWCE03	Referencia	
Escenario	Autentificación		
Identificación según Bass	Recuperación de contraseñas olvidadas (Retrieving Forgotten Passwords)		
Contexto	Al entrar al sistema el usuario debe llevar a cabo la autentificación en caso de estar registrado, pero si no recuerda alguno de los datos (usuario o login) no podrá entrar al sistema. En este momento no existe un mecanismo para la recuperación de datos olvidados.		
Propiedades Afectadas	Minimizar la carga cognitiva, accesibilidad		

Decisiones Arquitectónicas				
Decisiones	**S**	**R**	**M**	**N/A**
Generar un método y funcionalidad asociada, que permita el restablecimiento de una nueva contraseña, mediante el envío de un email con un una dirección especifica donde el usuario podrá elegir una nueva clave				

Problemas	Ponderación

(valor entre el 1-10)

Peso o Ponderación Global del escenario	4

(valor entre el 1-10)

		Leyenda
Explícito control del usuario = EU	Retroalimentación = RE	Sensibilidad = S
Minimizar la carga cognitiva = MC	Accesibilidad = AC	Riesgo = R
Mapeo Natural = MN	Manejo de errores = ME	Mejora = M
Orientación = OR	Consistencia = CO	No Afecta = N/A

Identificación	RATWCE04	Referencia	
Escenario	Registro en el sistema		
Identificación según Bass	Proporcionar una buena ayuda (Providing Good Help)		
Contexto	Al existir dos posibles perfiles de usuarios con registro en el sistema (arrendatario y propietario) .El usuario debe encontrar que los procedimientos de ayuda del sistema se adaptan al contexto.		
Propiedades Afectadas	Explicito control del usuario, retroalimentación, accesibilidad		

Decisiones Arquitectónicas				
Decisiones	**S**	**R**	**M**	**N/A**
Implementación de dos mecanismos de ayuda distintos, customizados, que serán mostrados de acuerdo al tipo de usuario que desee realizar el registro.				

Problemas	Ponderación

(valor entre el 1-10)

Peso o Ponderación Global del escenario	3

(valor entre el 1-10)

		Leyenda
Explícito control del usuario = EU	Retroalimentación = RE	Sensibilidad = S
Minimizar la carga cognitiva = MC	*Accesibilidad* = AC	*Riesgo* = R
Mapeo Natural = MN	Manejo de errores = ME	Mejora = M
Orientación = OR	Consistencia = CO	No Afecta = N/A

Identificación	RATWCE07	Referencia	
Escenario	Buscar Resorts		
Identificación según Bass	Soportar Búsquedas exhaustivas (Supporting Comprehensive Searching)		
Contexto	Cuando se realiza la búsqueda del resort quizás el usuarios sólo posea uno de los datos (parte de la ubicación, el nombre, el número de habitaciones), es importante permitir que el usuario pueda realizar una búsqueda poseyendo escasos datos y que brinde resultados acertados. El sistema debe permitir a los usuarios buscar datos de una manera integral y coherente por criterios pertinentes.		
Propiedades Afectadas	Minimizar la carga cognitiva, Explicito control del usuario		

Decisiones Arquitectónicas				
Decisiones	**S**	**R**	**M**	**N/A**
Implementar un mecanismo de búsqueda que permita al usuario seleccionar que parámetro desea utilizar para la realización de su búsqueda, insertando la palabra clave correspondiente, de no seleccionar ninguno el sistema realizara una búsqueda avanzada de esta palabra en cada uno de los parámetros existentes				

Problemas	**Ponderación**
Tiempo de respuesta del sistema	3
Recursos consumidos para búsqueda en grandes volúmenes de información	4

(valor entre el 1-10)

Peso o Ponderación Global del escenario	**8 (+7)**

(valor entre el 1-10)

		Leyenda
Explícito control del usuario = EU	Retroalimentación = RE	Sensibilidad = S
Minimizar la carga cognitiva = MC	*Accesibilidad = AC*	*Riesgo = R*
Mapeo Natural = MN	Manejo de errores = ME	Mejora = M
Orientación = OR	Consistencia = CO	No Afecta = N/A

Anexo 6: Resultados Tabulados de los cuestionarios aplicados a los usuarios

#	Pregunta	Posibles Respuesta	Resultados	Resultados %
1	Porcentaje de elementos funcionales utilizados sin error	Es menor que el 25%	2	10
		Mayor al 25% y menor que 50%	1	5
		Mayor al 50% y menor que 75%	13	65
		Mayor que 75%	4	20
2	Porcentaje de objetivos logrados cuando utiliza el sistema	Es menor que el 25%	1	5
		Mayor al 25% y menor que 50%	1	5
		Mayor al 50% y menor que 75%	17	85
		Mayor que 75%	1	5
3	Promedio de precisión de tareas completadas con el sistema	Es menor que el 25%	0	0
		Mayor al 25% y menor que 50%	2	10
		Mayor al 50% y menor que 75%	16	80
		Mayor que 75%	2	10
4	Tiempo promedio para completar una tarea	1 minuto o menos	17	85
		2 minutos a 5 minutos	3	15
		de 10 a 20 minutos	0	0
		Más de 20 minutos	0	0
5	Porcentaje de tareas completadas en un día	Es menor que el 25%	0	0
		Mayor al 25% y menor que 50%	0	0
		Mayor al 50% y menor que 75%	1	5
		Mayor que 75%	19	95
6	Porcentaje de tareas completadas en un día (sin problemas).	Es menor que el 25%	0	0
		Mayor al 25% y menor que 50%	0	0
		Mayor al 50% y menor que 75%	2	10
		Mayor que 75%	18	90
7	Existen mensajes de advertencia ante operaciones o comandos peligrosos.	Si	7	35
		No	13	65
8	Existen mensajes de advertencia ante operaciones irreversibles.	Si	8	40
		No	12	60
9	Errores detectados por el sistema	Es menor que el 25%	5	25
		Mayor al 25% y menor que 50%	12	60
		Mayor al 50% y menor que 75%	3	15
		Mayor que 75%	0	0
10	El sistema posee la posibilidad de recuperarse luego de un error	Si	9	45
		No	11	55

#	Pregunta	Posibles Respuesta	Resultados	Resultados %
11	Tiempo en que un usuario adquiere las destrezas necesarias para utilizar el sistema	6 días o menos	1	5
		7 - 14 días	15	75
		15 - 30 días	4	20
		2 meses o mas	0	0
12	Tiempo en que un usuario adquiere las destrezas necesarias para utilizar el sistema de forma experta.	6 días o menos	0	0
		7 - 14 días	1	5
		15 - 30 días	3	15
		2 meses o mas	16	80
13	Valor real invertido en la adquisición del producto	Es menor que el 25%	1	5
		Mayor al 25% y menor que 50%	4	20
		Mayor al 50% y menor que 75%	15	75
		Mayor que 75%	0	0
14	Valor real invertido en el mantenimiento del producto	Es menor que el 25%	0	0
		Mayor al 25% y menor que 50%	2	10
		Mayor al 50% y menor que 75%	18	90
		Mayor que 75%	0	0
15	Valor real invertido en el soporte del producto	Es menor que el 25%	6	30
		Mayor al 25% y menor que 50%	10	50
		Mayor al 50% y menor que 75%	4	20
		Mayor que 75%	0	0
16	Numero de términos técnicos utilizados en el elemento funcional.	Es menor que el 25%	14	70
		Mayor al 25% y menor que 50%	5	25
		Mayor al 50% y menor que 75%	1	5
		Mayor que 75%	0	0
17	Porcentaje de elementos funcionales con nombres relacionados a su funcionalidad.	Es menor que el 25%	0	0
		Mayor al 25% y menor que 50%	1	5
		Mayor al 50% y menor que 75%	1	5
		Mayor que 75%	18	90
18	Porcentaje de interfaces y métodos descritos en los manuales.	Es menor que el 25%	0	0
		Mayor al 25% y menor que 50%	0	0
		Mayor al 50% y menor que 75%	4	20
		Mayor que 75%	16	80
19	Porcentaje de elementos funcionales utilizados correctamente luego de leer el manual.	Es menor que el 25%	0	0
		Mayor al 25% y menor que 50%	0	0
		Mayor al 50% y menor que 75%	5	25
		Mayor que 75%	15	75
20	Porcentaje de elementos familiares en los elementos funcionales.	Es menor que el 25%	0	0
		Mayor al 25% y menor que 50%	1	5
		Mayor al 50% y menor que 75%	3	15
		Mayor que 75%	16	80

#	Pregunta	Posibles Respuesta	Resultados	Resultados %
21	Existen estándares a ser aplicados en el desarrollo de documentos.	Siempre	5	25
		Algunas veces	14	70
		Nunca	1	5
22	Es versátil la navegación entre pantallas.	Excelente	13	65
		Buena	2	10
		Promedio	4	20
		Debajo del promedio	1	5
		Inaceptable	0	0

Anexo 7: Clasificación de los Métodos de Evaluación de la Usabilidad

Nombre del Método	Descripción	Características
Evaluación Heurística	Analizar la conformidad de la interfaz con unos principios reconocidos de usabilidad (las heurísticas) mediante la inspección de varios evaluadores expertos. *Estrategia:* Se puede aplicar durante todo el proceso de desarrollo, cada etapa puede considerar la utilización de este método manejando diferentes enfoques y artefactos, por ejemplo durante la captura de requisitos, la evaluación podría ser aplicada a sistemas actuales o alternativas de la competencia.	*Prejuicio:* Alta subjetividad del experto. Los resultados y recomendaciones son criterio del evaluador. *Medida:* Típicamente cualitativa. *Costo:* bajo costo de recursos y tiempo. *Información:* Abundante información (problemas a bajo nivel: colores, fuente, etc.); baja realimentación (problemas de alto nivel). *Inmediatez de la Respuesta:* Rápido retorno de información. *Intromisión:* ninguna, la evaluación heurística no involucra la participación de usuarios reales.
Recorrido Pluralista	Se reúne tanto a desarrolladores y expertos como con usuarios, se trabaja sobre prototipos, pueden ser en papel, se pide que ejecuten ciertas tareas sobre el prototipo mientras anotan cada paso de la interacción. Luego se debaten los problemas y decisiones tomadas durante el proceso [BIA95]. Combina tres visiones diferentes del sistema: lo que se debe (experto), se quiere (usuarios) y se puede (desarrollador). *Estrategia:* se encuentra una mayor fortaleza para este método en las primeras fases del desarrollo, el impacto ante los cambios propuestos y la generación de un lineamiento de trabajo para futuros modelo de mayor nivel, es evidente.	*Prejuicio:* alta subjetividad de los participantes. *Medida:* cualitativa. *Costo:* mediana inversión de recursos, sobre todo de tiempo y organización. *Información:* alta información (problemas a bajo nivel: colores, fuente, etc.); baja realimentación (problemas de alto nivel), la atención no está centrada en la observación del usuarios sino en el debate entre las partes. *Inmediatez de la Respuesta:* rápido retorno de información. *Intromisión:* Baja, los usuarios realizan su observación y sacan sus propias conclusiones, luego viene el proceso de debate en el cual la estar frente a un equipo de desarrollo y un grupo de expertos muchas veces intimida al usuario

Nombre del Método	Descripción	Características
Caminata Cognitiva (Cognitive Walkthroughs)	Este método tiene por eje central evaluar el rápido aprendizaje del sistema. Está basado en los recorridos estructurales tradicionales que se usan en la comunidad de la ingeniería de software [YOU89]. Partiendo de una interfaz detallada, un escenario y una tarea, se pide a usuarios reales interactuar con el sistema. Cada paso que el usuario realice será anotado y contrastado con la secuencia de acciones bajo las cuales el analista planteo la interfaz. Las sesiones son grabadas en video y analizadas para descubrir el origen de fallos, errores o frustraciones de los usuarios. Se trata de reconstruir paso a paso la interacción del usuario con el sistema. *Estrategia:* se recomienda su utilización durante las fases de diseño, implementación pruebas e implantación.	*Prejuicio:* alta subjetividad de los usuarios. *Medida:* cuantitativa o cualitativa, se puede manejar como un grado de coincidencia entre la ruta esperada y los caminos de solución tomados por el usuario. *Costo:* mediana inversión de recursos. *Información:* media información; alta realimentación, se evidencian problemas críticos de la interfaz que vayan más allá del modelo estético. *Inmediatez de la Respuesta:* media, es necesario un análisis detallado de las conductas del usuario durante la interacción. *Intromisión:* Media, la sección se registra sobre video, lo cual puede condicionar ciertas decisiones del usuario.
Inspección de estándares	Este método se realiza por medio de un experto en un estándar que puede ser de facto o de iure de la interfaz. El experto realiza una inspección minuciosa a la interfaz para comprobar que cumple en todo momento y globalmente todos los puntos definidos en el estándar. *Estrategia:* fases avanzadas del desarrollo, pruebas, e implantación.	Prejuicio: objetivo. Medida: cuantitativa, porcentaje de cumplimiento con el estándar. Costo: baja inversión de recursos. Información: alta información; baja o nula realimentación. *Inmediatez de la Respuesta:* alta, resultados disponibles en cuanto termine el proceso. El tiempo en la ejecución de las verificaciones dependerá de las herramientas con que cuente el experto y la complejidad del estándar. *Intromisión:* nula, no hay presencia de usuario.

Nombre del Método	Descripción	Características
Estudio de Campo / Método Etnográfico	centra su atención en los aspectos tangibles del entorno de ejecución de la aplicación. Selecciona perfiles de usuarios reales, diseña entrevistas y organiza sesiones de observación directa del entorno de trabajo. Registro en fotos, archivos de trabajo, etc. El resultado final de este método, es el reflejo del modelo mental del usuario con respecto a la tarea y artefactos (objetos tangibles en la interacción). Información muy útil dentro de etapas de definición del sistema. *Estrategia:* fases de definición, captura de requisitos e implantación, vista como los requerimientos de una segunda versión del producto.	*Prejuicio:* subjetiva, las apreciaciones están a juicio del observador. *Medida:* cualitativa, el propósito es conocer, más que juzgar. *Costo:* mediana inversión de recursos. *Información:* baja información; alta realimentación, modelo mental de las tareas. *Inmediatez de la Respuesta:* media, los resultados requieren de un análisis de los datos observados para la generación del modelo mental. *Intromisión:* media, la observación y entrevistas interfieren con el usuario.
Grupo de discusión dirigido (focus group)	Reúne de 6 a 9 usuarios para discutir aspectos relacionados con el sistema. Un ingeniero de factores humanos hace las veces de moderador, que tiene que preparar la lista de aspectos a discutir y recoger la información que necesita de la discusión. Esto puede permitir capturar reacciones espontáneas del usuario e ideas que evolucionan en el proceso dinámico del grupo. *Estrategia:* pruebas e implantación.	*Prejuicio:* subjetiva, las apreciaciones se gestan en debate. *Medida:* cualitativa. Apreciaciones del sistema. *Costo:* mediana inversión de recursos. *Información:* media información; media realimentación. *Inmediatez de la Respuesta:* media, los datos recogidos tienden a tener una validez baja y son muy difíciles de analizar debido a su naturaleza no estructurada y de flujo libre. *Intromisión:* alta, el proceso de debate termina por alterar la opinión de los usuarios.
Entrevistas	Consta de una serie de preguntas y tópicos el cual el usuario debe responder a un revisor; aspectos tales como su experiencia con el sistema, actitud e impresiones. El revisor debe adaptar la entrevista tal que pueda obtener el máximo beneficio en cada sesión. *Estrategia:* pruebas e implantación.	*Prejuicio:* subjetiva, apreciaciones del usuario. *Medida:* cualitativa, apreciaciones del sistema. *Costo:* baja inversión de recursos. *Información:* baja información; alta realimentación. *Inmediatez de la Respuesta:* alta. *Intromisión:* media.

Nombre del Método	Descripción	Características
Cuestionarios	Es un procedimiento menos flexible que una entrevista pero permite llegar a más usuarios y generar informes estadísticos con mayor facilidad. *Estrategia:* pruebas e implantación.	*Prejuicio:* subjetiva y objetiva. Aun cuando los cuestionarios se llenen subjetivamente, un número considerables de muestras conlleva a una visión objetiva del sistema. *Medida:* cualitativa, apreciaciones del sistema. *Costo:* baja inversión de recursos, sobre todo aprovechando los recursos tecnológicos. *Información:* baja información; alta realimentación. *Inmediatez de la Respuesta:* baja. *Intromisión:* baja o nula.
Grabación de Uso (logging)	Implica disponer en el computador de una ampliación del sistema que recoja automáticamente estadísticas sobre el uso detallado del sistema. Es útil porque muestra cómo los usuarios realizan su trabajo real y porque es fácil recoger automáticamente datos de una gran cantidad de usuarios que trabajan bajo diversas circunstancias. *Estrategia:* pruebas e implantación.	*Prejuicio:* objetiva. *Medida:* cualitativa. *Costo:* alto, consecuencia del diseño y construcción de los elementos de captura de la información, mas el tiempo de análisis de los datos. *Información:* baja información; alta realimentación. *Inmediatez de la Respuesta:* baja, depende de la estrategia de análisis de los datos. *Intromisión:* baja, la captura puede hace de forma transparente al usuario.
Pensando en voz alta (thinking aloud)	Se pide a los usuarios que expresen en voz alta sus pensamientos, sentimientos y opiniones mientras que interaccionan con el sistema. Es muy útil en la captura de un amplio rango de actividades cognitivas. *Estrategia:* diseño, pruebas e implantación.	*Prejuicio:* subjetivo. *Medida:* cualitativa. *Costo:* alto. *Información:* media información; alta realimentación. Ayuda a comprender el modelo mental de la tarea y proporciona información como terminología empleada, funcionalidades deseables y necesidades en la documentación. *Inmediatez de la Respuesta:* alta. *Intromisión:* alta, no constituye un proceso natural para el usuario.

Nombre del Método	Descripción	Características
Interacción Constructiva (co-descubrimiento)	Es una derivación del método pensando en voz alta e implica el tener en vez de uno, dos usuarios que hagan el test al sistema conjuntamente. La principal ventaja es ser mucho más natural que el pensar en voz alta, ya que las personas normalmente verbalizan cuando tratan de resolver un problema conjuntamente y además hacen muchos más comentarios; la desventaja es que los usuarios pueden tener diferentes estrategias de aprendizaje. *Estrategia:* diseño, pruebas e implantación.	*Prejuicio:* subjetivo. *Medida:* cualitativa. *Costo:* alto. *Información:* media información; alta realimentación. Al tenerse estrategias de aprendizaje distintas se produce una interferencia entre las opiniones de los participantes, esto reduce la capacidad de abstraer el modelo mental de la tarea. *Inmediatez de la Respuesta:* alta. *Intromisión:* baja, convirtiendo el proceso de pensar en voz alta, en algo normal.
Examen Retrospectivo	Se parte del hecho que si se ha realizado una grabación en vídeo de la sesión de evaluación, es posible recoger más información haciendo que el usuario revise la grabación. De este modo, el usuario puede dar opiniones más amplias acerca de su interacción y el observador puede preguntar sin interferir el proceso. Demanda un mayor tiempo por sesión. *Estrategia:* diseño, pruebas e implantación.	*Prejuicio:* subjetivo. *Medida:* cualitativa. *Costo:* alto. *Información:* media información; alta realimentación. *Inmediatez de la Respuesta:* alta. *Intromisión:* baja, las preguntas se realizan examinando el video de la sesión.
Método del conductor (coaching method)	Contrario a muchos otros métodos, donde la interferencia del observador es evitada en lo posible, este método guía o conduce a los usuarios en la dirección correcta mientas usan el sistema. Este método se centra en el usuario inexperto y su propósito es descubrir las necesidades de información de los usuarios de tal manera que se proporcione un mejor entrenamiento y documentación al mismo tiempo que un posible rediseño de la interfaz para evitar la necesidad de preguntas. *Estrategia:* diseño, pruebas e implantación.	*Prejuicio:* subjetivo. *Medida:* cualitativa. *Costo:* alto. *Información:* alta información; alta realimentación. Inmediatez de la *Respuesta:* alta. *Intromisión:* alta, la sesión completamente guiada.

Nombre del Método	Descripción	Características
Método de Modelos	Utilización de los modelos GOMS (Goal, Operations, Methods and Selection Rules). Un modelo GOMS es una descripción del conocimiento que un usuario debe tener para poder llevar a cabo una tarea sobre un dispositivo o un sistema. Consiste de la descripción de los Métodos necesitados para culminar una Meta específica. Los Métodos están compuestos de Operadores que son una serie de pasos que el usuario interpreta. Un Método puede exigir el cumplimiento de sub.-Metas para ser culminado, por tanto los métodos tienen una estructura jerárquica. Si hay más de un método comprometido en una Meta, entonces Reglas de Selección escogen el Método apropiado dependiendo del contexto. Describir las Metas, Operadores, Métodos y Reglas de Selección para un conjunto de tareas de una manera formal constituye un análisis GOMS o define el denominado modelo GOMS. *Estrategia:* diseño, pruebas e implantación.	*Prejuicio:* objetivo. *Medida:* cuantitativa y cualitativo. Costo: bajo. *Información:* baja información; nula realimentación. Si por ejemplo debe seleccionarse entre dos sistemas, el GOMS es un buen criterio de elección. *Inmediatez de la Respuesta:* baja, complejidad en el uso de estos modelos. *Intromisión:* nula.