**Universidad de las Ciencias Informáticas**

**Facultad 3**


# Título: Desarrollo de las interfaces gráficas para la gestión del componente Estructura y Composición del marco de trabajo Bosón.

**Trabajo de Diploma para optar por el título de**

**Ingeniero en Ciencias Informáticas**

**Autora:**

Liliana Simón Figueredo

**Tutores:**

Ing. Dannel Jiménez Torres

Ing. Osnier Ramírez Alea

*"Lo fundamental es que seamos capaces de hacer cada día algo que perfeccione lo que hicimos el día anterior".*

*Ernesto Che Guevara*

## DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los _____ días del mes de _____ del año _____.

_____

Liliana Simón Figueredo

_____ _____

Ing. Dannel Jiménez Torres Ing. Osnier Ramírez Alea

## AGRADECIMIENTOS

**Q**uisiera comenzar agradeciendo a todas aquellas personas que de alguna forma u otra hicieron posible que me convirtiera en la mujer que soy.

**P**rimeramente, le agradezco a mi padre por ser el hombre más maravilloso del mundo, por creer en mí, por apoyarme en todo momento, por ser el motivo de inspiración de mi vida y por darme el honor de ser su hija. Te quiero mucho PAPI

**A** mis abuelos Luisa, Leonel y José por haber hecho de mí la mujer que soy, por sus sacrificios y porque sin ustedes no hubiese logrado cumplir hoy este sueño. Los amo.

**A** mi familia en especial a mi madre por el apoyo incondicional que me ha brindado durante el transcurso de mi vida. A mi prima Lisandra, a mi tía Leonela.

**A** mis hermanos Yoe y Omar por ser mi inspiración.

**A** mi novio Efraín por ser la persona más maravillosa del mundo, por quererme tanto, por ayudarme en todo, por soportar mi mal genio, porque sin ti no hubiese logrado mis sueños, te amo mucho.

**A** una persona muy especial para mí, que desde que la conocí en el primer año de la carrera ha sido como una madre para mí, gracias por brindarme tanto cariño te quiero Ana.

**A**gradezco a la Universidad de las Ciencias Informáticas por permitir que me formara como profesional en ella.

**A** todos los profesores que contribuyeron a mi preparación y formación profesional.

**A** todos mis amigos, los de ayer y los de hoy, que supieron ayudarme en los momentos difíciles cuando los necesité y brindarme su mano sin nada a cambio, en especial a Alexei, Ismaby, Rosy, Liannet, Liset, Dayani, Roly, Lenny, Lien, Jeidy, Robe, Ariel y Indra.

**A todos, ¡Gracias!**

# RESUMEN

La usabilidad es un tema que adquiere una importancia cada vez mayor en el desarrollo de software. En el entorno actual, en el que los sistemas de software están dirigidos a un público cada vez más amplio, a usuarios con menos experiencias en la interacción con los sistemas informáticos, la usabilidad está destacándose como atributo fundamental para el éxito de un producto de software.

Dada la necesidad del país de tener un mayor control de los recursos de la organización y lograr un comportamiento común en cada una de las empresas se buscan alternativas que permitan integrar y mejorar la competitividad de las mismas. En este sentido la investigación se realizó con el objetivo de desarrollar las interfaces gráficas para el componente Estructura y Composición del marco de trabajo Bosón, que brinda a los usuarios la posibilidad de definir la estructura organizativa en la cual se ubican las empresas.

Para la construcción de las interfaces gráficas se realiza un estudio de la metodología a utilizar para guiar el proceso de desarrollo de software, así como de las principales herramientas y tecnologías utilizadas en la implementación de la solución. Además, se describe la arquitectura que da soporte a la solución. El diseño, la implementación y la solución propuesta son validados a través de métricas de diseño, pruebas de caja negra y de aceptación, un pre-experimento y la lista de chequeo de usabilidad que propone calidad del Centro de Informatización de Entidades con el objetivo de mejorar la usabilidad del componente Estructura y Composición.

**PALABRAS CLAVES**

Componente, Estructura y Composición, interfaces gráficas, usabilidad.

# ÍNDICE DE CONTENIDOS

# ÍNDICE DE TABLAS

# ÍNDICE DE FIGURAS

# INTRODUCCIÓN

La informática es la ciencia que estudia el tratamiento automático de la información por medio de las computadoras. Desde sus inicios esta ciencia ha evolucionado vertiginosamente y actualmente casi todos los países del mundo basan en ella algunos de sus procesos más importantes. Cuba, a pesar de ser un país que no tiene gran desarrollo tecnológico, no ha estado ajeno al avance del mismo; para lograr esto se han creado estrategias que permitan tener acceso a las Tecnologías de la Información y las Comunicaciones (TIC).

Las TIC han logrado cambiar la forma tradicional que las organizaciones utilizaban para la toma de decisiones y la satisfacción de las necesidades de sus clientes. Por otra parte, se han convertido en un área de gran amplitud e impacto en todos los aspectos de la vida cotidiana, incluyendo la gerencia de cualquier empresa, en la cual hoy en día es casi indispensable. La integración de las TIC en los procesos de negocio, representa un objetivo significativo para aportar valor a las empresas, convirtiéndolo en un reto a nivel mundial.

En la actualidad los desarrolladores de software tienen como una de sus metas fundamentales que los sistemas informáticos sean lo más usables posible. Se pretende que la interacción del usuario con la aplicación sea sencilla y eficaz y por tanto sea aceptada por el mismo. Para lograr esto es recomendable hacer un diseño que se adapte al máximo a las características de los usuarios con el fin de obtener un mejor entendimiento y satisfacción, logrando la aceptabilidad y la comodidad de los usuarios con el sistema.

En Cuba se desarrollan una serie de proyectos informáticos, regidos en su mayoría, por la Universidad de las Ciencias Informáticas (UCI), la misma promueve un esquema de estudio-investigación-producción, en el que los estudiantes se vinculan directamente a la producción mediante los centros de desarrollo. Entre los centros productivos existentes en la UCI se encuentra el Centro de Informatización de Entidades (CEIGE) perteneciente a la Facultad 3.

En el departamento de Desarrollo de Componentes de CEIGE, se desarrolla el marco de trabajo Bosón, el cual se manifiesta como la solución que permite establecer un formato de trabajo común entre las diversas soluciones, al aportarles componentes previamente construidos y listos para ser empleados. Uno de los componentes desarrollados en este marco de trabajo es el de Estructura y Composición que tiene como objetivo manejar la estructura

organizacional y los nomencladores de forma dinámica, lo que le facilita al usuario personalizar sus estructuras en dependencia de sus requerimientos.

Para configurar y generar las estructuras y los nomencladores Bosón necesita de personas calificadas, con conocimientos de informática, debido a que todas estas gestiones se realizan por medio de comandos por consola, dificultándole al usuario:

- Visualizar e interactuar con los elementos del componente.
- Visualizar el resultado al buscar y listar las estructuras y nomencladores.
- La gestión de la información por la ausencia de interfaces.
- La satisfacción sea alta al no tener una interfaz gráfica intuitiva.

Aun cuando se prevén los errores por parte del usuario, es complejo minimizar las posibilidades de error, ya que, al necesitar ingresar texto, se puede presentar el problema de ingresar comandos con errores ortográficos u errores de escritura.

Se debe tener en cuenta que, aunque se tiene una uniformidad que busca estandarizar la interacción con el usuario, no se toma en cuenta las diferentes habilidades del mismo, lo que, para un usuario inexperto, puede ocasionar problemas para obtener una comunicación efectiva con el computador.

Por lo que surge el **problema a resolver:** ¿cómo mejorar la usabilidad del componente Estructura y Composición del marco de trabajo Bosón para proporcionar al usuario la visualización e interacción con los elementos que posee?

El problema planteado está contenido en el **objeto de estudio:** proceso de desarrollo de interfaces gráficas. Enmarcado en el **campo de acción:** proceso de desarrollo de interfaces gráficas en las aplicaciones web.

Para dar solución al problema de la investigación se trazó el siguiente **objetivo general:** desarrollar las interfaces gráficas para la gestión del componente Estructura y Composición del marco de trabajo Bosón para mejorar su usabilidad.

Del objetivo general se desglosan los siguientes **objetivos específicos:**

- Confeccionar el marco teórico de la investigación a partir de la búsqueda y revisión bibliográfica de las tecnologías que permiten construir interfaces de usuarios para las aplicaciones web.

- Realizar el análisis y diseño del componente de Estructura y Composición del marco de trabajo Bosón para proporcionar una solución al problema existente.
- Desarrollar las interfaces gráficas del componente de Estructura y Composición del marco de trabajo Bosón para mejorar su usabilidad.
- Validar el funcionamiento de las interfaces del componente de Estructura y Composición del marco de trabajo Bosón mediante pruebas de caja negra y de aceptación para lograr un correcto funcionamiento del mismo.

La **idea a defender** que se plantea es**:** si se desarrollan las interfaces gráficas del componente Estructura y Composición del marco de trabajo Bosón, se contribuye a mejorar la usabilidad para la gestión de los elementos que posee.

Entre los **métodos científicos** utilizados en la investigación se destacan los siguientes:

**Histórico-Lógico**: permitió comprender de forma más clara la esencia del objeto de estudio y su concepción histórica, así como la trayectoria real del funcionamiento y desarrollo de la gestión de estructuras y su composición en las entidades.

**Analítico-sintético**: posibilitó la descomposición del objeto de estudio en conceptos más pequeños y más fáciles de estudiar, así como conocer sus características generales. Para ello, se estudió la teoría y bibliografía relacionada con el problema.

**Métodos empíricos:**

**Entrevista:** posibilitó realizar conversaciones planificadas entre los especialistas funcionales y analista principal. Fue utilizado para la captura de datos que sirvió para la especificación de los requisitos del componente. Se utilizó para precisar el problema a resolver y los problemas existentes que actualmente existen en el componente Estructura y Composición.

**Encuesta:** se aplicó una encuesta a los profesionales del centro CEIGE del departamento de Desarrollo de Componentes que trabajan con el marco de trabajo Bosón, con el objetivo de determinar el porciento de usabilidad que tiene el componente a partir de los indicadores: facilidad de aprendizaje, eficiencia, recuerdo en tiempo, tasa de errores y satisfacción.

La **estructura de la investigación** quedó constituida en tres capítulos, conclusiones, anexos y bibliografía.

**Capítulo 1:** se realiza un estudio del estado del arte de los sistemas que gestionan estructuras y nomencladores, haciendo alusión a los principales sistemas de este tipo existentes partiendo de su especificación conceptual. Se analizan la metodología, herramientas y lenguaje de modelado a utilizar.

**Capítulo 2:** se definen los requisitos del sistema. Se realiza una descripción de las funcionalidades utilizando historias de usuarios. Se desarrollan las descripciones de la arquitectura de software y los patrones de diseño utilizados. Además, se confecciona un diagrama de diseño con estereotipos web, el cual fundamentalmente se emplea para representar y documentar el diseño.

**Capítulo 3:** se muestran los resultados de las pruebas realizadas a las interfaces gráficas del componente de Estructura y Composición. Además, se valida la investigación realizada.

# CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

## 1.1 Introducción

En siguiente capítulo se realiza un estudio del estado del arte acerca del desarrollo de las interfaces gráficas de usuario, sus características, ventajas y una descripción de los principales conceptos asociados al problema, para entender mejor la propuesta de solución. De igual forma se analiza la metodología, tecnologías y herramientas que se utiliza en el desarrollo de la investigación.

## 1.2 Principales conceptos asociados a la gestión de Estructura y Composición

**Estructura:** disposición o modo de estar relacionadas las distintas partes de un conjunto (Real academia española, 2014).

**Composición**: acción y efecto de componer (Real academia española, 2014).

**Componer**: dicho de varias partes: formar o constituir un todo (Real academia española, 2014).

**Nomenclador**: catálogo de nombres relativos a un asunto determinado (Diccionario de significados, 2015).

**Catálogo**: relación ordenada en la que se incluyen o describen de forma individual libros, documentos, personas, objetos, etc., que están relacionados entre sí (Real academia española, 2014).

Según el contexto de la investigación una **estructura** se define como: el concepto que se utiliza para hacer referencia a las partes que conforman el organigrama de una institución. A continuación, se muestra un ejemplo de estructura a partir del organigrama de la facultad 3:

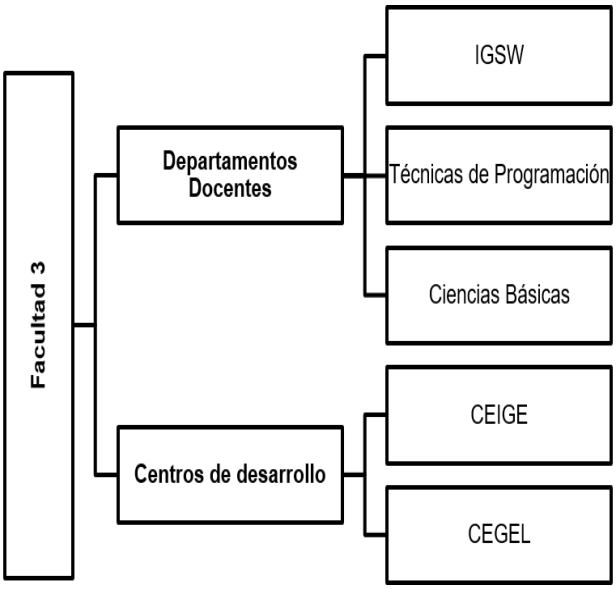


**Figura 1 Ejemplo del organigrama de la facultad 3.**

Fuente: (Elaboración propia)

Se define como **composición**: los elementos que puede tener una estructura que no sea concepto y **nomenclador:** forma de clasificar o agrupar una categoría determinada.

## 1.3 Interfaz de usuario

La interfaz de usuario (IU), es uno de los componentes más importantes de cualquier sistema computacional, pues funciona como el vínculo entre el humano y la máquina. La interfaz de usuario es un conjunto de protocolos y técnicas para el intercambio de información entre una aplicación computacional y el usuario (Verdecia Álvarez, y otros, 2010).

### 1.3.1 Clasificación de interfaces de usuario

Según su construcción, se clasifican de software o de hardware: (Verdecia Álvarez, y otros, 2010).

**Interfaces de hardware:** es un conjunto de controles o dispositivos que permiten la interacción hombre-máquina, de modo que permiten introducir o leer datos del equipo, mediante pulsadores, reguladores e instrumentos.

**Interfaces de software:** son programas o parte de ellos, que permiten expresar nuestros deseos al ordenador o visualizar su respuesta.

Atendiendo a cómo el usuario puede interactuar con una interfaz, se conocen varios tipos de interfaces de usuario: (Verdecia Álvarez, y otros, 2010).

**Interfaces alfanuméricas**: solamente presentan texto.

**Interfaces táctiles**: representan gráficamente un "panel de control" en una pantalla sensible que permite interactuar con él, de forma similar a si se operara un control físico.

**Interfaces gráficas de usuario:** permiten comunicarse con el ordenador de una forma muy rápida e intuitiva representando gráficamente los elementos de control y medida.

## 1.4 Interfaz Gráfica de Usuario

Las Interfaces Gráficas de Usuarios, en inglés Graphics User Interface (GUI), surgen dada la necesidad de hacer más simple el uso de los ordenadores para todo tipo de usuarios y no solo restringir el uso de estos a usuarios avanzados.

“Las GUI son una forma de representar la información procesada por la computadora de manera visual, concreta y manejable por el usuario final haciéndola más comprensible, más didáctica, más asimilable, más usable.” (Pantoja, 2009).

También existen otras definiciones como:

“La GUI es un tipo de interfaz de usuario que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz”. Habitualmente las acciones se realizan mediante manipulación directa para facilitar la interacción del usuario con la computadora (Babylon, 2014).

Una GUI es un programa de interfaz que aprovecha las capacidades de despliegue gráfico de la computadora que permite hacer más sencillo el uso de los programas (Marrero Expósito, 2010).

Carlos Marrero, en su investigación “Interfaz gráfica de usuario. Aproximación semiótica y cognitiva” define a la GUI como “el artefacto interactivo, que por su diseño y a través de ciertas interfaces humanas, posibilita la interacción de una persona con un sistema informático, haciendo uso de las gramáticas visuales y verbales (signos gráficos como íconos, botones, menús y verbales como tipografía)” (Alonso, 2010).

Básicamente una GUI es una representación gráfica en una pantalla de ordenador de los programas, datos y objetos, así como la interacción con ellos. Esta proporciona al usuario las

herramientas necesarias para realizar sus operaciones de forma concreta, manejable, más comprensible, más didáctica, más asimilable y más usable.

### 1.4.1 Características de las GUI

En la actualidad, una buena interfaz gráfica de usuario debe contar con un conjunto de características (Santiesteban Pérez, 2011).

- Poseer un dispositivo apuntador (típicamente un ratón).
- Promover la consistencia de la interfaz entre programas.
- Seguir el paradigma de la interacción objeto-acción.
- Permitir la transferencia de información entre programas.
- Poder manipular en la pantalla directamente los objetos y la información.
- Proveer elementos de interfaz estándar como menús y diálogos.
- Existir una muestra visual de la información y los objetos (íconos y ventanas).
- Proporcionar respuesta visual a las acciones del usuario.
- Existir información visual de las acciones y modos del usuario/sistema (menús, paletas).
- Existir controles gráficos (widgets) para la selección e introducción de la información.
- Proporcionar flexibilidad en el uso de dispositivos de entrada (teclado/ratón).

### 1.4.2 Etapas del desarrollo de GUI

El proceso de desarrollo de las GUI está determinado en 4 etapas (Verdecia Álvarez, y otros, 2010), las cuales se desarrollaron a lo largo de la investigación. A continuación, se expone un resumen de las diferentes actividades que se realizan en cada una de las etapas del desarrollo.

**Diseño**

- Análisis de requerimientos del producto.
- Análisis de las tareas.
- Conocimiento del usuario.
- Revisión de posibilidades para la implementación.

**Implementación**

- Generación de prototipos (profundos o amplios, para investigación general o de ajustes).
- Desarrollo de la aplicación, sitio o sistema.

**Medición (Test de usabilidad)**

- Planificación (desarrollo del plan, definición de las medidas, selección de participantes, formación de observadores, preparación de los materiales).
- Test (prueba piloto, tests con usuarios).

**Evaluación**

- Conclusión (análisis de los datos, elaboración del informe, resultados y recomendaciones).
- Verificación de las diferencias.
- Generación de nuevas metas.

### 1.4.3 Ventajas del uso de las GUI

Las principales ventajas de una interfaz gráfica son las siguientes (Nieblas Palau, y otros, 2008):

- La asimilación del contenido por parte del usuario es rápida y clara: el usuario al acceder al sistema es capaz de familiarizarse de forma amena con el mismo, le da sensación de seguridad y realismo, permitiéndole una rápida comprensión.
- Contenidos accesibles: la información que se muestra está disponible para que el usuario acceda a esta en el momento que lo necesite, sin grandes contratiempos.
- Usuarios sin experiencias pueden aprender el uso del sistema rápidamente: el usuario no tiene que ser experto en la materia para su interacción con el sistema.
- Retroalimentación garantizada: le permite al usuario navegar con más facilidad, siempre es posible volver atrás.
- Disponibilidad y portabilidad: se debe poder disponer de la información cuando se necesite, siempre que sea personal autorizado. Así mismo permite ser utilizado en múltiples ordenadores sin que necesiten de modificaciones de importancia.

## 1.5 Usabilidad

La norma internacional **ISO 9241-9** (Ferré Grau, 2012)**:** Guidance on Usability (2001) hace referencia a la usabilidad y ofrece una definición de su contenido y alcance:

La medida en la que un producto se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso especificado**.**

### 1.5.1 Atributos de usabilidad

La usabilidad es una cualidad demasiado abstracta como para ser medida directamente. Para su estudio se descompone habitualmente en los siguientes cinco atributos básicos (Ferré Grau, 2012):

**Facilidad de aprendizaje:** cuán fácil es aprender la funcionalidad básica del sistema, como para ser capaz de realizar correctamente la tarea que desea realizar el usuario. Se mide normalmente por el tiempo empleado con el sistema hasta ser capaz de realizar ciertas tareas en menos de un tiempo dado (el tiempo empleado habitualmente por los usuarios expertos). Este atributo es muy importante para usuarios noveles.

**Eficiencia:** el número de transacciones por unidad de tiempo que el usuario puede realizar usando el sistema. Lo que se busca es la máxima velocidad de realización de tareas del usuario. Cuanto mayor es la usabilidad de un sistema, más rápido es el usuario al utilizarlo, y el trabajo se realiza con mayor rapidez.

**Recuerdo en el tiempo:** para usuarios intermitentes (que no utilizan el sistema regularmente) es vital ser capaces de usar el sistema sin tener que aprender cómo funciona partiendo de cero cada vez. Este atributo refleja el recuerdo acerca de cómo funciona el sistema que mantiene el usuario, cuando vuelve a utilizarlo tras un periodo de no utilización.

**Tasa de errores:** este atributo contribuye de forma negativa a la usabilidad de un sistema. Se refiere al número de errores cometidos por el usuario mientras realiza una determinada tarea. Un buen nivel de usabilidad implica una tasa de errores baja. Los errores reducen la eficiencia y satisfacción del usuario, y pueden verse como un fracaso en la transmisión al usuario del modo de hacer las cosas con el sistema.

**Satisfacción:** este es el atributo más subjetivo. Muestra la impresión subjetiva que el usuario obtiene del sistema.

### 1.5.2 Relación entre la usabilidad y la GUI

En el desarrollo de software se identifica a menudo la usabilidad con las características de los elementos de una GUI, como puede ser su color, su disposición o el diseño gráfico de los íconos y animaciones. Sin embargo, la usabilidad no sólo tiene que ver con la GUI (Ferré Grau, 2012).

La usabilidad de un sistema está ligada principalmente a la interacción del mismo, al modo en que se realizan las operaciones con el sistema. Esta interacción no está definida en la interfaz gráfica, sino que está imbricada en el código que implementa la funcionalidad del sistema. La GUI es la parte visible de tal interacción. Es cierto que la interfaz gráfica es una parte importante del sistema, y un buen diseño de la misma puede hacer que un sistema aumente su nivel de usabilidad, pero un sistema con un diseño de la interacción pobre no puede mejorar su nivel de usabilidad tan solo cambiando la interfaz gráfica (Ferré Grau, 2012).

## 1.6 Herramientas de gestión de estructuras y nomencladores

En el epígrafe se explican algunas herramientas para la gestión de estructuras y nomencladores, las cuales fueron estudiadas para la solución del problema planteado. Se presentan sus principales características que son de gran importancia para el desarrollo de las interfaces gráficas del componente. Se expone en cada caso la tecnología utilizada para el desarrollo de las interfaces.

### 1.6.1 Sistema de gestión de nomencladores

Este sistema permite gestionar nomencladores de información común y poco variable en el tiempo; además posee características como la configuración y flexibilidad ante posibles cambios ( Mojena Alpizar, y otros, 2012).

Algunas de sus principales características son ( Mojena Alpizar, y otros, 2012):

- Gestiona los campos de los elementos nomenclados.
- Gestiona los grupos de los elementos a nomenclar.
- Asocia los campos a los nomencladores.
- Gestiona la información relacionada con cada nomenclador.

A continuación, se muestra una de las interfaces del sistema utilizando la tecnología Ext JS 3.1:


**Figura 2 Ejemplo de una interfaz del Sistema de gestión de nomencladores.**

Fuente: ( Mojena Alpizar, y otros, 2012)

### 1.6.2 Subsistema Estructura y Composición del sistema Cedrux

Cedrux es un paquete de soluciones integrales de gestión para las entidades presupuestadas y empresariales basada en los principios de independencia tecnológica y con funcionalidades generales de los procesos y las particularidades de la economía cubana.

Dentro de los subsistemas de Cedrux se encuentra el de Estructura y Composición, el cual permite la creación de forma jerárquica de las estructuras definidas en el organigrama de una institución con el objetivo de poder establecer posteriormente niveles de seguridad teniendo en cuenta los intereses de dicha institución. Permite además definir la plantilla de cargos que está asociada a las estructuras (internas), así como los recursos materiales con que cuentan para su funcionamiento (Rodríguez Sánchez, y otros, 2011).

Algunas características del subsistema son:

- Gestiona las estructuras según el nivel estructural.
- Permite establecer relaciones entre las estructuras.
- Gestiona nomencladores.

- Gestiona los valores de los campos y las instancias de las estructuras y los nomencladores.

A continuación, se muestra una de las interfaces del sistema utilizando la tecnología Ext JS 2.2:

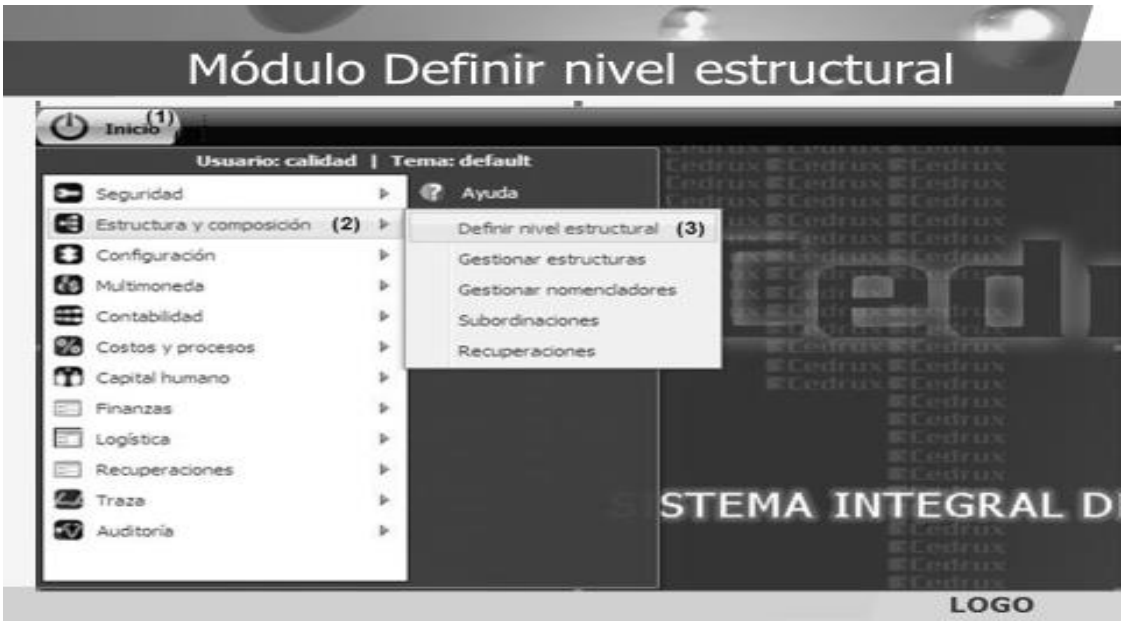


**Figura 3 Ejemplo de una interfaz del Sistema de gestión de nomencladores.**

Fuente: (Rodríguez Sánchez, y otros, 2011)

### 1.6.3 Gespro

Gespro es una herramienta utilizada y desarrollada en la UCI para la gestión de proyectos. Se encarga de organizar y administrar recursos de manera tal que se pueda culminar todo el trabajo requerido en el proyecto dentro del alcance, tiempo y coste definidos (Estevez Muñoz, y otros, 2013).

Dentro de esta herramienta se encuentra el modulo de configuración, el cual permite gestionar la integración entre los diferentes niveles de la organización. Garantiza, además, la integración del Gespro gerencial con los Gespro operacionales instalados en las sucursales de la organización.

Algunas características de Gespro son:

- Gestiona sucursales (departamentos), al adicionar una sucursal se le asigna el centro a la cual pertenece.

- Lista los departamentos para su gestión una vez creados.
- Gestiona los distintos niveles de la organización.

A continuación, se muestra una de las interfaces del sistema desarrolladas utilizando la tecnología JQuery:


**Figura 4 Ejemplo de una interfaz de Gespro.**

Fuente: (Estevez Muñoz, y otros, 2013)

### 1.6.4 Resultados del análisis de las herramientas

A partir del estudio realizado a las herramientas, se pudo corroborar que ninguna puede ser utilizada como base para el desarrollo de la solución, debido a que las interfaces de las herramientas no gestionan estructuras y nomencladores de forma dinámica, además no habían sido desarrolladas usando tecnología Angular JS, que es la que se emplea y requiere el proyecto; lo cual dificulta la integración con el marco de trabajo. Se decidió desarrollar interfaces gráficas para el componente de Estructura y Composición utilizando de las mismas:

- La forma de representar las estructuras jerárquicas según el organigrama de la organización.

- La facilidad de manejar las estructuras y los nomencladores al interactuar con la herramienta, lo que le proporciona al usuario que la interfaz sea más comprensible, más didáctica y más usable.

## 1.7 Metodología de desarrollo de software

La metodología empleada en la investigación es la variación de la metodología para la UCI "Proceso Unificado Ágil" (AUP por sus siglas en inglés) en unión con el modelo CMMI-DEV v1.3 que es la que propone la universidad para el desarrollo de software (Sánchez Rodríguez, 2015).


**Figura 5 Fases variación AUP.**

Fuente: (Elaboración propia)

La metodología cuenta con tres fases: inicio, ejecución y cierre por las que se deben transitar durante el desarrollo de las actividades productivas. La presente investigación se centra en la fase de ejecución, en la cual se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura.

Además, establece siete disciplinas y cuatro escenarios, de las disciplinas requeridas por la metodología se desarrollan solamente:

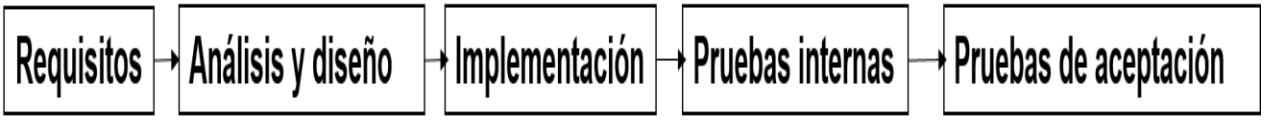


**Figura 6 Disciplinas de variación AUP.**

Fuente: (Elaboración propia)

El desarrollo de la investigación está enfocado en el cuarto escenario, el cual plantea: **Proyectos que no modelen negocio solo pueden modelar el sistema con historias usuarios (HU)**, donde se aplica en los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente está siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información. (Sánchez Rodríguez, 2015)

## 1.8 Técnicas para la captura y validación de requisitos

Para conocer las necesidades del cliente, es necesario realizar la captura de requisitos, en este proceso se extrae de diferentes fuentes de información, los datos que son necesarios para conocer las funcionalidades a desarrollar por el sistema. Una vez realizada la captura de requisitos se emplean técnicas de validación, para demostrar que la definición de los requisitos puntualiza realmente lo que el cliente desea y comprobar que los requisitos identificados no presenten ambigüedades, inconsistencias, omisiones o sean inalcanzables (Gracia, 2013).

Las técnicas para la captura de requisitos que se utilizan en la presente investigación son: **Entrevista:** consiste en establecer una conversación entre personas de ambas partes para obtener información sobre el negocio y a partir de éstos se definen los requisitos.

**Tormenta de ideas:** permitió recolectar nuevas ideas y/o información, proporcionadas por varios expertos.

Las técnicas utilizadas en la investigación para validar los requisitos son:

**Revisión técnica formal:** para la revisión técnica formal debe existir un equipo de revisión que incluya ingenieros del sistema clientes, usuarios, y otros interesados que examinan la especificación. Estos buscan errores en el contenido o en la interpretación, áreas donde se necesitan aclaraciones, información incompleta, inconsistencias, requisitos contradictorios, o requisitos irreales o inalcanzables. La revisión es esencial para asegurarse que el cliente y el desarrollador tienen el mismo concepto del sistema. Es un proceso manual que involucra a personas tanto de la organización del cliente como la del contratista, se verifica el documento de requisitos en cuanto a anomalías y omisiones (Sommerville, 2011).

**Construcción de prototipos:** en este enfoque de validación se muestra un modelo ejecutable del sistema a los usuarios finales y a los clientes. Estos pueden experimentar con este modelo para ver si cumple sus necesidades reales. El prototipo puede ser funcional o no (Pressman, 2006) .

## 1.9 Patrones de diseño

Los patrones de diseño en la investigación permiten facilitar el trabajo, al emplear un conjunto de buenas prácticas, además proporcionan una estructura conocida por todos los programadores de manera que la forma de trabajar no resulte distinta entre los mismos. Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del

diseño orientado a objetos, basadas en la experiencia, estos permiten reutilizar la experiencia de los desarrolladores, clasificar y describir formas de solucionar problemas que ocurren de forma frecuente en el desarrollo y están basados en la recopilación del conocimiento de los expertos en desarrollo de software (Pressman,2009).

### 1.9.1 GRASP

El uso de patrones proporciona una estructura conocida por todos los programadores, de manera que la forma de trabajar no resulte distinta. Los patrones GRASP es el acrónimo de "General Responsibility Assignment Software Patterns", en español “Patrones de Software de Asignación de Responsabilidades Generales” describen los principios fundamentales de la asignación de responsabilidades a objetos, son una serie de buenas prácticas de aplicación recomendable en el diseño de software. Entre ellos se encuentran el experto, creador, alta cohesión, bajo acoplamiento y el controlador (Larman, 2003) (Pressman, 2009).

**Experto:** cada clase dentro del sistema tiene la responsabilidad de utilizar únicamente la información que ella misma posee para realizar la labor para la que fue concebida. Tal es el caso de las clases del modelo que son las encargadas de toda la lógica del acceso a los datos (Pressman, 2009).

**Creador:** identifica quien debe ser el responsable de la creación de nuevos objetos o clases, donde la nueva clase debe ser creada por la clase que tiene toda la información necesaria para realizar la acción, que usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de clase y contiene o agrega la clase (Pressman, 2009).

**Bajo acoplamiento:** es la medida en que cada una de las clases realiza actividades independientes, además de poseer un conocimiento de las actividades que realizan las otras clases del sistema permitiendo la reutilización (Pressman, 2009)

**Alta cohesión:** asigna una responsabilidad de modo que la cohesión siga siendo alta. Se define que existe alta cohesión funcional cuando los elementos de un componente colaboran para producir algún comportamiento bien definido. El nivel de cohesión no debe ser independiente de otras responsabilidades ni de otros principios como los patrones: experto y bajo acoplamiento (Pressman,2009).

**Controlador:** se encarga de asignar la responsabilidad a una clase en el momento de manejar mensajes correspondientes a eventos en un sistema, facilitando la centralización de

actividades. Esta clase es la responsable de manejar la lógica de negocio, decidiendo así que clase es la encargada de ejecutar una tarea determinada (Pressman, 2009).

## 1.10 Métricas para validar el diseño

Las métricas, dentro del contexto de la ingeniería del software son un grupo de medidas efectuadas sobre programas, documentación, procesos de desarrollo o al mantenimiento, que permite una previa comparación con valores que proporcionan una indicación cuantitativa de extensión, cantidad, dimensión, capacidad y tamaño de algunos atributos de un proceso o producto. Las métricas permiten descubrir y corregir problemas potenciales, antes de convertirse en defectos catastróficos. Se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de software. Inclina sus objetivos a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo (Pressman, 2006).

Lorenz y Kidd dividen las métricas basadas en clases en tres grupos: tamaño de la clase, herencia y características internas de las clases. Dentro de las métricas propuestas por ellos se encuentran: número de métodos de instancia públicos (MIP), número de métodos de instancia (NMI), número de variables de instancia (NVI), número de métodos heredados (NMH)(Lorenz, 1994).

Chidamber y Kemerer también realizan una propuesta para medir la calidad del diseño, entre las cuales se encuentran: métodos ponderados por clases (MPC), árbol de profundidad de herencia (APH), número de descendientes (NDD), acoplamiento entre clases objeto (AECO), y falta de cohesión de los métodos (FCM) (Chidamber, 1994)(Pressman, 2009).

Para la investigación se considera oportuno aplicar la métrica tamaño operacional de clases (TOC), pues permite medir el total de atributos y operaciones encapsulados en una clase para valorar la sobrecarga de responsabilidades asignadas y la métrica relaciones entre clases (RC) para evaluar el grado de acoplamiento entre las clases. Fueron seleccionadas estas métricas a partir de un estudio realizado de diferentes fuentes de la comunidad científica de la universidad, en las cuales se reflejaba que las métricas TOC y RC son las que más atributos de calidad miden de conjunto para validar un correcto diseño de la solución.

### 1.10.1. Tamaño operacional de la clase (TOC)

Evalúa los siguientes atributos de calidad:

- **Responsabilidad:** consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación:** consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización:** consiste en el grado de reutilización de presente en una clase o estructura de clase, dentro de un diseño de software.

**Tabla 1 Métrica para el tamaño operacional de la clase.**

Fuente: (Rodríguez Landín, 2012)

| **Tamaño operacional de clase (TOC)** | |
|---|---|
| Descripción: | Está dado por el número de métodos asignados a una clase. |
| **Atributos que afecta** | **Modo en que lo afecta** |
| **Responsabilidad** | El aumento del TOC provoca un aumento de la responsabilidad asignada a la clase. |
| **Complejidad de implementación** | El aumento del TOC provoca un aumento de la complejidad de implementación de la clase. |
| **Reutilización** | Un aumento del TOC provoca una disminución en el grado de reutilización de la clase. |

### 1.10.1.1 Criterio de evaluación de la métrica TOC

**Tabla 2 Criterio de evaluación de las métricas TOC.**

Fuente: (Rodríguez Landín, 2012)

| **Atributo** | **Categoría** | **Criterio** |
|---|---|---|
| **Responsabilidad** | Baja | <=Promedio |
| | Media | Entre Promedio y 2* Promedio |
| | Alta | > 2* Promedio |
| **Complejidad de implementación** | Baja | <=Promedio |

| | | |
|---|---|---|
| | Media | Entre Promedio y 2* Promedio |
| | Alta | > 2* Promedio |
| **Reutilización** | Baja | > 2* Promedio |
| | Media | Entre Promedio y 2* Promedio |
| | Alta | <=Promedio |

### 1.10.2 Relaciones entre clases (RC)

Mide los siguientes atributos de calidad:

- **Reutilización:** consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento:** consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de "reutilización".
- **Complejidad del mantenimiento:** consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costos y la planificación del proyecto.
- **Cantidad de pruebas:** consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad del producto diseñado.

**Tabla 3 Métrica de relación entre clases RC.**

Fuente: (Rodríguez Landín, 2012)

| **Relaciones entre clases (RC)** | |
|---|---|
| **Descripción:** | Está dada por el número de relaciones de uso de una clase con otras. |
| **Atributos que afecta:** | **Modo en que lo afecta:** |
| **Acoplamiento** | El aumento del RC provoca un aumento del acoplamiento de la clase. |
| **Complejidad del mantenimiento** | El aumento del RC provoca un aumento de la complejidad del mantenimiento de la clase. |
| **Reutilización** | El aumento del RC provoca una disminución en el grado de reutilización de |

| | |
|---|---|
| | la clase. |
| **Cantidad de pruebas** | El aumento del RC provoca un aumento de la cantidad de pruebas de unidad necesarias para probar una clase. |

### 1.10.2.1 Criterio de evaluación de la métrica RC

**Tabla 4 Criterios de evaluación de las métricas RC.**

Fuente: (Rodríguez Landín, 2012)

| Atributo | Categoría | Criterio |
|---|---|---|
| **Acoplamiento** | Ninguno | 0 |
| | Bajo | 1 |
| | Medio | 2 |
| | Alto | >2 |
| **Complejidad del mantenimiento** | Baja | <=Promedio |
| | Media | Entre Promedio y 2* Promedio |
| | Alta | > 2* Promedio |
| **Cantidad de pruebas** | Baja | <=Promedio |
| | Media | Entre Promedio y 2* Promedio |
| | Alta | > 2* Promedio |
| **Atributo** | **Categoría** | **Criterio** |
| **Reutilización** | Baja | > 2* Promedio |
| | Media | Entre Promedio y 2* Promedio |
| | Alta | <=Promedio |

## 1.11 Herramientas y tecnologías

Para el desarrollo de la propuesta de solución se utilizaron diferentes herramientas y tecnologías definidas en el proyecto, en el documento *Vista entorno de desarrollo tecnológico* (Cálas Torres, 2016).

### 1.11.1 Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar cómputos que pueden ser llevados a cabo por las computadoras. Puede usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión o como modo de comunicación humana (Lorena Suárez, 2010).

**PHP 5.4**

Es un lenguaje de programación utilizado para la creación de sitio web. PHP es un acrónimo recursivo que significa “PHP Hypertext Pre-processor”, (inicialmente se llamó Personal Home Page). PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o IIS con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. Últimamente también se puede usar para la creación de otro tipo de programas incluyendo aplicaciones con interfaz gráfica usando la biblioteca GTK+ (Cowburn , 2001)(Cobo, y otros, 2005).

**JavaScript**

Es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo y dinámico. Es simple, no hace falta tener conocimientos avanzados de programación para poder hacer un programa en JavaScript. Maneja objetos dentro de la página web y sobre ese objeto se pueden definir diferentes eventos. Dichos objetos facilitan la programación de páginas interactivas, a la vez que se evita la posibilidad de ejecutar comandos que puedan ser peligrosos para la máquina del usuario, tales como formateo de unidades y modificar archivos (JavaScript, 2015).

**CSS 3**

Es un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML. El W3C es el encargado de formular la especificación de las hojas de estilo que sirven de estándar para los navegadores. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. Este lenguaje es un web entero, de modo que se puede definir la forma de todo el web de una sola vez. Un documento HTML o página, se puede definir la forma, en un pequeño trozo de código en la cabecera, a

toda la página. Una porción del documento, aplicando estilos visibles en un trozo de la página. Una etiqueta en concreto, llegando incluso a poder definir varios estilos diferentes para una sola etiqueta. Esto es muy importante ya que ofrece potencia en nuestra programación. Podemos definir, por ejemplo, varios tipos de párrafos: en rojo, en azul, con márgenes, y sin ellos (Guzmán Ojeda, y otros, 2013).

**HTML 5**

Es un lenguaje markup usado para estructurar y presentar el contenido para la web. Es uno de los aspectos fundamentales para el funcionamiento de los sitios, pero no es el primero. Es de hecho la quinta revisión del estándar que fue creado en 1990. A fines del año pasado, la W3C la recomendó para transformarse en el estándar a ser usado en el desarrollo de proyectos venideros.HTML 5 está relacionado también con la entrada en decadencia del viejo estándar HTML 4, que se combinaba con otros lenguajes para producir los sitios que podemos ver hoy en día. Con HTML 5, tenemos otras posibilidades para explotar usando menos recursos; también entra en desuso el formato XHTML, dado que ya no sería necesaria su implementación (Eguiluz, 2010).

### 1.11.2 Marcos de trabajo

Desde el punto de vista del desarrollo de software, un marco de trabajo es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Los cuales suelen incluir: soporte de programa, bibliotecas, lenguaje de scripting, software para desarrollar y unir diferentes componentes de un proyecto de desarrollo de programas. Permiten facilitar el desarrollo de software, evitar los detalles de bajo nivel, posibilitando concentrar más esfuerzo y tiempo en identificar los requerimientos de software (Alegsa, 2010).

**Symfony 2.7**

Es un proyecto PHP de software libre que permite crear aplicaciones y sitios web rápidos y seguros de forma profesional; además es un marco de trabajo para PHP completo construido con varios componentes independientes creados por el proyecto Symfony. La documentación del proyecto también es libre e incluye varios libros y decenas de tutoriales específicos. Aprender a programar con Symfony te permite acceder a una gran variedad de proyectos: el framework Symfony2 para crear aplicaciones complejas, el micro framework Silex para sitios web sencillos y los componentes Symfony para otras aplicaciones PHP. Aunque en su

desarrollo participan cientos de programadores de todo el mundo, las decisiones técnicas importantes siempre las toma Fabien Potencier, líder del proyecto. Esto evita el peligro de que surjan forks absurdos y la comunidad se fragmente (Eguiluz, 2013) (Symfony, 2007).

**Bosón 1.0**

El marco trabajo Bosón no es más que un conjunto de componentes desarrollados en Symfony2 que responden a la mayoría de los resquisitos tecnológicos de un amplio espectro de aplicaciones.Los componentes son los siguentes: Caché,Aspect, Seguridad, ADT, IUX, Portal, Excepciones, Trazas, Integrador y el de **Estructura y Composición** (Calas Torres, 2015).

**Angular JS**

Es un framework de JavaScript de código abierto, mantenido por Google, que ayuda con la gestión de lo que se conoce como aplicaciones de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles. La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript. Los valores de las variables de JavaScript se pueden configurar manualmente, o recuperados de los recursos JSON estáticas o dinámicas. (Solis, 2013)

Se seleccionó Angular JS ya que es un marco de trabajo estructural que permite generar las bases fundamentales para el desarrollo de aplicaciones web dinámicas, y es la que tiene definido el proyecto para el desarrollo de interfaces gráficas.

### 1.11.3 Servidor web

Un servidor web es un programa que sirve datos en forma de páginas web, hipertextos o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos. La comunicación de estos datos entre cliente y servidor se hace por medio del protocolo HTTP. Con esto, un servidor web se mantiene a la espera de peticiones HTTP, que son ejecutadas por un cliente HTTP; lo que se conoce como un navegador web (Pavón Mestras, 2010).

**Apache 2.4.7**

Está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Las diferentes plataformas y entornos, hacen que a menudo sean necesarias diferentes características o funcionalidades; Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular.

Este diseño permite a los administradores de sitios web elegir qué características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor. Este es el más común y más utilizado en todo el mundo.

Además, es gratuito, y de código abierto, así que se puede decir que se ejecuta sobre cualquier plataforma. Apache es una muestra, al igual que el sistema operativo Linux (un Unix desarrollado inicialmente para PC), de que el trabajo voluntario y cooperativo dentro de internet es capaz de producir aplicaciones de calidad profesional difíciles de igualar. (Bowen, 2000) (Foundation, 2007)

### 1.11.4 Entorno integrado de desarrollo

Un entorno integrado de desarrollo (IDE por sus siglas en inglés), es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de GUI (Fergarciac, 2013).

**PHPStorm 8.0**

Editor de código que ofrece un excelente soporte para PHP (incluyendo las últimas versiones de idioma y marcos), HTML, JavaScript, CSS, Sass, Menos, CoffeeScript, y muchos otros idiomas. Permite el completamiento de código sensible al contexto, detección de errores, y las inspecciones y correcciones sobre la marcha de código. PhpStorm es un descendiente de IntelliJ IDEA, la JetBrains IDE Java, y es básicamente una versión simplificada con soporte para PHP incrustado. Debido a esta naturaleza este IDE puede soportar otros idiomas con la misma facilidad, lo que le permite desarrollar NodeJS, Dart, Go y otras aplicaciones de idiomas en el mismo entorno - un beneficio invaluable. Es rápido teniendo en cuenta su tamaño, soporta muchos idiomas y marcos; es multiplataforma (Packt Publishing, 2013).

### 1.11.5 Herramienta de modelado

Las herramientas CASE comprenden un conjunto de programas de diferentes tipos empleados para ayudar a las actividades del proceso del software como el análisis de requisitos, el modelado de sistemas, la depuración y las pruebas (Sommerville, 2005).

**Visual Paradigm 8.0**

Es una de las herramientas CASE del mercado considerada muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Permite graficar los diferentes diagramas UML, revertir y generar código fuente para Java, C++, PHP, DotNet Exe/dll, XML, XML Schema, Python y Corba IDL. Esta herramienta incluye los objetos más recientes de UML además de diagramas de casos de uso, diagramas de clase y diagramas de componentes. La misma ofrece soporte para Rational Rose, integración con Microsoft Visio, además permite generar reportes y documentación en HTML/PDF. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además apoya los estándares más recientes de las notaciones de Java y de UML (Headquarters, 2006).

### 1.11.6 Mapeador Relacional de Objetos

**Doctrine 2.0**

Doctrine es un potente y completo sistema Mapeador Relacional de Objetos (por sus siglas en inglés ORM) para PHP con una capa de abstracción de base datos (DBAL) incorporada que permite exportar una base de datos a sus clases correspondientes y viceversa, o sea, a partir de las clases creadas y siguiendo las especificaciones de ORM, generar las tablas de la base de datos. Se encuentra en la parte superior de una poderosa DBAL. Una de sus principales características es la opción de escribir las consultas de base de datos en un objeto con una propiedad orientada al dialecto SQL llamado Doctrine Query Language (DQL), inspirada en Hibernate (HQL). Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria (JavaSMapper, 2012).

## 1.12 Pruebas de software

Como parte de la fase de ejecución de la metodología seleccionada se valida la propuesta de solución mediante la estrategia de prueba, la misma integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a la evaluación correcta del software. La estrategia de pruebas de software proporciona un mapa que describe los pasos que se dan para realizarla, indica cuando se planea y se dan dichos pasos, además cuanto tiempo, esfuerzo y recursos son consumidos. Un software se prueba para descubrir los errores

cometidos, si se realiza sin ningún plan seguramente se desperdicia tiempo, un esfuerzo innecesario y lo que es peor puede que no se detecten los errores (Pressman, 2009).

Las pruebas integran un elemento más amplio el cual se llama verificación y validación (Pressman, 2009). Las estrategias de prueba siguen varios objetivos entre los que se encuentran:

Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de unidad, integración, de alto nivel o pruebas de validación y las pruebas de sistema.

Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, cómo realizar las pruebas y creando, si es posible, componentes de prueba ejecutables para automatizar las pruebas (Pressman, 2009).

Realizar diferentes pruebas y manejar los resultados de cada prueba sistemáticamente (Pressman, 2009).

### 1.12.1 Niveles de prueba

A la hora de evaluar un sistema generalmente se comienza probando las partes más pequeñas y se continúa con las más grandes. El módulo (componente) se prueba primero y luego se continúa con la integración de módulos (Rodrígez Tello, 2006). Las pruebas se aplican en distintos niveles de trabajo, entre las que se encuentran:

**Pruebas de unidad:** se concentran en probar cada componente individualmente para asegurar que funcione de manera apropiada como unidad. Principalmente se utiliza el método de caja blanca o estructural que suelen llevarse a cabo con el acceso al código fuente, el cual emplea técnicas de prueba que recorren caminos específicos en la estructura de control de los componentes (Pressman, 2009) (Rodrígez Tello, 2006).

**Pruebas de integración:** las pruebas de integración tienen dos objetivos principales: descubrir errores asociados con las interfaces de los módulos y ensamblar sistemáticamente los módulos individuales para formar subsistemas y al final un sistema completo. Principalmente se utiliza el método de caja negra o funcional con la técnica de diseño de casos de prueba que verifican el correcto manejo de las entradas y salidas del software (Pressman, 2009) (Rodrígez Tello, 2006).

**Pruebas de validación:** se enfocan en los requerimientos, se validan los requisitos establecidos como parte del análisis de requisitos del software, comparándolos con el sistema que ha sido construido (Pressman, 2009) (Rodrígez Tello, 2006).

**Pruebas del sistema:** este nivel es más adecuado para comprobar requisitos no funcionales como seguridad, velocidad, exactitud, fiabilidad etc. Se enfoca en la integración del sistema (hardware, información, personas) (Pressman, 2009) (Rodrígez Tello, 2006).

Para detectar y corregir errores que no fueron encontrados en el transcurso de la implementación, se aplican las pruebas de integración para descubrir errores asociados con las interfaces del componete y las pruebas de aceptación como parte de las pruebas de validación de los requisitos establecidos, comparándolos con el componente que ha sido construido.

**Pruebas de aceptación**

Son pruebas que permiten que el cliente valide todos los requisitos. Las realiza el usuario final en lugar del responsable del desarrollo del sistema, una prueba de aceptación puede ir desde un informal caso de prueba hasta la ejecución sistemática de una serie de pruebas bien planificadas. La prueba de aceptación puede tener lugar a lo largo de semanas o meses, descubriendo así errores acumulados que pueden ir degradando el sistema (Pressman, 2006).

### 1.12.2 Métodos de prueba

Para realizar las pruebas seleccionadas, se hace uso de los diferentes métodos existentes en cada una de ellas. Los métodos de prueba tienen un enfoque sistemático, independiente del nivel en que se enmarque la prueba, que ayuda a encontrar buenos conjuntos de casos de prueba para detectar diferentes tipos de errores. Existen dos métodos básicos para diseñar casos de prueba: de caja blanca (o estructural) y de caja negra (o funcional) (Pressman, 2006).

**Pruebas de caja negra o funcionales**

Las pruebas de caja negra verifican el correcto manejo de funciones externas provistas o soportadas por el software y que el comportamiento observado se apegue a las especificaciones del producto y a las expectativas del usuario (Pressman, 2006).

La técnica que se emplea para encontrar errores de funciones incorrectas o faltantes, errores de interfaz, errores en estructuras de datos, de comportamiento o desempeño y errores de inicialización o término es el diseño de casos de prueba es particiones de equivalencia. Su

objetivo es probar todos los requisitos funcionales del sistema y analizar la respuesta que dan ante una ejecución de los mismos y recopilar las no conformidades obtenidas para su posterior corrección.

### 1.12.3 Tipos de pruebas

Entre los tipos de pruebas que pueden aplicarse se encuentran (Pressman, 2009):

**Funcionalidad:** verifican la habilidad del software para realizar el trabajo deseado.
**Usabilidad:** verifican la habilidad del software para satisfacer al usuario.
**Portabilidad:** verifican la habilidad del software para correr en diferentes entornos informáticos.

El principal objetivo del flujo de pruebas es evaluar la calidad del producto a través de la búsqueda y documentación de errores, la validación del cumplimiento de los requisitos y la validación del desempeño.

## 1.13 Conclusiones parciales

Con el estudio de los principales conceptos y herramientas informáticas vinculado a la gestión de Estructura y Composición de una institución y el análisis de tecnologías y herramientas de desarrollo, se llegó a las siguientes conclusiones:

- El análisis de las herramientas asociados a la gestión de Estructura y Composición, permitió corroborar que ninguna puede ser utilizada como base para el desarrollo de la solución, lo que ratificó la necesidad de crear las interfaces para darle solución al problema existente e incluir algunas de las buenas prácticas estudiadas de dichas herramientas.
- El análisis de las técnicas tormenta de ideas y entrevista permitió establecer una adecuada captura de requisitos. En cuanto a las técnicas para validar los requisitos se decidió utilizar la revisión técnica formal y la construcción de prototipos.
- Para el desarrollo de la solución fue necesario usar varias tecnologías y herramientas, las cuales fueron asignadas en el departamento de desarrollo de componentes del centro CEIGE. Además, se decidió aplicar pruebas de caja negra y de aceptación para verificar el correcto funcionamiento de la solución.

# CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

## 2.1 Introducción

En el presente capítulo se realiza una propuesta de solución para darle cumplimiento al objetivo general planteado y se describen las características que deben poseer las interfaces a desarrollar. Se enumeran además los requisitos funcionales y no funcionales con los que debe cumplir el mismo, así como la construcción del diagrama de clases del diseño con estereotipos web.

## 2.2 Requisitos funcionales

Una vez aplicadas las técnicas de captura de requisitos se obtuvo un total de 29 requisitos agrupados por la técnica de agrupación funcional. A continuación, se representan los requisitos funcionales del componente, expresados en lenguaje natural. Los mismos son identificados con las siglas RF_ más el número del requisito (Ej. RF-1.). Estos requisitos permiten definir las interfaces gráficas del componente de Estructura y Composición enfocándose en las necesidades y aspiraciones de los usuarios.

**Tabla 5 Descripción de los requisitos funcionales.**

Fuente: (Elaboración propia)

| Gestionar nomenclador | | |
|---|---|---|
| **Identificador** | **Nombre** | **Descripción** |
| RF_1 | Adicionar nomenclador | Permite adicionar un nuevo nomenclador al sistema |
| RF_2 | Modificar nomenclador | Permite modificar un nomenclador existente en el sistema. |
| RF_3 | Eliminar nomenclador | Permite eliminar un nomenclador existente en el sistema a partir de su identificador. |
| RF_4 | Listar nomencladores | Permite mostrar en un listado todos los nomencladores existentes en el sistema. |

| **Gestionar estructura** | | |
|---|---|---|
| **Identificador** | **Nombre** | **Descripción** |
| RF_5 | Adicionar estructura | Permite adicionar una nueva estructura al sistema. |
| RF_6 | Modificar estructura | Permite modificar una estructura existente en el sistema. |
| RF_7 | Eliminar estructura | Permite eliminar una estructura existente en el sistema a partir de su identificador. |
| RF_8 | Buscar estructura | Permite mostrar la estructura existente en el sistema a la cual corresponde un identificador dado. |
| RF_9 | Listar estructura | Permite mostrar en un listado todas las estructuras existentes en el sistema. |
| RF_10 | Crear relación entre estructuras | Permite crear una nueva relación entre dos estructuras existente en el sistema. |
| RF_11 | Eliminar relación entre estructuras | Permite eliminar una relación entre dos estructuras existente en el sistema. |
| **Gestionar instancia de nomenclador** | | |
| **Identificador** | **Nombre** | **Descripción** |
| RF_12 | Adicionar instancia a nomenclador | Permite adicionar una nueva instancia a nomenclador al sistema. |
| RF_13 | Modificar instancia a nomenclador | Permite modificar una instancia a nomenclador existente en el sistema. |
| RF_14 | Eliminar instancia a nomenclador | Permite eliminar una instancia a nomenclador existente en el sistema a partir |

| | | |
|---|---|---|
| | | de su identificador. |
| RF_15 | Buscar instancia de nomenclador | Permite mostrar la instancia de nomenclador existente en el sistema a la cual corresponde un identificador dado |
| RF_16 | Listar instancias a nomenclador | Permite mostrar en un listado todas las instancias a nomenclador existentes en el sistema. |
| **Gestionar instancia estructura** | | |
| **Identificador** | **Nombre** | **Descripción** |
| RF_17 | Adicionar instancia a estructura | Permite adicionar una nueva instancia a estructura al sistema. |
| RF_18 | Modificar instancia a estructura | Permite modificar instancia a estructura existente en el sistema. |
| RF_19 | Eliminar instancia a estructura | Permite eliminar una instancia a estructura existente en el sistema a partir de su identificador. |
| RF_20 | Buscar instancia de estructura | Permite mostrar la instancia de estructura existente en el sistema a la cual corresponde un identificador dado. |
| RF_21 | Listar instancias a estructura | Permite mostrar en un listado todos las instancia a estructura existentes en el sistema |
| **Gestionar campo de nomenclador** | | |
| **Identificador** | **Nombre** | **Descripción** |
| RF_22 | Adicionar campo a nomenclador | Permite adicionar un nuevo campo a nomenclador al sistema. |

| | | |
|---|---|---|
| RF_23 | Modificar campo a nomenclador | Permite modificar campo a nomenclador existente en el sistema. |
| RF_24 | Eliminar campo a nomenclador | Permite eliminar campo a nomenclador existente en el sistema a partir de su identificador. |
| RF_25 | Listar campos a nomenclador | Permite mostrar en un listado todos los campo a nomenclador existentes en el sistema |
| **Gestionar campo de estructura** | | |
| **Identificador** | **Nombre** | **Descripción** |
| RF_26 | Adicionar campo a estructura | Permite adicionar un nuevo campo a estructura al sistema. |
| RF_27 | Modificar campo a estructura | Permite modificar campo a estructura existente en el sistema. |
| RF_28 | Eliminar campo a estructura | Permite eliminar campo a estructura existente en el sistema a partir de su identificador. |
| RF_29 | Listar campos a estructura | Permite mostrar en un listado todos los campo a estructura existentes en el sistema |

### 2.2.1 Historias de usuario

De los requisitos funcionales identificados fueron diseñadas sus historias de usuario. A continuación, se muestra en la Tabla 6, el ejemplo HU del requisito “Modificar estructura”, el resto de ellas son mostradas en los anexos.

**Tabla 6 HU del requisito funcional “Modificar estructura”.**

Fuente: (Elaboración propia)

| | |
|---|---|
| **Número:** 53 | **Nombre del requisito:** Modificar estructura |
| **Programador:** Liliana Simón Figueredo | **Iteración Asignada:** 1 |
| **Prioridad:** Alta | **Tiempo Estimado:** 16 horas |
| **Riesgo en Desarrollo:** Poca experiencia de los estudiantes en las tecnologías de desarrollo del proyecto. | **Tiempo Real:** N/A |
| **Descripción:**<br>Este requisito se encarga de modificar una estructura de las existentes en el sistema. Se le puede modificar a la estructura el nombre y si es raíz. Cuando se modifica en caso que los valores sean incorrectos el sistema muestra un mensaje indicando que existen valores erróneos, en caso contrario muestra un mensaje indicando el éxito de la operación. | |
| **Observaciones:** N/A | |
| **Prototipo de interfaz:**<br>  | |

## 2.3 Requisitos no funcionales

Los requerimientos no funcionales (RNF) son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Normalmente, están vinculados a requerimientos funcionales, es decir una vez se conozca lo que el sistema debe hacer se puede determinar

cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser (Sommerville, 2011). A continuación, se exponen el requisito no funcional de las interfaces a desarrollar.

**Tabla 7 Descripción de los requisitos no funcionales.**

Fuente: (Elaboración propia)

| Identificador | Clasificación | Requisito |
|---|---|---|
| RNF_1 | Usabilidad | El sistema debe brindar un acceso fácil y rápido, para facilitar el uso del mismo por usuarios con pocos conocimientos en el campo de la informática. |
| RNF_2 | Interfaz | La interfaz de la aplicación a desarrollar debe ser sencilla para reducir el tiempo de capacitación de los usuarios. |
| RNF_3 | Software | El sistema funciona sobre las plataformas Windows y Linux. |
| RNF_4 | Hardware | Para el servidor los requerimientos mínimos deben ser un servidor Core-i3 a 2.2 GHz de velocidad de procesamiento y más de 2Gb de memoria RAM, con al menos 40Gb de espacio libre en disco duro, además de una tarjeta de red. |
| RNF_5 | Hardware | Para el cliente los requerimientos mínimos deben ser una computadora Pentium III a 1.0 GHz con 1 Gb de memoria RAM, además de una tarjeta de red. |

## 2.4 Arquitectura del software

La arquitectura de software se define, a grandes rasgos, como una vista del sistema que incluye los componentes principales del mismo, su conducta, y las formas en que interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones (Clements, 1996).

El desarrollo del componente de Estructura y Composición está definido a partir del marco de trabajo Symfony2, una arquitectura basada en cuatro capas: (Capa de Presentación, la Capa de Negocio, la Capa de Acceso a Datos y la Capa de Datos), con el uso del patrón

arquitectónico **Modelo-Vista-Controlador (MVC),** muy beneficioso en aplicaciones que manejan gran cantidad de datos y transacciones complejas, donde se requiere una mejor separación de conceptos facilitando la programación en diferentes capas de manera paralela e independiente (Gómez, 2011).

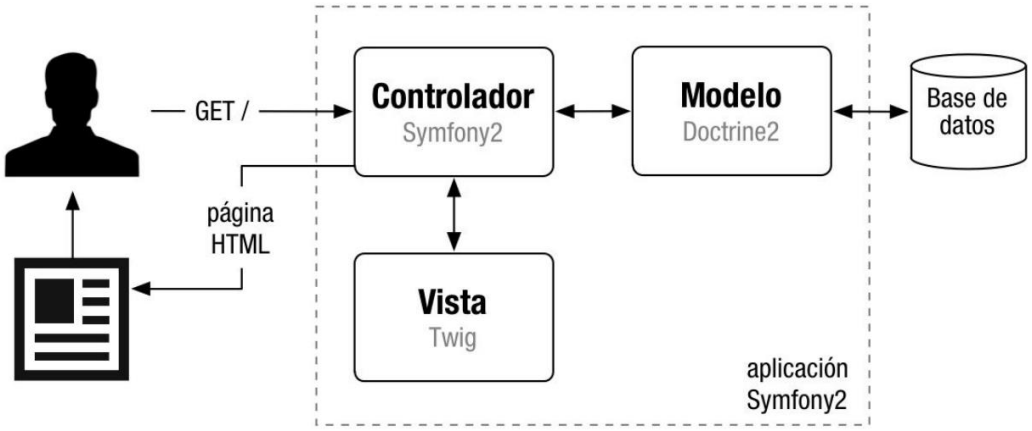


**Figura 7 Funcionamiento general del patrón MVC.**

Fuente: (Symfony, 2007)

El desarrollo de las interfaces gráficas del componente está centrado en la **Capa de Presentación:** en esta capa están los componentes con los que el usuario va a interactuar. Los mismos permiten utilizar las funcionalidades que brinda el controlador y mostrar o capturar la información a través de los diferentes elementos que comprende, es decir: formularios, grids, entre otros.

Esta se divide en dos sub-capas: cliente y servidor. En la primera se visualiza mediante el navegador web (utilizando tecnologías como HTML5, CSS3, JavaScript y Angular JS) datos procesados. En la segunda se maneja la lógica de control, así como la construcción de páginas y formularios (Gómez, 2011).

## 2.5 Modelo de datos

Uno de los artefactos generados en la disciplina de análisis y diseño es el modelo de datos. En la Figura 8 se representa el modelo de datos del componente Estructura y Composición. El mismo está compuesto por un total de 9 tablas donde se describen las estructuras de datos, su tipo, la forma en que se relacionan y las restricciones de integridad entre estas tablas.

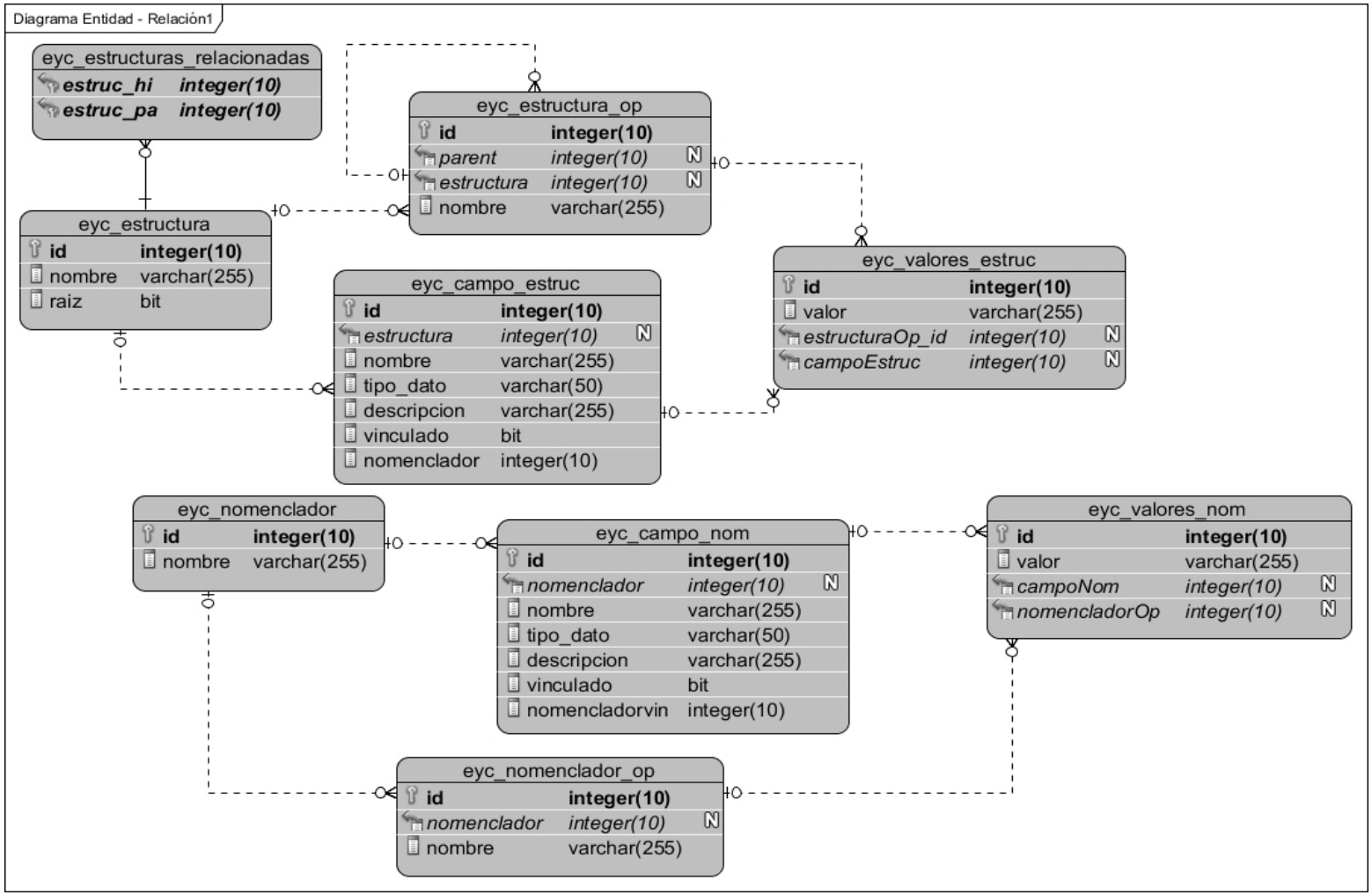


**Figura 8 Modelo de datos del componente Estructura y Composición.**

Fuente: (Elaboración propia)

## 2.6 Modelo del diseño

El modelo de diseño ha sido definido como “una abstracción del modelo de implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño” (Pressman, 2006). El modelo de diseño puede contener: diagramas, clases, paquetes, interfaces, entre otros, que son utilizados como entrada esencial en las actividades relacionadas a la implementación. En el presente epígrafe se muestran los elementos que se tuvieron en cuenta para el diseño de la solución.

**Diagrama de clases del diseño con estereotipos web**

El diagrama de clases muestra un conjunto de clases, interfaces y las relaciones entre éstas. Los diagramas de clases muestran el diseño de un sistema desde un punto de vista estático; un diagrama estático es el que muestra una colección de elementos (estáticos) declarativos (Pressman, 2006). En la Figura 9 se muestra el diagrama de clases del diseño del componente de Estructura y Composición, representado por las clases servidoras EstructuraController y EyCEstructuraCtrl las cuales se encargan de procesar las interacciones del usuario y realizan los cambios apropiados en el modelo o en la vista, el formulario del componente envía los datos al código servidor para ser procesados, el código servidor solicita y envía la información a través de la capa modelo, representa la información con la que trabaja la aplicación, es decir, su lógica de negocio. En la construcción de los formularios se utiliza la librería Angular JS para las vistas, la cual transforma el modelo en una página web que permite al usuario interactuar con ella.

A continuación, se muestra el diagrama de clases del diseño del componente de Estructura y Composición “Gestionar estructura”, los diagramas de clases “Gestionar nomenclador” y “Nivel estructural “se encuentran en los anexos:

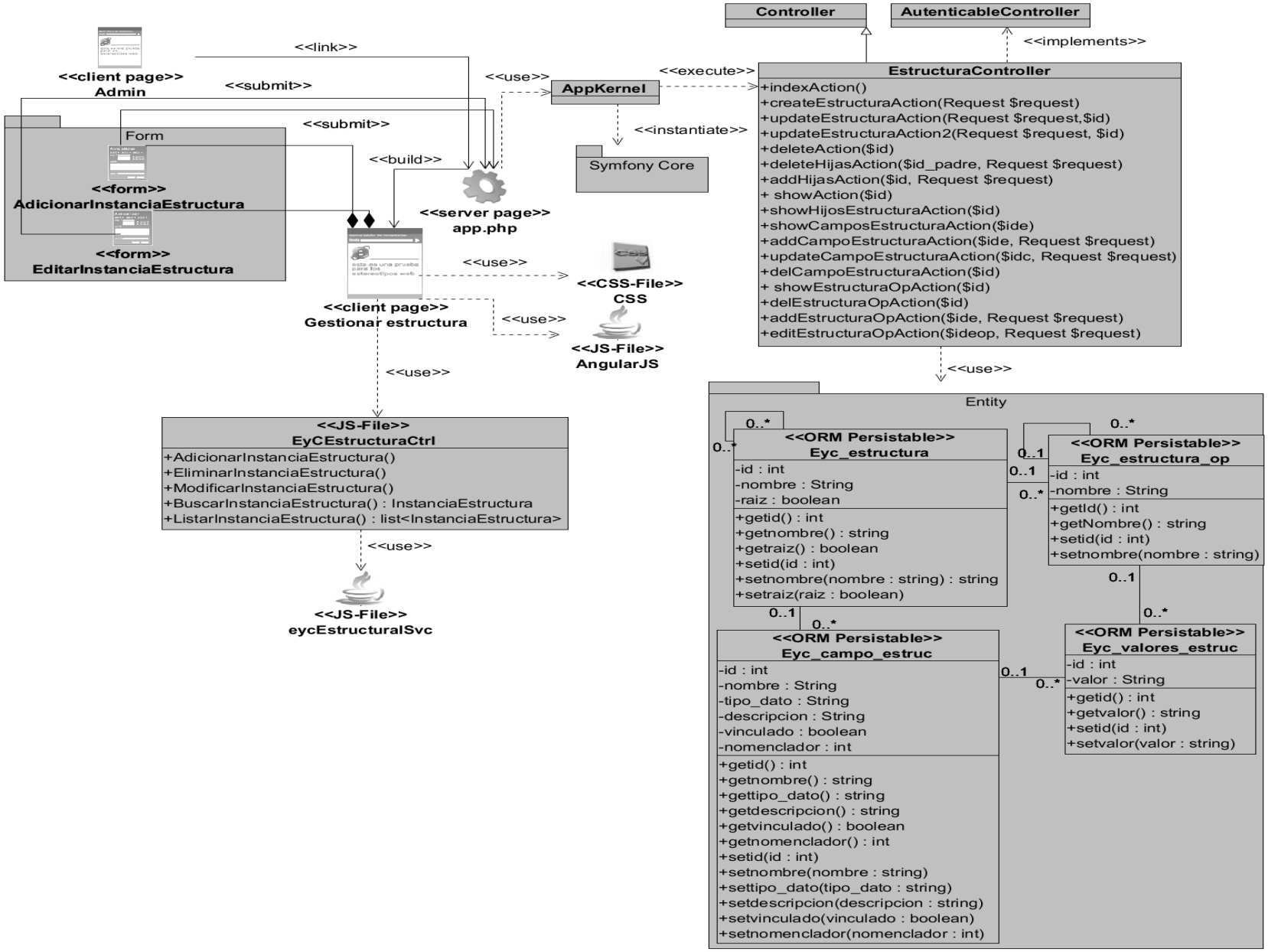
Controller
AutenticableController
<<implements>>
<<client page>>
Admin
<<link>>
<<submit>>
<<use>>
AppKernel
<<execute>>
EstructuraController
+indexAction()
+createEstructuraAction(Request $request)
+updateEstructuraAction(Request $request,$id)
+updateEstructuraAction2(Request $request, $id)
+deleteAction($id)
+deleteHijasAction($id_padre, Request $request)
+addHijasAction($id, Request $request)
+ showAction($id)
+showHijosEstructuraAction($id)
+showCamposEstructuraAction($ide)
+addCampoEstructuraAction($ide, Request $request)
+updateCampoEstructuraAction($idc, Request $request)
+delCampoEstructuraAction($id)
+ showEstructuraOpAction($id)
+delEstructuraOpAction($id)
+addEstructuraOpAction($ide, Request $request)
+editEstructuraOpAction($ideop, Request $request)
<<instantiate>>
Symfony Core
Form
<<form>>
AdicionarInstanciaEstructura
<<form>>
EditarInstanciaEstructura
<<build>>
<<server page>>
app.php
<<client page>>
Gestionar estructura
<<use>>
<<CSS-File>>
CSS
<<JS-File>>
AngularJS
<<JS-File>>
EyCEstructuraCtrl
+AdicionarInstanciaEstructura()
+EliminarInstanciaEstructura()
+ModificarInstanciaEstructura()
+BuscarInstanciaEstructura() : InstanciaEstructura
+ListarInstanciaEstructura() : list<InstanciaEstructura>
<<JS-File>>
eycEstructuralSvc
Entity
<<ORM Persistable>>
Eyc_estructura
-id : int
-nombre : String
-raiz : boolean
+getid() : int
+getnombre() : string
+getraiz() : boolean
+setid(id : int)
+setnombre(nombre : string) : string
+setraiz(raiz : boolean)
<<ORM Persistable>>
Eyc_estructura_op
-id : int
-nombre : String
+getId() : int
+getNombre() : string
+setid(id : int)
+setnombre(nombre : string)
<<ORM Persistable>>
Eyc_campo_estruc
-id : int
-nombre : String
-tipo_dato : String
-descripcion : String
-vinculado : boolean
-nomenclador : int
+getid() : int
+getnombre() : string
+gettipo_dato() : string
+getdescripcion() : string
+getvinculado() : boolean
+getnomenclador() : int
+setid(id : int)
+setnombre(nombre : string)
+settipo_dato(tipo_dato : string)
+setdescripcion(descripcion : string)
+setvinculado(vinculado : boolean)
+setnomenclador(nomenclador : int)
<<ORM Persistable>>
Eyc_valores_estruc
-id : int
-valor : String
+getid() : int
+getvalor() : string
+setid(id : int)
+setvalor(valor : string)
0..*
0..1

**Figura 9 Diagrama de clase del diseño del componente de Estructura y Composición "Gestionar estructura".**

Fuente: (Elaboración propia)

Angular JS está basado en el patrón **MVC**. A diferencia de otras librerías, nos proporciona una herramienta sencilla para hacer que el código HTML5 sea dinámico, es decir, que se actualice en tiempo real (Wonderbits, 2012).

**Modelo Vista Controlador en Angular JS**

El patrón MVC se sigue en Angular JS, separando la parte visual de la funcionalidad y las estructuras de datos (Wonderbits, 2012):

El archivo HTML de la aplicación representa la vista y debe ser separada del controlador y el modelo.

El controlador es un objeto JavaScript que debe ser importado en el HTML que se encarga de capturar los eventos de la vista y realizar las acciones pertinentes sobre el modelo para modificar los datos.

El modelo es un objeto de JavaScript nombrado como elemento del controlador "$scope". Contiene los datos a los que va a acceder la vista y debe contener también métodos de acceso y modificación para separar totalmente la forma a la que se acceden o modifican los datos del controlador.

A continuación, se muestra el diagrama de clases del diseño con estereotipos web de la vista del componente de Estructura y Composición de "Gestionar estructura "basándose en patrón MVC que sigue el marco de trabajo Angular JS:

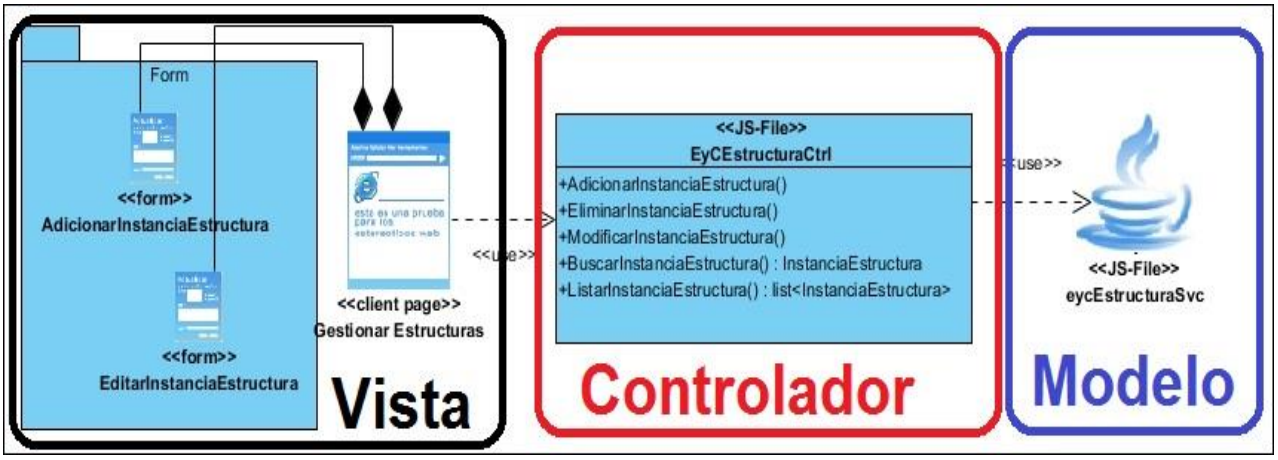

**Figura 10 Diagrama de clase del diseño de la vista del componente de Estructura y Composición.**

Fuente: (Elaboración propia)

## 2.7 Patrones de diseño

Para evitar la sobrecarga de las clases y contribuir a que el componente proporcione un mayor nivel de reutilización y agilicen el proceso de desarrollo de software se utilizaron los patrones GRASP.

### 2.7.1 Experto en información

Es el patrón más utilizado debido a su particularidad, por lo cual es empleado en todas las clases entidades que contiene el componente. A continuación, se muestra en Figura 11 un ejemplo del patrón:

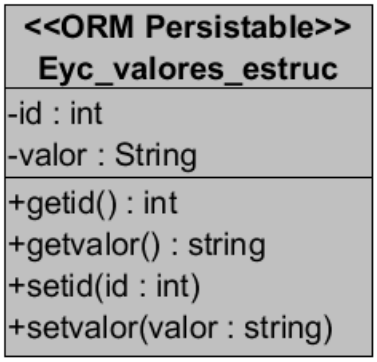


**Figura 11 Ejemplo de patrón experto.**

Fuente: (Elaboración propia)

### 2.7.2 Controlador

Dicho patrón se encuentra presente en todas las clases controladoras y sirve de intermediario entre las interfaces y el algoritmo que las implementa. Es el encargado de recibir los datos del usuario y enviarlos a las distintas clases según el método solicitado. Su filosofía principal se basa en que la lógica de negocio debe estar separada de la capa de presentación, con el propósito de aumentar la reutilización de código y tener un mayor control sobre los cambios.

A continuación, se muestra a través de la Figura 12 un ejemplo donde se refleja el patrón controlador:

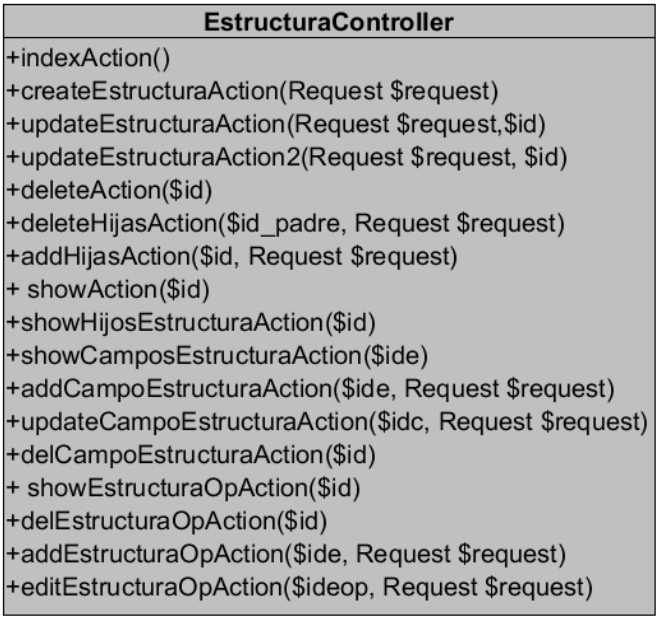


**Figura 12 Ejemplo de patrón controlador.**

Fuente: (Elaboración propia)

### 2.7.3 Creador

La utilización de este patrón se aprecia en las clases controladoras, las cuales se encargan de crear los objetos de las clases que representan las entidades.

### 2.7.4 Alta cohesión

El patrón tiene como propósito asignar responsabilidades a las clases de forma tal que la cohesión siga siendo alta, de tal manera que las clases se encuentren estrechamente relacionadas entre sí y no lleguen a realizar un trabajo excesivo. En el componente se evidencia la alta cohesión en las clases entidades y controladoras.

### 2.7.5 Bajo acoplamiento

El patrón bajo acoplamiento está estrechamente relacionado con los patrones experto y alta cohesión, el cual plantea la baja dependencia que debe existir entre las clases. Esto ocurre porque a cada clase se le asignan solamente las responsabilidades necesarias de manera que no dependan en gran medida de otras. Esto favorece la escalabilidad del componente, la reutilización y el mantenimiento futuro del mismo. Se evidencia el uso de este patrón en las clases entidades y controladoras.

## 2.8 Validación del diseño

Con el propósito de evaluar la calidad del diseño se utilizaron las métricas de software siguientes que permiten medir de forma cuantitativa la calidad de los atributos internos de las interfaces gráficas para componente propuesto:

### 2.8.1 Resultados de la aplicación de la métrica TOC

En el gráfico de la Figura 13 se puede observar el resultado de la evaluación de los atributos de la métrica TOC, el instrumento de evaluación de la métrica TOC es mostrado en los anexos:

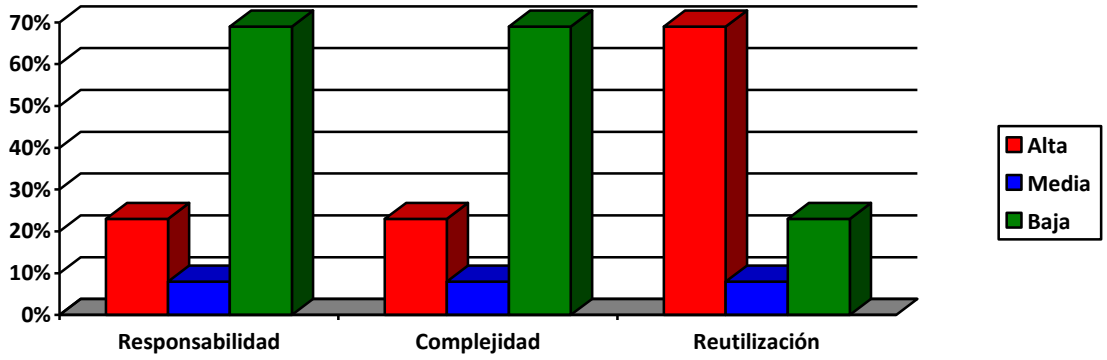


**Figura 13 Evaluación de los atributos de la métrica TOC.**

Fuente: (Elaboración propia)

Dado un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir, que el diseño realizado para los requisitos identificados en el desarrollo de las interfaces gráficas del componente de Estructura y Composición tiene una calidad aceptable; teniendo en cuenta que se obtuvieron resultados satisfactorios en los diferentes atributos de calidad medidos, estos resultados van unidos al hecho de que el 69.23% de las 13 clases medidas (9 clases) tienen 6 o menos operaciones. Por lo que se puede afirmar que se obtuvo una solución eficiente con altos niveles de reutilización (69%) garantizando que las operaciones realizadas puedan ser utilizadas en otras aplicaciones. También se evidencia que el diseño realizado fue correcto ya que se obtuvieron bajos niveles de responsabilidad (69%) y complejidad de implementación (69%), pues se les asignaron correctamente las responsabilidades a las clases involucradas en la solución.

### 2.8.2 Resultados de la aplicación de la métrica RC

En el gráfico de la Figura 14 se puede observar el resultado de la evaluación de los atributos de la métrica RC, el instrumento de evaluación de la métrica RC es mostrado en los anexos.

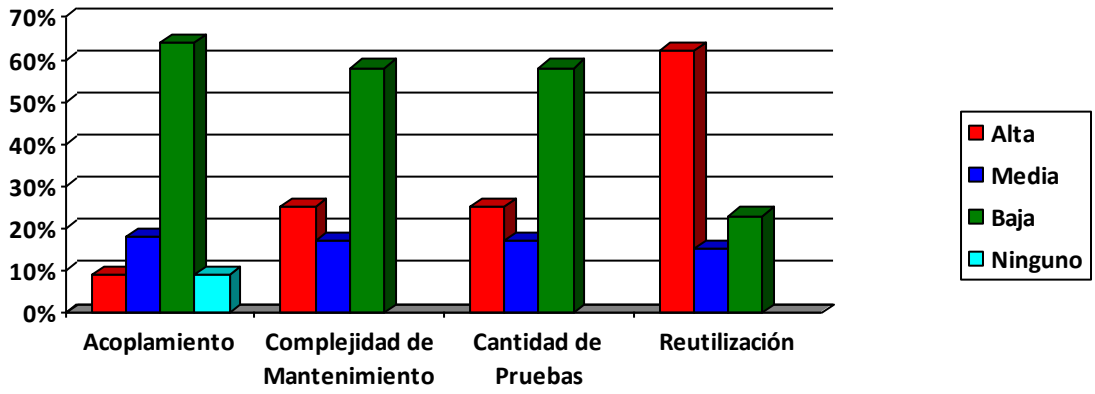


**Figura 14 Evaluación de los atributos de la métrica RC.**

Fuente: (Elaboración propia)

Dado un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir, que el diseño realizado para los requisitos identificados en el desarrollo de las interfaces gráficas del componente de Estructura y Composición tiene una calidad aceptable teniendo en cuenta que el 61.54% de las clases incluidas en el sistema poseen 1 o menos dependencias de otras clases. Por lo que, al ocurrir un cambio en alguna de las clases, la afectación en las restantes sea de poca importancia. El 64% de las clases poseen acoplamiento bajo, lo cual demuestra que una clase solo depende de las clases necesarias. También la aplicación de la métrica RC arrojó resultados satisfactorios en el atributo complejidad de mantenimiento con un 58% de las clases con índices bajo, lo que facilita las tareas de corrección, modificación y mantenimiento de las clases. El número o el grado de esfuerzo para realizar las pruebas de calidad del producto diseñado es bajo dado que la métrica aplicada arroja un 58% de bajo cantidad de pruebas fomentando así un alto índice de reutilización con un 62%.

## 2.9 Conclusiones parciales

Según lo expuesto en el capítulo se arriba a las siguientes conclusiones:

- A partir del uso de las técnicas para la validación de los requisitos funcionales se logró comprobar que los requisitos identificados se encuentran definidos de forma correcta y consistente, además que se corresponden con las perspectivas del cliente.
- La arquitectura de software permitió una mejor separación de conceptos, facilitando la programación en diferentes capas de manera paralela e independiente basándose en el patrón arquitectónico MVC.
- La aplicación de las métricas TOC y RC permitió verificar el correcto diseño demostrando que el mayor por ciento de las clases era reutilizable, tenían poca responsabilidad, así como baja complejidad en la realización de las pruebas y complejidad en la implementación de las clases. Los patrones GRASP permitieron obtener un sistema mantenible, escalable y flexible ante posibles cambios en la implementación de las interfaces graficas del componente.

# CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

## 3.1 Introducción

En el presente capítulo se describen los tipos de pruebas a realizar al componente de Estructura y Composición del marco trabajo Bosón, para comprobar el correcto funcionamiento de las interfaces, así como los estándares de codificación que presenta la herramienta desarrollada.

## 3.2 Diagrama de componentes

Los diagramas de componentes representan cómo un sistema de software es dividido en componentes y las dependencias entre ellos, los cuales son utilizados para modelar la vista estática y dinámica de un sistema lo que permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que los componentes proporcionan y utilizan a través de las interfaces (Microsoft, 2007).

El diagrama de componentes está compuesto por un componente principal EyCBundle que representa el sistema en general. Dentro de este se encuentra el paquete Vista que agrupa los componentes que son utilizados en las interfaces y con los cuales el usuario puede interactuar directamente, el paquete Modelo que agrupa los componentes relacionados con las clases de acceso a datos, las cuales son utilizadas por el controlador para dar respuesta a la vista y el componente Controller relacionado con la lógica del negocio, el cual obtiene las peticiones del usuario y da respuesta a la misma mediante su interacción con el modelo. El componente Repository del paquete Modelo utiliza la biblioteca Doctrine para el mapeo a la base de datos y el componente Resource del paquete Vista utiliza la biblioteca Angular JS.

A continuación, se representa el diagrama de componentes correspondiente al componente de Estructura y Composición.

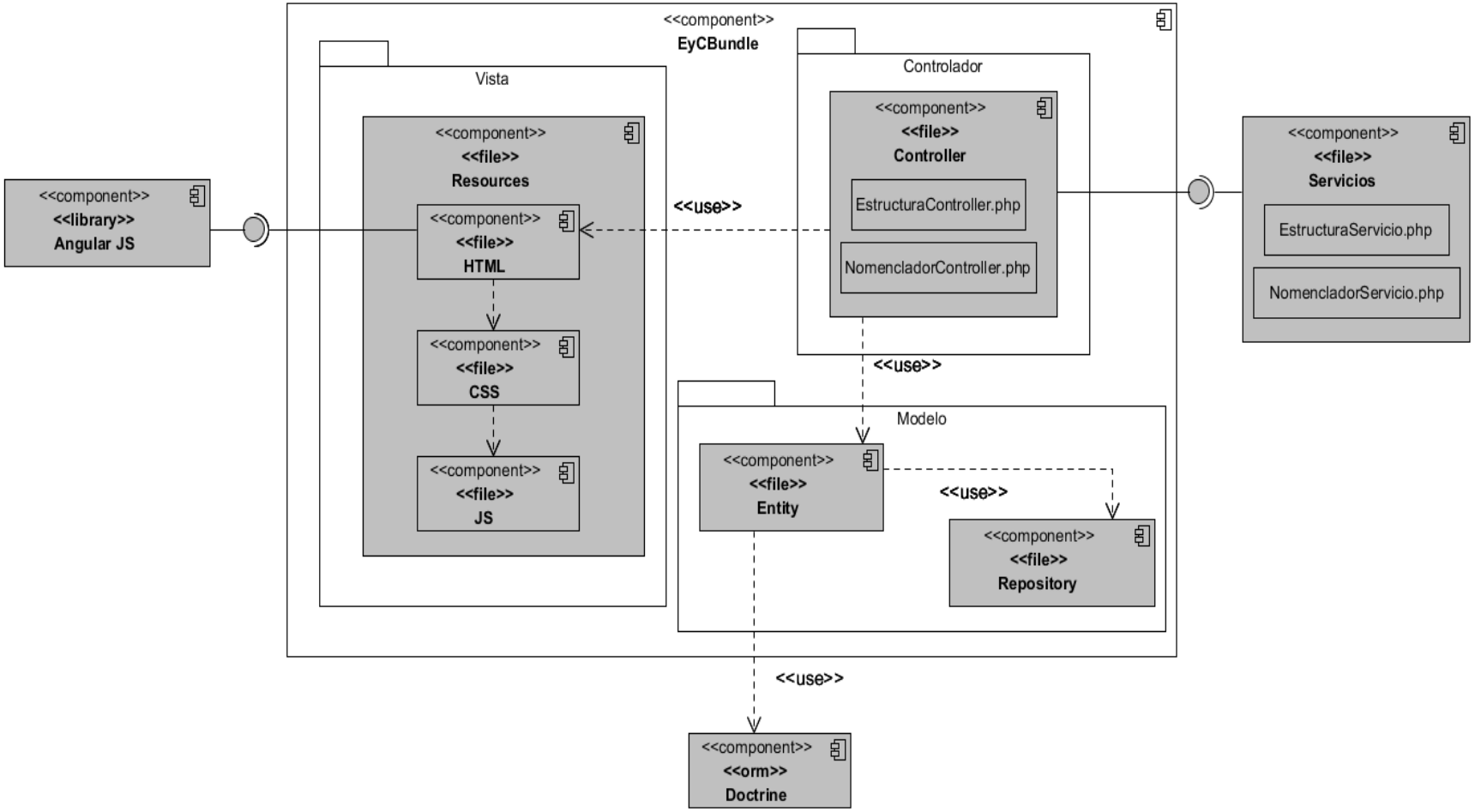


**Figura 15 Diagrama de componentes.**

Fuente: (Elaboración propia)

## 3.3 Estándares de codificación

Un factor importante para la implementación lo constituye la forma en que se construye el código fuente, esencialmente su organización y el estilo de codificación utilizado en el desarrollo. Una buena estructuración del código, mejora la lectura del software, permitiendo entender el código nuevo rápidamente y más a fondo (Robert Lobo, 2012).

El uso de estándares de codificación permite lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su mantenibilidad a lo largo del tiempo (Pérez Alfonso, 2012).

Para el desarrollo de la solución, se utilizan estándares y normas de codificación, los cuales se muestran a continuación:

- **Notación Húngara**: definir prefijos para cada tipo de datos y según el ámbito de las variables. La idea de esta notación es la de dar mayor información al nombre de la variable, método o función definiendo en ella un prefijo que indique su tipo de dato o ámbito (Sperberg, 2012).
- **Notación PascalCasing**: es como la notación húngara, pero sin prefijos. En este caso los identificadores y nombres de las variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula (Svensk, 2012).
- **Notación CamelCasing**: es parecido al PascalCasing con la excepción que la letra inicial del identificador no debe estar en mayúscula (Svensk, 2012).

### 3.3.1 Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se emplea la notación PascalCasing. Ejemplo: Estructura.

**Clase controladora**: después del nombre llevan la palabra “Controller”. Ejemplo: EstructuraController.

### 3.3.2 Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se emplea la notación CamelCasing, y en caso de ser

una acción de la clase controladora se debe especificar el nombre de dicha acción en minúscula y seguido el sufijo “Action”.

Ejemplo: updateEstructuraAction.

### 3.3.3 Nomenclatura de las variables

El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se emplea la notación CamelCasing. Ejemplo: $nombreEstructura.

## 3.4 Validación de la solución

A continuación, se muestra los resultados obtenidos en las pruebas de caja negra y de aceptación.

### 3.4.1 Método de prueba de caja negra

Para aplicar el método de prueba de caja negra se utilizó la técnica de particiones de equivalencia diseñando por cada requisito funcional un caso de prueba (Pressman, 2006).

#### 3.4.1.1 Diseño de caso de prueba

A continuación, se muestra el diseño de caso de prueba del requisito “Eliminar estructura”, el resto son mostrados en los anexos:

**Tabla 8 Diseño de caso de prueba del requisito “Eliminar estructura”.**

Fuente: (Elaboración propia)

| Escenario | Descripción | Estructura | Respuesta del sistema | Flujo central |
|---|---|---|---|---|
| EC 1.1 Eliminar estructura. | Se elimina una estructura de las que existen en el sistema. | V<br>Ministerio | El sistema elimina una estructura y muestra el mensaje: “La estructura ha sido eliminada satisfactoriamente”. | - Se selecciona una estructura de las listadas en la parte izquierda de la interfaz.<br>- Se presiona el botón **Eliminar** en la parte superior izquierda de la interfaz.<br>- Se muestra un mensaje de confirmación **“¿Está seguro que** |

| | | | | |
|---|---|---|---|---|
| | | | | **desea eliminar el elemento seleccionado?"**.<br>- Presiona el botón **Aceptar**.<br>- Se muestra un mensaje indicando el éxito de la operación.<br>- Se presiona el botón **Aceptar** del mensaje de notificación o se espera hasta que el mismo desaparece. |
| EC 1.2 Presionar botón **Cancelar.** | Se cancela la operación de eliminar una estructura. | V<br>Universidad | El sistema cancela la acción. | - Se selecciona una estructura de las listadas en la parte izquierda de la interfaz.<br>- Se presiona el botón **Eliminar** en la parte superior izquierda de la interfaz.<br>- Se muestra un mensaje de confirmación **"¿Está seguro que desea eliminar el elemento seleccionado?"**.<br>- Presiona el botón **Cancelar** para abortar la operación. |

### 3.4.2 Pruebas de aceptación

Con el propósito de evaluar las interfaces del componente desarrolladas por parte de los usuarios finales, le fueron aplicadas pruebas de aceptación por el arquitecto del departamento Ing. Abraham Calas Torres, obteniendo los resultados que se muestran en la Figura 16.

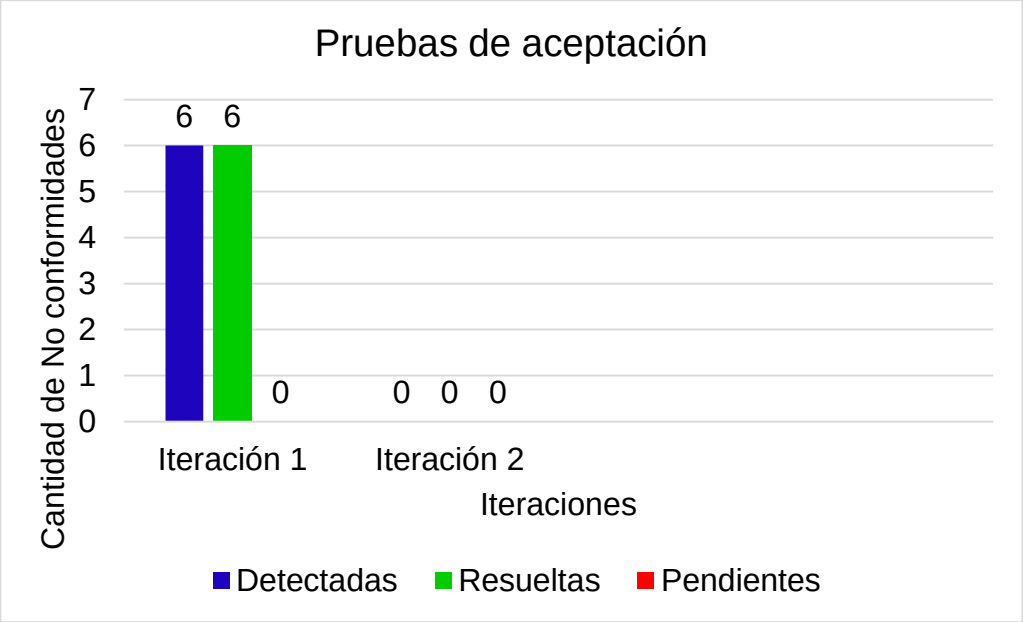


**Figura 16 Resultados de la prueba de aceptación por iteraciones.**

Fuente: (Elaboración propia)

### 3.4.3 Tipos de pruebas

Se aplicaron tres tipos de pruebas:

**Funcionalidad**: esta prueba fue aplicada en todos los niveles de prueba mediante el uso del método de caja negra.

**Portabilidad:** las interfaces del componente fueron probada en diferentes plataformas (Ubuntu 14.4, Windows 8.1 y 10), mostrando un desempeño aceptable.

**Usabilidad:** calidad del centro CEIGE aplicaron pruebas al componente, a partir de la lista de chequeo de usabilidad, estas pruebas que se realizaron mostraron su satisfacción con respecto al producto, reflejando resultados satisfactorios.

## 3.5 Validación de la investigación

### 3.5.1 Pre-experimento

Históricamente al experimento se le ha atribuido una importancia decisiva en la demostración del vínculo causal entre dos fenómenos, llegando a considerarse solamente como científicas las demostraciones que se realizaban por vía experimental.

El experimento es el método empírico para el estudio de un objeto en el cual el investigador crea las condiciones o adapta las existentes para el esclarecimiento de las propiedades, leyes y relaciones del objeto, para verificar una hipótesis, una teoría o un modelo (Hernández León, y otros, 2011).

El experimento como método de investigación tiene determinada estructura básica que da lugar a muchas alternativas pero que de forma general consta de las partes siguientes: constatación del estado inicial, introducción del factor de cambio, constatación del estado final y comparación del estado inicial con el final (Hernández León, y otros, 2011).

Los pre-experimentos presentan un grado de control mínimo. Estos tipos de diseño suelen ser útiles para una primera aproximación al problema (estudios exploratorios) de investigación (Hernández Sampieri, y otros, 2006).

Dentro de los pre-experimentos pueden ser aplicados los tipos de diseño:

**Estudio de caso con una sola medición:** se administra un tratamiento a un grupo y después se aplica una medición de una o más variables para observar cuál es el nivel del grupo en estas variables. No se puede establecer causalidad con certeza ni se controlan las fuentes de invalidez.

**Diseño de Pre-prueba y Post-prueba con un solo grupo:** a un grupo se le aplica una prueba previa al tratamiento, después se le administra el tratamiento y se pasa de nuevo una prueba (Hernández Sampieri, y otros, 2006).

Para la validación de la presente investigación se realizó un pre-experimento, mediante el diseño de pre-prueba y post-prueba con un solo grupo; para comprobar cómo se comporta el grado de usabilidad del componente de Estructura y Composición, analizando un antes (sin interfaces gráficas) y un después (con interfaces gráficas), además se mantuvo la heterogeneidad de los participantes con el propósito de asegurar la validez de los resultados.

Teniendo en cuenta para la pre-prueba y la post-prueba los resultados obtenidos mediante el instrumento de medición (encuesta, ver Anexo 9) realizado en el centro CEIGE, aplicadas a los profesionales del departamento de Desarrollo de Componentes que trabajan con el marco de trabajo Bosón para medir el grado de usabilidad que tiene el componente Estructura y Composición.

**Análisis de los resultados**

A continuación, se muestran los resultados del análisis, apoyado de gráficos de barras aplicado a las 6 preguntas realizadas en la encuesta para caracterizar el grado de usabilidad del componente a partir de los atributos de usabilidad: facilidad de aprendizaje, eficiencia, recuerdo en el tiempo, tasa de errores, satisfacción del usuario.

**Antes (sin interfaces gráficas)**

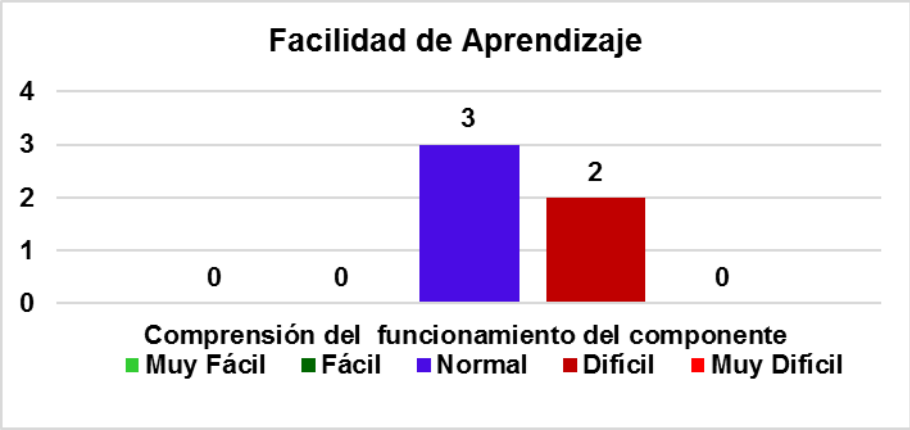


**Figura 17 Resultados del atributo Facilidad de aprendizaje.**

Fuente: (Elaboración propia)

La comprensión del funcionamiento del componente ayuda a medir el atributo de usabilidad facilidad de aprendizaje, arrojando como resultado que 3 encuestados respondieron normal (60%) y 2 que era difícil (40%).

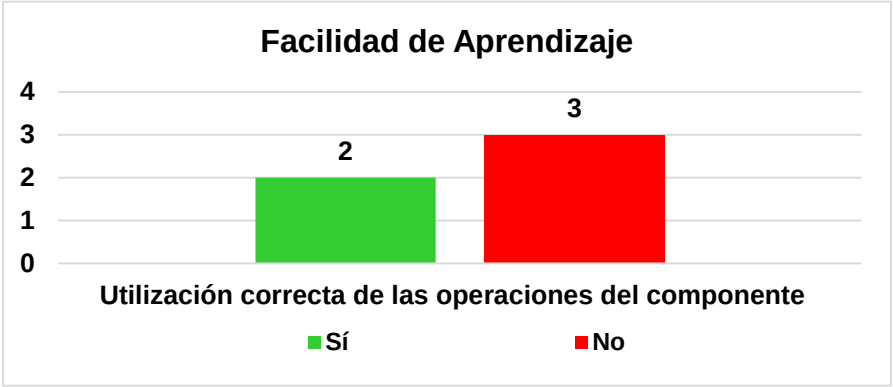


**Figura 18 Resultados del atributo Facilidad de aprendizaje.**

Fuente: (Elaboración propia)

La utilización correcta de las operaciones del componente ayuda medir el atributo facilidad de aprendizaje, arrojando como resultado que 2 encuestados respondieron sí (40%) y 3 que no (60%).

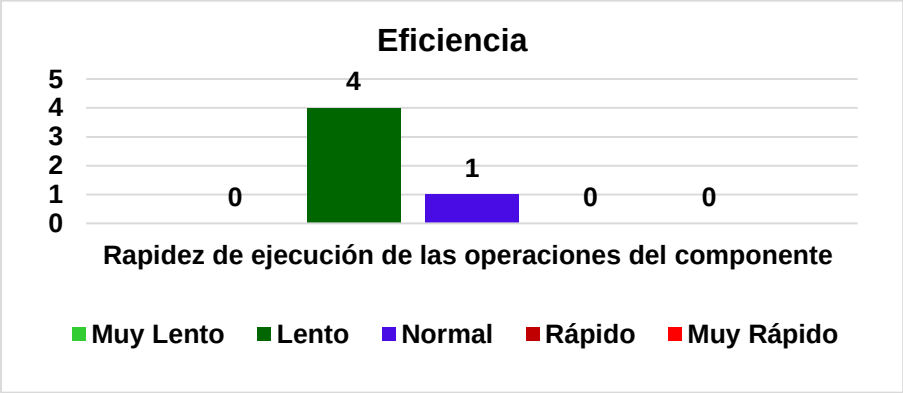


**Figura 19 Resultados del atributo Eficiencia.**

Fuente: (Elaboración propia)

La rapidez de ejecución de las operaciones del componente ayuda a medir el atributo de usabilidad eficiencia, arrojando como resultado que 4 encuestados respondieron lento (80%) y 1 que era normal (20%).

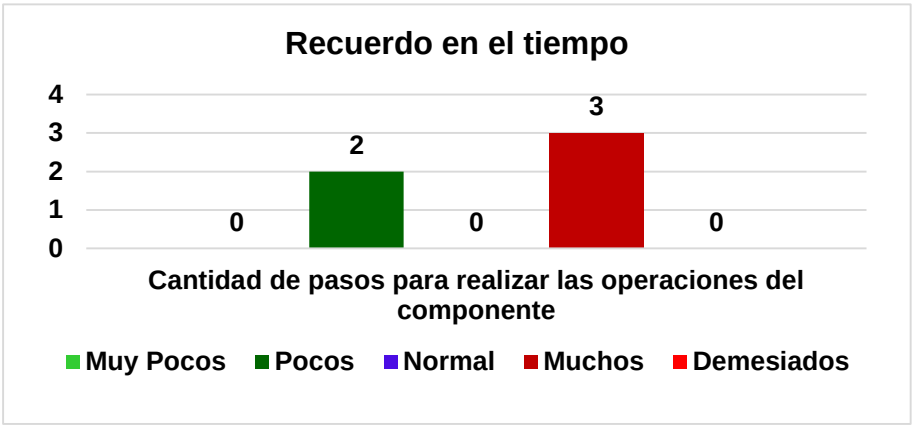


**Figura 20 Resultados del atributo Recuerdo en tiempo.**

Fuente: (Elaboración propia)

La cantidad de pasos para realizar las operaciones del componente ayuda a medir el atributo de usabilidad recuerdo en el tiempo, arrojando como resultado que 2 encuestados respondieron pocos (40%) y 3 que era muchos (60%).

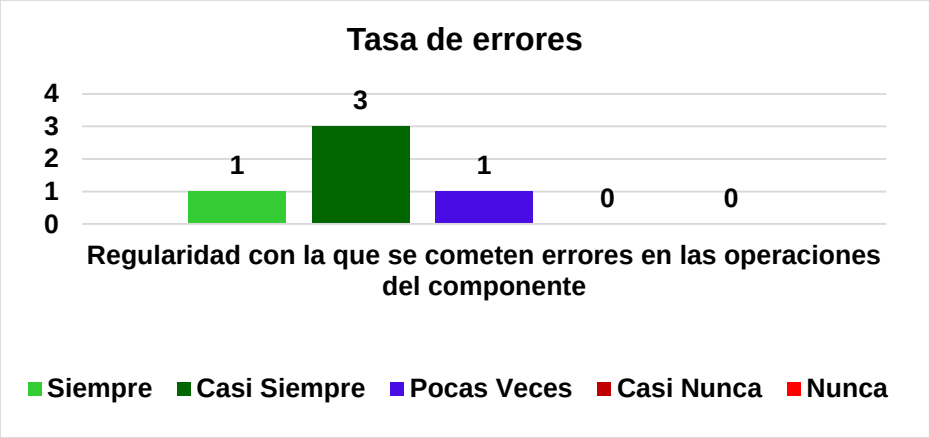


**Figura 21 Resultados del atributo Tasa de errores.**

Fuente: (Elaboración propia)

La regularidad con la que se cometen errores en las operaciones del componente ayuda a medir el atributo de usabilidad tasa de errores, arrojando como resultado que 3 encuestados respondieron siempre (60%), 1 que se cometen pocas veces (20%) y otro que casi siempre (20%).

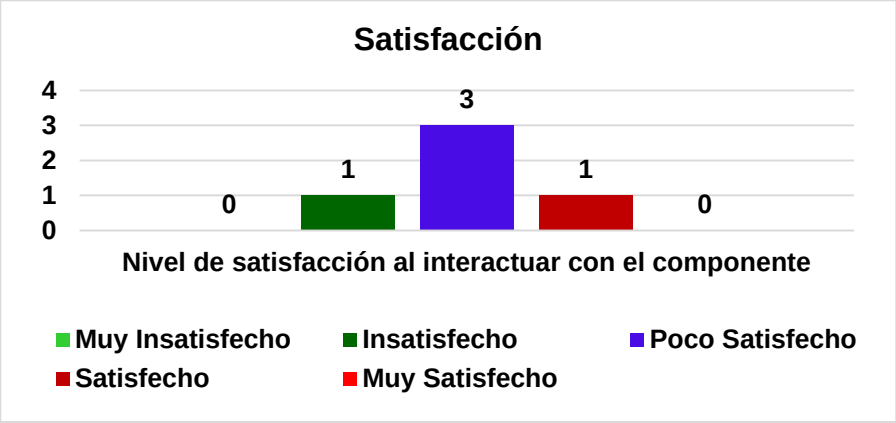


**Figura 22 Resultados del atributo Satisfacción.**

Fuente: (Elaboración propia)

El nivel de satisfacción al interactuar con el componente ayuda a medir el atributo de usabilidad satisfacción, arrojando como resultado que 3 encuestados respondieron que estaban poco satisfechos (60%), 1 insatisfecho (20%) y otro que satisfecho (20%).

**Después (con interfaces gráficas)**

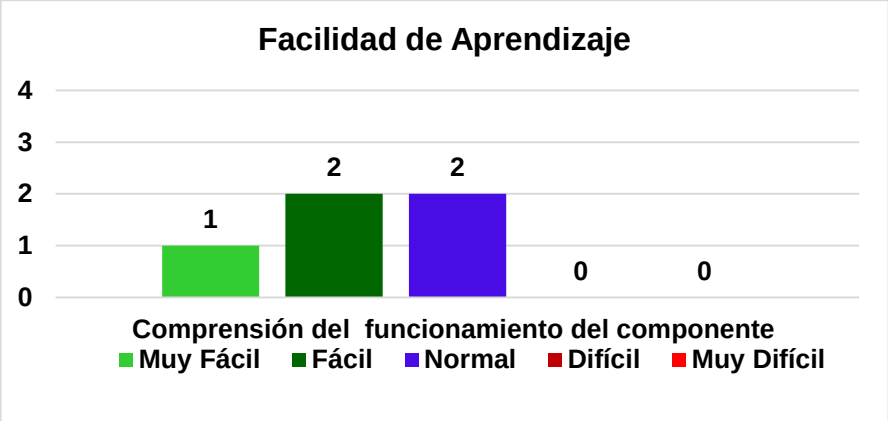


**Figura 23 Resultados del atributo Facilidad de Aprendizaje.**

Fuente: (Elaboración propia)

La comprensión del funcionamiento del componente ayuda a medir el atributo de usabilidad facilidad de aprendizaje, arrojando como resultado que 2 encuestados respondieron fácil (40%), 1 que muy fácil (20%) y 2 que era normal (40%).

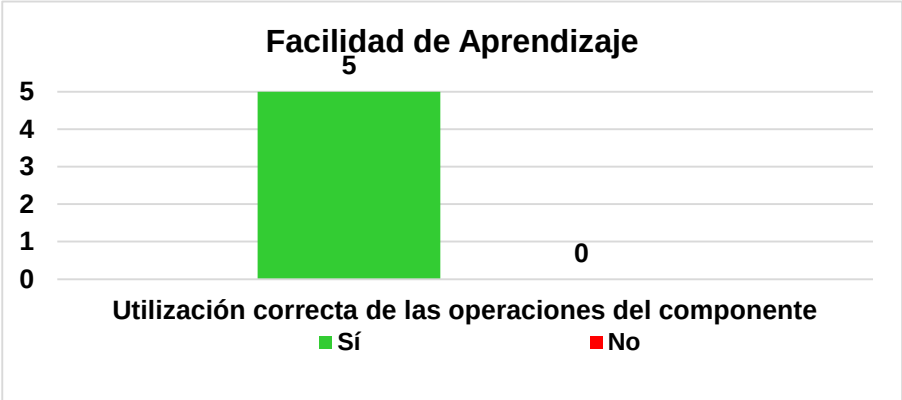


**Figura 24 Resultados del atributo Facilidad de Aprendizaje.**

Fuente: (Elaboración propia)

La utilización correcta de las operaciones del componente ayuda medir el atributo facilidad de aprendizaje, arrojando como resultado que 5 encuestados respondieron sí (100%).

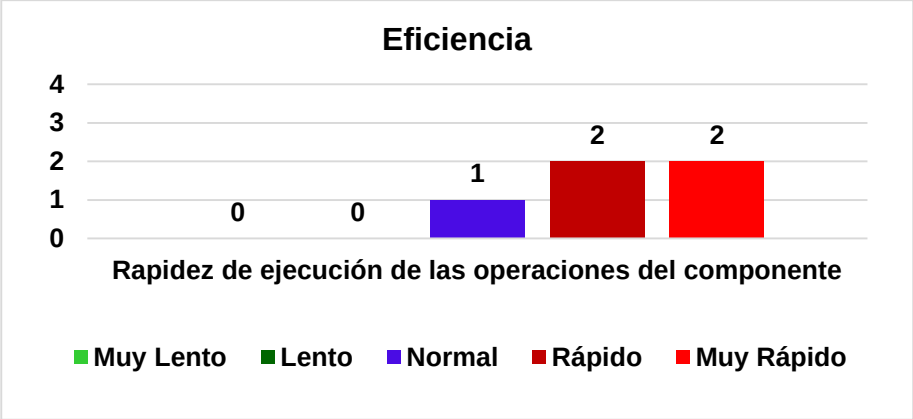


**Figura 25 Resultados del atributo Eficiencia.**

Fuente: (Elaboración propia)

La rapidez de ejecución de las operaciones del componente ayuda a medir el atributo de usabilidad eficiencia, arrojando como resultado que 2 encuestados respondieron que era rápido (40%), 2 que muy rápido (40%) y 1 que era normal (20%).

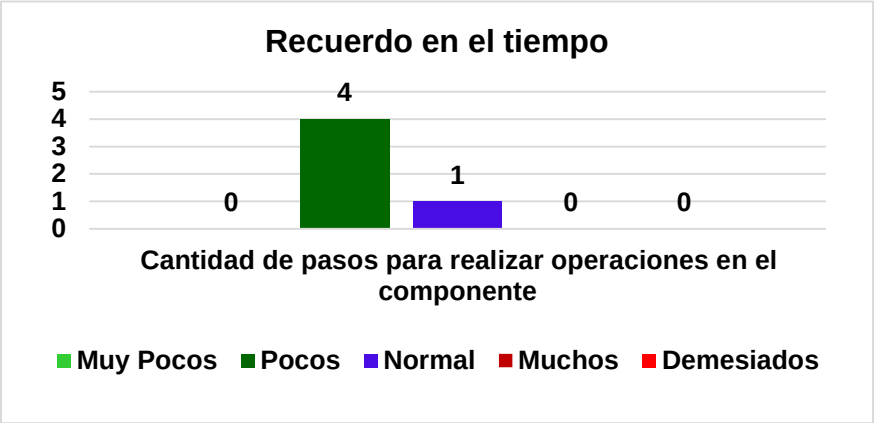


**Figura 26 Resultados del atributo Recuerdo en el tiempo.**

Fuente: (Elaboración propia)

La cantidad de pasos para realizar las operaciones del componente ayuda a medir el atributo de usabilidad recuerdo en el tiempo, arrojando como resultado que 4 encuestados respondieron pocos (80%) y 1 que era normal (20%).

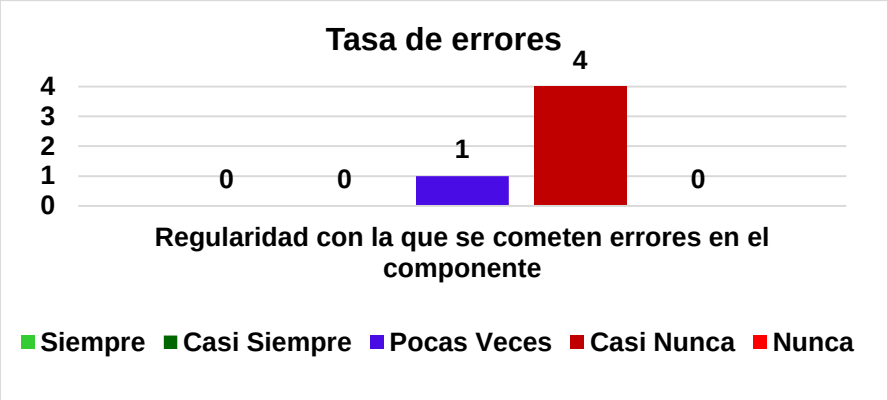


**Figura 27 Resultados del atributo Tasa de errores.**

Fuente: (Elaboración propia)

La regularidad con la que se cometen errores en las operaciones del componente ayuda a medir el atributo de usabilidad tasa de errores, arrojando como resultado que 4 encuestados respondieron casi nunca (80%) y 1 que se cometen pocas veces (20%).

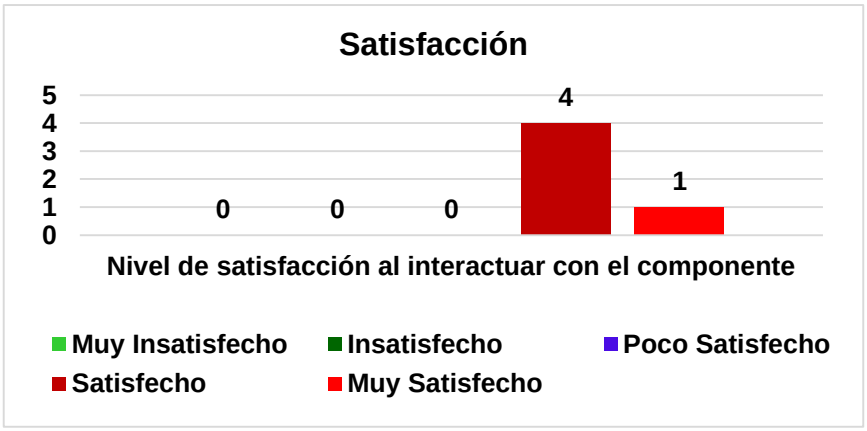


**Figura 28 Resultados del atributo Satisfacción.**

Fuente: (Elaboración propia)

El nivel de satisfacción al interactuar con el componente ayuda a medir el atributo de usabilidad satisfacción, arrojando como resultado que 4 encuestados respondieron que estaban muy satisfechos (80%) y 1 que muy satisfechos (20%).

Para poder realizar una comparación de cómo se comportaban los atributos de usabilidad antes y después de la creación de las interfaces gráficas se le dio una ponderación a cada respuesta

por pregunta y luego se sumaron las ponderaciones en dependencia de las respuestas elegidas por cada encuestado. Quedando las ponderaciones de la siguiente forma:

**Tabla 9 Resultados de las Ponderaciones.**

Fuente: (Elaboración propia)

| Ponderaciones | Pregunta 1 | Pregunta 2 | Pregunta 3 | Pregunta 4 | Pregunta 5 | Pregunta 6 |
|---|---|---|---|---|---|---|
| **1** | Muy difícil | No | Muy lento | Demasiados | Siempre | Muy insatisfecho |
| **2** | Difícil | - | Lento | Muchos | Casi siempre | Insatisfecho |
| **3** | Normal | - | Normal | Normal | Pocas veces | Poco satisfecho |
| **4** | Fácil | - | Rápido | Pocos | Casi nuca | Satisfecho |
| **5** | Muy fácil | Sí | Muy rápido | Muy pocos | Nunca | Muy satisfecho |

A continuación, se muestra en la Figura 29 el resultado de la comparación de los atributos de usabilidad medidos al sumar las ponderaciones, representándolo en por ciento con respecto al máximo valor posible por cada atributo. Lo que permite obtener un acercamiento al grado de usabilidad antes de tener interfaces gráficas y después de la propuesta de solución. Demostrando por cada atributo que luego de realizada la propuesta de solución se le dio cumplimiento al objetivo planteado en la investigación mejorando el grado de usabilidad del componente de Estructura y Composición.

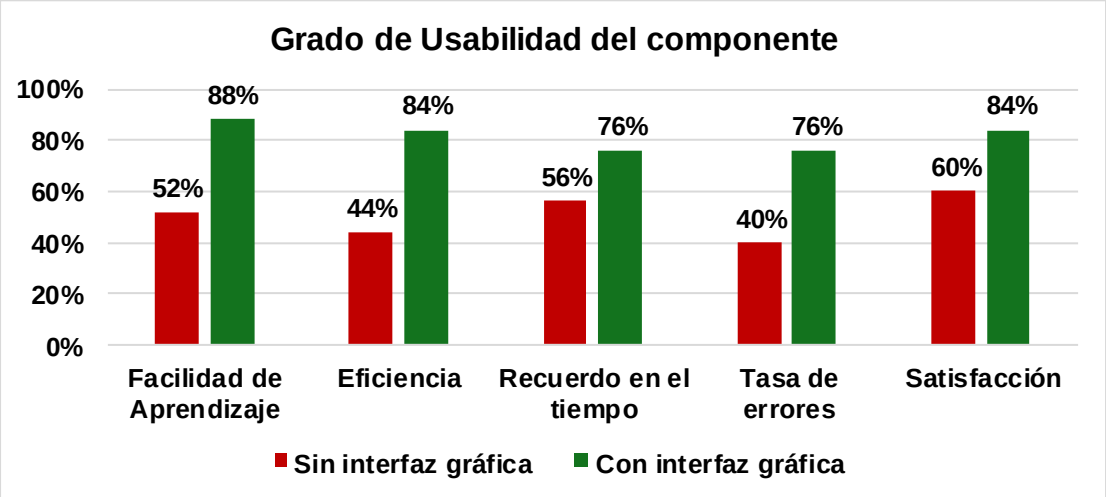


**Figura 29 Resultados del grado de usabilidad del componente.**

Fuente: (Elaboración propia)

### 3.5.2 Lista de chequeo usabilidad de sitios web

El uso de las listas de chequeos en el laboratorio de pruebas de software ha agilizado las pruebas, teniendo en cuenta que constituyen una guía básica para el probador en la revisión de los artefactos. Constituyen un apoyo en la ejecución de evaluaciones estáticas garantizando una mayor calidad en los artefactos de apoyo a los sistemas desarrollados por la UCI. Las listas de chequeo establecen un orden de revisión por subtítulos y acápites que coinciden con la organización del artefacto a evaluar.

**Forma de uso**

Evaluación**:** es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal (cuando la respuesta al indicador sea “No”) y 0 en caso que elemento revisado no presente errores (cuando la respuesta al indicador sea “Sí”).

La lista de chequeo (Ver Anexo 10) cuenta con varios puntos, los cuales se evalúan según los atributos de usabilidad. La misma fue aplicada en dos iteraciones de trabajo para lograr un resultado final de cero no conformidades.

Una vez aplicada la lista de chequeo al componente de Estructura y Composición, arrojó un resultado satisfactorio demostrando la usabilidad del componente.

## 3.6 Conclusiones parciales

- Los estándares de codificación empleados en la implementación permitieron ganar en legibilidad, claridad y mayor entendimiento del código por parte de los desarrolladores.
- El tratamiento de errores posibilitó evitar que datos incorrectos fuesen introducidos en los diferentes campos de los formularios mediante el uso de determinados métodos de validación.
- Para verificar el correcto funcionamiento de la solución obtenida se aplicaron pruebas de caja negra y pruebas de aceptación, durante las cuales se detectaron no conformidades en una primera etapa, las cuales fueron solucionadas, y posteriormente no se encontraron no conformidades.
- Se efectuó la validación de la investigación realizando un pre-experimento y la aplicación de la lista chequeo de usabilidad de calidad del centro CEIGE que permitió medir el grado usabilidad del componente, donde se evidenció que, por cada atributo, se le dio cumplimiento al objetivo planteado en la investigación mejorando la usabilidad del mismo.

# CONCLUSIONES GENERALES

Con el propósito de darle cumplimiento al objetivo general y a la problemática planteada en la presente investigación, se han llevado a cabo satisfactoriamente los objetivos específicos trazados.

- El estudio y análisis de las principales herramientas informáticas asociadas a la gestión de Estructura y Composición, evidenció que las mismas no brindan una solución completa al problema existente en el departamento de Desarrollo de Componentes de CEIGE, lo cual constituyó motivación para el desarrollo de la investigación, que utiliza como framework a Symfony 2, empleando la metodología de desarrollo aprobada en el plan de mejoras de la UCI, garantizando un ambiente visual ameno y organizado con el empleo de Angular JS.
- La verificación del diseño realizado, mediante la aplicación de las métricas TOC y RC, evidenció una baja responsabilidad, cantidad de pruebas y complejidad de implementación, la cual favoreció la reutilización corroborándose así una correcta asignación de responsabilidades.
- Con la implementación se obtuvieron las interfaces gráficas del componente de Estructura y Composición que responden a las necesidades, especificidades y flexibilidad que debe brindar el componente, facilitándole al usuario un mejor entendimiento y satisfacción con el mismo.
- Las pruebas de caja negra y de aceptación permitieron validar el correcto funcionamiento de las interfaces desarrolladas y el cumplimiento de los objetivos de la investigación.
- La realización del pre-experimento y la aplicación de la lista de chequeo de usabilidad del centro CEIGE, corroboró la mejora del nivel de usabilidad del componente, reflejando una gran satisfacción por parte del cliente.

## BIBLIOGRAFÍA

**Mojena Alpizar José y Vázquez Moreno Renán** Sistema para la gestión de nomencladores [Publicación periódica] // Revista Cubana de Ciencias Informáticas (RCCI). - La Habana : Revista Cubana de Ciencias Informáticas (RCCI), 2012. - 2 : Vol. 6.

**Alegsa Leandro** alegsa.com.ar [En línea]. - 2010. - 15 de 04 de 2016. - http://www.alegsa.com.ar/Dic/framework.php.

**Alonso Daniel Bolaños** Pruebas de Software y JUnit. Un análisis en profundidad y ejemplos [Informe]. - [s.l.] : 2, 2010.

**Babylon** Diccionario Babylon [En línea] // Diccionario Babylon. - 2014. - 03 de 05 de 2016. - http://diccionario.babylon-software.com/.

**Bowen Rich** Servidor apache al descubierto [Libro]. - Madrid : Prentice Hall, 2000.

**Calas Torres Abraham** Bosón: naciente Arquitectura de Referencia para la UCI. [Publicación periódica]. - 2015.

**Cálas Torres Abraham** Vista Entorno de Desarrollo Tecnológico [Informe] / Desarrro de componentes. - 2016.

**Chidamber Shyam R. y Kemerer, Chris F** A Metrics Suite for Object Oriented Design. Software Engineering, IEEE Transactions [Informe]. - [s.l.] : 0098-5589 , 1994.

**Clements, P.** “A Survey of Architecture Description Languages”, in Proceedings of the International Workshop on Software Specification and Design. 1996: Alemania. [Publicación periódica]. - 1996.

**Cobo Ángel [y otros]** PHP y MySQL Tecnologías para el desarrollo de aplicaciones web [Libro]. - España : Ediciones Díaz de Santos, 2005.

**Cowburn Peter** PHP [En línea]. - 2001. - 3 de 2016. - https://secure.php.net/manual/es/index.php.

**Dicionario de significados** Diccionario de significados [En línea]. - 2015. - 5 de 2016. - http://que-significa.com/significado.php?termino=nomenclador.

**Eguiluz Javier** Desarrollo web ágil con Symfony2 [Libro]. - [s.l.] : easybook, 2013.

**Eguiluz Javier** Introducción a CSS [Libro]. - 2010.

**Estevez Muñoz Yudaimy y Gómez Fal Ariel** Solución para la gestión de la estructura investigativa de la Universidad de las Ciencias Informáticas [Informe] / Facultad 1 ; Universidad de las Ciencias Informáticas. - 2013.

**Fergarciac** Fergarciac [En línea]. - 25 de enero de 2013. - 20 de 05 de 2016. - https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

**Ferré Grau Xavier** Principios Básicos de Usabilidad para Ingenieros Software [Informe] / Facultadde Informática ; Universidad Politécnicade Madrid. - 2012.

**Foundation The Apache Software** [En línea] // “How it works”. - 2007. - http://www.apache.org/foundation/how-it-works.html.

**Gómez José Jorge Márquez** Arquitectura del Software [En línea]. - 2011. - http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rivera_l_a/capitulo2.pdf..

**Gracia Luis Miguel** Un poco de Java. Técnicas para la captura de Requisitos. [En línea] // Un poco de Java. Técnicas para la captura de Requisitos.. - 9 de 5 de 2013. - http://unpocodejava.wordpress.com/2013/05/09/tecnicas-para-la-captura-de-requisitos.

**Guzmán Ojeda José Roberto y Betancourt Santana Reisel** Biblioteca JavaScript para el desarrollo de interfaces gráficas de usuario de RIA [Informe] / Facultad 6 ; Universidad de las Ciencias Informáticas. - Habana : [s.n.], 2013.

**Headquarters Company** Visual Paradigm [En línea] // 10 Reasons to Choose Visual Paradigm. - 2006. - http://www.visual-paradigm.com/aboutus/10reasons.jsp..

**Hernández León Rolando Alfredo y Coello Gonz Sayda** El proceso de investigación científica [Libro]. - La Habana : Editorial Universitaria, 2011.

**Hernández Sampieri Roberto, Fernández Collado Carlos y Baptista Lucio Pilar** Metodología de la investigación [Libro]. - México : McGraw-Hill, 2006. - Vol. Cuarta Edición.

**JavaScript** Aprender a programar. [En línea]. - 2015. - 5 de 4 de 2016. - http://aprenderaprogramar.com..

**JavaSMapper** Object Relational [En línea]. - 2012. - 3 de 2016. - http://www.doctrine-project.org/projects/orm.html..

**Larman Craig** UML y Patrones [Informe]. - [s.l.] : Pearson, 2003.

**Lorena Suárez María** Programación en diversos lenguajes [Publicación periódica]. - [s.l.] : 4, 2010.

**Lorenz Mark y Kidd, Jeff** Object-Oriented Software Metrics [Informe]. - 1994.

**Marrero Expósito Carlos** Interfaz Gráfica de Usuario.Aproximación semiótica y cognitiva [En línea]. - 30 de 8 de 2010. - http://www.chr5.com/investigacion/investiga_igu/igu_aproximacion_semio - cognitiva_by_chr5.pdf.

**Microsoft** Microsoft [En línea] // Developer Network. - 2007. - 1 de 06 de 2016. - https://msdn.microsoft.com/es-es/library.

**Nieblas Palau Leonardo Antonio y Cubela Medina Yasmany** Interfaz Visual para la configuración de Entornos Virtuales desarrollados con la Herramienta Scene Tool Kit. [Informe] / Facultad 5 ; Universidad de las Ciencias Informáticas. - La Habana : [s.n.], 2008.

**Packt Publishing** Instant PhpStorm Starter [Libro]. - Birmingham, UK : Livery Place, 2013.

**Pantoja Juan Carlos** Introducción al proceso de desarrollo en el Diseño de Interfaces de Usuario [Publicación periódica]. - 30 de septiembre de 2009.

**Pavón Mestras Juan** Servidores Web [Informe]. - Madrid : [s.n.], 2010.

**Pérez Alfonso Damián** Normas y estándares de codificación de Sauxe. [Informe] / Universidad de las Ciencias Informáticas. - La Habana. Cuba : [s.n.], 2012.

**Pressman Roger S** Ingeniería del Software Un enfoque práctico [Publicación periódica]. - 2006.

**Pressman Roger** Software Engineering: A Practitioner's Approach [Libro]. - [s.l.] : 1, 2009. - Vol. 2.

**Real academia española** Real academia española [En línea]. - 10 de 2014. - 5 de 2016. - http://dle.rae.es/?id=QZxyZyj.

**Robert Lobo Armando** Lycan-Génesis,Component Builder. Manual del Desarrollador. [Informe] / Facultad 6 ; Universidad de las Ciencias Informáticas. - La Habana. Cuba : Centro DATEC, 2012.

**Rodrígez Tello Dr. Eduardo** Cinvestap Tampulinas [En línea]. - 2006. - 2 de 5 de 2016. - http://www.tamps.cinvestav.mx/~ertello/swe/sesion15.pdf..

**Rodríguez Landín Liset** Integración del Archivo Universitario con el módulo Estructura y Composición del Sistema de Gestión Universitaria [Informe] / Facultad 1 ; Universidad de las Ciencias Informáticas. - La Habana : [s.n.], 2012.

**Rodríguez Sánchez Ing. Tamara , Fernández González Ing. Mairelys y Cabrera Casas Ing. Eliecer** La gestión empresarial de las entidades cubanas. Cedrux a la vuelta de la esquina. [Publicación periódica]. - La Habana : Revista Infociencia, 2011. - 2 : Vol. 15.

**Sánchez Rodríguez Tamara** Metodología de desarrollo para la Actividad productiva de la UCI. [Informe]. - [s.l.] : UCI, 2015.

**Santiesteban Pérez Yusimi** Interfaz gráfica de usuarios para la biblioteca SceneToolKit [Informe] / Facultad 5 ; Universidad de las Ciencias Informáticas. - La Habana : [s.n.], 2011.

**Solis Carlos** AngularJs. De 0 a 100. - video2brain, 2013.

**Sommerville Ian** Ingeniería del Software [Libro]. - Madrid, España : [s.n.], 2005. - 84-7829-074-5.

**Sommerville** Software Engineering [Libro]. - Boston : 8va, 2011. - Vol. 2.

**Sperberg Camilo** unreal4u´s Personal Network [En línea] // Sobre convenciones y notaciones (húngara, CamelCase, etc). - 2012. - http://blog.unreal4u.com/2011/03/sobre -convenciones-ynotaciones-hungara-camelcase-etc/.

**Svensk Magnus** DiVA [En línea]. - 2012. - 10 de 05 de 2016. - http://urn.kb.se/resolve?urn=urn:nbn:se:hig:diva-12010..

**Symfony** Symfony [En línea]. - 2007. - 17 de 04 de 2016. - http://symfony.es/pagina/que-es-symfony/.

**Verdecia Álvarez Yamila L. y Garcell Fundora Leyanis** Propuesta de Interfaz Gráfica de Usuario para el proyecto VISMEDIC [Informe] / Facultad 5 ; Universidad de las Ciencias Informáticas. - Ciudad de La Habana : [s.n.], 2010.

**Wonderbits** wonderbits [En línea] // wonderbits. - 2012. - 25 de 04 de 2016. - https://www.wonderbits.net/es/posts/5-mvc-para-apps-web-con-angularjs.

# ANEXOS

A continuación, se muestra tres Historias de Usuarios, el resto se encuentran en expediente de proyecto:

**Anexo 1** Historia de usuario del requisito “Listar Estructura”

| | |
|---|---|
| **Número:** 56 | **Nombre del requisito:** Listar estructuras |
| **Programador:** Liliana Simón Figueredo | **Iteración Asignada:** 1 |
| **Prioridad:** Alta | **Tiempo Estimado:** 16 horas |
| **Riesgo en Desarrollo:** Poca experiencia de los estudiantes en las tecnologías de desarrollo del proyecto. | **Tiempo Real:** N/A |
| **Descripción:**<br>Este requisito se encarga de listar las estructuras que existan en el sistema. Esto permitirá al usuario poder consultar el número de las estructuras que existen, así como poder ejecutar las acciones de modificación, eliminación y búsqueda de los mismos. | |
| **Observaciones:** N/A | |
| **Prototipo de interfaz:** | |

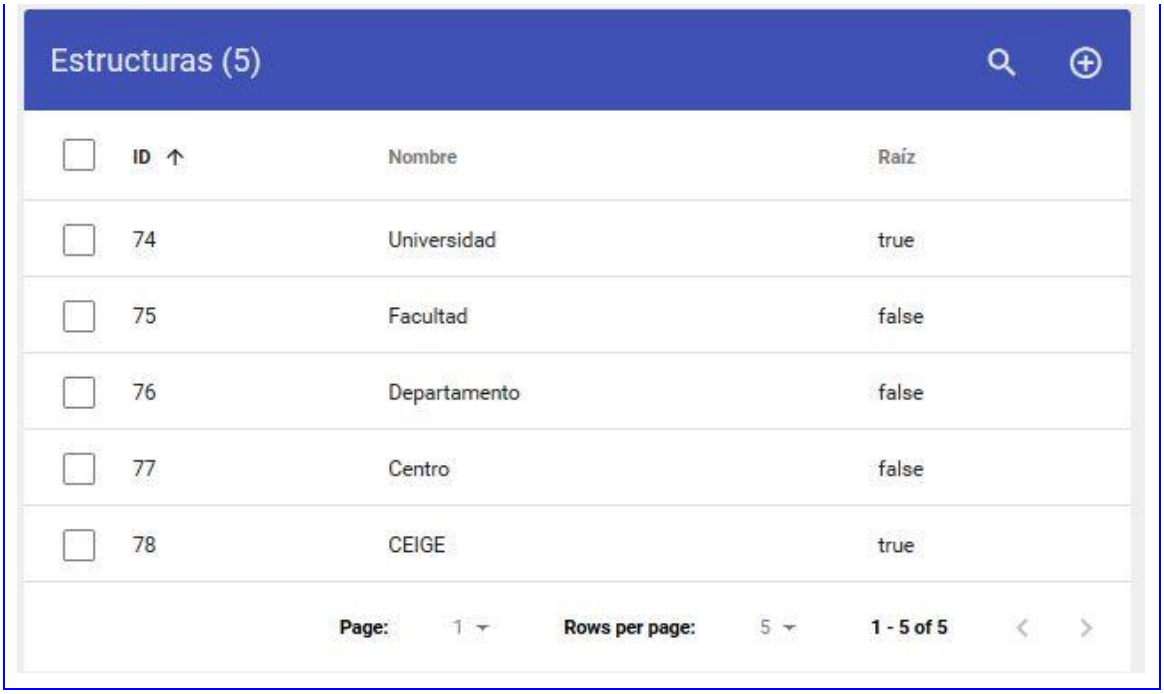


Tabla A.1 HU del requisito “Listar Estructura”.

**Anexo 2** Historia de usuario del requisito “Adicionar Estructura”

| | |
|---|---|
| **Número:** 52 | **Nombre del requisito:** Adicionar estructura |
| **Programador:** Liliana Simón Figueredo | **Iteración Asignada:** 1 |
| **Prioridad:** Alta | **Tiempo Estimado:** 12 horas |
| **Riesgo en Desarrollo:** Poca experiencia de los estudiantes en las tecnologías de desarrollo del proyecto. | **Tiempo Real:** N/A |
| **Descripción:**<br>Este requisito se encarga de adicionar una nueva estructura. De una estructura se debe conocer el nombre y si es raíz. Luego de insertado los campos el sistema efectuará las validaciones pertinentes, en caso que existan errores muestra un mensaje indicando los campos incorrectos. Si toda la información brindada es correcta se muestra un mensaje indicando el éxito de la operación. | |
| **Observaciones:** N/A | |
| **Prototipo de interfaz:** | |

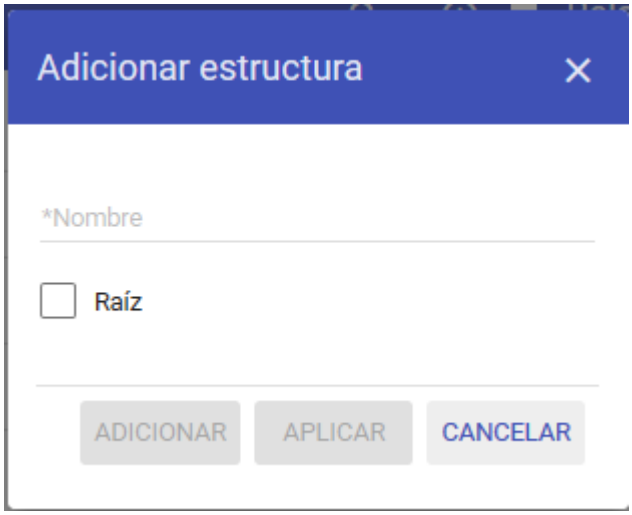


Tabla A.2 HU del requisito “Adicionar Estructura”.

**Anexo 3** Historia de usuario del requisito “Modificar campo de nomenclador”

| | |
|---|---|
| **Número:** 60 | **Nombre del requisito:** Modificar campo a nomenclador |
| **Programador:** Liliana Simón Figueredo | **Iteración Asignada:** 1 |
| **Prioridad:** Alta | **Tiempo Estimado:** 16 horas |
| **Riesgo en Desarrollo:** Poca experiencia de los estudiantes en las tecnologías de desarrollo del proyecto. | **Tiempo Real:** N/A |
| **Descripción:**<br>Este requisito se encarga de modificar un campo a un nomenclador de los existentes en el sistema. Cuando se modifica un campo a un nomenclador en caso que los valores sean incorrectos el sistema mostrará un mensaje indicando que existen valores erróneos, en caso contrario mostrará un mensaje indicando el éxito de la operación. | |
| **Observaciones:** N/A | |
| **Prototipo de interfaz:** | |


Tabla A.3 HU del requisito “Modificar campo de nomenclador”.

**Anexo 4** Diagrama de clases del diseño “Nivel estructural”

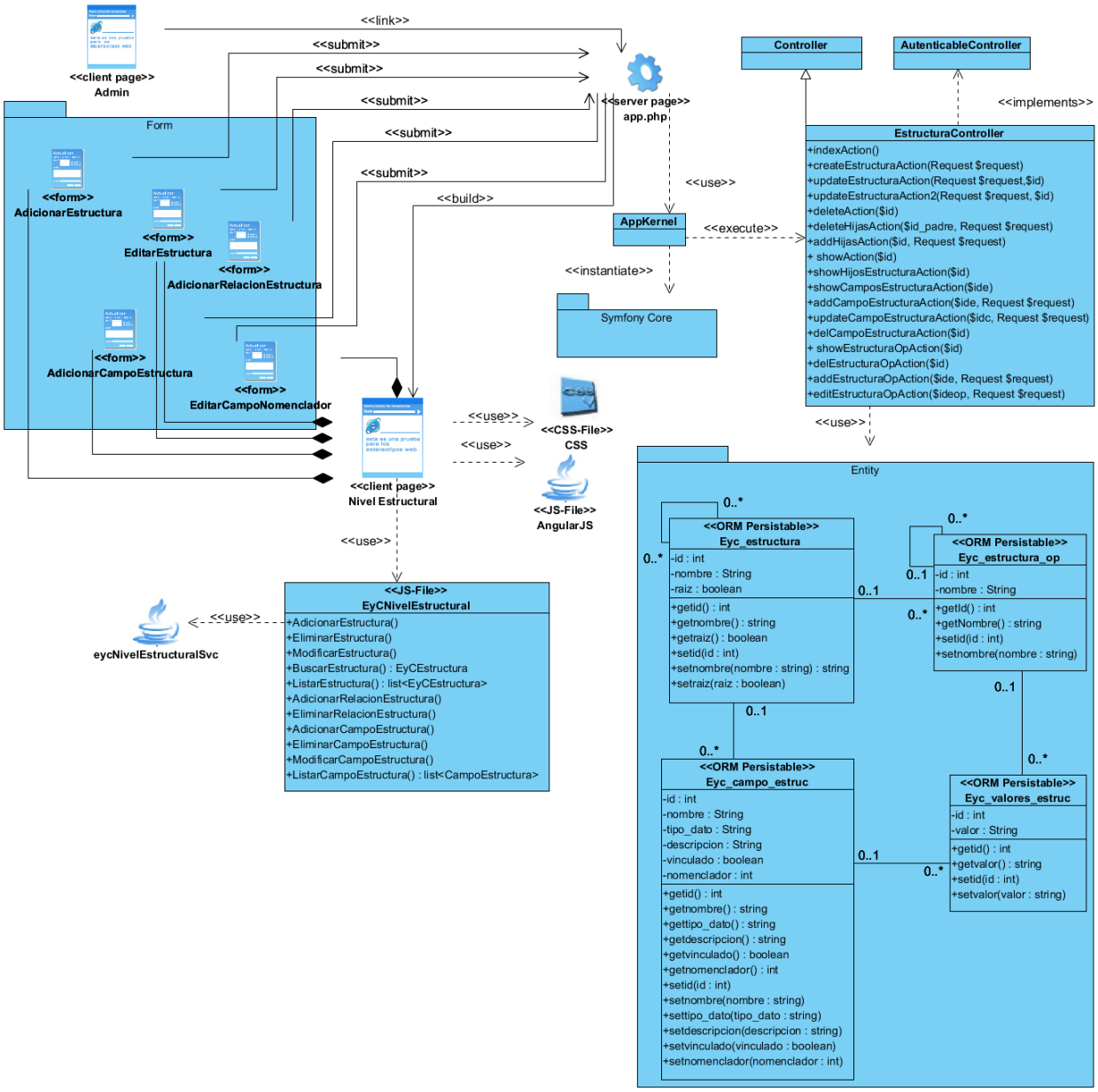


Figura A.4 Diagrama de clases del diseño “Nivel Estructural”.

**Anexo 5** Diagrama de clases del diseño “Gestionar nomenclador”

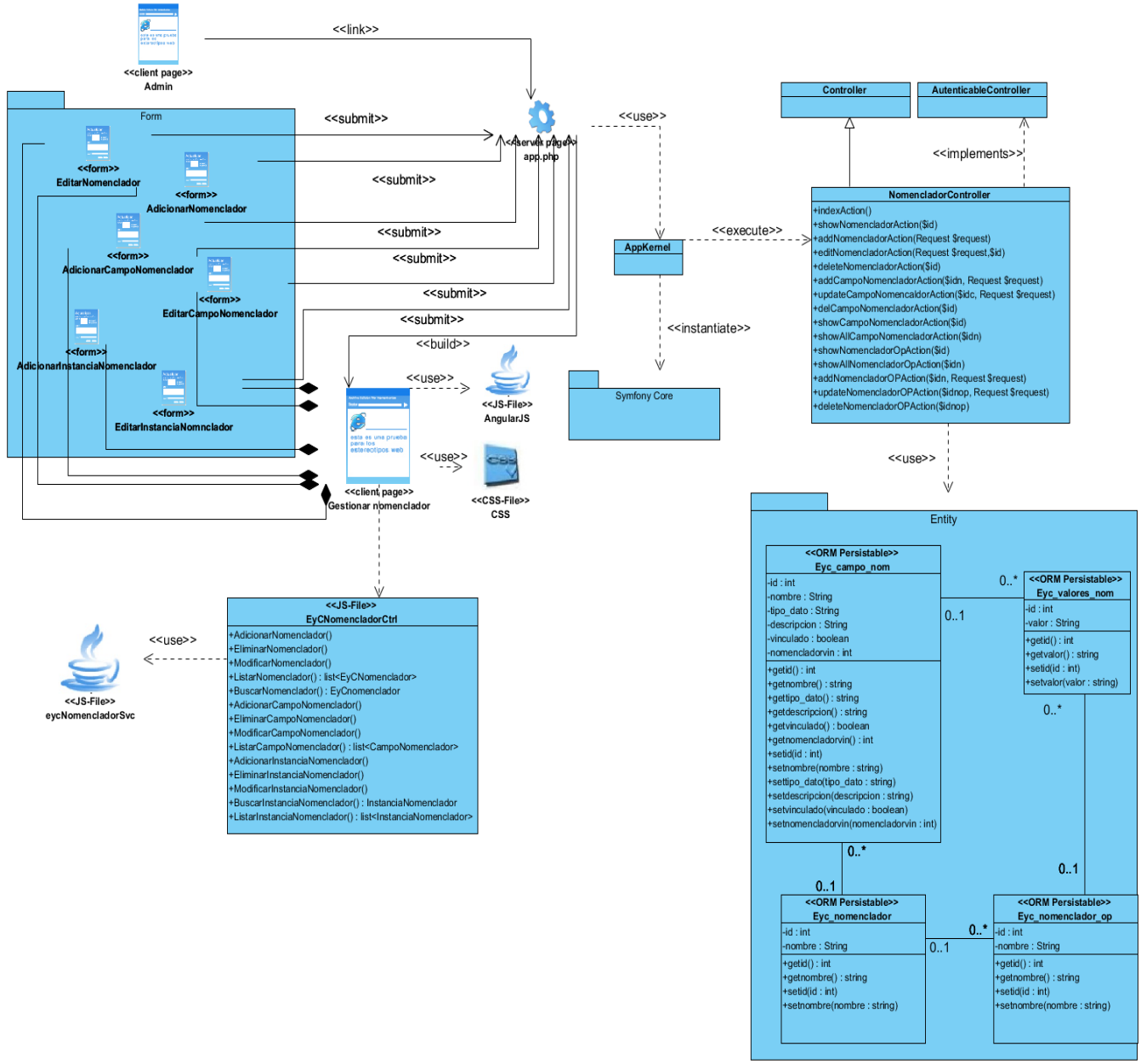


Figura A.5 Diagrama de clases del diseño “Gestionar Nomenclador”.

A continuación, se muestran tres diseños de casos de pruebas el resto se encuentra en expediente de proyecto:

**Anexo 6** Diseño de caso de prueba “Listar Estructura”

| Escenario | Descripción | Nombre | Raíz | Respuesta del sistema | Flujo central |
|---|---|---|---|---|---|
| EC 1.1 Listar estructuras. | Se muestra un listado de las estructuras que existen en el sistema. | V<br>N/A | V<br>N/A | El sistema muestra el listado de las estructuras insertadas. | -Al abrir la interfaz se muestra el listado de las estructuras que existen en el sistema en la parte izquierda de la interfaz. |

Tabla A.6 Diseño de caso de prueba “Listar Estructura”.

**Anexo 7** Diseño de caso de prueba “Eliminar relación entre estructuras”

| Escenario | Descripción | Padre | Hija | Respuesta del sistema | Flujo central |
|---|---|---|---|---|---|
| EC 1.1 Cancelar la eliminación de relación entre estructuras. | Se cancela la eliminación de las relaciones entre las estructuras. | V<br>N/A | V<br>N/A | Se cierra la interfaz sin registrar la eliminación de la relación entre las estructuras. | -Se selecciona la estructura padre de las listadas en la parte izquierda de la interfaz.<br>-Se habilita la parte derecha de la interfaz denominada **Relaciones.**<br>-Se muestra el listado de las estructuras hijas relacionadas con la padre seleccionada.<br>-Se selecciona la estructura hija que se desea eliminar de la relación.<br>-Se presiona el botón **Eliminar** en la parte superior derecha de la interfaz. |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | - Se muestra un mensaje de confirmación **"¿Está seguro que desea eliminar el elemento seleccionado?"**<br>- Presiona el botón **Cancelar** para abortar la operación. |
| EC 1.2 Eliminar relación entre estructuras. | Se eliminar las relaciones entre las estructuras. | V<br>Universidad | V<br>Facultad | El sistema elimina una relación entre las estructuras y muestra el mensaje: "La relación fue eliminada satisfactoria mente". | -Se selecciona la estructura padre de las listadas en la parte izquierda de la interfaz.<br>-Se habilita la parte derecha de la interfaz denominada **Relaciones.**<br>-Se muestra el listado de las estructuras hijas relacionadas con la padre seleccionada.<br>-Se selecciona la estructura hija que se desea eliminar de la relación.<br>-Se presiona el botón **Eliminar** en la parte superior derecha de la interfaz.<br>- Se muestra un mensaje de confirmación **"¿Está seguro que desea eliminar el elemento seleccionado?"** |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | - Se presiona el botón **Aceptar**.<br>- Se registra la acción y se muestra un mensaje indicando el éxito de la operación.<br>- Se presiona el botón **Aceptar** del mensaje de notificación o se espera hasta que el mismo desaparece. |

Tabla A.7 Diseño de caso de prueba “Eliminar relación entre estructuras Estructura”.

**Anexo 8** Diseño de caso de prueba “Adicionar Nomenclador”

| **Escenario** | **Descripción** | **Nombre** | **Respuesta del sistema** | **Flujo central** |
|---|---|---|---|---|
| EC 1.1 Adicionar nomenclador insertando valores correctos | Se adiciona un nomenclador al sistema insertando | V<br>Provincia<br>V | El sistema adiciona un nuevo nomenclador y muestra el mensaje: “El nomenclador ha sido | - Se selecciona el botón **Adicionar** en la parte izquierda de la interfaz. |

| | | | | |
|---|---|---|---|---|
| presionando el botón **Aceptar**. | valores correctos. | Municipio | adicionado satisfactoriamente". | - Se abre la interfaz **Adicionar nomenclador**.<br>- El usuario inserta el valor deseado en el campo mostrado.<br>- Presiona el botón **Aceptar**.<br>- El sistema valida los datos.<br>- Se registran los datos insertados y se muestra un mensaje indicando el éxito de la operación.<br>- Se presiona el botón **Aceptar** del mensaje de notificación o se espera hasta que el mismo desaparece. |
| EC 1.2 Adicionar nomenclador insertando valores correctos presionando el botón **Aplicar**. | Se adiciona un nomenclador al sistema insertando valores correctos. | V<br>Raza<br>V<br>Color | El sistema adiciona un nuevo nomenclador y muestra el mensaje: "El nomenclador ha sido adicionado satisfactoriamente". | - Se selecciona el botón **Adicionar** en la parte izquierda de la interfaz.<br>- Se abre la interfaz **Adicionar nomenclador**.<br>- El usuario inserta el valor deseado en el campo mostrado.<br>- Presiona el botón **Aplicar**.<br>- El sistema valida los datos. |

| | | | | |
|---|---|---|---|---|
| | | | | - Se registran los datos insertados y se muestra un mensaje indicando el éxito de la operación.<br>- Se presiona el botón **Aceptar** del mensaje de notificación o se espera hasta que el mismo desaparece.<br>- Se mantiene la ventana abierta para continuar adicionando nomencladores. |
| EC 1.3 Adicionar nomenclador dejando campos vacíos presionando el botón **Aceptar/Aplicar.** | Se dejan campos vacíos al adicionar un nomenclador. | I<br>Vacío | El sistema muestra el mensaje de error: "Existen campos vacíos". | - Se selecciona el botón **Adicionar** en la parte izquierda de la interfaz.<br>- Se abre la interfaz Adicionar nomenclador.<br>- Presiona el botón **Aceptar** o **Aplicar**.<br>- El sistema valida los datos.<br>- El sistema muestra un mensaje indicando que existen campos vacíos que deben ser llenados. |
| EC 1.4 Adicionar | Se insertan | I | El sistema muestra el | - Se selecciona el |

| | | | | |
|---|---|---|---|---|
| nomenclador introduciendo valores incorrectos presionando el botón **Aceptar/Aplicar.** | campos con valores incorrectos para adicionar un nomenclador. | Provincia*+ | mensaje de error: “Existen campos con valores incorrectos”. | botón **Adicionar** en la parte izquierda de la interfaz.<br>- Se abre la interfaz Adicionar nomenclador.<br>- El usuario inserta el valor deseado en el campo mostrado.<br>- Presiona el botón **Aceptar** o **Aplicar**.<br>- El sistema valida los datos.<br>- El sistema muestra un mensaje indicando que existen campos con valores incorrectos. |
| EC 1.5 Presionar botón **Cancelar.** | Se cancela la operación de adicionar un nuevo nomenclador. | V<br>N/A | El sistema cancela la acción. | - Se selecciona el botón **Adicionar** en la parte izquierda de la interfaz.<br>- Se abre la interfaz **Adicionar nomenclador**.<br>- El usuario inserta el valor deseado en el campo mostrado.<br>- Presiona el botón **Cancelar** para abortar la operación. |
| EC 1.6 Adicionar | Se inserta un | I | El sistema muestra el | - Se selecciona el |

| | | | | |
|---|---|---|---|---|
| nomenclador registrando un nombre que ya existe presionando el botón **Aceptar/Aplicar.** | nomenclador con un nombre que ya existe en el sistema. | Color | mensaje de error: "Ya existe un nomenclador con el mismo nombre". | botón **Adicionar** en la parte izquierda de la interfaz.<br>- Se abre la interfaz Adicionar nomenclador.<br>- El usuario inserta el valor deseado en el campo mostrado.<br>- Presiona el botón **Aceptar** o **Aplicar**.<br>- El sistema valida los datos.<br>- El sistema muestra un mensaje indicando que existe el mismo nombre en el sistema. |

Tabla A.8 Diseño de caso de prueba "Adicionar Nomenclador".

**Anexo 9** Encuesta sobre el uso del componente de Estructura y Composición del marco trabajo Bosón.

## Encuesta sobre el uso del componente de Estructura y Composición del marco trabajo Bosón.

Encuesta aplicada a los profesionales del centro CEIGE del Departamento de Desarrollo de componente que trabajan con el marco de trabajo Bosón, para determinar el grado de usabilidad que tiene el componente.

**Homologías**

**Facilidad de aprendizaje:** Cuán fácil es aprender la funcionalidad básica del sistema.

**Eficiencia:** El número de transacciones por unidad de tiempo que el usuario puede realizar usando el sistema.

**Recuerdo en el tiempo:** Para usuarios intermitentes (que no utilizan el sistema regularmente) es vital ser capaces de usar el sistema sin tener que aprender cómo funciona partiendo de cero.

**Tasa de errores:** El número de errores cometidos por el usuario mientras realiza una determinada tarea.

**Satisfacción:** Muestra la impresión subjetiva que el usuario obtiene del sistema.

## Sin interfaces Gráficas

1. **¿Qué tan fácil es comprender el funcionamiento del componente?**

__Muy fácil __Fácil __Normal __Difícil __Muy difícil

2. **¿Se pueden utilizar correctamente las operaciones del componente?**

__Sí __No

3. **¿Qué tan rápido se pueden ejecutar las operaciones del componente?**

__Muy lento __Lento __Normal __Rápido __Muy Rápido

4. **¿Cuántos pasos se necesitan para realizar una operación en el componente?**

__Muy pocos __Pocos __Normal __Varios __Demasiados

5. **¿Con que regularidad se cometen errores en la ejecución de las operaciones del componente?**

__Siempre __Casi siempre __Pocas veces __Casi nunca __ Nunca

6. **¿Cuál es el nivel de satisfacción al interactuar con el componente?**

__ Muy insatisfecho __ Insatisfecho __ Poco satisfecho __ Satisfecho __Muy Satisfecho

## Con interfaces Gráficas

1. **¿Qué tan fácil es comprender el funcionamiento del componente?**

__Muy fácil __Fácil __Normal __Difícil __Muy difícil

2. **¿Se pueden utilizar correctamente las operaciones del componente?**

__Sí __No

3. **¿Qué tan rápido se pueden ejecutar las operaciones del componente?**

__Muy lento __Lento __Normal __Rápido __Muy Rápido

4. **¿Cuántos pasos se necesitan para realizar una operación en el componente?**

__Muy pocos __Pocos __Normal __Varios __Demasiados

5. **¿Con que regularidad se cometen errores en la ejecución de las operaciones del componente?**

__Siempre __Casi siempre __Pocas veces __Casi nunca __ Nunca

6. **¿Cuál es el nivel de satisfacción al interactuar con el componente?**

__ Muy insatisfecho __ Insatisfecho __ Poco satisfecho __ Satisfecho __Muy Satisfecho

**Anexo 10** Lista de Chequeo Usabilidad de Sitios Web aplicada al componente de Estructura y Composición

| **Elementos definidos por la metodología** | | | | |
|---|---|---|---|---|
| **No** | **Indicador a evaluar** | **Evaluación** | **NP** | **Observación** |
| **Visibilidad del sistema** | | | | |
| 1 | ¿La página refleja la identidad de la empresa logos, compañía...)? | **0** | | |
| 2 | ¿Cada pantalla empieza con un título que describe su contenido? | **0** | | |
| 3 | ¿Cuándo se selecciona un icono se diferencia de los no seleccionados? | **0** | | |
| 4 | ¿Los enlaces del menú se resaltan cuando se seleccionan? | **0** | | |
| 5 | ¿Los iconos que aparecen se identifican claramente con lo que representan? | **0** | | |
| 6 | ¿El menú de navegación aparece en un lugar destacado? | **0** | | |
| 7 | ¿No utiliza más de siete opciones principales en el menú de navegación? | **0** | | |
| 8 | ¿Si la respuesta a una acción se retrasa, aparece un mensaje o indicio como que el sistema está | **0** | | |

| | | | | |
|---|---|---|---|---|
| | procesando la acción? | | | |
| 9 | ¿El sitio le indica al usuario en que parte de la estructura del sitio web se encuentra, es decir si muestra 'migas de pan'? | **0** | | |
| 10 | ¿El nombre de los enlaces es el mismo que el título de la página a la que dirige? | **0** | | |
| 11 | ¿El logo de la organización está ubicado en el mismo lugar en todas las páginas, y hacer clic en el logo retorna al usuario a la página más lógica (Ejemplo: la página de inicio)? | **NP** | | |
| 12 | ¿Los títulos de las páginas, tablas e imágenes son descriptivos y distintivos? | **0** | | |
| 13 | ¿Las etiquetas de las categorías describen con precisión la información de las mismas? | **0** | | |
| 14 | ¿Cuándo una tarea involucra documentos fuente, la interfaz es compatible con las características del documento fuente? | **NP** | | |
| 15 | ¿Las imágenes se muestren con buena resolución? | **NP** | | |
| 16 | ¿No se muestran errores ortográficos? | **NP** | | |
| 17 | ¿No hay ninguna imagen | **NP** | | |

| | | | | |
|---|---|---|---|---|
| | con información relevante? | | | |
| **Lenguaje común entre sistema y usuario** | | | | |
| 18 | ¿El lenguaje es simple, con un tono adecuado? | **0** | | |
| 19 | ¿La información que se presenta en la aplicación es fácil de entender y memorizar? | **0** | | |
| 20 | ¿Utiliza los conceptos establecidos para las funciones estándar? (“buscar" para las búsquedas, etc.) | **0** | | |
| 21 | ¿Evita el lenguaje técnico: términos informáticos o propios de Internet? | **0** | | |
| 22 | ¿Se utiliza siempre la misma nomenclatura para las mismas funciones? | **0** | | |
| 23 | ¿Los acrónimos y abreviaturas son definidos al ser usados por primera vez? | **NP** | | |
| 24 | ¿No hace uso de términos extranjeros? | **NP** | | |
| 25 | ¿Utiliza un texto específico y descriptivo en los vínculos? | **0** | | |
| 26 | ¿La información es de rápida lectura, y con una disposición asequible? | **0** | | |
| 27 | ¿Los vínculos basados en nombres de la gente, conducen a las biografías cortas o a sus propios blogs, no a un correo electrónico? | **NP** | | |
| 28 | ¿Si se desea incluir un enlace de correo electrónico, se muestra el correo y no el nombre de la persona? | **NP** | | |
| **Libertad y control por parte del usuario** | | | | |
| 29 | ¿Existe una manera lógica | **NP** | | |

| | | | | |
|---|---|---|---|---|
| | de acceder a páginas relacionadas o a otras secciones? | | | |
| 30 | ¿Tras una acción relevante hay una opción de vuelta atrás? | **NP** | | |
| 31 | ¿Si una acción tiene consecuencias, el sistema proporciona información y pide confirmación antes de continuar? | **0** | | |
| 32 | ¿En las páginas internas hay un acceso a la página de inicio en una zona visible y reconocible? | **NP** | | |
| 33 | ¿El Sitio cuenta con un mapa o buscador que facilite el acceso directo a los contenidos? | **NP** | | |
| 34 | ¿Al dar clic en el botón “Atrás” siempre lleva al usuario de vuelta a la página de dónde vino? | **0** | | |
| 35 | ¿El sitio no deshabilita el botón “Atrás” y dicho botón aparece activo en la barra de herramientas del navegador en todas las páginas? | **0** | | |
| 36 | ¿El sitio evita que los usuarios se registren de manera innecesaria? | **NP** | | |
| 37 | ¿La página se ve con cualquier resolución de pantalla? | **0** | | |
| 38 | ¿Es posible aumentar y disminuir el tamaño de letra? | **NP** | | |
| 39 | ¿Las imágenes de los | **NP** | | |

| | | | | |
|---|---|---|---|---|
| | productos se pueden ampliar? | | | |
| 40 | ¿Ofrece el contenido en otros formatos, como dispositivos móviles? | **NP** | | |
| 41 | ¿Es posible imprimir la web sin perder información? | **NP** | | |
| 42 | ¿Se pueden guardar las páginas web? | **0** | | |
| 43 | ¿El sitio provee una clara retroalimentación cuando una tarea ha sido completada exitosamente? | **0** | | |
| 44 | ¿Una imagen que sirve como enlace es fácilmente distinguible? | **NP** | | |
| 45 | ¿En caso que se muestre información relacionada con registros obtenidos de la base de datos existe un sistema de navegación donde el usuario pueda especificar cuantos elementos desea ver en la página? | **NP** | | |
| 46 | ¿Los usuarios son informados si es necesario un plug-in del navegador o resolución específico? | **NP** | | |
| 47 | ¿Las páginas que utilizan nuevas tecnologías siguen funcionando cuando dicha tecnología no está presente (por ejemplo, los plug-ins de Flash)? | **NP** | | |
| 48 | ¿Cuándo es necesaria la descarga de un plug-in, hay un enlace a la página | **NP** | | |

| | | | | |
|---|---|---|---|---|
| | donde obtenerlo? | | | |
| 49 | ¿El motor de búsqueda maneja correctamente (No arroja ningún resultado) las búsquedas vacías (cuando no se introduce nada)? | **0** | | |
| 50 | ¿Las etiquetas de navegación y links contienen las "palabras clave" que los usuarios necesitan para alcanzar su objetivo? | **0** | | |
| 51 | ¿Los usuarios pueden ordenar y filtrar los resultados? | **NP** | | |
| 52 | ¿Los enlaces que invocan acciones (ej. descargas, nuevas ventanas) están claramente distinguidos de los links que cargan otras páginas? | **0** | | |
| 53 | ¿La página de resultados de una búsqueda indica claramente cuántos resultados tuvo la búsqueda? | **0** | | |
| 54 | ¿La página de resultados de una búsqueda no muestra resultados duplicados (ni duplicados reales ni duplicados muy parecidos)? | **0** | | |
| 55 | ¿La caja de búsqueda es suficientemente grande para manejar la longitud de las consultas más comunes? | **0** | | |
| 56 | ¿Las búsquedas cubren todo el sitio, no una porción de él? | **NP** | | |
| 57 | ¿La interfaz de búsqueda está ubicada donde los | **0** | | |

| | | | | |
|---|---|---|---|---|
| | usuarios esperan encontrarla (en la parte superior derecha de la página)? | | | |
| **Consistencia y estándares** | | | | |
| 58 | ¿Las imágenes tienen tamaños adecuados que no dificultan el acceso a las páginas? | **NP** | | |
| 59 | ¿Existe un cambio visible cuando el ratón apunta a algo clickeable (excluyendo los cambios de cursor)? | **0** | | |
| 60 | ¿Todos los botones“clickeables” son efectivamente presionables? | **0** | | |
| 61 | ¿Los íconos son visualmente y conceptualmente distintos pero mantienen una armonía? | **0** | | |
| 62 | ¿Para tareas similares, los diálogos, formularios son similares? | **0** | | |
| 63 | ¿Existe una clara distinción entre campos “requeridos” y “opcionales” en los formularios? | **1** | | NO existe una clara distinción entre campos “requeridos” y “opcionales” en los formulario |
| 64 | ¿Las preguntas en los formularios están agrupadas de manera lógica y cada grupo tiene un título descriptivo? | **NP** | | |
| 65 | ¿Los campos en los formularios contienen ayudas, ejemplos o modelos de respuestas para demostrar el dato que se debe introducir? | **0** | | |

| | | | | |
|---|---|---|---|---|
| 66 | ¿El nombre de los botones de un formulario es adecuado, aplicado a la acción, no general (Ej.: utilizar “Enviar” en vez de “OK”)? | **0** | | |
| 67 | ¿Las listas de opciones, botones de radio y casillas son preferibles a las cajas de texto en los formularios? | **0** | | |
| 68 | ¿Se mantiene una navegación consistente y coherente en todas las pantallas? | **0** | | |
| 69 | ¿La distribución y ubicación de los elementos estructurales que contienen las páginas se mantiene constante a lo largo de la aplicación? | **0** | | |
| 70 | ¿Se usa la misma fuente para todos los navegadores? | **0** | | |
| 71 | ¿Puede utilizarse en cualquier navegador? | **0** | | |
| 72 | ¿Se mantiene una tipografía coherente en todo el sitio web? | **0** | | |
| 73 | ¿Existen enlaces redundantes? Enlaces redundantes: Son enlaces con rótulos diferentes que llevan a una misma página. | **NP** | | |
| 74 | ¿El link al mapa del sitio aparece en todas las páginas del sitio? | **NP** | | |
| 75 | ¿Añade una descripción en las imágenes? | **NP** | | |
| 76 | ¿Señala claramente los vínculos a archivos PDF | **NP** | | |

| | | | | |
|---|---|---|---|---|
| | como tal? | | | |
| 77 | ¿Especifica el tamaño de los archivos PDF? | **NP** | | |
| 78 | ¿El sitio tiene una URL correcta, clara y fácil de recordar? | **NP** | | |
| 79 | ¿Tiene un tiempo de respuesta rápida? | **0** | | |
| 80 | ¿Los vínculos llevan a donde prometen ir? | **0** | | |
| 81 | ¿El sistema de navegación es amplio y sencillo (muchos ítems en un menú) en vez de un menú profundo? | **0** | | |
| 82 | ¿Se usan nombres estandarizados (“mapa web”, “acerca de…”)? | **NP** | | |
| 83 | ¿Se proporciona un texto equivalente para todo elemento no textual, tales como imágenes, para explicar su contenido a discapacitados visuales? | **NP** | | |
| 84 | ¿El documento está estructurado para que pueda ser leído con o sin una hoja de estilo, utilizando adecuadamente los tags de HTML? | **0** | | |
| 85 | ¿Las presentaciones multimedia, en caso de existir están sincronizadas con sus subtítulos? | **NP** | | |
| 86 | ¿El cambio de idioma en los textos está identificado?<br>El texto que se encuentre en otro idioma debe estar en cursiva o en un formato diferente al del resto del texto. | **NP** | | |
| 87 | ¿El sitio funciona correctamente en un lector de pantalla y/o navegador de voz? | **NP** | | |
| 88 | ¿Es posible la navegación sin ratón? | **NP** | | |
| 89 | ¿Se han creado atajos de teclado? | **NP** | | |

| | | | | |
|---|---|---|---|---|
| 90 | ¿Todos los estilos se han creado en hojas CSS? | **0** | | |
| **Estética y diseño minimalista** | | | | |
| 91 | ¿Cumple el sitio con el principio de usabilidad de realizar las operaciones con un máximo de tres click? | **0** | | |
| 92 | ¿Existe suficiente contraste entre el color del fondo y el del texto? | **0** | | |
| 93 | ¿Los tipos y tamaños de letra son legibles y distinguibles? | **0** | | |
| 94 | ¿Añade color de fondo a los div que llevan imagen de fondo? (Para los usuarios que desactivan las imágenes, desaparece el contraste entre texto y fondo, convirtiéndose en texto ilegible.) | **0** | | |
| 95 | ¿Poseen las páginas animaciones innecesarias? | **0** | | |
| 96 | ¿El sitio puede ser usado sin desplazamiento horizontal? | **0** | | |
| 97 | ¿El uso de los colores es moderado? | **0** | | |
| 98 | ¿Se usan los estilos (negritas, cursivas...) con moderación? Si todo está resaltado con negrita o cursiva, el cerebro se acostumbra y deja de parecerle destacado. | **0** | | |
| 99 | ¿Utiliza la misma tipografía los mismos aspectos, ejemplo: mantener una tipografía estándar por elementos? | **0** | | |

| | | | | |
|---|---|---|---|---|
| 100 | ¿Utiliza un interlineado adecuado para una buena lectura? | **0** | | |
| 101 | ¿Se usan frases breves y concisas: que resuman los puntos clave y vayan al grano? | **0** | | |
| 102 | ¿Los párrafos son cortos? | **NP** | | |
| 103 | ¿Los textos están corregidos? | **NP** | | |
| 104 | ¿Resalta en negrita los conceptos principales de los textos densos? | **NP** | | |
| 105 | ¿Utiliza listas de boliche y numeradas? | **NP** | | |
| 106 | ¿El sitio evita el uso excesivo del texto en mayúsculas? | **0** | | |
| 107 | ¿El texto dentro del sitio no debe estar justificado? | **0** | | |
| 108 | ¿No se utiliza el texto subrayado en el cuerpo del texto para resaltar a menos que sea un hipervínculo? | **NP** | | |
| **Prevención de errores** | | | | |
| 109 | ¿Existe suficiente espacio entre los elementos de acción (links, botones, etc.) para prevenir que el usuario haga click en el elemento incorrecto? | **0** | | |
| 110 | ¿Hay ausencia de enlaces rotos o que no lleven a ninguna página? | **NP** | | |
| 111 | ¿Se dan indicaciones para completar campos problemáticos? | **NP** | | |
| 112 | ¿Los botones de acción, (tales como “Enviar”) siempre son invocados por el usuario y no automáticamente invocados por el sistema cuando el último campo de un formulario ha sido lleno? | **0** | | |

| | | | | |
|---|---|---|---|---|
| 113 | ¿El espacio entre los campos del formulario es suficiente como para distinguirlos unos de otros? | **0** | | |
| 114 | ¿Se evita el contenido importante del sitio en ventanas emergentes? | **0** | | |
| 115 | ¿La zona de acción del vínculo esta ampliada para acertar a la primera? | **0** | | |
| 116 | ¿El buscador (si existe) permite errores tipográficos y ortográficos (tildes)? | **1** | | El buscador No permite errores tipográficos y ortográfico |
| **Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de los errores** | | | | |
| 117 | ¿En caso de errores de consistencia dentro del sitio, ¿se ofrece un mensaje de personalizado mediante una página explicativa?, (Por ejemplo: Error 404 para página inexistente) | **0** | | |
| 118 | ¿El mensaje de error permite volver a la situación anterior? | **0** | | |
| 119 | ¿Entrega información de contacto con múltiples opciones dentro y fuera de Internet? (Por ejemplo: Formulario, Teléfono, Fax, Correo electrónico, Chat, y hasta Wave, Buzz, Skype…, teléfono institucional, mesa de ayuda) | **NP** | | |
| 120 | ¿Ofrece área de Preguntas Frecuentes con datos de ayuda a usuarios? | **NP** | | |
| 121 | ¿Ofrece páginas de ayuda que explican cómo usar el | **NP** | | |

| | | | | |
|---|---|---|---|---|
| | Sitio? | | | |
| 122 | ¿Si la ayuda obliga a salir de la zona principal, se proporciona un medio para moverse entre esa ventana y la ayuda? | **NP** | | |
| 123 | ¿La ayuda no interrumpe la tarea del usuario? | **NP** | | |
| 124 | ¿El sitio está diseñado para necesitar el mínimo de ayuda y de instrucciones? | **NP** | | |
| 125 | ¿La ayuda está organizada en pasos? | **NP** | | |
| 126 | ¿Se dan ejemplos para facilitar la tarea? | **NP** | | |
| 127 | ¿Se utilizan explicaciones cortas en la ayuda? | **NP** | | |
| **Ayuda y documentación** | | | | |
| 128 | ¿Existe un vínculo a los datos de contacto en un lugar bien visible en todas las páginas web del sitio? | **NP** | | |
| 129 | ¿Información necesaria en página de contacto? (Atención al cliente, Consultas, Soporte técnico, Solicitudes de empleo.) | **NP** | | |
| 130 | ¿Existe una página con las indicaciones sobre la protección de datos en el sitio web? | **NP** | | |
| 131 | ¿Incluye un vínculo a los datos legales en todas las páginas? | **NP** | | |
| 132 | ¿Puede el usuario ponerse en contacto con el encargado del Sitio Web para hacer sugerencias o comentarios? | **NP** | | |

| | | | | |
|---|---|---|---|---|
| 133 | ¿Funcionan correctamente los formularios de contacto? | **NP** | | |
| 134 | ¿Hay alguien encargado de recibir y contestar estos mensajes? | **NP** | | |
| 135 | ¿El sitio soporta a los usuarios novatos y expertos brindando diferentes niveles de explicación? (ej. en páginas de ayuda y mensajes de error) | **NP** | | |
| 136 | ¿La política de privacidad del sitio es fácil de encontrar, especialmente esas páginas que piden información personal? | **NP** | | |
| 137 | ¿Cuándo existen múltiples pasos en una tarea, el sitio muestra todos los pasos que deben ser completados y provee una retroalimentación al usuario indicándole la posición actual en toda la ruta de la tarea? | **NP** | | |
| 138 | ¿La funcionalidad de los controles para nuevos dispositivos es exactamente la misma que para los otros dispositivos? | **NP** | | |
| **Flexibilidad y eficiencia de uso** | | | | |
| 139 | Se defina de manera correcta gráficos y tablas utilizando atributos (leyendas, unidades de medida, etc.) | **0** | | |
| 140 | ¿Las partes o secciones más importantes de los sitios son accesibles desde la página de inicio? | **NP** | | |
| 141 | ¿Las páginas no requieren volver a escribir la información solicitada en páginas anteriores? | **0** | | |

| | | | | |
|---|---|---|---|---|
| 142 | ¿Existen aceleradores, accesos rápidos a operaciones frecuentes? | **NP** | | |
| 143 | ¿El cursor se desplaza adecuadamente en un formulario al presionar “tabulador”? | **0** | | |
| 144 | ¿Se implementen validaciones antes de que el usuario envíe información? | **1** | | No se implementen validaciones antes de que el usuario envíe información |

Tabla A.9 Lista de Chequeo Usabilidad de Sitios Web.

Anexo 10 Instrumento de evaluación de la métrica TOC.

| Clase | Cantidad de Procedimientos | Responsabilidad | Complejidad | Reutilización |
|---|---|---|---|---|
| EyCNomencladorCtrl | 14 | Alta | Alta | Baja |
| EyCEstructuraCtrl | 5 | Baja | Baja | Alta |
| EyCNivelEstructuralCtrl | 11 | Media | Media | Media |
| EyC_estructura | 3 | Baja | Baja | Alta |
| EyC_campo_estructura | 2 | Baja | Baja | Alta |
| EyC_estructura_operacion | 2 | Baja | Baja | Alta |
| EyC_valores_estructura | 2 | Baja | Baja | Alta |
| EyC_campo_nomenclador | 6 | Baja | Baja | Alta |
| EyC_campo_nomenclador | 2 | Baja | Baja | Alta |
| EyC_nomenclador | 2 | Baja | Baja | Alta |
| EyC_nomenclador_opracion | 2 | Baja | Baja | Alta |
| NomencladorController | 15 | Alta | Alta | Baja |
| EstructuraController | 17 | Alta | Alta | Baja |
| EyCNomencladorCtrl | 14 | Alta | Alta | Baja |

| | | | | |
|---|---|---|---|---|
| EyCEstructuraCtrl | 5 | Baja | Baja | Alta |
| EyCNivelEstructuralCtrl | 11 | Media | Media | Media |
| EyC_estructura | 3 | Baja | Baja | Alta |
| EyC_campo_estructura | 2 | Baja | Baja | Alta |

Tabla A.10 Instrumento de evaluación de la métrica TOC.

Anexo 11 Instrumento de evaluación de la métrica RC.

| Clase | Cantidad de relaciones de uso | Acoplamiento | Complejidad del mantenimiento | Cantidad de pruebas | Reutilización |
|---|---|---|---|---|---|
| EyCNomencladorCtrl | 0 | Ninguno | Baja | Alta | Baja |
| EyCEstructuraCtrl | 1 | Bajo | Baja | Alta | Baja |
| EyCNivelEstructuralCtrl | 1 | Bajo | Baja | Alta | Baja |
| EyC_estructura | 1 | Bajo | Baja | Alta | Baja |
| EyC_campo_estructura | 1 | Bajo | Baja | Alta | Baja |
| EyC_estructura_operacion | 2 | Media | Media | Media | Media |
| EyC_valores_estructura | 3 | Alta | Alta | Baja | Alta |
| EyC_campo_nomenclador | 1 | Bajo | Baja | Alta | Baja |
| EyC_campo_nomenclador | 1 | Bajo | Bajo | Alta | Bajo |
| EyC_nomenclador | 3 | Alto | Alta | Baja | Alta |
| EyC_nomenclador_opracion | 1 | Bajo | Baja | Alta | Baja |
| NomencladorController | 3 | Alta | Alta | Baja | Alta |
| EstructuraController | 2 | Media | Media | Media | Media |

Tabla A.11 Instrumento de evaluación de la métrica TOC.