CATÁLOGO PARA LA REUTILIZACIÓN DE COMPONENTES DESARROLLADOS CON EL MARCO DE TRABAJO MOON

FABIO ATUESTA OSORIO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2011

CATÁLOGO PARA LA REUTILIZACIÓN DE COMPONENTES DESARROLLADOS CON EL MARCO DE TRABAJO MOON

Presentado por:
FABIO ATUESTA OSORIO

Proyecto de Grado para optar al título de
Ingeniero de Sistemas

Director:
MSc. FERNANDO ROJAS MORALES
Profesor Titular,
Universidad Industrial de Santander

Co-Director:
MSc. Luz Elena Gutiérrez López
Docente Investigador,
Unidades Tecnológicas de Santander

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2011

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

TITULO:
CATÁLOGO PARA LA REUTILIZACIÓN DE COMPONENTES DESARROLLADOS CON EL MARCO DE TRABAJO MOON[*].

Autores:
ATUESTA OSORIO, Fabio[**].

Palabras Claves:
Proceso de Desarrollo, Marco de Trabajo, Componente, Reutilización, Clases.

Descripción:
El documento está organizado en seis (6) capítulos, Aspectos Generales donde se plantean cuatro (4) objetivos a desarrollar a lo largo de la investigación, Contextualización Teórica relacionado con el referente teórico, el núcleo de la propuesta se encuentra en los apartados: Componente Software Reutilizable Propuesto, Aplicación del Proceso de Desarrollo Propuesto por Vera Rivera y Propuesta del Catálogo para la Construcción de Componentes Software Reutilizables, y para finalizar se describen los Resultados Obtenidos.

El presente proyecto de investigación tiene como finalidad proporcionar un instrumento que soporte el proceso de selección de componentes reutilizables. La línea de base de este trabajo es el proyecto de maestría denominado "*Propuesta de un Proceso de Desarrollo de Componentes software Reutilizable*" realizado por Freddy Vera en 2009, el trabajo de maestría proporcionó los criterios de valoración para identificar cuando un componente es reutilizable y la metodología de desarrollo del mismo. El alcance del proyecto de grado consistió en la aplicación de las escalas de valoración y la metodología de desarrollo en la construcción de un componente software que fue implementado con un marco de trabajo reconocido a nivel académico y empresarial denominado MOON. El valor adicional del proyecto se puede apreciar en un componente implementado haciendo uso del marco de trabajo y además integrado como parte de las librerías del Framework MOON. La arquitectura de software utilizada es una arquitectura N-Capas. El producto final del proyecto tiene dos fortalezas el componente reutilizable y portable y el catálogo para seleccionar componentes reutilizables sin necesidad de conocer el código fuente. Esta última característica hace parte de los requisitos de vanguardia necesarios para la certificación de las empresas con modelos como CMMi.

Los principios metodológicos para la realización del presente trabajo de grado, están soportados en las técnicas y estándares de la Ingeniería del Software.

[*] Trabajo de Grado.
[**] Facultad de Ingenierías Físico-Mecánicas, Escuela de Ingeniería de Sistemas e Informática, Director: Fernando Rojas Morales, Co-Director: Luz Elena Gutiérrez López.

ABSTRACT

TITLE:
CATALOG FOR THE REUTILIZATION OF COMPONENTS DEVELOPED WITHIN THE MOON FRAMEWORK[*].

Authors:
ATUESTA OSORIO, Fabio[**].

Key words:
Development Process, Framework, Reutilization, Classes.

Description:
This document is organized in six (6) chapters, general aspects where four (4) objectives to develop along the research are suggested: theoretic contextualizing related to the theoretic reference, the nucleus of the proposal is to be found in the sections Proposed Reusable Software Component, Application Of The Development Process Proposed by Vera Rivera, and Proposal of a Catalog for the Construction of Reusable Software Components; closing with a description of Results Obtained.

The present research project has as its goal to furnish a tool to support the selection process of reusable components. The basic guide to the research is the Master's degree thesis called "Proposal of a process to develop reusable software components" ("Propuesta de un proceso de desarrollo de components software reutilizables") by Freddy Vera in 2009; this M.Sc. thesis supplied the criteria to evaluate when a component is reusable or not, and the methodology to develop it. The scope of that thesis includes the application of evaluation scales, and the methodology to develop the construction of a software component that was implemented within a work frame called MOON, recognized at both academic and business levels. The value added by the project can be appreciated in a component implemented by using the work frame, that has deserved to be included as part of the library of the MOON Framework. The software architecture utilized is N-layers architecture. The final product of the project shows two strengths: the reusable and portable component, and the catalog that allows selecting reusable components without the need to know the source code. This last characteristic is part of the vanguard requisites demanded by models like CMMi to certify companies.

The methodological principles for this thesis are supported by the techniques and standards of software engineering.

[*] Thesis work.
[**] College of Physical-mechanical Engineering, Department Of Systems Engineering and Informatics Director: Fernando Rojas Morales, Co-Director: Luz Elena Gutierrez Lopez.

INTRODUCCIÓN

El presente documento tiene como propósito presentar los resultados obtenidos a partir del desarrollo del trabajo de investigación, "*CATÁLOGO PARA LA REUTILIZACIÓN DE COMPONENTES DESARROLLADOS CON EL MARCO DE TRABAJO MOON*", como requisito para optar al título de Ingeniero de Sistemas de la Universidad Industrial de Santander.

Dicho trabajo de investigación tiene por objeto la definición y construcción de un catálogo que sirva de soporte al proceso de desarrollo en proyectos del área de sistemas. El modelo diseñado será descrito por medio de UML como lenguaje de representación.

Como línea de base para el desarrollo del catálogo, se utilizó el proyecto de maestría denominado "*Propuesta de un Proceso de Desarrollo de Componentes software Reutilizable*" autoría del Ing. Fredy Vera. Para la verificación del catálogo se plantea el desarrollo de un prototipo. Aunque es un trabajo de investigación aplicada, el enfoque principal será la investigación descriptiva, pues ésta ayuda a definir bases, identificar características y propiedades del objeto de estudio de la presente propuesta. Cómo marco metodológico para el desarrollo del prototipo se utilizará el prototipado evolutivo con el fin de presentar avances tempranos del proyecto.

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

GLOSARIO

Arquitectura: es la estructura y organización fundamental de un sistema, formada por sus componentes y 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.

Diagrama: es una representación gráfica de una colección de elementos del modelo, construida a menudo como un gráfico conexo de arcos (relaciones) y de vértices (otros elementos modelo).

HTML: lenguaje para marcado de hipertexto. Lenguaje para estructurar documentos a partir de texto en World Wide Web. Se basa en etiquetas (instrucciones que le dicen al texto como deben mostrarse) y atributos (parámetros que dan valor a la etiqueta).

Ingeniería del Software: es una aproximación sistemática, disciplinada y cuantificable aplicada al desarrollo, operación y mantenimiento del software. IEEE STD Glosario.

Internet: red global de redes de ordenadores cuya finalidad es permitir el intercambio libre de información entre todos sus usuarios. Con Internet se pueden enviar mensajes, programas ejecutables, ficheros de texto, consultar catálogos de bibliotecas, pedir libros y hacer compras.

PHP: lenguaje de programación de estilo clásico, es decir, es un lenguaje de programación con variables, sentencias condicionales, bucles, funciones. El programa PHP es ejecutado en el servidor y el resultado enviado al navegador. El resultado es normalmente una página HTML.

PostgresSQL: es un administrador de base de datos relacionales; que soporta instrucciones de SQL. Este manejador es uno de los más populares y funcionales dentro de lo que es el software libre, esto funciona para Solaris y Linux.

TCP/IP: Transfer Control Protocol/Internet Protocol. Paradigma de red que permite la comunicación de sistemas en todo el mundo como una sola red, Internet. Desarrollado por la DARPA (Defense Advanced Research Projects Agency; Agencia de proyectos avanzados de investigación de defensa) a finales de la década de los 70's. TCP corresponde a la capa de transporte del modelo OSI (Modelo de referencia OSI) y ofrece la transmisión de datos, e IP corresponde a la capa de red y ofrece servicios de data gramas sin conexión.

UML: Lenguaje Unificado de Modelado (Unified Modeling Language), es una notación estándar para el modelado de objetos del mundo real, como un primer paso en el desarrollo de una metodología de diseño orientada a objetos. Su notación unifica las notaciones de tres

metodologías de análisis y diseño orientados a objetos: Grady Booch, Object-Modeling Technique (OMT), y Jacobson.

XML: Extensible Markup Language o Lenguaje extensible de marcas es un conjunto de reglas que sirven para definir etiquetas semánticas para organizar un documento. Además, el XML es un metalenguaje que permite diseñar un lenguaje propio de etiquetas.

1. ASPECTOS GENERALES

1.1 Información General del Proyecto

Titulo	Catálogo para la Reutilización de Componentes Desarrollados con el Marco de Trabajo MOON
Director	MSc. Fernando Rojas Morales, *frojas@uis.edu.co*
Autor	Fabio Atuesta Osorio
Duración proyecto (en meses)	Cinco (5)
Presupuesto	$ 5.025.000

1.2 Descripción de Objetivos

1.2.1 Objetivo General

Desarrollar un catálogo para la reutilización de componentes que utilicen el Marco de Trabajo (Framework) *MOON*[1], soportado en los lineamientos y criterios definidos en el trabajo de Maestría "*Propuesta de un Proceso de Desarrollo de Componentes software Reutilizable*[2]", basándose en el uso de patrones de software y en los principios de la Ingeniería del Software.

1.2.2 Objetivos Específicos

La tabla 1 describe los objetivos específicos planteados que servirán para dar cumplimiento al objetivo general.

Tabla 1. Objetivos Específicos del proyecto

No.	Descripción del Objetivo
1	Aplicar el proceso de desarrollo propuesto en el trabajo de maestría "*Propuesta de un Proceso de Desarrollo de Componentes software Reutilizable*", para la construcción de un componente reutilizable que utilice la filosofía de trabajo y el Marco de Trabajo *MOON*.
2	Implementar un componente software reutilizable y utilizando el lenguaje PHP versión 5.3.3, haciendo uso de los patrones GOF[3] definidos en el Marco de Trabajo *MOON*, adoptando el proceso de desarrollo propuesto en el trabajo de maestría que se referencia en el objetivo 1.
3	Verificar el componente software con el fin de identificar el estado de reutilización del componente, utilizando como base los lineamientos y pesos cuantitativos definidos en el trabajo de maestría referenciado en el objetivo 1.
4	Diseñar un instrumento que permita construir un catálogo software que referencie componentes reutilizables desarrollados con el Marco de Trabajo MOON, teniendo en cuenta para esto como mínimo seis (6) criterios: Explicación, Escenarios de Aplicación, Representación en UML, Participantes, Beneficios, Problemas asociados al uso de este patrón, Implementación del Componente.

[1] Marco de Trabajo soportado en Patrones de Diseño GOF, de orientación académica para el desarrollo de aplicaciones empresariales en entornos de código abierto. Andrés Guerrero Alarcón, Unidades Tecnológicas de Santander.
[2] Trabajo de grado para optar por el título de Magíster en Ingeniería Área de Informática realizado por Fredy Humberto Vera Rivera, Universidad Industrial de Santander.
[3] Gang of Four es el nombre con el que se conoce comúnmente a los autores del libro Design Patterns, el cual es la referencia más importante en los procesos de desarrollo con orientación a objetos.

1.3 Justificación

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

1.3.1 Empresarial

Las empresas de desarrollo Software de la región, y en general en Colombia, han optado por mejorar sus procesos de calidad en producción, por ende han asimilado estándares y modelos de calidad en sus fases de desarrollo. Entre todos los estándares, la adopción de la Norma ISO es la más común, ésta indica los "Debes" que la empresa debe cumplir en los procesos que desea asegurar calidad, aunque es adecuada como guía, en software es preferible utilizar un Modelo que no solo indique lo que debe cumplir la empresa, sino que oriente en la posible forma de hacerlo. Este último caso corresponde al Modelo de Madurez de la Capacidad CMMi, pues el cual define niveles de madurez para que la empresa identifique las oportunidades de mejoramiento por áreas de conocimiento.

Ahora bien, en CMMi uno de los puntos de evaluación para el área de ingeniería son los repositorios, el área de proceso denominada "Technical Solution -TS-" es la que solicita criterios y bibliotecas de reutilización claramente definidas, en otras palabras, Cuáles y Cómo las bibliotecas de código pueden llegar a ser reutilizadas en otros proyectos. Los evaluadores no verifican el código fuente como tal, pues eso es competencia de la empresa, sin embargo, si verifican aspectos como: ¿Cómo la empresa accede a esos repositorios?, ¿Cómo sabe que repositorios tiene?, ¿Cómo toman una decisión para reutilizar un repositorio sin mirar el código fuente? Lo anterior indica que las empresas deberían tener a su disposición documentos que guíen paso a paso a un diseñador y desarrollador en el proceso de selección de un componente a reutilizar, ahorrándole el tiempo de verificar el código.

Este proyecto plantea la construcción de un Catálogo Software que le permita a un desarrollador identificar la posibilidad de reutilización de un componente siempre y cuando haya sido desarrollado con un Marco de trabajo específico (MOON), razón por la cual se justifica la oportunidad de realizar el presente proyecto en funcionan de las necesidades del sector productivo.

1.3.2 Tecnológica

La evolución tecnológica que viven las empresas en Colombia ha permitido un acelerado crecimiento de la producción de Software, los sistemas tradicionales ya están obsoletos, y las nuevas versiones de esos sistemas tienen su soporte en sistemas orientados a servicios. Sin embargo, los sistemas orientados a servicios tienen a su vez un núcleo base, el cual debe estar desarrollado con paradigmas que soporten y permitan hacer extensibles y escalables cada componente, en otras palabras, éste núcleo debe estar soportado en paradigmas como la programación orientada a objetos y la reutilización de software, de otra forma un servicio computacional puede ser obsoleto sin siquiera llegar a ser utilizado.

Este proyecto busca obtener como resultados directos productos que sirvan para soportar los procesos de desarrollo bajo desarrollos por Patrones, Reutilización de código y Estándares de calidad en producción de software.

1.3.3 Académica

Desde el punto de vista académico este proyecto se sustenta desde dos perspectivas:

1. *La actualidad de la temática a tratar*: En las ciencias computacionales, un concepto, una teoría o una técnica se consideran estables si han superado la barrera de los "15 años de tradición en el contexto"[4], por esta razón, los temas de Patrones, Arquitecturas y Marcos de Trabajo están a la vanguardia, son una realidad en los procesos de desarrollo actuales.

2. *La relación directa entre academia y sector productivo*: Este trabajo de investigación busca aplicar conceptos actuales en Ingeniería del Software a procesos reales de desarrollo, por tanto, el resultado directo del proyecto puede ser utilizado en procesos empresariales de desarrollo, inclusive por profesionales recién egresados que conozcan o estén involucrados con la temática del proyecto.

[4] Artículo "The Golden Age of Software Architecture", Mary SHAW y Paul CLEMENTS.

2. CONTEXTUALIZACIÓN TEÓRICA

2.1 Repositorios de Componentes

Teniendo en cuenta la contextualización informática del proyecto, un repositorio es una bodega que sirve para almacenar información de todo tipo, ya sean diseños, desarrollos, documentos o cualquier otro artefacto software.

Un componente software por su parte es un conjunto de código diseñado y desarrollado para cumplir con un fin particular, que puede funcionar aislado o en conjunción con otros componentes. Se caracterizan por ser independientes y funcionales, por tal motivo tienen su propia estructura interna en cuanto a organización de archivos fuentes y forma de implementación. En general, la bibliografía que relaciona los conceptos de componentes los visualiza como entidades pre-compiladas que definen formas de comunicarse (interfaces) entre ellos mismos.

Teniendo en cuenta lo anterior, se puede inferir que los “repositorios de componentes” son bodegas de datos para almacenar código pre-compilado (en algunos casos el mismo fuente) organizados de acuerdo a algunas características. Los repositorios sirven para optimizar los tiempos de desarrollo de proyectos a gran escala pues utilizando técnicas de reutilización se pueden desarrollar nuevos componentes en función de los ya existentes:

Ventajas:

1. Mantiene una memoria permanente de los desarrollos realizados.
2. Pueden almacenar tantos componentes como sean desarrollados.
3. Analizando la estructura de un componente almacenado se puede identificar si es útil su reutilización, si es mejor desarrollar uno nuevo o en su defecto si es mejor adquirir un producto de un tercero.

Desventajas

1. A medida que crece el repositorio se hace más complejo recordar y tener presente que componentes puede llegar a ser reutilizados en un contexto particular.
2. Los repositorios por si solos son mecanismos de almacenamiento, no brindan una estructura flexible para tomar una decisión a priori a la hora de reutilizar un componente.
3. Para que un repositorio tenga éxito requiere de estrategias y lineamientos adicionales para la indexación de los mismos, sistemas de clasificación y por ende documentación adicional para que cada componente pueda llegar a ser reutilizado con facilidad.

2.2 Catálogos de componentes

Los catálogos de componentes son una estrategia para mantener los repositorios actualizados y funcionales. Antes de los catálogos, los repositorios almacenaban cantidades y cantidades de componentes, en muchos casos el mismo componente era almacenado con una variante muy simple y a la hora de reutilizarlo era complejo identificar cual era la diferencia sustancial entre uno y otro.

Con el tiempo, los modelos y estándares para la mejora en el desarrollo de software llegaron a conclusiones tácitas sobre las posibles estrategias a seguir para mantener repositorios que en realidad cumplieran su objetivo final. Una de estas estrategias es la creación y utilización de Catálogos de Software que permitan identificar claramente qué componentes almacena un repositorio, cómo fue desarrollado, cuál es su diseño, qué tipo de interfaces permite, todo esto sin necesidad de ingresar físicamente a buscar un componente a un repositorio.

Se pude definir entonces un Catálogo de Software como la agrupación de un conjunto de características que permiten a un diseñador y desarrollador tomar la decisión de reutilizar un componente sin necesidad de analizar el código o su posible implementación. Son herramientas de apoyo a los repositorios que permiten mantenerlos actualizados, pues así como en los catálogos de artículos se presentan nuevos productos, los cuales siempre van a referenciar las últimas tendencias, en términos de software un catálogo reflejaría las últimas versiones de cada componente.

Los Catálogos no son la competencia de los repositorios, por el contrario son el complemento ideal para mantener un repositorio actualizado.

2.3 Reutilización

Es el proceso de la Ingeniería del Software para la creación de software desarrollando componentes para ser utilizados en otros desarrollos.

La reutilización a nivel individual o de pequeños grupos de desarrollo no es nueva y desde la primera generación los programadores han copiado segmentos de código para ahorrar trabajo de programación. En otros casos se han utilizado bibliotecas de rutinas, principalmente matemáticas que han dado buenos resultados pero sin lograr una disminución substancial de los costos de desarrollo y mantenimiento.

La reutilización a nivel de empresa es sistemática y requiere de un conocimiento y modelado del dominio para la aplicación de interés, sea del sector financiero, control de procesos industriales o gestión de redes. Para realizar el modelado del dominio del problema se utilizan técnicas de análisis de dominio, algunas de ellas se basan en los principios de adquisición y representación del conocimiento desarrollado dentro de la Inteligencia Artificial, otras utilizan principios parecidos a los del análisis de sistemas donde existe mucha experiencia en la industria. La representación del dominio en el ámbito del problema, lleva a la búsqueda de sus soluciones, entonces se desarrollan arquitecturas de referencia que permiten crear implementaciones de aplicaciones en el ámbito del dominio. Estas arquitecturas de referencia se pueden apoyar en bibliotecas de programas sean estas rutinas o clases dependiendo de las metodologías de desarrollo software que se utilicen [2*].

2.3.1 Reutilización de Caja negra

La reutilización de caja negra es conseguida cuando los detalles de la implementación de un componente reutilizable son escondidos al cliente. Los clientes saben sólo sobre un conjunto disponible de capacidades (el que). Los componentes nunca exponen detalles internos de la su implementación (el cómo).

Ejemplo: Muchas tecnologías hoy apoyan la reutilización de caja negra. Los componentes COM+, Assembly de Microsoft y la tecnología .NET soportan la creación de componentes que son usados vía su conocida interfaz sin cualquier conocimiento del usuario sobre la estructura interna del componente. Los artefactos de GUI, Controles UI, Add-In son un ejemplo de tales componentes.

2.3.2 Reutilización de Caja blanca

La reutilización de caja blanca es la reutilización de un componente de software completo o parcial, con el entendimiento profundo de su estructura interna, algoritmos, módulos etc.

Ejemplo: En la Programación OO la reutilización de caja blanca se refiere al proceso de desarrollar software escribiendo subclases con el conocimiento y entendiendo de las clases internas de las clases padres. Un ejemplo de esto es la utilización de patrones.

2.3.3 Reutilización Horizontal

La reutilización horizontal proporciona componentes reutilizables genéricos que pueden apoyar una variedad de productos. La reutilización horizontal se refiere a componentes de software usados a través de una amplia variedad de aplicaciones.

Ejemplo: En términos de activos de código, la reutilización horizontal incluye bibliotecas de componentes, como clases genéricas, rutinas de manipulación de cadenas, o funciones de interfaz gráfica de usuario (GUI). La reutilización horizontal también puede referirse al uso de una aplicación de terceros dentro de un sistema más grande, como un paquete de correo electrónico o un programa de procesamiento de textos.

2.3.4 Reutilización Vertical

El concepto de Reutilización Vertical es la reutilización de áreas funcionales de un sistema, o dominios, que puede ser usado por una familia de sistemas con la funcionalidad similar. El estudio y la aplicación de esta idea han engendrado otras dos disciplinas de ingeniería llamadas: ingeniería de dominio e ingeniería de aplicación. Ingeniería de Dominio se enfoca en la creación y el mantenimiento de repositorios de reutilización de áreas funcionales, mientras en la ingeniería de aplicación, hace el uso de aquellos repositorios para poner en práctica nuevos productos.

Ejemplo: En el alcance de las aplicaciones de base de datos, un generador de informe general definido por el usuario es un ejemplo para la reutilización vertical.

2.3.5 Solución Específica

Es un acercamiento sistemático para identificar las similitudes, semejanzas y variabilidad necesaria para caracterizar y estandarizar una línea de productos como un conjunto del alcance de la solución, basado generalmente en componentes comerciales.

Ejemplo: Un convertidor de formato de imágenes general, es un ejemplo para una línea de productos (o una solución de digitalización de imágenes) de reutilización de software [8*].

2.4 Marcos de Trabajo

Un Marco de Trabajo es un conjunto de componentes físicos y lógicos estructurados de tal forma que permiten ser reutilizados en el Diseño y Desarrollo de nuevos Sistemas de Información [4*].

Un Marco de Trabajo se puede clasificar según su extensibilidad en:

Caja negra: Conocido como Marco de Trabajo de Arquitectura de Dato-Conducido. Para este Marco de Trabajo no es necesario conocer los detalles internos, pues se utiliza cada elemento a nivel de componentes. Los objetos son creados por medio de secuencias que utilizan y/o envían mensajes a cada clase para implementar el código.

Caja blanca: Conocido como Marco de Trabajo de Arquitectura de Configuración-Conducida. Las instancias de cada clase deben realizarse por medio de la creación de nuevas clases (a través de la herencia) que utilizan la extensión del Marco de Trabajo [3*].

2.4.1 Características de los Marcos de Trabajo

A continuación se enuncian las principales características que podemos encontrar en la mayoría de los Marcos de Trabajo existentes:

- *Abstracción de URLs y Sesiones:* No es necesario manipular directamente las URLs ni las sesiones, el Marco de Trabajo ~~ya~~ se encarga de hacerlo.
- *Acceso a Datos:* Incluyen las herramientas e interfaces necesarias para integrarse con herramientas de acceso a datos. Disponen de conectores a las bases de datos más conocidas.
- *Controladores:* La mayoría de Marcos de Trabajo e interfaces implementa una serie de controladores para gestionar eventos, como una introducción de datos mediante un formulario o el acceso a una página. Estos controladores pueden ser fácilmente adaptables a las necesidades de un proyecto concreto.
- *Autenticación y control de acceso:* Incluye mecanismos para la identificación de usuarios, mediante la solicitud de nombre de usuario y una contraseña cifrada, para restringir el acceso a determinados componentes y usuarios.
- *Formularios:* Cuentan con herramientas para crear formularios, permitiendo la aplicación de controles personalizados a cada uno de los objetos que son utilizados en formularios.
- *Código Abierto (Open Source):* Reducción de costos en licencias, pues la mayoría de componentes reutilizables son de código abierto.
- *Reducción de esfuerzo en el desarrollo:* Permiten la creación de plantillas para la reutilización de código. Permite la simplificación de problemas comunes en el proceso de desarrollo, por ende se optimizan los tiempos de implementación.
- *Solución definida y controlada:* Satisface requerimientos claramente definidos en un proyecto [2*].

2.4.2 Selección del Marco de trabajo a Utilizar

Antes de iniciar un proceso de desarrollo se deben tener en cuenta algunos aspectos como: Experiencia en desarrollos similares, Plataformas tecnológicas a utilizar, Líneas de base para el diseño y desarrollo, entre otras. Aunque lo anterior no garantiza la calidad del proyecto, si

aumenta la productividad en un proceso de desarrollo, por tanto, es clave seleccionar un Marco de Trabajo que permita tener una línea de base sólida al inicio del proyecto, no sólo en aspectos de desarrollo, sino en la elección de la Arquitectura a utilizar y el diseño.

La selección de Marcos de Trabajo no es una tarea sencilla, es un proceso que requiere la definición de criterios y variables que permitan elegir la mejor alternativa para un contexto particular. En el caso del presente proyecto, el contexto es estrictamente académico e investigativo, con fines de aprendizaje y formación de competencias, por tanto, el Marco de Trabajo seleccionado para ser utilizado debe cumplir con esas características.

Como fuente primaria para realizar el proceso de selección se tomó como referencia el libro titulado "*FrameWorks Para el desarrollo de Aplicaciones Web que utilizan código abierto. Herramientas necesarias para el desarrollo de Aplicaciones Web*" [2*]. En el capítulo 1 define el concepto de marco de trabajo utilizado en esta propuesta, expone la arquitectura de un marco de trabajo y aclara algunos conceptos en cuanto al uso de licencias y el desarrollo Open Source. En el capítulo 2, los autores hacen referencia a tres (3) estudios previos que se realizaron, sin embargo, los criterios que utilizaron esos trabajos no eran completos y dejaban algunos interrogantes. El groso del libro se centra en los capítulos 3 y 4, en donde primero define los criterios de comparación y los sustenta desde la perspectiva académica, empresarial y tecnológica, cubriendo así gran parte del contexto que tiene esta propuesta de investigación, en segundo lugar realiza el comparativo de los Marcos de Trabajo y plantea una dicotomía implícita en los Marcos Actuales: Complejidad-Potencia ó Sencillez-Debilidad.

2.4.3 Línea de Base para el proyecto: Marco de Trabajo MOON

MOON es un Marco de Trabajo en código Abierto, que tiene las siguientes características:

1. **Modelado por Objetos**: Las entidades que se modelan tienen las mismas características de JAVA (esta versión del marco de trabajo está construida con PHP en su versión 5.3.3), es decir, la relación clase y modelo se mantiene, se requiere una clase por cada entidad asociada al modelo.

2. **Arquitectura definida**: MOON le brinda al desarrollador una Arquitectura soportada en capas, inicia en cuatro (4) capas pero el desarrollador puede llegar a definir capas adicionales. A pesar de usar el concepto de capas, la Arquitectura puede usar conceptos de servicios, lo que hace que el desarrollador use varias arquitecturas sin de forma transparente.

3. **Multiplataforma Transparente**: MOON funciona en plataformas Windows y Linux sin necesidad de requerir configuración adicional. El marco de trabajo configura automáticamente el entorno para que el desarrollador se preocupe de los requisitos funcionales.

4. **Validación Integrada:** Los componentes de validación están integrados en el Marco de Trabajo, el desarrollador solo tiene que usar las librerías que requiera.

5. **Integración con AJAX:** El marco de Trabajo usa como línea de base la biblioteca denominada "*Prototype*", la cual hace parte del núcleo de java script para el soporte a AJAX.

6. **Utilidades gráficas**: Desde el uso de menús automáticos contra funcionalidades hasta los patrones de interacción más conocidos: "Calendar Picker" y "Time PIcker". Provee un conjunto de componentes que le permiten a un usuario desarrollar rápidamente sin tener conocimientos previos de interfaces Web.

7. **Conectividad con Bases de Datos**: Utiliza la librería nativa denominada PDO (*PHP Data Objects*), la cual hace parte integral del núcleo de Zend (el motor más actualizado de PHP). PDO tiene soporte para múltiples bases de datos relacionales.

Adicionalmente posee una potente interfaz para operaciones a una base de datos utilizando el modelo de objetos propuesto en las fases de diseño, éste permite que un desarrollador implemente funcionalidades de manera rápida sin necesidad de hacer consultas SQL directamente, todo lo realizan los objetos. Es una capa intermedia entre el concepto ORM (*Mapeo relacional de objetos*) y el modelo tradicional de conectividad a bases de datos.

2.4.4 ¿Por qué usar MOON?

Los Marcos de Trabajo comerciales, aunque sean de tipo Open Source siempre tienen una orientación técnica que marca la diferencia a la hora de ser utilizado. Algunos están construidos pensando en la escalabilidad de los sistemas pero dejan de lado el aspecto visual, por el contrario otros son ideales para desarrollar aplicaciones rápidas pero no tienen un modelo de desarrollo que sea escalable. La razón fundamental para tomar como referencia este marco de trabajo se fundamente en:

1. Las conclusiones en [2*], en donde expresan que los marcos de trabajo requieren de una curva de aprendizaje alta y en muchos casos es mejor adecuar uno a las necesidades que trabajar con uno terminado por terceros.

2. MOON es un Marco de Trabajo con dos connotaciones:
    a. ***Académica***: Esta desarrollado para que un estudiante implemente aplicaciones empresariales y pueda visualizar el uso correcto de patrones de diseño. Provee capas de abstracción para que el desarrollador no se entere de detalles tan técnicos, pero provee la filosofía para que el estudiante adquiera competencias que le permitan desarrollar más adelante y de manera rápida en plataformas como .NET y JAVA.
    b. ***Empresarial***: El uso del marco de trabajo en componentes que se encuentran en máquinas de producción han permitido la mejora continua del Marco, optimizando tareas y procesos para que su filosofía de desarrollo sea similar a los estándares que propone Java.

3. La curva de Aprendizaje de este Marco de Trabajo es relativamente baja comparada con otros marcos que requieren en promedio un año. Adicionalmente, en producción un componente puede ser evacuado con un buen diseño en semanas, mientras que en otros marcos de trabajo pueden llevar meses. Para el caso de JAVA el Marco de Trabajo más importante se denomina **SEAM**, y su curva de trabajo es tan alta que

requiere profesionales para su uso. En el caso de .NET la ausencia de un modelo orientado por estándares o patrones lo hace poco escalable.

2.5 Patrones de Software

El concepto de "patrón de diseño" que tenemos en Ingeniería del Software se ha tomado prestado de la arquitectura. En 1977 se publica el libro "A Pattern Language: Towns/Building/Construction", de Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King y Shlomo Angel, Oxford University Press. Contiene numerosos patrones con una notación específica de Alexander.

Alexander comenta que “Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma”. El patrón es un esquema de solución que se aplica a un tipo de problema, esta aplicación del patrón no es mecánica, sino que requiere de adaptación y matices. Por ello, dice Alexander que los numerosos usos de un patrón no se repiten dos veces de la misma forma.

La idea de patrones de diseño estaba "en el aire", la prueba es que numerosos diseñadores se dirigieron a aplicar las ideas de Alexander a su contexto. El catálogo más famoso de patrones se encuentra en “Design Patterns: Elements of Reusable Object-Oriented Software”, de Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, 1995, Addison-Wesley, también conocido como el libro GOF (Gang-Of-Four).

Siguiendo el libro de GOF los patrones se clasifican según el propósito para el que han sido definidos:

- Creacionales: solucionan problemas de creación de instancias. Nos ayudan a encapsular y abstraer dicha creación.
- Estructurales: solucionan problemas de composición (agregación) de clases y objetos.
- De Comportamiento: soluciones respecto a la interacción y responsabilidades entre clases y objetos, así como los algoritmos que encapsulan.

Según el ámbito se clasifican en patrones de clase y de objeto, como se puede apreciar en la tabla 2.

Tabla 2. Tipos de patrones

	Creación	**Estructural**	**De Conducta**
Clase	Método de Fabricación	Adaptador (clases)	Interprete Plantilla
Objeto	Fábrica, Constructor, Prototipo, Singleton	Adaptador (objetos), Puente, Composición, Decorador, Fachada, Flyweight	Cadena de Responsabilidad, Comando (orden), Iterador, Intermediario, Observador, Estado, Estrategia, Visitante, Memoria

2.6 Concepto de patrón de diseño

"Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles". (Grady Booch)

Los patrones de diseño son descripciones de clases cuyas instancias colaboran entre sí. Cada patrón es adecuado para ser adaptado a un cierto tipo de problema. Para describir un caso debemos especificar:

- Nombre
- Propósito o finalidad
- Sinónimos (otros nombres por los que puede ser conocido)
- Problema al que es aplicable
- Estructura (diagrama de clases)
- Participantes (responsabilidad de cada clase)
- Colaboraciones (diagrama de interacciones)
- Implementación (consejos, notas y ejemplos)
- Otros patrones con los que está relacionado

2.6.1 Ventajas de los patrones

La clave para la reutilización es anticiparse a los nuevos requisitos y cambios, de modo que los sistemas evolucionen de forma adecuada. Cada patrón permite que algunos aspectos de la estructura del sistema puedan cambiar independientemente de otros aspectos. Facilitan la reusabilidad, extensibilidad y mantenimiento. Un patrón es un esquema o microarquitectura que supone una solución a problemas (dominios de aplicación) semejantes (aunque los dominios de problema pueden ser muy diferentes e ir desde una aplicación CAD a un cuadro de mando empresarial). Interesa constatar una vez más la vieja distinción entre dominio del problema (donde aparecen las clases propias del dominio, como cuenta, empleado, coche o beneficiario) y el dominio de la solución o aplicación (donde además aparecen clases como ventana, menú, contenedor o listener). Los patrones pertenecen al dominio de la solución.

También conviene distinguir entre un patrón y una arquitectura global del sistema. Por decirlo en breve, es la misma distancia que hay entre el diseño de un componente (o módulo) y el análisis del sistema. Es la diferencia que hay entre el aspecto micro y el macro, por ello, en ocasiones se denomina a los patrones como "microarquitecturas".

En resumen, un patrón es el denominador común, una estructura común que tienen aplicaciones semejantes. Esto también ocurre en otros órdenes de la vida. Por ejemplo, en nuestra vida cotidiana aplicamos a menudo el esquema saludo-presentación-mensaje-despedida en ocasiones diversas, que van desde un intento de ligar hasta dar una conferencia (si todavía no cree que existan patrones o que no son útiles intente ligar con el esquema despedida-mensaje-presentación-saludo, a ver qué ocurre) [7*].

3. COMPONENTE SOFTWARE REUTILIZABLE PROPUESTO

El presente proyecto tiene entre sus objetivos específicos el siguiente alcance: "*Implementar un componente software reutilizable utilizando el lenguaje PHP versión 5.3.3, haciendo uso de los patrones GOF definidos en el Marco de Trabajo MOON, adoptando el proceso de desarrollo propuesto en el trabajo de Maestría que se referencia en el objetivo 1*", como consecuencia, la selección del componente debe realizarse teniendo en cuenta los criterios de reutilización definidos por Szyperski, Meyer, Sun Microsystem, e Iribarne Martínez, puesto que fueron los criterios utilizados en el trabajo de maestría que sirve como fuente primaria al presente proyecto.

Los conceptos de reutilización no sólo se refieren al código fuente, también hacen referencia a los requisitos, diseño en incluso las pruebas. El componente a implementar fue seleccionado pensando en función de la reutilización desde las perspectivas anteriormente enunciadas. Esto quiere decir que el proceso de reutilización inicia desde los requisitos básicos del componente, prosigue con el diseño de clases y el modelo relacional, hasta finalizar con los métodos básicos implementados en el código.

3.1 Funcionalidad del Componente reutilizable

El Marco de trabajo MOON está en constante crecimiento y tiene a disposición de los desarrolladores más de 30 componentes que han sido utilizados con éxito en múltiples desarrollos, incrementar el repositorio de MOON es entonces una estrategia que puede permitir una constante evolución y actualización del Marco de Trabajo.

Teniendo en cuenta que las necesidades a nivel de componentes del Marco de Trabajo MOON ya han sido identificadas con anterioridad, para este proyecto se seleccionó para diseñar y desarrollar el componente que más solicitudes de los usuarios tiene. Inicialmente se obtuvo una visión inicial del componente requerido y se encontraron las siguientes necesidades:

1. Listar un conjunto de elementos u objetos, permitiendo la edición, el borrado y la inserción de nuevos elementos.
2. Debe permitir la relación con otros objetos del mismo tipo y debe mantener la integridad de cada una de esas relaciones.
3. Debe permitir hacer replicas, de tal manera que se pueda extender su usabilidad en cualquier otro contexto, esto implica una alta cohesión y un bajo acoplamiento, de esta forma se permite su integración en cualquier otra parte del desarrollo.
4. La implementación de las clases debe estar orientada bajo el modelo EJB3, independientemente que se desarrolle con una tecnología diferente, llámese .NET, PHP, RUBBY, etc.
5. La interfaz que provea el componente debe ser independiente, no debe tener dependencia de otro componente.
6. Debe tener un conjunto de pruebas unitarias que le permitan garantizar su correcto funcionamiento.

3.2 Evaluación de necesidades para implementar el componente

La evaluación de requisitos del componente se realizó de acuerdo al trabajo realizado en el proyecto de maestría "*Propuesta de un proceso de Desarrollo de componentes software reutilizables*", el cual determina seis (6) criterios para analizar la posibilidad de reutilización de la solicitud del cliente. La tabla 3 presenta un análisis inicial de algunos de los criterios propuestos por [6*] y la justificación que indica el nivel de aprobación de cada criterio.

Tabla 3. Evaluación de requisitos del componente

No.	Criterio	Evaluación	Análisis
1	Dependencia estructural	Ok	El componente es independiente, no depende de otras configuraciones y/o requerimientos externos, y sólo gestiona la información que le corresponde para el conjunto de entidades u objetos que se estén utilizando.
2	Especificación de funcionalidades	Ok	Los requerimientos iniciales definidos en el apartado anterior delimitan el componente. Todas las funcionalidades estarán documentadas de tal manera que hagan parte de una biblioteca de reutilización.
3	Modelo de componentes	Ok	El modelo de componentes utilizado es el del marco de trabajo MOON, sin embargo este modelo hereda toda su filosofía de los EJB3[5].
4	Ensamble con otros componentes	Ok	Para el ensamble de componentes se utiliza como referencia el "Modelo Vista controlador - MVC", esto le permite a un desarrollador utilizar las clases en cualquier otro desarrollo que utilice el modelo propuesto por MOON.
5	Interfaz	Ok	Cada clase define sus interfaces y utiliza una fachada para manejar el nivel de abstracción aceptable para el desarrollador.
6	Certificado y probado	Ok	Tiene la certificación del Marco de Trabajo MOON en el desarrollo de pruebas Unitarias.

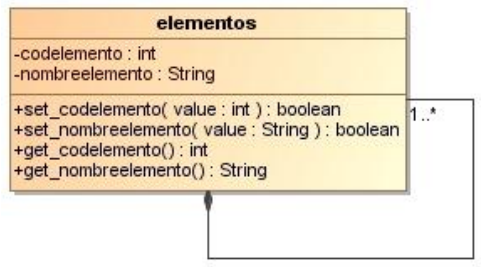
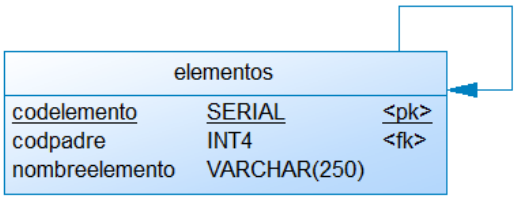
Fuente: Autores

3.3 Diseño reutilizable propuesto

La solución de diseño propuesta cumple con los requerimientos iniciales y puede ser reutilizado en otros contextos, a continuación se presenta el modelo de clases, su vista relacional (ver figura 1) y los posibles escenarios en donde puede llegar a ser utilizado (ver tabla 4).

[5] Ver apartado "*4.3 Alternativas de arquitecturas del componente software*"

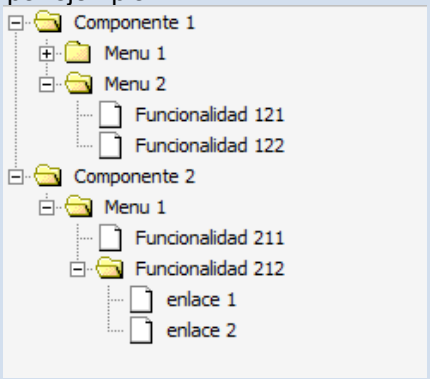
Figura 1. Modelo de clases

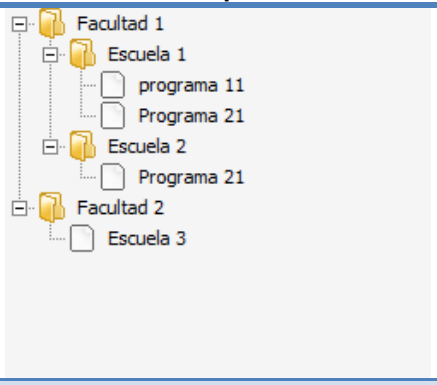
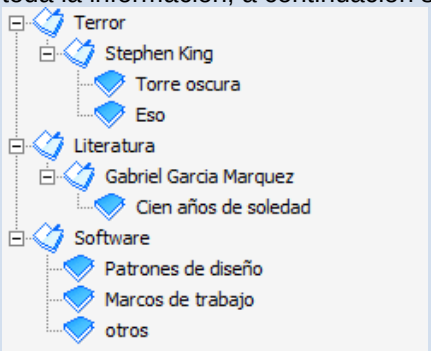
	
La clase elementos define las propiedades privadas y métodos de acceso a cada una de ellas de acuerdo al modelo EJB3, o lo que es lo mismo bajo el modelo de desarrollo del marco de trabajo MOON.	La tabla elementos puede tener muchas implementaciones en motores de bases de datos relacionales, obsérvese que existe una relación recursiva que obedece al diseño de clases para la relación de composición.

Fuente: Autores

Los escenarios de implementación a nivel de código para este diseño son muchos. La tabla 4 presenta tres (3) de los escenarios posibles, de esta manera el lector puede visualizar en cuantos contextos posibles puede llegar a reutilizar los requerimientos, el diseño, el código y las pruebas para todo el desarrollo propuesto en el presente proyecto.

Tabla 4. Escenarios aplicación componente

Escenarios	Descripción
Escenario 1	Este componente responde a un requerimiento de un sistema que opere por funcionalidades, la recursividad del modelo relacional le permite trabajar con múltiples niveles, de esta forma un administrador puede definir árboles de permisos por usuarios, por ejemplo: 
Escenario 2	Si el componente a realizar tiene que ver con dependencias, la propuesta de este proyecto es útil en un 100%. El modelo le permite administrar dependencias u oficinas definiendo Metadatos y datos en la misma tabla. En el siguiente ejemplo se puede observar la estructura libre que puede llegar a manejar este componente:

Escenarios	Descripción
	
Escenario 3	Otro escenario posible es en la creación de sistemas que gestionan información editorial, el caso de los libros es un ejemplo típico, un libro está compuesto por capítulos, sin embargo, se pueden agrupar los libros en categorías y permitir al usuario utilizar el sistema como prefiera, lógicamente proveyéndole una interfaz de usuario accesible a toda la información, a continuación se ilustra el escenario: 

Fuente: Autores

Una vez identificado el componente y de justificar sus bondades desde el punto de vista de reutilización de requisitos, diseño, código y pruebas, se procede con la implementación del componente teniendo como base el "*Proceso de Desarrollo de Componentes Software Reutilizable*". En el apartado "*4. Aplicación del proceso de desarrollo propuesto por [Vera Rivera]*" se puede observar paso a paso el proceso de desarrollo.

4. APLICACIÓN DEL PROCESO DE DESARROLLO PROPUESTO POR [VERA RIVERA]

4.1 Análisis de requisitos del dominio de la aplicación

Establecer el dominio de la aplicación, según [Vera Rivera], es un trabajo que debe realizar un arquitecto de software pues es la persona que más tiene conocimiento de la empresa y de los sistemas que esta requiere. Dado que el componente a desarrollar va hacer parte integral del núcleo del Marco de Trabajo MOON lo ideal era entrevistar al Arquitecto y diseñador del Marco de Trabajo, con el fin de tener una idea clara y precisa en cuanto a las necesidades del desarrollo.

La tabla 5 presenta el análisis de los criterios de evaluación del componente (***propuestos por [VERA RIVERA]***), pensando en el uso que pueda llegar a tener en una biblioteca de componentes para el Marco de Trabajo MOON.

Tabla 5. Criterios de evaluación áreas de aplicación

No.	Criterio	Evaluación	Justificación
1	Grado de utilización	5	Moon no posee en la actualidad un componente que de soporte a tablas recursivas, es ideal para dar soluciones a funcionalidad y permisos.
2	Dificultad de implementación	5	El componente puede ser desarrollado si se conoce el modelo propuesto en el trabajo de maestría y se conoce el Marco de Trabajo MOON.
3	Complejidad	5	El sistema en si es complejo, pero el uso de MOON disminuye ese factor, MOON posee estrategias para desarrollar modelos recursivos de manera rápida.
4	Cambiabilidad	3	Existe la posibilidad de cambiar los requerimientos, toda vez que el componente puede aplicarse en múltiples contextos que requieran de un modelo recursivo.
5	Nivel de abstracción	5	Tiene un nivel de abstracción alto pues usa el patrón de diseño FACADE para encapsular la lógica del componente.
6	Necesidad de Implementación	5	El componente no existe y es una necesidad para el Marco de Trabajo.
7	Posibilidad de componente ya desarrollado	5	No existe este componente en MOON.
	Total	**33**	

Fuente: Autores

Sobre el análisis inicial, y sin especificar requerimientos formales se tuvo obtuvo un valor de 33 puntos para el análisis de evaluación de componentes, este valor supera el umbral de 30, el cual es el valor mínimo según [Vera Rivera] para iniciar un proceso de desarrollo de este tipo.

4.1.1 Modelo de requisitos

En esta fase se define el detalle del dominio de la aplicación y los diagramas de casos de uso, para obtener este producto se utilizan plantillas para el levantamiento de requerimientos en procesos de desarrollo reales.

Diagrama de casos de uso

La figura 2 presenta el diagrama de casos de uso para el componente especificado.

Figura 2. Casos de uso componente

Fuente: Autores

Requisito 001

Caso de uso	Crear Perfil
Actor	Administrador
Propósito	Permitir al actor crear un perfil del sistema.
Precondiciones	Teniendo en cuenta que es una asociación simple, no está condicionado a otro requisito.
Flujo de eventos	Una vez el usuario ingresa a la aplicación encuentra el menú Administración => Perfiles y selecciona la opción Nuevo, el sistema debe presentar una caja de texto y un botón de acción para ingresar el nuevo perfil.
Interfaz de usuario	La interfaz del usuario debe heredar todo el modelo de menús del sistema que esté utilizando el requisito. A continuación se ilustra un ejemplo de una posible salida admitida para este requisito: **Formulario Creación** Nombre del perfil: Crear Perfil
Respuesta del sistema	Al hacer clic en el menú que enlaza a este requisito el sistema debe realizar lo siguiente: 1. Debe presentar una caja de texto y un botón para permitir el

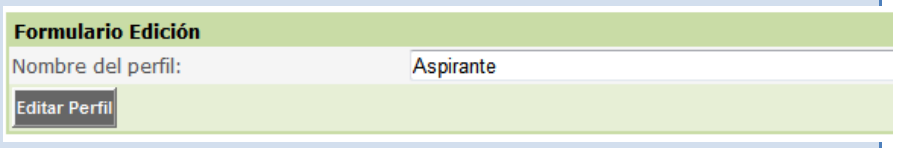
Caso de uso	Crear Perfil
	proceso de grabación. 2. Agregar un perfil haciendo clic en el botón "Crear Perfil".
Excepciones	Ninguna en fase de desarrollo.

Requisito 002

Caso de uso	Administrar Perfiles
Actor	Administrador
Propósito	El sistema debe imprimir el listado de perfiles registrados en el la base de datos.
Precondiciones	Deben existir perfiles registrados en la base de datos.
Flujo de eventos	Una vez el usuario ingresa a la aplicación encuentra el menú Administración => Perfiles y selecciona la opción Administrar, el sistema debe presentar un listado de perfiles con dos opciones: eliminar perfil y editar perfil (en caso de tener esos permisos).
Interfaz de usuario	La interfaz del usuario debe heredar todo el modelo de menús del sistema que esté utilizando el requisito. La figura ilustra un ejemplo de una posible salida admitida para este requisito:  Nombre del Perfil Buscar: Iniciar búsqueda Mostrando del 1 al 7 de 7 registros. En 1 páginas. Cod | Nombre Perfil | Usuarios | Funcionalidades 7 Aspirante 0 0 6 Administrativo 4 0 5 Estudiante 53,305 0 4 Docente 1,699 0 3 Coordinador 0 0 2 Directivo 0 0 1 Administrador 0 0
Respuesta del sistema	Al hacer clic en el menú que enlaza a este requisito el sistema debe realizar lo siguiente: 1. Consultar todos los perfiles que existen en la base de datos. 2. Si el usuario tiene permisos necesarios debe incluir dos (2) requisitos adicionales.
Excepciones	Ninguna en fase de desarrollo

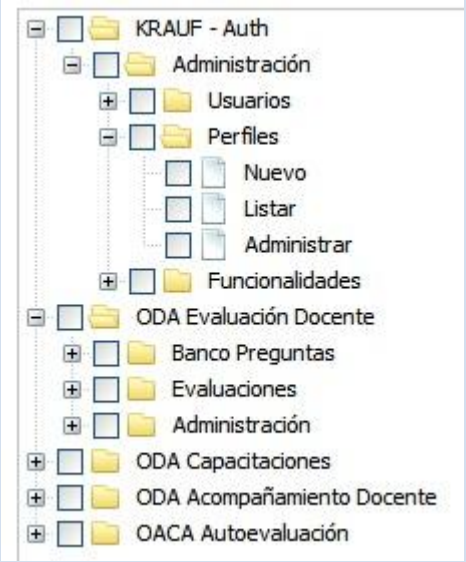
Requisito 003

Caso de uso	Editar Perfil
Actor	Usuario
Propósito	Permitir al actor actualizar los datos de un perfil del sistema.
Precondiciones	Teniendo en cuenta que es un "*include*" debe estar activado el Requisito 002.
Flujo de eventos	Debe seleccionar la opción editar perfil y el sistema debe presentar una caja de texto y un botón de acción para ingresar los nuevos datos del perfil.
Interfaz de usuario	La interfaz del usuario debe heredar todo el modelo de menús del sistema que esté utilizando el requisito. A continuación se ilustra un

Caso de uso	Editar Perfil
	ejemplo de una posible salida admitida para este requisito: 
Respuesta del sistema	Al hacer clic en el menú que enlaza a este requisito el sistema debe realizar lo siguiente: 1. Debe presentar una caja de texto y un botón para permitir el proceso de actualización. 2. Editar un perfil haciendo clic en el botón "Editar Perfil".
Excepciones	Ninguna en fase de desarrollo.

Requisito 004

Caso de uso	Visualizar Árbol
Actor	Administrador
Propósito	El sistema debe imprimir un árbol completo con las funcionalidades existentes en la base de datos, debe mantener una visión multinivel de todos los elementos.
Precondiciones	Deben existir funcionalidades registradas en la base de datos. El inicio de sesión en el sistema es necesario pues se requiere identificar el perfil del usuario para asignar funcionalidades.
Flujo de eventos	Una vez el usuario ingresa a la aplicación encuentra el menú Administración => Perfiles y selecciona la opción Administrar, el sistema debe presentar un listado de perfiles con dos opciones: eliminar perfil y editar perfil (en caso de tener esos permisos). Debe seleccionar la opción editar perfil y el sistema debe presentar dos pestañas: Datos Básicos y Funcionalidades, debe hacer clic en funcionalidades y debe visualizarse el árbol de esos elementos.
Interfaz de usuario	La interfaz del usuario debe heredar todo el modelo de menús del sistema que esté utilizando el requisito. La figura ilustra un ejemplo de una posible salida admitida para este requisito:

Caso de uso	Visualizar Árbol
	
Respuesta del sistema	Al hacer clic en el menú que enlaza a este requisito el sistema debe realizar lo siguiente: 1. Consultar de manera recursiva todas las funcionalidades que existen en la base de datos. Se requiere que el llamado de niveles se realice mediante AJAX para no cargar el servidor en una sola petición. 2. Aunque no es obligatoria la vista de árbol si es necesario que la apariencia sea similar para facilidad del usuario final.
Excepciones	Ninguna en fase de desarrollo

Requisito 005

Caso de uso	Seleccionar Funcionalidad
Actor	Administrador
Propósito	Permitir al actor seleccionar mediante objetos tipos ***CheckBox*** cualquier funcionalidad del árbol.
Precondiciones	Debe estar activado el Requisito 004.
Flujo de eventos	Una vez el usuario despliega el árbol de funcionalidades, éste debe disponer de casillas de verificación para poder seleccionar un nodo padre o sus nodos hijos.
Interfaz de usuario	La interfaz del usuario debe heredar todo el modelo de menús del sistema que esté utilizando el requisito. A continuación se ilustra un ejemplo de una posible salida admitida para este requisito.

Caso de uso	Seleccionar Funcionalidad
	
Respuesta del sistema	Al hacer clic en el menú que enlaza a este requisito el sistema debe realizar lo siguiente: 1. Consultar de manera recursiva todos los elementos que existen en la base de datos. Se requiere que el llamado de niveles de realice mediante AJAX para no cargar el servidor en una sola petición. 2. Presentar casillas de verificación para la selección de los elementos. 3. La selección debe ser sin recursividad, así el usuario selecciona lo que desea. 4. Debe extender dos (2) requisitos adicionales.
Excepciones	Ninguna en fase de desarrollo

Requisito 006

Caso de uso	Agregar Funcionalidad
Actor	Administrador
Propósito	Permitir al actor seleccionar mediante objetos tipos ***CheckBox*** cualquier funcionalidad del árbol.
Precondiciones	Debe estar activado el Requisito 005.
Flujo de eventos	Una vez el usuario despliega el árbol de funcionalidades, éste debe disponer de casillas de verificación para poder seleccionar un nodo padre o sus nodos hijos.
Interfaz de usuario	La interfaz de usuario es la misma del requisito 005, sin embargo, la implementación de este caso de uso se diferencia del 005 sólo a nivel de código.
Respuesta del sistema	Al hacer clic en el menú que enlaza a este requisito el sistema debe realizar lo siguiente: 1. Consultar de manera recursiva todos los elementos que existen en la base de datos. Se requiere que el llamado de niveles de realice mediante AJAX para no cargar el servidor en una sola petición. 2. Presentar casillas de verificación para la selección de los elementos. 3. La selección debe ser sin recursividad, así el usuario selecciona lo

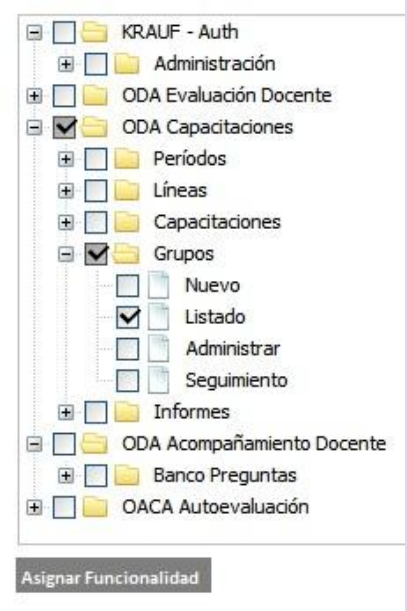
Caso de uso	Agregar Funcionalidad
	que desea.
Excepciones	Ninguna en fase de desarrollo

Requisito 007

Caso de uso	Eliminar Funcionalidad
Actor	Administrador
Propósito	Permitir al actor seleccionar mediante objetos tipos ***CheckBox*** cualquier funcionalidad del árbol.
Precondiciones	Debe estar activado el Requisito 005.
Flujo de eventos	Una vez el usuario despliega el árbol de funcionalidades, éste debe disponer de casillas de verificación para poder eliminar un nodo padre o sus nodos hijos.
Interfaz de usuario	La interfaz de usuario es la misma del requisito 005, sin embargo, la implementación de este caso de uso se diferencia del 005 sólo a nivel de código, debido a que en este requisito se quitan funcionalidades que estaban seleccionadas con anterioridad.
Respuesta del sistema	Al hacer clic en el menú que enlaza a este requisito el sistema debe realizar lo siguiente: 1. Consultar de manera recursiva todos los elementos que existen en la base de datos. Se requiere que el llamado de niveles de realice mediante AJAX para no cargar el servidor en una sola petición. 2. Presentar casillas de verificación para la eliminación de los elementos. 3. La eliminación debe ser sin recursividad, así el usuario quita lo que desea.
Excepciones	Ninguna en fase de desarrollo

Requisito 008

Caso de uso	Asignar Funcionalidad
Actor	Administrador
Propósito	Permite asignar o quitar una funcionalidad al perfil seleccionado.
Precondiciones	Teniendo en cuenta que es un "*include*" debe estar activado el Requisitos 005.
Flujo de eventos	Una vez el usuario ingresa a la aplicación encuentra el menú Administración => Perfiles y selecciona la opción Administrar, el sistema debe presentar un listado de perfiles con dos opciones: eliminar perfil y editar perfil (en caso de tener esos permisos). Debe seleccionar la opción editar perfil y el sistema debe presentar dos pestañas: Datos Básicos y Funcionalidades, debe hacer clic en funcionalidades y debe visualizarse el árbol de esos elementos, debe estar habilitado la opción de chequear los nodos del árbol.
Interfaz de usuario	La interfaz del usuario debe heredar todo el modelo de menús del sistema que esté utilizando el requisito. La figura ilustra un ejemplo de una posible salida admitida para este requisito:

Caso de uso	Asignar Funcionalidad
	
Respuesta del sistema	Al hacer clic en el menú que enlaza a este requisito el sistema debe realizar lo siguiente: 1. Consultar de manera recursiva todas las funcionalidades que existen en la base de datos. Se requiere que el llamado de niveles se realice mediante AJAX para no cargar el servidor en una sola petición. 2. Aunque no es obligatoria la vista de árbol si es necesario que la apariencia sea similar para facilidad del usuario final. 3. Debe estar habilitada la opción de chequear cada una de las funcionalidades, debe incluir un requisito adicional. 4. Debe presentar un botón con el letrero “Asignar Funcionalidad” y el actor debe hacer clic para asignar las funcionalidades seleccionadas en el paso 3.
Excepciones	Ninguna en fase de desarrollo

Requisito 009

Caso de uso	Eliminar Perfil
Actor	Administrador
Propósito	Permitir al actor eliminar un perfil.
Precondiciones	Teniendo en cuenta que es un “*include*” debe estar activado el Requisito 002.
Flujo de eventos	Una vez el sistema despliega el listado de perfiles debe existir una opción que permita eliminar el perfil que esté seleccionado. La confirmación se hace necesaria, pues el usuario puede cometer errores a la hora de eliminar el elemento.
Interfaz de usuario	La interfaz del usuario debe heredar todo el modelo de menús del sistema que esté utilizando el requisito. Para eliminar un elemento se debe disponer de una única opción para eliminar.
Respuesta del sistema	Al hacer clic en el menú que enlaza a este requisito el sistema debe realizar lo siguiente:

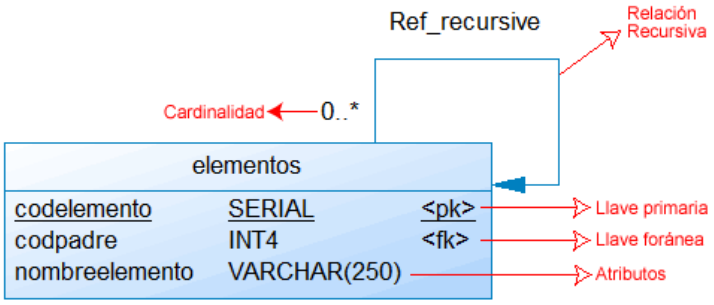
Caso de uso	Eliminar Perfil
	1. Consultar todos los perfiles que existen en la base de datos. 2. En la parte derecha del listado debe existir una opción que permita eliminar un elemento seleccionado
Excepciones	Ninguna en fase de desarrollo

4.2 Modelo de selección de componentes reutilizables

4.2.1 Modelo del dominio problema

En esta fase del proceso de desarrollo [Vera Rivera] propone determinar las entidades, atributos, llaves primarias, relaciones con otras entidades y la multiplicidad de las relaciones. Para el desarrollo del presente componente existe una entidad llamada “Elementos” la cual tiene una relación de recursividad, esto hace que cualquier otra entidad se elimine del modelo de dominio. La figura 3 presenta cada uno de estos elementos identificados en esta fase.

Figura 3. Modelo entidad-relación

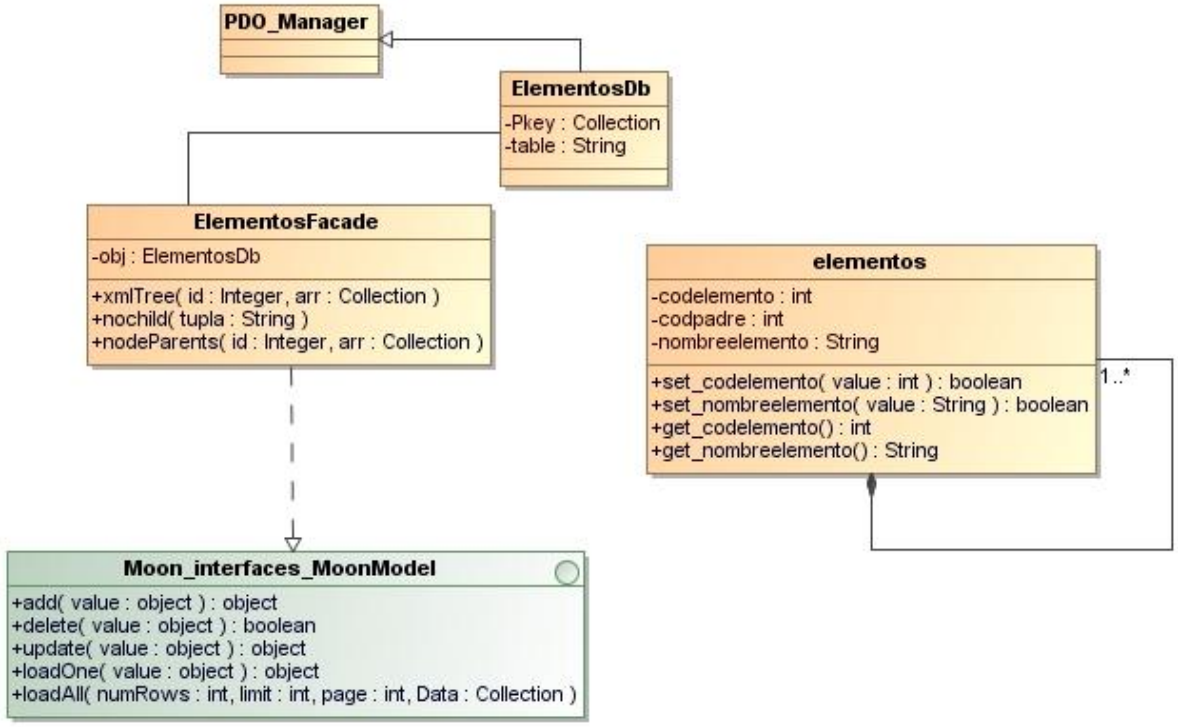


Fuente: Autores

4.2.2 Modelo de análisis

En esta fase del proyecto se definen las clases a utilizar en el desarrollo, de acuerdo a la propuesta, este proyecto utiliza un Marco de Trabajo con una tecnología diferente a JAVA. La figura 4 presenta las interfaces, la clase ***Base*** del modelo de persistencia, las fachadas y las entidades a utilizar en el componente a desarrollar.

Figura 4. Modelo de clases

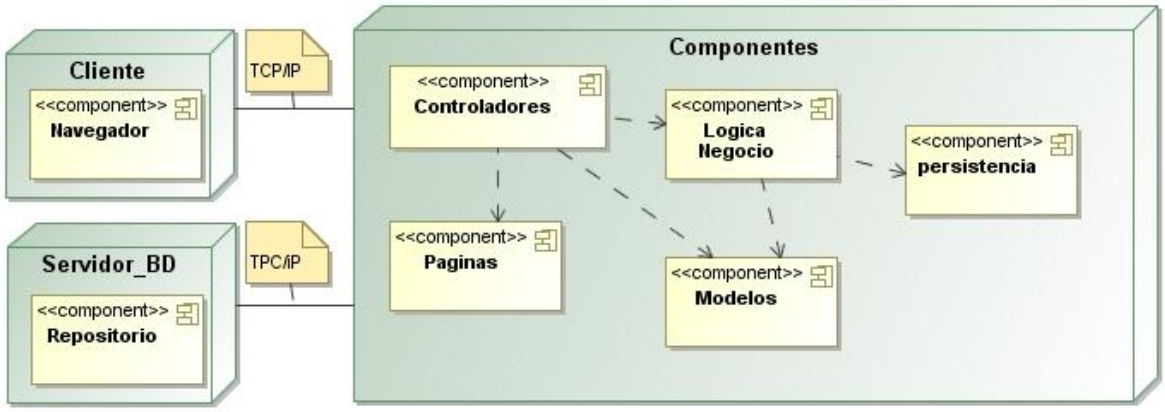


Fuente: Autores

4.2.3 Modelo de componentes

El modelo de componentes hereda del Marco de Trabajo el Modelo de Capas, en la figura 5 se presenta el modelo de componentes para el presente proyecto.

Figura 5. Modelo de componentes



Fuente: Autores

Como se puede observar en la figura 5, la capa cliente y la capa servidor BD pueden ser de cualquier tipo, en el caso de los clientes un navegador, un celular, o cualquier dispositivo que se comunique con el componente vía TCP/IP. Para el caso de los repositorios es igual, la base de datos puede ser Oracle, Postgres, MySql o cualquier otra, esto se debe a que la

persistencia no la manejará el componente a desarrollar sino un conjunto de entidades propias del marco de trabajo MOON.

4.2.4 Análisis de reutilización

Teniendo en cuenta que el componente a desarrollar fue definido y evaluado desde las primeras fases, se procede entonces a analizar el factor de reutilización de tal manera que este componente al integrarse con el Núcleo de MOON pueda ser utilizado en otros desarrollos. Los criterios y las escalas son definidos por [VERA RIVERA], el detalle se puede apreciar en la tabla 6.

Tabla 6. Criterios de selección de componentes

No.	Criterios	Calificación	Justificación
1	Dependencia estructural	5	El componente es independiente, no depende de otras configuraciones y/o requerimientos externos, y sólo gestiona la información que le corresponde para el conjunto de entidades u objetos que se estén utilizando.
2	Facilidad de Mantenimiento	5	EL componente se soporta en el modelo MOON que hereda la filosofía de trabajo de JAVA, esto quiere decir que el mantenimiento que se debe realizar será sobre clases depuradas y probadas.
3	Grado de adaptabilidad	5	El componente debe integrarse primero con el marco de trabajo, una vez logrado este proceso el grado de adaptabilidad hacia cualquier desarrollo será del 100%
4	Grado de extensibilidad	4	El grado de dificultad para este ítem es bajo pues el impacto de añadir nuevas funcionalidades implica la creación de nuevos métodos, en el caso de los ya existentes se deben implementar sobrecargas.
5	Grado de utilización	5	Cualquier desarrollo que utilice MOON puede utilizar este componente.
6	Portabilidad	5	Al ser integrado en un Marco de Trabajo el componente hereda las características de portabilidad, esto quiere decir que puede funcionar sobre arquitecturas de 132 y 64 bits y sobre plataformas Windows y/o Linux.
7	Complejidad	2	La complejidad a la hora de reutilizar el componente es baja, se requiere la inclusión de librerías y el manejo de objetos con mensajes definidos y estructurados.
8	Cambiabilidad	3	Para requerimientos nuevos el cambio debe implementarse como una adición pues el núcleo del componente opera con características previas ya definidas.
9	Posibilidad de componente ya desarrollado	5	Este componente es totalmente nuevo, inicialmente se pensó en la posibilidad de un COTS, sin embargo, el problema de integrarlo al marco de trabajo dio como resultado el requerimiento para un nuevo desarrollo.
	Total	**39**	

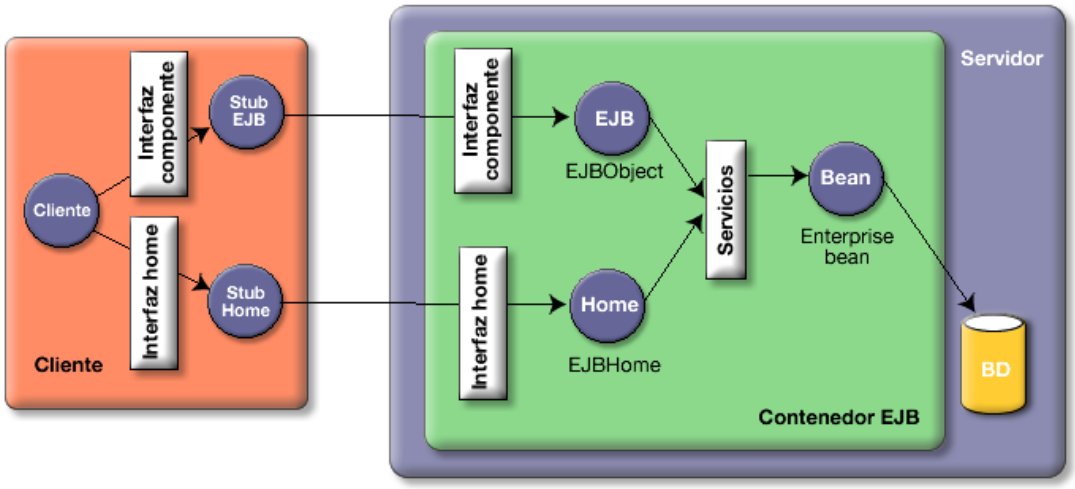
Fuente: Autores

El resultado del análisis de reutilización entrego un puntaje de **39** puntos, sobrepasando el umbral definido por [VERA RIVERA], el cual establece un puntaje mínimo de 30 puntos para que el componente tenga potencial de reutilización.

4.3 Alternativas de arquitecturas del componente software

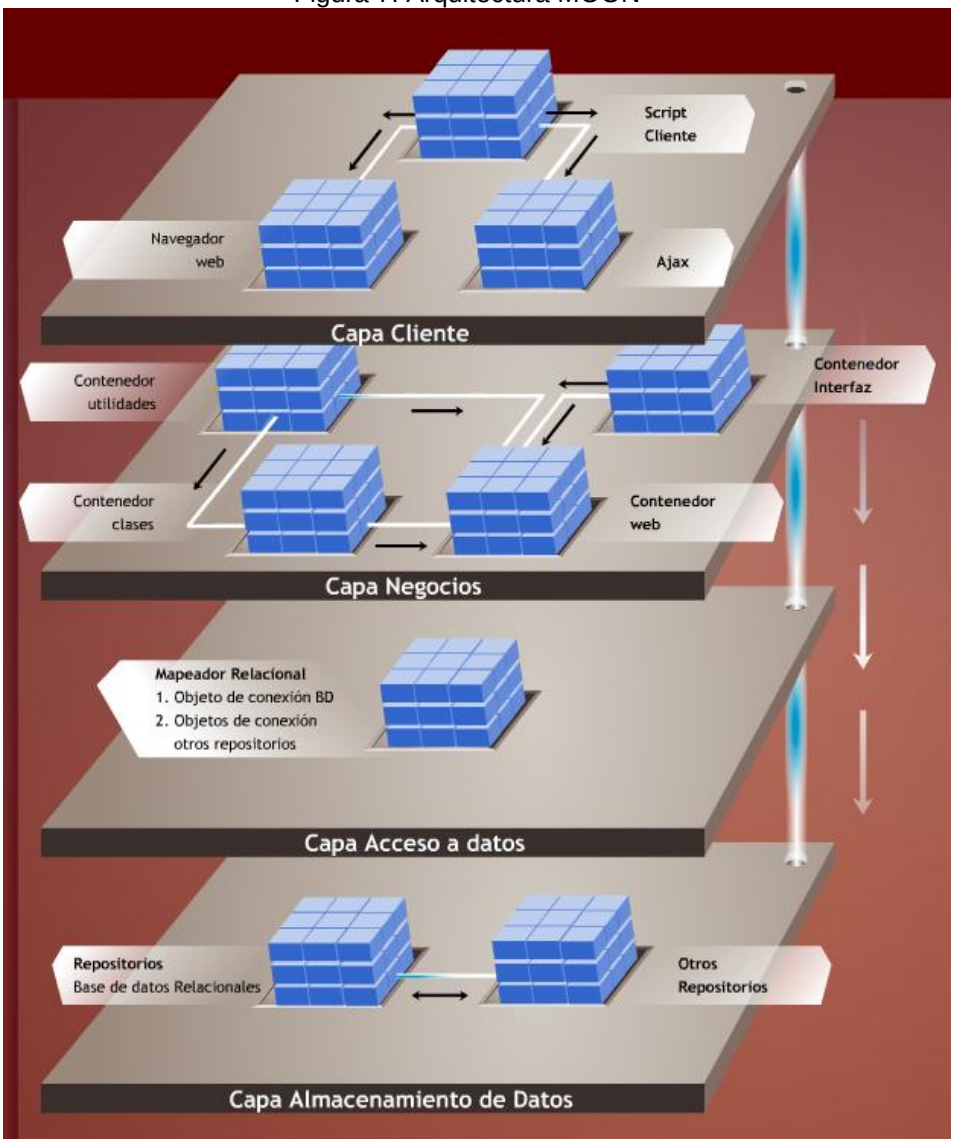
La arquitectura utilizada en el componente tiene como referencia la propuesta de JAVA EE 5, con algunas adecuaciones como por ejemplo: El patrón EAO de java (Entity Access Objetc) es reemplazado por el patrón AO (Access Object) definido por MOON. La comunicación entre las capas de presentación y las clases contenedores se realiza por medio del patrón de diseño FACADE, esto es tanto en JAVA como en MOON. La arquitectura del componente sin estar desarrollado con JAVA cumple con la propuesta de ORACLE para el desarrollo de aplicaciones Web. La figura 6 presenta la arquitectura base del componente.

Figura 6. Arquitectura JAVA



En la figura 6 se puede observar en color amarillo el *repositorio*, posteriormente en color verde se encuentran los *contenedores* y en color rojo claro *el cliente*. Hay una relación directa entre esta arquitectura y la propuesta por MOON, en la figura 7 se pueden observar los mismos componentes: Capa de almacenamiento también llamada repositorios, la persistencia y el conjunto de contenedores que prestan todos los servicios necesarios al aplicativo a desarrollar, por último se encuentra la capa cliente, encargada de presentar las vistas que el usuario requiere. A simple vista los componentes coinciden en organización sin embargo el detalle técnico y la forma de conectar cada elemento hace parte de la filosofía de desarrollo de cada Arquitectura. En la figura 7 se puede observar la arquitectura propuesta por el Marco de Trabajo utilizado.

Figura 7. Arquitectura MOON



Fuente: Autores

4.4 Procedimiento para el desarrollo, pruebas y especificación del componente

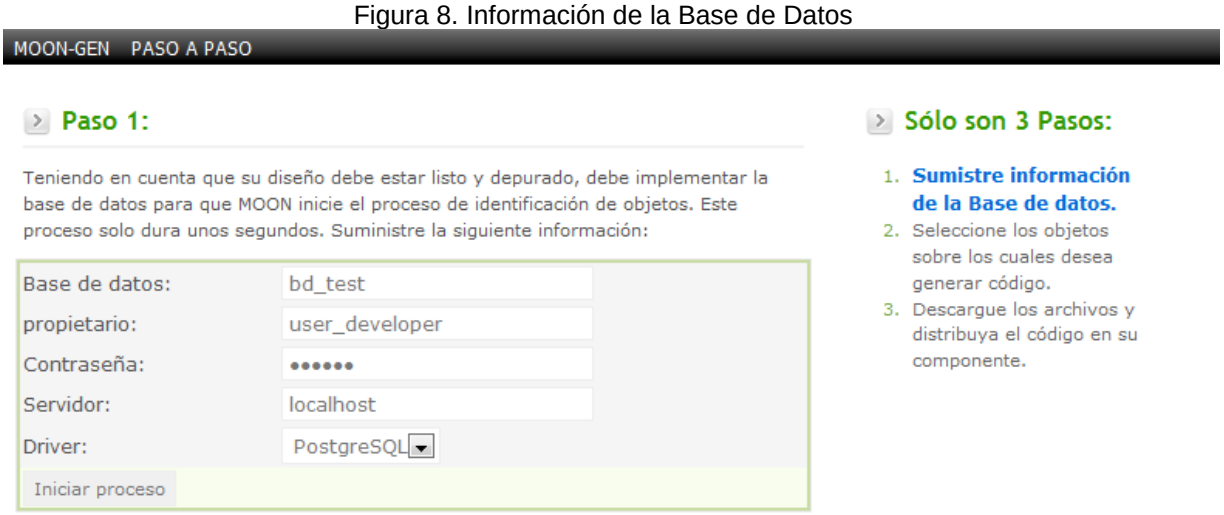
Hay una frase muy común entre los desarrolladores: "*Java está hecho en Java*", pues bien, sin ser MOON un lenguaje de programación, provee todos los mecanismos para que un componente pueda ser diseñado y desarrollado bajo estándares de código, utilización de patrones de diseño y un nivel de abstracción que facilita la reutilización de todo el desarrollo.

En cuanto a las herramientas, se seleccionó POSTGRES como motor de bases de datos relacional de tipo OPEN SOURCE, como interfaz de desarrollo NETBEANS el cual es el entorno de desarrollo OPEN SOURCE para JAVA, como creador y validador de entidades se utilizaron los componentes MOON_GEN y MOON_UNIT.

4.4.1 Implementación de entidades

Para realizar la implementación de las entidades se debe definir la persistencia y el mapeo de las entidades contra cada una de las tablas que tienen la base de datos relacional. Para realizar este proceso se hace uso del componente MOON-GEN el cual provee un asistente en línea (el proceso se puede realizar desde internet) que guía al desarrollador en la generación de clases. La figura 8 presenta el primer paso del proceso de implementación de entidades.

Figura 8. Información de la Base de Datos



Fuente: Autores

Todos los campos de la figura 8 son obligatorios, primero se debe suministrar el nombre del repositorio, el propietario de la base de datos, la contraseña del propietario, el servidor en donde se encuentra la base (si reside en el servidor de producción no es necesario indicar la dirección IP), por último el tipo de manejador de base de datos.

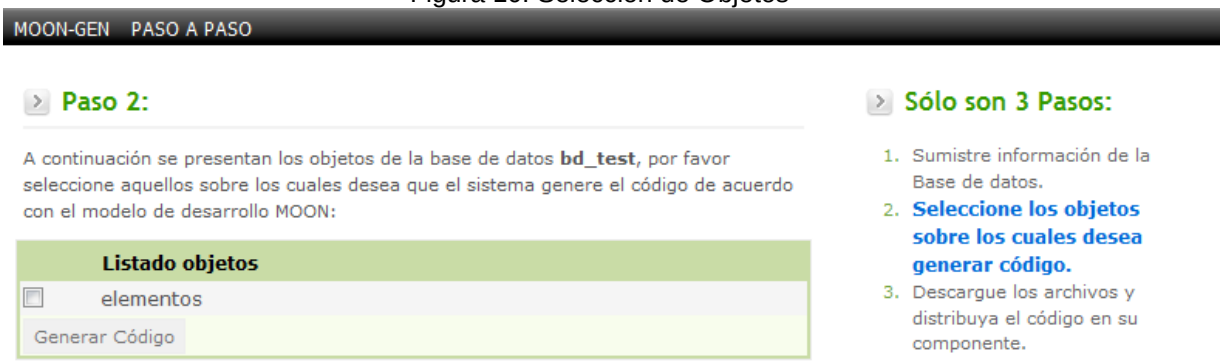
Si la información es correcta el sistema permite avanzar al siguiente paso, en caso contrario presenta el mensaje respectivo indicando el tipo de fallo ocurrido. La figura 9 presenta un ejemplo con los tipos de error de MOON-GEN:

Figura 9. Tipificación de errores

Fuente: Autores

El segundo paso consiste en la selección de los objetos sobre los cuales se desea generar el código. La figura 10 presenta los elementos disponibles para el componente en construcción.

Figura 10. Selección de Objetos

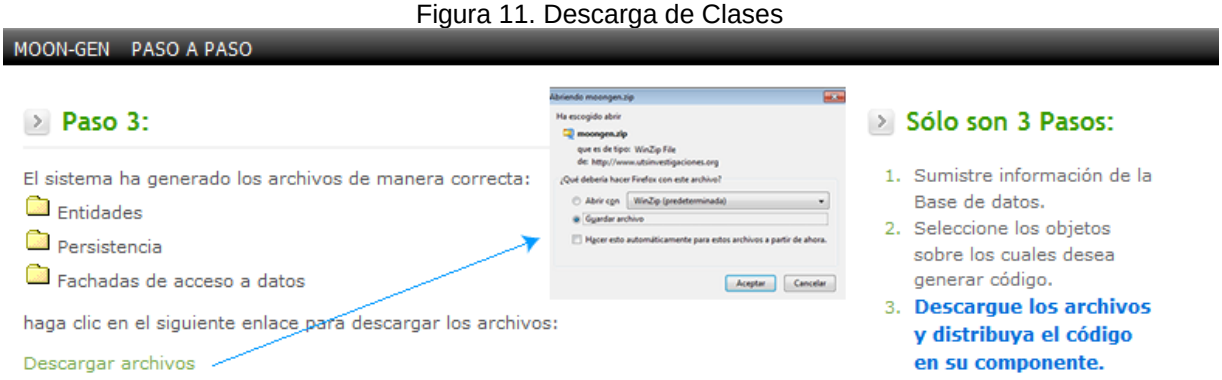


Fuente: Autores

Teniendo en cuenta que el diseño definió una clase de forma recursiva, es lógico que la implementación de la base de datos tenga la misma representación. La figura 1 presenta la relación entre el diagrama de clases y el modelo entidad relación del componente realizado. En este paso no es necesario asignar claves ni suministrar información, sencillamente se selecciona el objeto y se procede a generar el código.

El paso final consiste en la descarga del archivo empaquetado que contiene el código listo para ser utilizado en el desarrollo. La figura 11 presenta el resultado del proceso.

Figura 11. Descarga de Clases

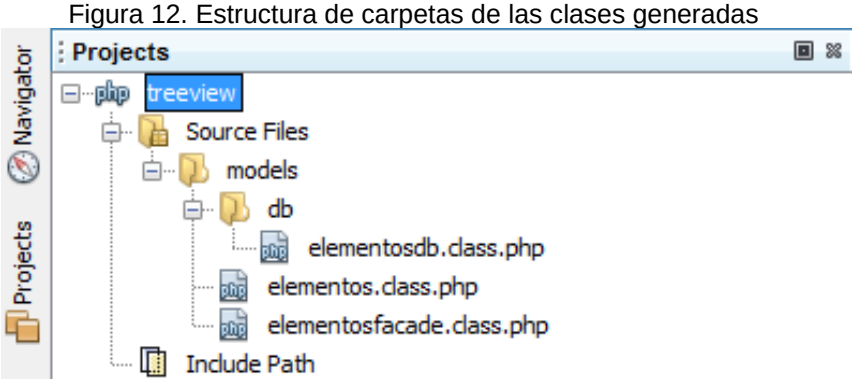


Fuente: Autores

La compilación de los archivos generados, así como el uso de aplicativos adicionales para compilar o empaquetar el código generado no son necesarios. JAVA utiliza ANT para realizar operaciones finales en el proceso de creación de archivos, en MOON esto es automático.

Como herramienta de desarrollo se utilizó NETBEANS, por ende la estructura de carpetas, la organización del proyecto y en general la presentación de la información sigue los requerimientos básicos de la misma. La figura 12 presenta la estructura del código realizado:

Figura 12. Estructura de carpetas de las clases generadas



Fuente: Autores

El modelo de referencia para el desarrollo de aplicaciones con MOON es el "Modelo, Vista, Controlador -MVC-", en la figura 12 se aprecia claramente como las clases generadas están organizadas en la carpeta "***models***".

4.4.2 Prueba de Entidades - Clases

Para las pruebas se utilizó el proyecto creado en NETBEANS, y se realizaron de la siguiente manera:

1. Integrar el código generado con el Marco de trabajo para crear el escenario de desarrollo y pruebas.
2. Crear una página para la ejecución de las pruebas.
3. Crear un controlador para solicitar los servicios que podrá prestar el componente.
4. Ingresar a la página de prueba.
5. Verificar los resultados obtenidos.
6. Documentar los resultados.

En un controlador se creó un objeto del tipo "***Elementos***", y cada uno de los métodos de la clase fue probado. Los resultados de la prueba se presentan en la tabla 7.

Tabla 7. Resultados del mapeo de clases

Clases	Métodos probados	Resultado
Elementos	set_codelemento($value)	ok
	set_codpadre($value)	ok
	set_nombreelemento($value)	ok
	get_codelemento()	ok
	get_codpadre()	ok
	get_nombreelemento()	ok
ElementosFacade	add($obj)	ok
	update($obj)	ok
	loadOne($obj)	ok
	delete($obj)	ok
	add_fieldSearch($key, $field)	ok
	load_all(&$rsNumRows, $limit_numrows, $page, $Data=array())	ok

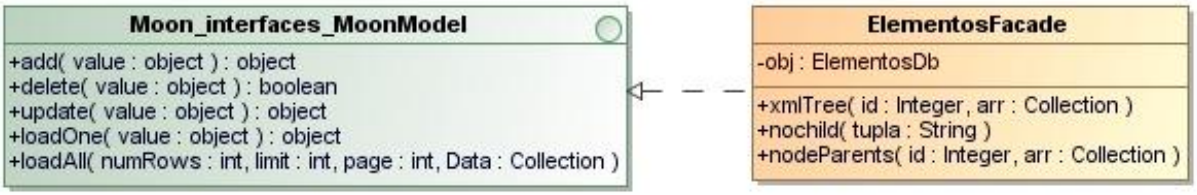
Fuente: Autores

El resultado de las pruebas fue satisfactorio, se concluyó que el mapeo de las clases que conforman el componente a desarrollar se generó de manera correcta de acuerdo al diseño realizado previamente.

4.4.3 Implementación de Servicios

Como se pudo apreciar en la fase de pruebas unitarias, el componente ya tiene servicios disponibles, esto se debe a que MOON-GEN generó las entidades, las fachadas y las clases de persistencia para proveer una forma de acceso de los datos a través del modelo de objetos. El patrón de acceso a datos se denomina AO (Access Object), éste define la forma como los objetos mapean la información a la base de datos. La figura 13 presenta los métodos que implementados en el patrón AO.

Figura 13. Implementación del Patrón AO



Fuente: Autores

Para implementar la totalidad de los servicios se crearon tres (3) métodos nuevos, con esto se dio cumplimiento a los requerimientos propuestos. El detalle de la implementación de los métodos se encuentra en la tabla 8.

Tabla 8. Métodos Adicionales

Clases	Definición	Nivel de Acceso	Descripción
Elementos	xmlTree($id, &$arrFunc)	public	Método **recursivo** que obtiene los nodos de la estructura jerárquica que sean solicitados
	noChild($tupla)	public	Método que verifica si uno o varios nodos tiene hijos. Con este se busca tener el control final del árbol recursivo a construir
	nodeParent($id, &$arrParents)	public	Método **recursivo** que obtiene toda la rama de una estructura jerárquica empezando en orden ascendente

Fuente: Autores

La implementación de los tres nuevos métodos permite extender la funcionalidad del componente para cualquier tipo de estructura jerárquica. Aunque en Postgres existe la posibilidad de realizar consultas recursivas, está funcionalidad no puede ser extendida a otras bases de datos, de esta forma el código se puede aplicar a cualquier base de datos relacional. La figura 14 presenta el código fuente del método ***xmlTree*** el cual posee dos parámetros, el primero es el identificador del nodo padre que va iniciar la estructura jerárquica y el segundo es una colección que permite identificar cuando una rama puede o no estar seleccionada.

Figura 14. Método recursivo xmlTree

```
public function xmlTree($id, &$arrFunc){
    $params = array($id);
    $arr = $this->GetAssoc($sql, $params);
    foreach($arr as $cod_func => $fields){
        $cod_padre = $fields["codpadre"];
        $nombre = utf8_decode($fields["nombre"]);
        $hijos = $fields["cant"];
        $checked = "";
        $keys = array_keys($arrFunc);
        if (in_array($cod_func, $keys)){
            $checked = " checked=\"1\"";
        }
        if ($hijos>0){
            echo "<item text=\"{$nombre}\" id=\"{$cod_func}\">\n";
            $this->xmlTree($cod_func, $arrFunc);
            echo "</item>\n";
        }else{
            echo "<item text=\"{$nombre}\" id=\"{$cod_func}\"{$checked} />\n";
        }
    }
}
```

Recursividad

Fuente: Autores

La figura 15 presenta la implementación de los otros des métodos que dan soporte a este componente.

Figura 15. Métodos Adicionales noChild y NodeParent

```
public function noChild($tupla){
    $arr = array();
    $arr_tmp = $this->GetAll($sql);
    foreach($arr_tmp as $index => $fields){
        if ($fields["cant"]==0){
            $arr[] = $fields["codfunc"];
        }
    }
    return $arr;
}
```

```
public function nodeParent($id, &$arrParents){
    $arrParents[$id] = $id;
    $parent = $this->GetOne($sql, array($id));
    if ($parent!=1){
        $arrParents[$parent] = $parent;
        $this->nodeParent($parent, $arrParents);
    }
}
```

Fuente: Autores

4.4.4 Prueba Funcional de los Servicios

La prueba funcional de los servicios es muy similar en su contexto a las pruebas realizadas a las clases generadas por MOON-GEN. Para estas pruebas se utilizó el proyecto creado en NETBEANS, y se realizaron de la siguiente manera:

1. Integrar el código generado con las clases anteriormente probadas y con el Marco de trabajo para crear el escenario de desarrollo y pruebas.
2. Crear una página para la ejecución de las pruebas.
3. Crear un controlador para solicitar los servicios que podrá prestar el componente.
4. Ingresar a la página de prueba.

5. Verificar los resultados obtenidos.
6. Documentar los resultados.

En un controlador se creó un objeto del tipo "***Elementos***", y cada uno de los tres métodos fue probado. Los resultados de la prueba se presentan en la tabla 9.

Tabla 9. Pruebas unitarias de los servicios implementados

Clases	Métodos probados	Resultado
Elementos	xmlTree($id, &$arrFunc)	ok
	noChild($tupla)	ok
	nodeParent($id, &$arrParents)	ok

Fuente: Autores

Una vez realizadas las pruebas unitarias de Clases y las pruebas unitarias de los servicios implementados se procedió a realizar la verificación del componente contra los requerimientos iniciales.

Tabla 10.Verificación contra requerimientos

Nro	Requisito	Servicio implementado	Resultado
1	01 Listar Elementos	load_all(&$rsNumRows, $limit_numrows, $page, $Data=array())	Ok
2	02 seleccionar elementos	xmlTree($id, &$arrFunc)	Ok
3	03 crear elementos	add($obj)	Ok
4	04 actualizar elementos	update($obj)	Ok
5	05 eliminar elementos	delete($obj)	Ok
6	06 obtener nodos	nodeParent($id, &$arrParents)	Ok
7	07 obtener árbol	xmlTree($id, &$arrFunc)	Ok

Fuente: Autores

Con las pruebas realizadas hasta el momento se puede concluir que:

1. Las clases (entidades) funcionan correctamente y son la representación del diagrama de clases realizado inicialmente.
2. Los servicios implementados funcionan correctamente.
3. El componente desarrollado cumple con los requerimientos propuestos.

4.4.5 Implementación de los Servicios Web

La implementación de los servicios Web utiliza como referencia el modelo MVC aplicado en el patrón MOON, esto quiere decir que la organización y la independencia se debe reflejar no sólo en la forma de organizar el código (estructura de carpetas), sino también en la forma como cada una de ella accede a la otra. Para implementar los servicios Web MOON propone en su modelo de desarrollo la creación de un controlador que instancie los objetos requeridos de las clases y que posteriormente le entregue la información de alguna manera a la vista. La figura 16 presenta el controlador que obtiene como resultado una estructura jerárquica para luego ser visualizada en el componente que gestiona la vista.

Figura 16. Controlador que implementa los servicios

```php
<?php
$ids_controller = array("AUTH_ADM_PER_ADM");
$Func = Modules_Auth_Model_Func::get_Instance($SESION3);
$Func->verify_id($ids_controller, $PATH_CONFIG["APP"].$PATH_CONFIG["FOLDER"]["share"]);

$Parameters = new Moon_Params_Parameters();
$Parameters->verify("GET",true);
$action = $Parameters->get_parameter("action", "");

$ElementoFacade = new Modules_Auth_Model_FuncFacade();
echo $ElementoFacade->xmlTree("1", array());
?>
```

Fuente: Autores

En (1) se encuentran los lineamientos generales de MOON para la creación de un controlador: Primero el tipo de permiso que debe tener el usuario para acceder al controlador, en segunda instancia obtiene los permisos sobre los cuales el usuario tiene acceso, y en tercer lugar verifica que realmente existan permisos para manipular el controlador. Este paso no es obligatorio si no se utiliza el Modelo de Seguridad por funcionalidades que provee MOON.

En (2) se realiza la gestión de peticiones al controlador, este controlador particularmente solo acepta peticiones de tipo GET y tiene como parámetro un "***action***" el cual define lo que debe realizar el controlador.

En (3) se implementa el servicio Web solicitado este paso es muy similar a la prueba funcional de los servicios, la diferencia radica en que en este paso el controlador gestiona políticas de seguridad y está listo para entregarle la información a la Vista.

4.4.6 Pruebas de los Servicios Web

Para la prueba del servicio Web es necesario permitir que un usuario consuma el servicio, los pasos son los siguientes:

1. Declarar la referencia Web (dirección URL) al controlador que le permitirá cargar el servicio, para este caso la ruta será relativa al punto en donde se encuentre el controlador creado para consumir el servicio.
2. Se utiliza proxy transparente, por tanto, no es necesario el uso de un método que obtenga el puerto sobre el cual se consumirá el servicio.
3. Invocar el método "xmlTree" con los parámetros requeridos.

Para este caso el resultado de la prueba puede ser visualizado sin implementar los componentes a nivel de interfaz, esto se debe a que el servicio provee un archivo en formato XML, el cual puede ser visualizado en cualquier navegador o editor de texto. La figura 17 presenta el resultado de la prueba.

Figura 17. Resultado de la prueba sin implementar el componente UI

```
<?xml version='1.0' encoding='iso-8859-1'?>
<tree id="0">
        <item text="Paises" id="paises" open="1" call="1" select="1">
                <item text="Estados Unidos" id="usa" >
                        <item text="California" id="cal" >
                                <item text="Fresno" id="cal_1" />
                                <item text="Los Angeles" id="cal_2" />
                                <item text="San Diego" id="cal_3" />
                        </item>
                        <item text="Florida" id="flo" >
                                <item text="Fort Lauderdale" id="flo_1" />
                                <item text="Miami" id="flo_2" />
                                <item text="Orlando" id="flo_3" />
                                <item text="Tallahassee" id="flo_4" />
                        </item>
                        <item text="Texas" id="texa" >
                                <item text="El Paso" id="tex_1" />
                                <item text="Houston" id="tex_2" />
                                <item text="San Antonio" id="tex_3" />
                        </item>
                </item>
        </item>
</tree>
```

Fuente: Autores

A diferencia de otros servicios que proveen información en formato tipo “s*tring*” o “*array*”, este componente provee una salida con formato XML, esta es una de las fortalezas del componente pues desde este punto hasta la presentación al usuario la información puede ser procesada en cualquier tecnología, llámese Java, .NET, PHP, RUBBY, etc.

4.4.7 Implementación de los Componentes de interfaz

Para la implementación de los componentes personalizados de interfaz se utilizó el patrón “*ViewManager*” definido por MOON. Este patrón provee un conjunto de objetos que gestionan los requerimientos de salida al usuario, esto permite que el desarrollador no se preocupe por las interfaces gráficas, ocupando toda su energía en la lógica del negocio.

Una vez finalizado el proceso de creación y prueba de las clases que componen en núcleo del presente desarrollo, y teniendo en cuenta que sólo resta implementar los componentes de interfaz se procede a integrar el desarrollo realizado con el Marco de trabajo MOON. A continuación se enuncian los pasos para realizar la integración.

PASO 1: Extensión del patrón ViewManager

MOON provee una clase llamada “*page.class*”, en ella se encuentran las propiedades y métodos que gestionan la interfaz de usuario, por lo tanto de deben adicionar los siguientes componentes para que el Marco de Trabajo provea el soporte para la presentación de estructuras jerárquicas.

Propiedades adicionadas: **1**

A. Nombre de la propiedad: *_tree*
B. Tipo de dato: Booleano
C. Nivel de acceso: Privado

Ajuste en el constructor: **1**

A. La nueva propiedad debe ser inicializada, su valor inicial es FALSE

Métodos tipo setter adicionados: **1**

A. Nombre del método: *treeSupport*
B. Parámetros del método: *value* de tipo Booleano
C. Nivel de acceso: Publico

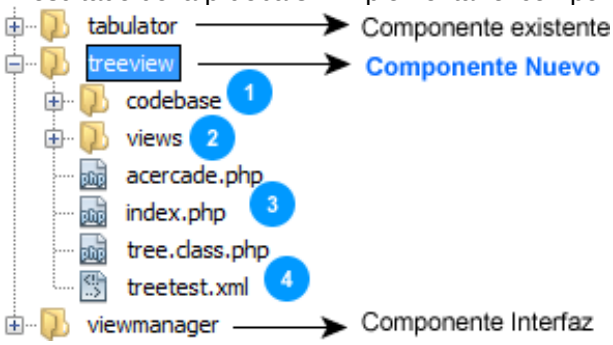
Ajustes al servicio que provee la presentación de páginas. **2**

1. Inclusión de función para la carga de archivos JavaScript y CSS
    a. Nombre función: *xhtml_tree*
    b. Parámetros*: includeTree* de tipo booleano y *path_treeview* de tipo string
    c. Objetivo: Cargar dos librerías javascript integradas en MOON y cargar la hoja de estilos que dará soporte a la estructura jerárquica
2. Llamado de la función *xhtml_tree* al momento de utilizar el servicio *open* del patrón "ViewManager"

PASO 2: Integración del nuevo componente a MOON

Los componentes en MOON están organizados en carpetas independientes, cada carpeta representa un componente con una funcionalidad particular. Para integrar el componente realizado al Marco de Trabajo se requiere la creación de una nueva carpeta al interior del contenedor de componentes del Marco, adicionalmente se deben seguir los lineamientos en cuanto a nombres, tipos de datos y creación de servicios definidos en MOON. La figura 18 presenta la forma como está organizado el componente para su integración.

Figura 18. Resultado de la prueba sin implementar el componente UI



Fuente: Autores

Como se puede observar en la figura 18 hay tres carpetas, la primera denominada "*tabulator*" el cual es un componente que permite adicionar pestañas en la interfaz de usuario final. La carpeta "*treeview*" contiene el nuevo componente a integrar y la carpeta "*viewmanager*" que contiene el patrón de interfaz de MOON.

Al interior de la carpeta "*treeview*" hay dos carpetas, la primera (1) se denomina "*codebase*" es la encargada de almacenar las librerías que tienen MOON para la presentación de estructuras jerárquicas, la segunda carpeta (2) se denomina "*views*", su uso es obligatorio pues debe permitir la presentación de la funcionalidad de manera rápida. Los archivos (3) y (4) contienen los ejemplos respectivos para permitir que un usuario conozca el modelo de objetos del "*treeView*" y visualice su implementación.

PASO 3: Clase interface entre el Marco de Trabajo y el nuevo componente

La integración del nuevo componente con el Marco de trabajo finaliza con la creación y prueba de una clase que sirva como interface entre los dos. La clase se denomina "*treeview.class*" y se encuentra ubicada en la carpeta raíz del nuevo componente. La tabla 11 presenta la descripción funcional de la clase.

Tabla 11.Detalle de la clase treeview

Componentes	Elementos	Nivel de Acceso
Propiedades	_id	private
	_style	private
	_iconStyle	private
	_appPath	private
	_codPath	private
	_imgPath	private
	_enableCheckBox	private
	_enableInherit	private
	_pageLoad	private
Constructor	Inicializa todas las propiedades y carga las rutas especificadas por el marco de trabajo.	public
Setters y Getters	No se realizaron *Getters* pues la funcionalidad que provee la clase no devuelve valores para cada propiedad.	public
Métodos adicionales	loadScript()	public

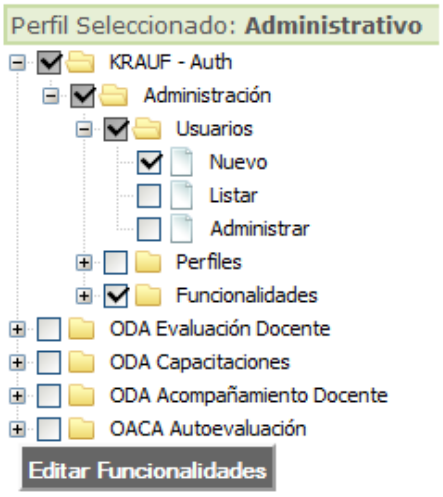
PASO 4: Implementación del nuevo componente en un desarrollo

Para dar cumplimento a los requerimientos funcionales se construyó un componente que gestiona los perfiles en un sistema orientado por funcionalidades, el componente provee un listado de perfiles como se puede apreciar en la figura 19.

Figura 19. Listado de perfiles

El componente permite eliminar perfiles que no estén asignados a usuarios, grupos y/o funcionalidades, de igual forma provee la funcionalidad para actualizar la información de cada perfil. En la figura 20 se presenta el componente "*treeview*" funcionando, posteriormente se ilustran los pasos para crear la interfaz.

Figura 20. Componente *treeview* en uso



Fuente: Autores

Para lograr visualizar la interfaz presentada en la figura 20 se requieren un conjunto de pasos que permitan instanciar adecuadamente los objetos requeridos. La tabla 12 presenta los pasos a seguir con el código respectivo.

Tabla 12.Detalle de la clase treeview

Implementación del componente	Código

Se deben crear los objetos que van a interactuar en la interfaz. Para este caso se instancian tres objetos, el primero es el encargado de controlar la interfaz de todos el componen te, el segundo se encarga de los mensajes y el último se encarga de visualizar el árbol con las funcionalidades	`$Page = new Moon_Viewmanager_Pages();`
	`$Message = new Moon_Layer_Messages();`
	`$treeView = new Moon_TreeView_Tree();`

El objeto Page provee un servicio que permite dar soporte al componente "treeView", de igual forma el componente es totalmente configurable permitiendo asignar: identificador, habilitar casillas de chequeo, posibilitar la herencia entre casillas de chequeo, selección de estilos, etc.	`$Page->treeSupport(true);`
	`$style = "csh_dhx_skyblue";`
	`$treePage = "controllers/xmltree.php?{$str_url}";`
	`$treeView->set_id("tree_div_01");`
	`$treeView->set_enableCheckBox(1);`
	`$treeView->set_enableInherit(true);`
	`$treeView->set_iconStyle($style);`
	`$treeView->set_pageLoad($treePage);`

Implementación del componente	Código

El paso anterior invocó un controlador denominado “xmltree”, este controlador es el encargado de obtener la información que se desea visualizar en el árbol.	`$cod_perfil = $Parameters->get_parameter("codperfil", 0);`
	`$FuncFacade = new Modules_Auth_Model_FuncFacade();`
	`$PerfilFacade = new Modules_Auth_Model_PerfilesFacade();`
	`$arr_func_seleccionadas = $PerfilFacade->obtenerFuncionalidades($cod_perfil);`
	
	`echo "<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>\n";`
	`echo "<tree id=\"0\">\n";`
	`echo $FuncFacade->xmlTree("1", $arr_func_seleccionadas);`
	`echo "</tree>";`

5. PROPUESTA DE CATÁLOGO PARA LA CONSTRUCCIÓN DE COMPONENTES SOFTWARE REUTILIZABLES

5.1 Definición del Catálogo

El catálogo propuesto toma como referencia el modelo de catálogo descrito por Gang of Four en su libro "*Desing Patterns: Elements of Reusable Object - Oriented Software*". Del modelo *GOF* se reutilizan cinco (5) características y se adicionan dos (2) que buscan conformar un modelo de catálogo ideal para un proceso de reutilización de requerimientos. La tabla 13 presenta cuatro (4) columnas para describir el catálogo propuesto:

1. Característica: identificación del criterio a explicar.
2. Descripción: explicación del alcance y utilidad de la característica.
3. Base: presenta cual es el origen de la característica.
4. Criterio: identificación utilizada en los objetivos específicos del proyecto.

Tabla 13.Plantilla catálogo propuesto

Característica	Descripción	Base	Criterio
Intención	Describe que hace el componente	GOF	Explicación
Aplicabilidad	Presenta casos o escenarios en donde puede ser utilizado el componente	GOF	Escenarios de Aplicación
Estructura	Presenta el modelo de clases	GOF	Representación en UML
Participantes	Describe cada uno de los componentes de la estructura	GOF	Participantes
Componentes relacionados	Identifica posibles componentes relacionados	GOF	-
Entradas	Describe las entradas requeridas por el componente	**Nuevo**	-
Salidas	Define el tipo de salida que tiene el componente	**Nuevo**	Implementación del Componente

Fuente: Autores

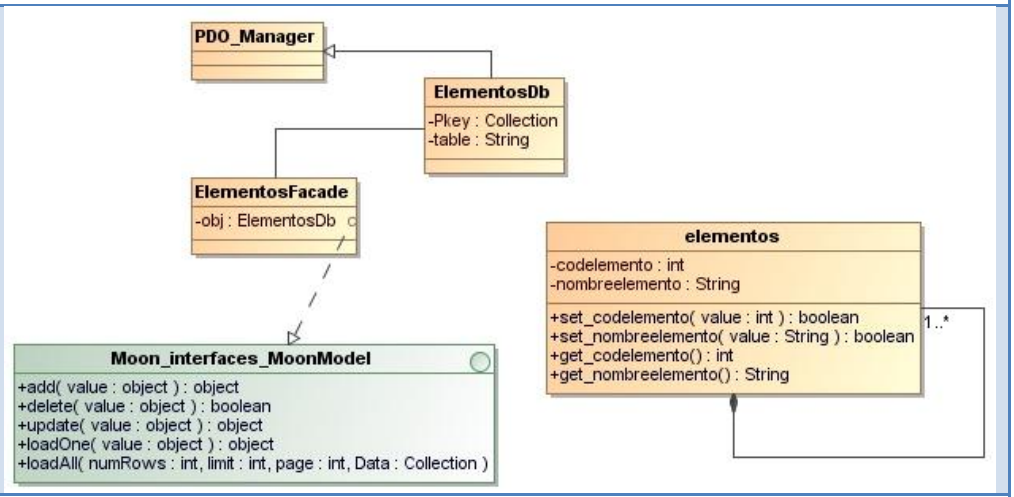
Las características reutilizadas del catálogo GOF son utilizadas para definir catálogos de Patrones de Diseño, si se tiene en cuenta que un patrón de diseño es una solución software que puede ser utilizada en diferentes contextos, se puede inferir que esas características son útiles para definir un catálogo de reutilización de componentes.

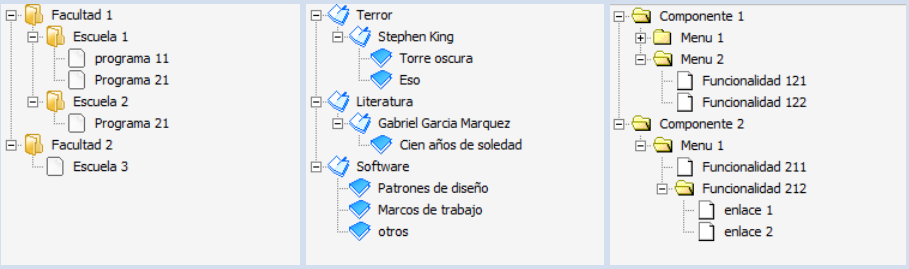
Aunque las características propuestas por GOF son apropiadas se ha considerado que se requieren criterios adicionales que permitan comunicar el componente definido con otros. Para lograr lo anterior es útil conocer la tipología de los datos de entrada y salida, razón por la cual en el catálogo propuesto se evidencian dos características adicionales: Entradas y Salidas

5.2 Componente TREEVIEW

En este apartado se presenta la aplicación del catálogo para un componente específico, en este caso el componente TreeView construido en el presente proyecto. Se espera que para todos los componentes reutilizables construidos y a implementar para el Marco de Trabajo MOON, se utilice la plantilla presentada en la tabla 14.

Tabla 14.Catálogo con el componente realizado

Característica	Descripción
Intención	El componente permite visualizar estructuras jerárquicas tomando como base una estructura de documento XML. La estructura XML puede estar almacenada en un archivo o puede ser construida en tiempo de ejecución como es el caso de una consulta a una base de datos. El objetivo principal es proveer una estructura en forma de árbol que permita manipular sus elementos para brindar una interfaz visual sencilla y práctica. La selección de elementos puede realizarse para la totalidad del árbol o por cada uno de sus elementos
Aplicabilidad	El componente puede ser utilizado en un porcentaje alto de estructuras jerárquicas que permitan agrupar datos bajo el modelo de herencia simple. Entre los escenarios más comunes se encuentran: • Permisos y Funcionalidades: Aquellos sistemas en donde se requiera la selección de permisos y funcionalidades pueden adaptar este componente para asignar los permisos y para visualizarlos. • Ubicación Geográfica: Los componentes geográficos almacenan información sobre países, departamentos, municipios, barrios, etc. Este componente permite organizar esa información de manera visual. Este componente no tiene aplicabilidad en modelos de datos que proveen un tipo de herencia múltiple.
Estructura	
Participantes	**PDO_Manager:** Provee los servicios para la conexión a bases de datos. La manipulación y el tipo de datos dependen de esta Meta Clase. Hace parte del Núcleo base de MOON lo que significa que no es modificable. **ElementosDB:** Es la implementación que permite codificar los tipos de solicitudes a la base de datos. Esta clase no puede ser instanciada desde el software, su uso requiere de un patrón del tipo Facade. **ElementosFacade:** Define las primitivas para el acceso a los servicios que provee la clase ElementosDB.

Característica	Descripción
	MOON_Interface_MoonModel: Define los métodos requeridos que deben ser implementados a la hora de instanciar la Facade de ElementosDB. **Elementos:** Esta clase provee todos los servicios necesarios para la carga de información de acuerdo al modelo recursivo utilizado por el componente.
Componentes relacionados	**Componente Menús:** Su relación se establece por el uso de XML como mecanismo para el intercambio de información. **Componente Security:** Su relación está dada por la serialización de objetos que hace este componente y por la recursividad con la que carga los datos.
Entradas	Este componen te permite dos tipos de entrada: 1. Archivo físico en formato XML con codificación iso-8859-1 2. Vector asociativo de datos con los siguientes campos: a. Identificador b. Código padre que relaciona el identificador c. Texto que se desea visualizar
Salidas	La salida del componente es un árbol con los datos solicitados en la característica entrada: 

Fuente: Autores

6. RESULTADOS OBTENIDOS

La tabla 11 describe cada uno de los productos entregables resultado del presente proyecto como respuesta a los objetivos específicos planteados en la propuesta inicial. Este análisis se realiza con el propósito de evidenciar el cumplimiento satisfactorio de los cuatro (4) objetivos propuestos.

Tabla 15.Resultados obtenidos contra objetivos

Objetivo Relacionado	Resultados Obtenidos	Cumplimiento
1,2	Un (1) documento con la aplicación del proceso de desarrollo software propuesto por el trabajo de maestría "*Propuesta de un Proceso de Desarrollo de Componentes software Reutilizable*".	100%
1,2	Diagramas de casos de uso para el modelo de requisitos, Diagrama de clases para el modelo de análisis y Diagrama de componentes para el modelo de componentes.	100%
1,2	Construcción de un (1) componente software implementado dos escenarios de aplicación, con el fin de evidenciar la reutilización del componente.	100%
1,2,3,4	Un (1) catálogo software que permita identificar componentes reutilizables desarrollados con MOON.	100%
1,2,3,4	Participación de los autores de la propuesta en un proceso de investigación formativa, apoyado por la escuela de Ingeniería de sistemas e informática.	100%

Fuente: Autores

CONCLUSIONES

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

- En los últimos años el uso de marcos de trabajo en el proceso de desarrollo software se ha convertido en una buena práctica. Convirtiéndose los marcos de trabajo en herramientas fundamentales en la implementación de aplicaciones de manera rápida y óptima. En el presente proyecto se utilizó un marco de trabajo de reconocimiento académico y empresarial denominado MOON. Actualmente MOON está siendo utilizado por el Ministerio de Educación Nacional, la Registraduría Nacional y algunas instituciones de educación superior, demostrando su aplicabilidad y reconocimiento.

- La integración del componente desarrollado en este trabajo de investigación al marco de trabajo MOON contribuye de manera significativa al enriquecimiento de las funcionalidades del Framework adicionando características nuevas que no estaban presentes en el mismo y que eran necesarias. Es importante, aclarar que el acople del componente nuevo al marco de trabajo fue transparente y no tuvo repercusiones negativas debido al uso de los lineamientos de la Ingeniería del Software Basada en Componentes y de la Ingeniería del Software en general.

- El componente software construido como resultado del presente proyecto de investigación cuenta con un alto grado de reutilización de acuerdo a los criterios establecidos en el proyecto de maestría utilizado como referencia. El resultado del análisis de reutilización entrego un puntaje de 39 puntos, siendo el puntaje mínimo para garantizar que el componente tiene potencial de reutilización de 30 puntos.

RECOMENDACIONES

- Luego de la experiencia en la realización del presente proyecto se puede sugerir dar continuidad al desarrollo del mismo, mediante la implementación de nuevos componentes que permitan ampliar el funcionamiento del Framework MOON. La caracterización del proceso de desarrollo sugerida y aplicada en el presente proyecto, cubre gran parte de variables; lo que facilitará futuros desarrollos, debido a que el catálogo para la correcta reutilización de componentes ya está construido.

- La línea de base para el desarrollo del presente proyecto fue la investigación de maestría del Ing. Freddy Vera denominada "*Propuesta de un Proceso de Desarrollo de Componentes Software Reutilizable*", durante la aplicación del proceso de desarrollo propuesto en ese trabajo de maestría se pudieron identificar algunas características por mejorar como: el instrumento utilizado para la evaluación de los requisitos a cumplir por un componente para clasificar como componente reutilizable, cuenta con cinco (5) criterios de evaluación y tres (3) columnas de descripción. La columna requisito y análisis están definidas de manera clara, sin embargo, la descripción de la columna evaluación se limita a colocar si el requisito cumple o no cumple. Obviando el detalle de porque el proceso de evaluación del criterio resulto favorable o por el contrario porque no cumplió.

REFERENCIAS BIBLIOGRÁFICAS

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

[2] Guerrero Alarcón, Andrés. Frameworks para el desarrollo de aplicaciones Web que utilizan código abierto. Sic Editorial. Proyecto Cultural de Sistemas y Computadores S.A. ISBN 978-958-708-406-1, 2009

[3] Markiewicz, Marcus Eduardo y J.P. de Lucena, Carlos. El Desarrollo del Framework Orientado al Objeto. [Consultado noviembre de 2008]. Disponible en: < http://www.willydev.net/descargas/prev/FrameworkJM.pdf>

[4] Minetto, Elton Luís. Frameworks para Desenvolvimento em PHP. Brasil: Novatec, 2007; p: 17-18.

[5] Orós, Juan Carlos. Diseño de páginas Web interactivas con JavaScript y CSS. Alfaomega, 4ª Edición, 2004.

[6] Vera Rivera, Freddy. Propuesta de un proceso de desarrollo de componentes software reutilizables. UIS. 2009.

[7] http://www.proactiva-calidad.com/java/patrones/index.html

[8] http://webcache.googleusercontent.com/search?q=cache:sxEcRcLXWr4J:blogs.timovil.com/efrenca/archive/2007/04/18/m%C3%A9todos-de-reutilizaci%C3%B3n-de-software.aspx+Es+un+acercamiento+sistem%C3%A1tico+para+identificar+las+similitudes,+semejanzas+y+variabilidad+necesaria+para+caracterizar+y+estandarizar+una+l%C3%ADnea+de+productos+como+un+conjunto+del+alcance+de+la+soluci%C3%B3n,+basado+generalmente+en+componentes+comerciales&cd=1&hl=es&ct=clnk&gl=co