Universidad de las Ciencias Informáticas

Facultad 3

Título: Generación de código JavaScript para construir interfaces gráficas en Ext JS 4 para aplicaciones web de gestión.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor (es):

Efraín F. Ruiz Zamora

Anabel Laza Tarafa

Tutores:

Ing. René R. Bauta Camejo.

Ing. Julio César Ocaña Bermúdez.

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Autores:	Tutores:
Efraín Francisco Ruiz Zamora	Ing. Julio Cesar Ocaña Bermúdez
Anabel Laza Tarafa	Ing. René Bauta Camejo

_____	_____
Firma	Firma
_____	_____
Firma	Firma

"El primer 90% del código está reservado para el primer 90% del tiempo de desarrollo. El 10% de código restante es para el otro 90% del tiempo de desarrollo".

Tom Cargill

Agradecimientos

Yo Efraín Francisco Ruiz Zamora le agradezco a:

La Revolución por darme la oportunidad de convertirme en el hombre que soy y permitirme estudiar en la Universidad de las Ciencias Informáticas.

Mi familia, en especial a mi mamá, mi papá y mi hermano por apoyarme en cada paso que he dado en mi vida.

Mi novia Liliana por brindarme su ayuda y apoyo en todo cuanto ha podido y por tolerarme durante tanto tiempo.

A todos mis compañeros del grupo 3507, en especial a mi familia de aquí de la UCI, la gente de "La Dinastía" (Michel, Lian, Luis Ángel y Rigo).

A mi hermanito de aquí de la UCI: Alexei.

RESUMEN

En la Universidad de Ciencias Informáticas se lleva una intensa tarea de producción de software, por lo que buscar ideas que agilicen el proceso de desarrollo de software es una tarea fundamental. Las herramientas de generación de código fuente juegan un papel primordial en la producción de software; el funcionamiento de estas herramientas se basa en la obtención de un modelo y la transformación del mismo en código fuente útil en el desarrollo de una aplicación. Con ellas se disminuye el tiempo en que se desarrolla un producto y las posibilidades de errores en la elaboración del mismo; además se aumenta la calidad del software creado y se facilita su mantenimiento. En este trabajo se propone construir una herramienta sencilla, práctica y adaptable a los estilos de programación de cualquier marco de trabajo que utilice Doctrine 2.0 para generar las entidades y utilice PHP como lenguaje de programación; mediante la cual se pueda generar el código JavaScript para construir interfaces gráficas en Ext JS 4 de un CRUD (Create, Read, Update, Delete). La herramienta se clasificará en pasiva según su interacción con el código generado y se desarrollará usando la técnica de generación de plantillas.

PALABRAS CLAVES

Entidad, Ext JS, generación de código, herramienta, interfaz, JavaScript.

Índice de contenidos

Índice de figuras

Índice de tablas

INTRODUCCIÓN

En la Universidad de las Ciencias Informáticas, donde radican centros de desarrollo de software, según (Osorio Leyva, 2008) existe una vinculación a la creación de productos con alta calidad, trazándose como meta avanzar en el desarrollo de soluciones informáticas. Una particularidad esencial en la producción de software, es concluir los proyectos en la fecha acordada con el cliente; pero debido al cúmulo de proyectos, y los disímiles módulos que estos contienen, en repetidas ocasiones se atrasa la fecha pactada; a pesar de que se han perfeccionado los procesos y metodologías de software, se han capacitado a los recursos humanos y se utilizan las tecnologías y marcos de trabajo que están en constante desarrollo y perfeccionamiento (Sosa Marín, 2009).

En los centros de desarrollo de la universidad se desarrollan aplicaciones con el uso de las tecnologías web, en las cuales la interfaz gráfica según (Guzmán Ojeda, y otros, 2013) juega un papel importante, ya que es con la que interactúan el cliente y los usuarios finales del sistema. Entre los marcos de trabajos usados para el desarrollo de estas interfaces visuales en las aplicaciones web de gestión (Sistemas de información[1]) se encuentra Ext JS, que es una librería JavaScript, desarrollado por la organización Sencha. Ext JS usa estructuras definida para el desarrollo de cada componente visual, por lo que, al utilizarlo el código resultante para la creación de interfaces gráficas se vuelve repetitivo (varían sólo algunos atributos o datos).

Luego de una encuesta realizada en el centro CEIGE, la cual se muestra en los anexos en la fig. A.1, aplicada a una muestra aleatoria de 30 desarrolladores que utilizan el marco de trabajo EXT JS 4 para el desarrollo de interfaces gráficas, con el objetivo de obtener el tiempo promedio que se demoran en el desarrollo de las interfaces de un CRUD (Create, Read, Update, Delete); se obtuvo como resultado que sólo el 80% de los programadores del centro que utilizan este marco de trabajo han desarrollado interfaces de un CRUD y de ellos el 65% considera que la creación

[1] En informática, un **sistema de información** es cualquier sistema computacional que se utilice para obtener, almacenar, manipular, administrar, controlar, procesar, transmitir o recibir datos, para satisfacer una necesidad de información.

de estas interfaces es compleja. En la actualidad a pesar de que el 36% de los profesionales se demoran de 1 a 3 horas en el proceso de desarrollo de las vistas de un CRUD, todavía existe un 58% de estos que necesitan entre 1 y 3 días. A raíz de la encuesta realizada se pudo corroborar que más del 50% del tiempo empleado por los desarrolladores de este marco de trabajo en el centro, es utilizado en la creación de estas interfaces; convirtiéndose en uno de los factores que provoca el aumento del tiempo planificado para el proyecto. A partir de la problemática planteada se define como **problema a resolver** ¿Cómo disminuir el tiempo de desarrollo de interfaces gráficas en Ext JS 4 para aplicaciones web de gestión?

Objeto de estudio:

Generación de código fuente.

Campo de acción:

Generación de código JavaScript a partir de clases autogeneradas por Doctrine 2.0.

Objetivo general:

Desarrollar una herramienta que permita la generación de código JavaScript para disminuir el tiempo de desarrollo de interfaces gráficas en Ext JS 4 para aplicaciones web de gestión.

Objetivos específicos:

- Construir el marco teórico de la investigación relacionada con la generación de código fuente.
- Realizar el análisis y diseño del componente para la generación de código JavaScript a partir de clases autogeneradas por Doctrine 2.0.
- Implementar un componente para la construcción de interfaces gráficas en Ext JS 4 a partir de clases autogeneradas por Doctrine 2.0.
- Validar la solución mediante la aplicación de métricas, pruebas de caja blanca y pruebas de caja negra.
- Validar que se disminuye el tiempo de desarrollo de las interfaces gráficas con el uso de la herramienta.

Idea a defender:

Si se desarrolla y utiliza una herramienta que permita la generación de código JavaScript para construir interfaces gráficas en Ext JS 4, a partir clases autogeneradas por Doctrine 2.0, se reducirá el tiempo de desarrollo de interfaces gráficas en Ext JS 4 para aplicaciones web de gestión.

Para dar cumplimiento al objetivo general propuesto en el presente trabajo, se aplican los métodos de investigación científicos que a continuación se mencionan:

Del nivel teórico se selecciona:

Análisis Histórico-Lógico: permitió comprender de forma más clara la esencia del objeto de estudio y su concepción histórica. Para ello se realizó un análisis de las herramientas de generación de código fuente existentes.

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. Una vez concluido el análisis, se logró un conocimiento concreto sobre herramientas para la generación de código fuente y su necesidad, se formularon conceptos y definiciones fundamentales relacionados con el tema y se ofreció una propuesta de solución.

Inductivo-deductivo: permitió arribar a conclusiones particulares para la selección de las herramientas y la metodología a utilizar en el desarrollo del sistema

Del nivel empírico se selecciona:

Encuesta: permitió mediante un cuestionario pre-elaborado, obtener información sobre el tiempo real que demoran los desarrolladores en realizar una interfaz de un CRUD con Ext JS.

La **estructura del trabajo** quedó constituida en tres capítulos, conclusiones, recomendaciones, bibliografía, anexos y glosario.

Capítulo I. Fundamentación teórica: en este capítulo se hace un estudio del estado del arte de las herramientas actuales utilizadas para la generación de código. Las tendencias existentes, las técnicas utilizadas, la tecnología y herramientas empleadas para el análisis, diseño e implementación de la herramienta.

Capítulo II. Características del sistema: se trata el objeto de informatización, logrando una propuesta del sistema, especificando requisitos funcionales y no funcionales que debe cumplir el sistema, diseño de la propuesta de solución, así como patrones de diseños empleados en la misma.

Capítulo III. Implementación y Prueba: aborda los estándares de codificación empleados; además de las pruebas realizadas a la aplicación para evaluar la calidad del producto de software desarrollado.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hace un estudio del estado del arte, donde se brinda una visión general de los aspectos relacionados con las herramientas generadoras de código, sus características, y una descripción de los principales conceptos asociados al problema para entender mejor la propuesta de solución. Se describen las tecnologías, herramientas y técnicas que serán empleadas para la construcción del software.

1.2 Acerca de los generadores de código

1.2.1 ¿Qué es un generador de código?

Según (Rúa Suárez, Julio, 2010), los generadores de códigos son herramientas que permiten generar automáticamente aquellos fragmentos de código que pueden repetirse más de una vez en las iteraciones de construcción de un sistema y en proyectos diferentes de software. Estos generan código en algún lenguaje predefinido por el desarrollador listo para compilar o interpretar. Ellos parten de una estructura, que puede ser un diagrama de clases o un modelo de base de datos, para generar el código fuente. El código generado por estos va a depender del modelo que se le entregue y de los algoritmos y patrones que tengan implementado los mismos. Las herramientas generadoras de código poseen plantillas para representar los algoritmos y patrones implementados, las cuales el usuario puede modificar y adaptar según sus necesidades (Herrington, 2006).

1.2.2 Tipos de generadores de código

Según su interacción con el código generado se clasifican en: (Herrington, 2006)

- **Activos**: son aquellos que permiten generar varias veces sobre el mismo código generado a partir de cambios en la entrada. Estos generadores definen espacios de código seguros donde el programador puede hacer los cambios que desee sin que éstos se pierdan en las sucesivas generaciones de código.

Pasivos: generan el código una vez y no vuelven a tener interacción con él. Si se corrige un error en los mecanismos de generación o se cambia el diseño y se vuelve a generar se pierde lo que se codificó manualmente.

Según la aproximación que usan para generar el código se clasifican en: (Herrington, 2006) (Molina Moreno, 2003)

- **Estructural**: generan bloques de código, desde modelos estáticos y relaciones entre objetos. Las primitivas de trabajo en estos modelos son clases, atributos, tipos y asociaciones. Algunas herramientas usan un motor de traducción y plantillas preexistente para especificar correspondencias con un código fuente en particular. La generación de código estructural es incompleta pero ahorra esfuerzo de codificación manual y proporciona un marco de trabajo inicial consistente con los modelos.
- **De comportamiento**: generan código completo a partir de modelos de máquinas de estados y la especificación de acciones en un lenguaje de alto nivel. Algunos métodos que modelan comportamiento con máquinas de estados, añaden código (como C++ o un lenguaje propietario) para representar las acciones que ocurren durante la transición de estado. Junto con modelos de estructuras de objetos y mecanismos de comunicación, esta técnica permite a las herramientas generar código para el modelo completo de la aplicación. Un beneficio de esta técnica es la capacidad para simular y verificar el comportamiento del sistema basado en modelos antes de que el código sea generado.
- **Traductivos:** se basan en que los modelos de aplicación y de arquitectura son independientes uno del otro. Un modelo de aplicación completo, con estructura de objetos, comportamiento y comunicaciones es creado usando el método de análisis orientado a objetos. Un modelo de arquitectura (un conjunto de patrones llamados plantillas o arquetipos) es desarrollado con una herramienta que soporte esta aproximación. Entonces, un motor de traducción genera el código para la aplicación de acuerdo con las reglas de correspondencia en la arquitectura. Las aproximaciones traductivas ofrecen una reutilización significativa debido a que la aplicación y el modelo de arquitectura son independientes.

1.2.3 Creación de generadores de código.

Para la creación de generadores de código se deben considerar los siguientes aspectos: (Informática, 2004)

- La arquitectura de software para la cual se va a desarrollar el generador.
- Las características específicas del lenguaje de programación para el que se va a desarrollar.
- El lenguaje con el que se desarrollará el propio generador.

1.2.4 Tipos de generaciones de código.

Existen tres tipos fundamentales para la generación de código, enumerados a continuación (Sosa Marín, 2009):

- **Plantillas (Templating):** se genera un armazón (o esbozo) de código fuente no funcional para ser editado, con el que se evita tener que escribir la parte más repetitiva del código. Generalmente poco complejo, recomendable.
- **Parcial:** se genera código fuente que implementa parcialmente la funcionalidad requerida, pero que el programador usará como base para modificar, integrar y/o adaptar a sus necesidades. No suele ser recomendable.
- **Total**: se genera código fuente funcionalmente completo pero que no va a ser modificado por el programador, sino que si es necesario se vuelve a regenerar. Por lo general tampoco suele ser un código excesivamente complejo, recomendable.

1.2.5 Funcionamiento de los generadores de código

Los generadores implementan las siguientes fases (Molina Moreno, 2003):

- **Carga**: la fase de carga consiste en la lectura desde un repositorio (puede ser un fichero binario, XML, una base de datos, o un diagrama UML) del modelo a traducir y crear una representación de éste en memoria. La representación de este modelo en memoria, no tiene porqué ser completa, ni tampoco seguir la misma estructura del modelo. Al contrario, la carga y las estructuras pueden ser adaptadas para cargar sólo la información necesaria y disponerla del modo que sea más conveniente para la tarea de traducción a realizar.
- **Inferencia**: si se han definido una serie de mecanismos de inferencia, que completan la información de modelado, éstos se ejecutan. Las estructuras del modelo en memoria son completadas y extendidas. Es necesario llevar a cabo este proceso antes de proceder a la generación propiamente dicha. El proceso es responsable de realizar pre cálculos útiles y de disponer adecuadamente la información para la fase posterior.
- **Generación**: la última fase es la de generación. En función del código destino a producir, se recorren secuencialmente, y de modo anidado, los elementos de modelo en sucesivas pasadas. Por ejemplo: para cada clase, para cada servicio y para cada argumento. Como resultado de esta fase se obtiene el código generado.

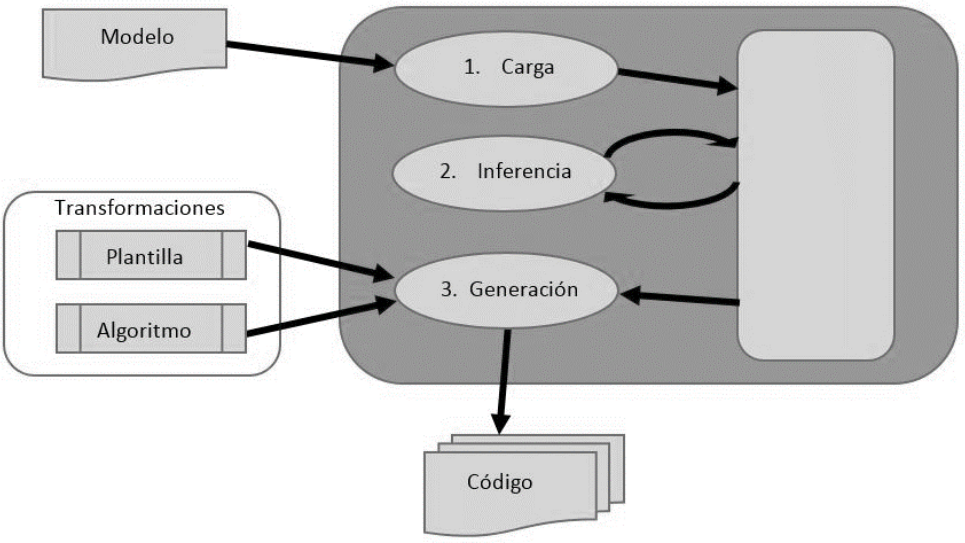

Fig. 1.1 Funcionamiento de un generador de código. (Molina Moreno, 2003)

1.2.6 Técnicas de generación de código

Se reconocen cuatro técnicas generales utilizadas por los generadores de códigos (Molina Moreno, 2003):

- **Clonación**: si el código destino a producir contiene ficheros que permanecen constantes independientemente del modelo a traducir, la traducción puede llevarse a cabo por medio de clonación, o copia directa del fichero origen al directorio de generación. Las librerías de funciones genéricas o ficheros binarios representando imágenes o iconos constantes pueden ser tratados de este modo.
- **Concatenación de cadenas**: la concatenación de cadenas es un modo sencillo de ir construyendo el código destino desde un lenguaje de programación clásico. El nombre proviene del proceso de ir concatenando cadenas de texto que contienen el código a producir. Finalmente, la cadena es volcada a un fichero. Se dispone de la potencia del lenguaje de programación para determinar que código debe ser generado, permitiendo cálculos, sustituciones y procesados complejos.
- **Plantillas**: la generación mediante plantillas puede ser empleada cuando el código destino a producir tiene un patrón de repetición bien caracterizado, de modo que los ficheros generados son idénticos salvo los datos procedentes directamente del modelo a generar. En este caso puede

definirse una plantilla genérica a partir de ejemplares del código objetivo. En esta plantilla, las dependencias del modelo son sustituidas por marcadores con nombre único. Aparejado a la plantilla, se puede definir un proceso de generación que, dado un elemento del modelo, la plantilla sea instanciada a código mediante un proceso de sustitución de cadenas: los marcadores son sustituidos por los datos del modelo correspondiente.

- **Análisis de expresiones mediante gramáticas**: existen ocasiones donde las estrategias previas de generación se vuelven insuficientes. En particular, un caso muy claro es el del tratamiento de expresiones que siguen una gramática dada. Aquí las técnicas de compilación clásicas son las más adecuadas para producir el código necesario. La expresión puede ser convertida en una estructura con forma arbórea en memoria: Un analizador léxico, sintáctico y semántico realizan este trabajo. Después, un algoritmo puede recorrer dicho árbol que representa la expresión para analizarla y decidir el tipo de código a producir.

1.2.7 Ventajas de la generación de código

- **Calidad**: la calidad del código generado está en correspondencia con la calidad de las plantillas y del proceso que se usa para la generación. A medida que son detectados errores y es mejorado el código de las plantillas la calidad del código generado aumenta. Se pueden imponer reglas de estilo al código generado para aumentar su homogeneidad y legibilidad. Se puede generar la documentación del código así como comentarios que faciliten su mantenimiento.
- **Consistencia**: el código generado es extremadamente consistente. El nombre de las variables, métodos y clases es formado de la misma forma en el código. La relación entre el modelo y el código generado permite que también haya consistencia entre la documentación y el código, la cual es muy difícil de mantener en un proceso de desarrollo tradicional.
- **Productividad**: es fácil reconocer los beneficios de la generación de código respecto a la productividad. Se comienza con un diseño de entrada e instantáneamente se obtiene una implementación de salida que responde al diseño. Por otra parte estos beneficios son mucho más apreciados cuando se genera nuevamente el código debido a un cambio en el diseño y no se pierde el trabajo realizado. Por otra parte libera a los programadores del trabajo tedioso y repetitivo permitiéndoles enfocarse en los aspectos que sí requieren de su creatividad e inteligencia.
- **Abstracción**: los generadores construyen el código basados en modelos abstractos. Por ejemplo, se puede generar una capa de acceso a datos, a través de un XML que represente las tablas, los

atributos y sus relaciones. Entre las ventajas que esto brinda está la portabilidad a distintas plataformas porque elevando el nivel de abstracción no se cae en especificaciones únicas de una plataforma, sino que permite centrarse en el modelado de los datos o del negocio (Durán, 2014) (Molina Moreno, 2003).

1.2.8 Desventajas de la generación de código

- **Esfuerzo extra en educación**: se necesita educar a los desarrolladores en las ventajas que ofrece el generador y de qué forma deben emplearlo.
- **Mantenimiento**: cuando se usa un generador hay que darle mantenimiento constantemente. Si es desarrollado por terceros se tiene que estar al tanto de las versiones nuevas que salen y de los errores que se le van detectando. En cualquier caso hay que dedicarle esfuerzos a resolver los errores que pueda traer el generador y a agregarle las nuevas funcionalidades que van surgiendo en el mercado. Si es un generador poco usado se corre además el riesgo de que sus desarrolladores dejen de darle soporte y que por tanto en poco tiempo se vuelva obsoleto.
- **Dominios reducidos**: la generación de código tradicionalmente se ha aplicado a dominios muy específicos y bien conocidos donde es posible anticiparse a la variabilidad de problemas que pueden encontrarse en ese dominio. Los dominios o áreas de aplicación demasiado grandes o fuera de ámbito no se benefician de la aplicación de técnicas de generación de código.
- **Resistencia por parte de los desarrolladores**: hay programadores convencionales que ven la generación de código como una amenaza para su trabajo. Ante lo cual, toman una postura defensiva o de rechazo. Otros afirman que el uso de generadores va contra las buenas prácticas de diseño y critican mucho la idea de copiar y pegar sobre plantillas (Abon Cepeda, 2008) (Herrington, 2006).

1.2.9 Resultado del análisis del estudio de los generadores de código

Una vez analizada la definición de generador de código y los principales aspectos vinculados a las herramientas de esta naturaleza, se determinó que era factible desarrollar un componente que permitiera la generación de código para darle solución a la problemática descrita; decidiendo utilizar en el presente trabajo para la creación del componente, la técnica de plantillas para desarrollar un generador de tipo pasivo. Fue elegida esta técnica, ya que permite crear plantillas genéricas con el código que se desea generar y en caso de cambiar la tecnología o el lenguaje de salida, poder

modificarlas permitiéndole al componente seguir siendo reutilizable. Se seleccionó el tipo pasivo porque no se necesita tener interacción con el código una vez que se ha generado.

1.3 Herramientas de generación de código existente

Luego de una búsqueda en las bases de datos académicas: Microsoft Academic Search, The Collection of Computer Science Bibliographies, CiteSeerX y Google scholar fueron seleccionadas para el análisis las herramientas generadoras de código: CodeSmith, Codejay, PHP Object Generator, PHPMyEdit, Sencha Architect y Ext Designer, teniendo en cuenta que las mismas permitieran crear interfaces en EXT JS u obtener como salida el código fuente de un CRUD, en algún lenguaje predefinido; para realizar un análisis y comparación de cada una de ellas con el objetivo de ver si es factible utilizarlas o de lo contrario buscar aquellos aspectos que puedan ser reutilizados en el desarrollo de la solución de este trabajo.

1.3.1 CodeSmith

El CodeSmith es un generador de código fuente para distintos lenguajes (C#, Java, VB, PHP, ASP.Net, SQL, etc.), a partir de plantillas. Las cuales se pueden crear usando un diseñador de plantillas. Posee plantillas predefinidas para la generación de arquitecturas reconocidas como (NetTiers, CSLA, NHibernate, PLINGO, y más). Se pueden modificar las plantillas o escribir plantillas propias para generar el código exactamente como se requiere. Es reconocida por sus prestaciones, y si tomando en cuenta el alto coste de las licencias se puede decir que los desarrolladores le dan mucha importancia a este tipo de herramientas. Pero el uso del CodeSmith es bastante complejo, la creación y edición de plantillas requiere de altos conocimientos de programación y específicamente del lenguaje XML (CodeSmith, 2010) (CodeSmith, 2008). No es recomendable usarlo dado el alto coste de las licencias necesarias para su uso, además de ser una herramienta muy compleja de usar.

1.3.2 PHP Object Generator

PHP Object Generator (POG), es un generador libre para aplicaciones WEB realizadas con PHP. Permite la generación de los métodos elementales (select, insert, delete, update) en forma de funciones. Es compatible con PHP4/PHP5. Brinda soporte para la codificación de caracteres. Es una aplicación de interfaz WEB. Según su interacción con el código se clasifica en pasivo y utiliza una aproximación estructural para generar código (Gallego Vázquez, 2003). No es adecuado usarlo ya que sólo genera código en forma de funciones.

1.3.3 PHPMyEdit

Herramienta orientada al manejo de bases de datos MYSQL desde PHP. Permite la creación de formularios para el manejo de la base de datos. Genera funciones para la manipulación de tablas. Permite el paginado de las consultas. Obtiene la estructura de la tabla directamente de la base de datos. Es un software libre y de código abierto. En cuanto a su interacción con el código se clasifica en pasivo y utiliza una aproximación estructural para generar código (Informática, 2004). No es práctico usarlo, dado que el código que genera no es de fácil comprensión por los desarrolladores porque no establece una división por capas del mismo.

1.3.4 Codejay

Es una herramienta diseñada para ayudar a los desarrolladores en la creación de aplicaciones WEB. Permite el trabajo con varias plataformas e incorpora mecanismos de administración para la aplicación. Soporta varios sistemas gestores de base de datos (SQL Server, MS Access, MYSQL). Soporta varios lenguajes de programación (ASP, PHP). Permite la creación automática de reportes WEB. Inserta código JavaScript para la validación dentro de los campos del formulario. En cuanto a su interacción con el código se clasifica en activo y utiliza una aproximación estructural para generar código (Montero Ayala, 2001). Su uso se ve limitado, ya que este es propietario y el precio por las licencias necesarias para su utilización son demasiado alto.

1.3.5 Sencha Architect

Es una herramienta de los mismos creadores del marco de trabajo Ext JS, es bastante poderosa, y permite desarrollar aplicaciones web y móviles de manera visual. Sencha Architect es la herramienta que más se asemeja a un IDE para desarrollo de aplicaciones nativas (como Xcode o Visual Studio). Las ventajas de esta herramienta son claras: es posible crear las vistas de las aplicaciones directamente, arrastrando componentes; de la misma forma que herramientas para desarrollo nativo, esto ahorra tiempo al momento de desarrollar (Groner, 2013). Esta herramienta no es posible usarla porque para su uso es necesario pagar el precio de las licencias el cual es muy elevado.

1.3.6 Ext Designer

Esta aplicación de escritorio facilita de una manera muy rápida la creación de las interfaces gráficas de usuario para poder dedicar el tiempo restante a la programación. Con EXT Designer se puede diseñar elegantes interfaces muy rápidamente con un poco de conocimientos usando simplemente un “Arrastrar y soltar”.

Entre sus características más sobresalientes están:

- Cambio entre la vista diseño/código.
- La lista de todos los componentes a la mano.
- Exportar el proyecto a los archivos fuentes correspondientes (Informer Technologies, 2010).

Esta herramienta no es posible usarla porque para su uso es necesario pagar el precio de las licencias el cual es muy elevado.

1.3.7 Comparación de las herramientas

Para la comparación de las herramientas seleccionadas se utilizaron los indicadores: propiedad, interacción con el código, plataforma y características; los cuales se utilizan para verificar si es necesario adquirir licencias para el uso de las mismas, comprobar cuantas de estas están basadas en la misma clasificación seleccionada para el desarrollo del componente (pasivo), analizar cuantas de estas fueron desarrolladas con el uso de las tecnologías web y observar particularidades propias de cada herramienta respectivamente.

Nombre de la herramienta	Propiedad	Interacción con el código	Plataforma	Características
CodeSmith	Privada	Pasivo	Escritorio	La creación y edición de plantillas requiere de altos conocimientos de programación.
PHP Object Generator	Libre	Pasivo	Web	Sólo genera código en forma de funciones.
PHPMyEdit	Libre	Pasivo	Web	El código generado no es de fácil comprensión por los desarrolladores porque no establece una división por capas.
Codejay	Privada	Activo	Escritorio	Permite el trabajo con varias plataformas. Soporta varios sistemas

				gestores de base de datos. Soporta varios lenguajes de programación.
Nombre de la herramienta	**Propiedad**	**Interacción con el código**	**Plataforma**	**Características**
Sencha Architect	Privada	Pasivo	Escritorio	Es posible crear las vistas de las aplicaciones directamente, arrastrando componentes.
Ext Designer	Privada	Pasivo	Escritorio	Es posible crear las interfaces de las aplicaciones directamente, arrastrando y soltando los componentes.

Tabla 1.1 Comparativa entre las principales herramientas de generación de código.

1.3.8 Resultado del análisis del estudio de las herramientas generadoras de código

Se realizó un análisis de cada una de las herramientas y se llegó a la conclusión de que no es factible utilizar ninguna de ellas para la realización del trabajo, ya que estas son de propiedad privada y los precios por las licencias para su utilización son demasiado altos o no permiten generar código fuente en Ext JS 4 o son aplicaciones de escritorio. Debido a estas deficiencias se va a desarrollar una nueva herramienta que permita la generación de código JavaScript para disminuir el tiempo de desarrollo de interfaces gráficas en Ext JS 4 para aplicaciones web de gestión. En la que se reutilizará la forma de crear la estructura de carpetas de Sencha Architect (una carpeta con el nombre entrado por el usuario, la cual contiene un fichero principal “app.js” y una carpeta app, que a su vez contiene las carpetas controller, model, store y view, las que guardan los controladores, modelos, almacenes y vistas respectivamente) y el modo de utilizar los valores de entrada del PHP Object Generator (usando los nombre de los atributos y sus tipos de datos, en los que estos últimos pueden ser modificados).

1.4 Tecnologías y herramientas

1.4.1 Metodología de desarrollo de software

La metodología a emplear será una variación de la metodología AUP-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 UCI

para el desarrollo de software (Sánchez Rodríguez, 2015), de ahí que esta sea la seleccionada para la realización de este trabajo.

1.4.2 Lenguajes de programación

PHP

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). Surgió en 1995, desarrollado por PHP Group.

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. Los archivos cuentan con la extensión (php). Últimamente también se puede usar para la creación de otro tipo de programas incluyendo aplicaciones con interfaz gráfica usando la biblioteca GTK+ (Dondo, 2006) (Cobo, y otros, 2005).

ASP.NET

Este es un lenguaje comercializado por Microsoft, y usado por programadores para desarrollar entre otras funciones, sitios web. ASP.NET es el sucesor de la tecnología ASP, fue lanzada al mercado mediante una estrategia de mercado denominada .NET.

El ASP.NET fue desarrollado para resolver las limitantes que brindaba su antecesor ASP. Creado para desarrollar tanto aplicaciones web sencillas como grandes. Para el desarrollo de ASP.NET se puede utilizar C#, VB.NET o J#. Los archivos cuentan con la extensión (aspx). Para su funcionamiento de las páginas se necesita tener instalado IIS con el Framework .Net. Desde Microsft Windows 2003 se incluye este, solo se necesita instalarlo en versiones anteriores (Serrano Pérez, 2002).

JSP

Es un lenguaje para la creación de sitios web dinámicos, acrónimo de Java Server Pages. Está orientado a desarrollar páginas web en Java. JSP es un lenguaje multiplataforma. Creado para ejecutarse del lado del servidor.

JSP fue desarrollado por Sun Microsystems. Comparte ventajas similares a las de ASP.NET, desarrollado para la creación de aplicaciones web potentes. Posee un motor de páginas basado en los

servlets de Java. Para su funcionamiento se necesita tener instalado un servidor Tomcat (Falkner, y otros, 2002).

Python

Es un lenguaje de programación creado en el año 1990 por Guido van Rossum, es el sucesor del lenguaje de programación ABC. Python es comparado habitualmente con Perl. Los usuarios lo consideran como un lenguaje más limpio para programar. Permite la creación de todo tipo de programas incluyendo los sitios web.

Su código no necesita ser compilado, por lo que se llama que el código es interpretado. Es un lenguaje de programación multi-paradigma, lo cual fuerza a que los programadores adopten por un estilo de programación particular:

- Programación orientada a objetos.
- Programación estructurada.
- Programación funcional.
- Programación orientada a aspectos (González Duque, 2011).

Lenguaje de programación seleccionado

Se seleccionó el lenguaje de programación PHP ya que este es muy fácil de aprender, es un lenguaje muy rápido, soporta en cierta medida la orientación a objeto (clases y herencia), es un lenguaje multiplataforma (Linux, Windows, entre otros) y permite capacidad de conexión con la mayoría de los manejadores de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otras; además de que este lenguaje es libre, incluye una valiosa cantidad de funciones, no requiere definición de tipos de variables ni manejo detallado del bajo nivel y se encuentra entre los lenguajes de programación del lado del servidor más usados, como se puede apreciar en los anexos en la fig. A.2.

1.4.3 Servidores Web

Apache

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 (Foundation, 2007) (Bowen, 2000).

Microsoft IIS

Es el Servidor Web de Microsoft, el IIS (Internet Information Server), es el motor que ofrece esta compañía a modo profesional, con él es posible programar en ASP (Active Server Pages, Páginas de Servidor Activo) las cuales vienen a ser algo similares al PHP, este servidor posee componentes programables desde ASP accediendo a cada uno de sus módulos para una función específica (Windows Server, 2012).

Lighttp

Es un servidor Web para los Sistemas operativos Unix/Linux y Microsoft Windows. Este servidor también conocido como Lighty, es una alternativa para el Servidor de páginas Web Apache. Está diseñado para ser seguro, rápido, compatible con los estándares y flexible; a la vez que esta optimizado para entornos en los cuales la velocidad es crítica.

Su huella de memoria es muy pequeña (en comparación a otros servidores Web), una ligera carga en el CPU y su enfoque en velocidad hacen de lighttpd perfecto para servidores con demasiada carga. Este servidor Web es otro de los más ligeros que hay en el mercado. Está especialmente pensado para hacer cargas pesadas sin perder balance, utilizando poca RAM y poca de CPU. Algunas páginas populares que lo usan son Youtube, Wikipedia y otras que soportan gran tráfico diariamente. También es gratuito y se distribuye bajo Licencia BSD (López, 2007).

Ngnix

Es un servidor http y proxy inverso gratuito, de código abierto y de alto rendimiento, además de ser servidor proxy para IMAP y POP3.Este servidor está actualmente manejando entre el 1% y el 4% de todos los dominios globales. Nginx es conocido por su estabilidad, su gran conjunto de características, una configuración sencilla y por consumir pocos recursos. Como este servidor no provee actualmente

de un adaptador directo para Merb, se requiere utilizar un proxy reverso en Nginx con el fin de direccionar peticiones hacia uno o varios procesos Merb distintos. Esto puede ejecutarse con cualquier Servidor de aplicaciones basado en Rack, como Mongrel, Thin, Ebb o Glassfish (ServidoresAdmin, 2015).

Servidor web seleccionado

Se seleccionó el servidor web Apache por su configurabilidad y robustez, por la posibilidad de estar en una multitud de Sistemas Operativos y tener una gran variedad de módulos que se van instalando a medida que se necesitan; además de que es una tecnología gratuita de código fuente abierto y permite el trabajo con gran cantidad de lenguajes como: Perl, PHP y otros lenguajes de script, Java y páginas JSP, teniendo todo el soporte que se necesita para tener páginas dinámicas. También se optó por este servidor por las facilidades que brinda con la creación y gestión de logs, permitiendo tener el control máximo sobre todo lo que pasa en el servidor; además de que es uno de los más usados, como se puede observar en los anexos en la fig. A.3.

1.4.4 Entornos integrados de desarrollo (IDE, por sus siglas en inglés)

NetBeans

Programa que sirve como IDE que permite programar en distintos lenguajes, es ideal para trabajar con el lenguaje de desarrollo JAVA (y todos sus derivados), además ofrece un excelente entorno para programar en PHP. La IDE de NetBeans es perfecta y muy cómoda para los programadores. Tiene un excelente balance, entre una interfaz con variadas opciones y un aceptable completamiento de código. Es gratis y de código libre (Open Source), cuenta con una comunidad de desarrolladores alrededor de todo el mundo. Existen versiones para Linux, Mac y Windows (Estrada, 2012) (Boudreau, y otros, 2003).

Eclipse

Entorno de desarrollo integrado de código abierto multiplataforma para desarrollar proyectos. Esta plataforma ha sido usada para desarrollar entornos de desarrollo integrados, como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). También se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus. En Eclipse se pueden usar diferentes lenguajes de programación como: Java, ANCI C, C++, JSP, sh, perl, php, sed (Chen, y otros, 2005).

PHPStorm

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 súper rápido teniendo en cuenta su tamaño, soporta muchos idiomas y marcos; es multiplataforma (Packt Publishing, 2013).

IDE seleccionado

Se determinó emplear el PhpStorm, dado que este IDE soporta múltiples sistemas operativos, es rápido y fácil de usar, presenta todas las funciones IDE de desarrollo web PHP, JS, HTML y editor de CSS, permite el completamiento de código, presenta una valiosa documentación y se puede ampliar con un cúmulo de plugins útiles; además de que se ubica entre los mejores IDEs PHP de la actualidad, como se puede observar en los anexos en la fig. A.4, y tiene soporte para varios marcos de trabajo como: ZendFramework, Yii, Symfony, entre otros.

1.4.5 Herramienta de modelado

Visual Paradigm

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).

1.4.6 Marcos de trabajo

CodeIgniter

Es un marco de trabajo de aplicaciones web de código abierto para ayudar a escribir los programas de PHP. El objetivo de la aplicación es ayudar a los desarrolladores a culminar más rápido los proyectos. Esto se logra ofreciendo un amplio conjunto de bibliotecas para tareas comúnmente necesarias, así como una interfaz sencilla y estructura lógica para acceder a estas bibliotecas. CodeIgniter se basa libremente en el patrón de desarrollo Modelo-Vista-Controlador (MVC) popular. CodeIgniter a menudo se destaca por su velocidad en comparación con otros marcos de trabajo PHP (Upton, 2007).

CakePHP

Es un flexible y rápido marco de trabajo, basado en el patrón modelo-vista-controlador y de código abierto para aplicaciones PHP, inspirado en Ruby. CakePHP utiliza comúnmente conocido patrones de diseño como Active Record, Asociación Asignación de datos, Front Controller y MVC. El objetivo principal es proporcionar un marco estructurado que permite a los usuarios de PHP, en todos los niveles, desarrollar rápidamente aplicaciones web robustas, sin ninguna pérdida de flexibilidad (Golding, 2008).

Zend Framework

Es un marco de trabajo simple, sencillo, orientado a objetos y de código abierto, para PHP 5 diseñado para eliminar los tediosos detalles de codificación y permitir centrarse en los puntos principales. Este marco proporciona funcionalidad para el 80% del código de aplicación que es común en muchas aplicaciones, por lo que el usuario puede centrarse en la personalización del otro 20% de sus aplicaciones para satisfacer sus necesidades de negocio.
Uno de sus puntos fuertes es el diseño altamente modular Modelo-Vista-Controlador (MVC), lo que hace el código más reusable y fácil de mantener (Padilla, 2009).

Symfony

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 (symfony.es, 2007).

Principales características:

- Su código, y el de todos los componentes y librerías que incluye, se publican bajo la licencia MIT de software libre.
- 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.
- Los componentes de Symfony son tan útiles y están tan probados, que proyectos tan gigantescos como Drupal 8 están construidos con ellos (symfony.es, 2007) (Eguiluz, 2013).

Marco de trabajo seleccionado

Se escogió el marco de trabajo Symfony 2, ya que este mejora las funcionalidades orientadas a objetos encontradas en PHP 5, sigue el paradigma Modelo-Vista-Controlador (MVC) y utiliza ORM para interactuar con la base de datos; además de que utiliza el lenguaje PHP y las plantillas twig, permite el almacenamiento en caché y la validación de formularios. También se optó por este marco porque presenta una valiosa documentación, es orientado a componentes (los que pueden ser reutilizados) y se encuentra entre los marcos de trabajo más utilizados en el desarrollo web, como se puede apreciar en los anexos en la fig. A.5.

1.4.7 Código de salida

EXT JS

Es un conjunto de librerías JavaScript que permite el desarrollo de aplicaciones RIA basadas en un navegador. EXT JS ofrece al desarrollador un conjunto de widgets (componentes como por ejemplo grids, ventanas de dialogo, etc) plenamente integrados y un API (Application Program Interface) para conseguir interfaces web más dinámicas e interactivas con el usuario. Usa el lenguaje JavaScript junto con HTML para la creación de las interfaces de usuario, así como para el manejo de eventos en cada una de las páginas que comprende una aplicación desarrollada con EXT JS. Permite la orientación a objetos, la manipulación del DOM y el soporte de navegadores como Internet Explorer, Opera, Safari,

Mozilla Firefox, entre otros. Este es usado en importantes organizaciones, como se evidencia en los anexos en la fig. A.6 (Correa Lozano, 2010).

1.4.8 Resultado del análisis de las tecnologías y herramientas

Después del análisis realizado se decidió emplear la variación de la metodología AUP-UCI, como metodología de desarrollo, PHP v5.4.3 como lenguaje de programación, Apache v.2.4 para utilizarlo como servidor web; además de usar PhpStorm v.8.0 como entorno integrado de desarrollo, Symfony v2.3 como marco de trabajo, Visual Paradigm v.8.0 para el modelado de los diagramas y teniendo EXT JS v.4.2.1 como código fuente de salida.

1.5 Conclusiones del capítulo

Después de haber analizado y realizado un estudio del arte, se llegó a la conclusión de que con el uso de un generador de código se puede solucionar el problema presentado; pero que es preciso desarrollar una nueva aplicación, ya que ninguna de las herramientas estudiadas responde a las necesidades de disminuir el tiempo de desarrollo de softwares en la universidad en las aplicaciones web que utilicen la librería Ext JS 4 para la creación de sus interfaces; dado que estas son propietarias y con elevados precios en sus licencias de desarrollo, el código que generan no siempre es de fácil comprensión por parte de los desarrolladores, no cuentan con la posibilidad de definir nuevos mecanismos de generación de código y/o no brindan la posibilidad de generar interfaces para probar el buen funcionamiento del código generado. Para darle solución al problema se implementará un generador de código JavaScript de tipo pasivo, aplicando la técnica de generación por plantillas.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

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 debe poseer el sistema a desarrollar; además se enumeran los requisitos funcionales y no funcionales con los que debe cumplir el mismo, así como su diseño de clases.

2.2 Propuesta de solución

Con la generación de la nueva herramienta se disminuirá el tiempo de desarrollo de interfaces gráficas en Ext JS 4 para aplicaciones web de gestión, para luego aplicarla en los proyectos productivos del centro y así hacer más extensible el uso de la misma. La misma tomará como entrada una entidad generada por Doctrine 2.0 y realizará las transformaciones necesarias (obtener los atributos y tipos de datos de la entidad) para generar el código fuente de las interfaces de un CRUD en EXT JS 4.

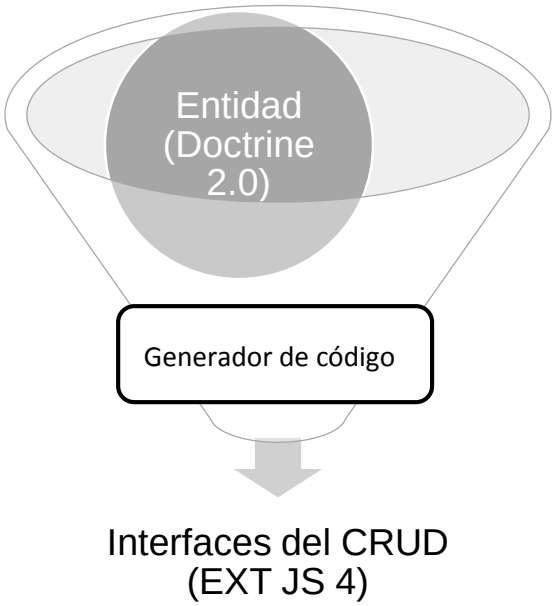

Fig. 2.1 Entrada y salida del componente generador de código.

2.3 Funcionamiento del generador de código

Para poder utilizar el generador de código fuente una vez desarrollado es necesario seguir una secuencia de pasos:

- Copiar el componente desarrollado (GeneradorEXTJSBundle), en el directorio origen del sistema en el cual se quiera generar las interfaces del CRUD en EXT JS 4.
- Buscar todos los directorios que sean componentes y contengan al menos una entidad dentro del sistema.
- Buscar todas las entidades para cada uno de los componentes que sean seleccionados.
- Obtener los atributos y tipos de datos de cada una de las entidades elegidas.
- Generar el CRUD en EXT JS 4, con los valores obtenidos.

En la fig. 2.2 se muestra la estructura de una entidad generada por Doctrine 2.0, en la cual los nombres de los atributos y sus tipos de datos son mostrados mediante anotaciones; como se puede apreciar por ejemplo en la línea resaltada.

```php
<?php
namespace UCI\Boson\SeguridadBundle\Entity;
use Doctrine\ORM\Mapping as ORM;
/**
 * User
 * @ORM\Table(name="seg_usuario")
 * @ORM\Entity(repositoryClass="UCI\Boson\SeguridadBundle\Entity\UsuarioRepository")
 */
class Usuario extends BaseUser
{
    /**
     * @var integer
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;
      /**
     * Set dominio
     * @param integer $dominio
     * @return Usuario
     */
    public function setDominio($dominio)
    {
        $this->dominio = $dominio;

        return $this;
    }
```

Fig. 2.2 Estructura de entidad generada por Doctrine 2.0.

Para poder obtener los datos (atributos y sus tipos de datos) de la entidad es necesario transformar la misma en una cadena de texto y descomponerla en tokens, los cuales serán interpretados y llevados a valores entendibles por el usuario.

Cuando se obtienen los datos de la entidad, se configuran los parámetros del CRUD (alias de los atributos y expresiones regulares), los cuales son opcionales, y se manda a generar las interfaces del CRUD en EXT JS 4. En la fig. 2.3 se puede observar un ejemplo de cómo quedarían las interfaces.

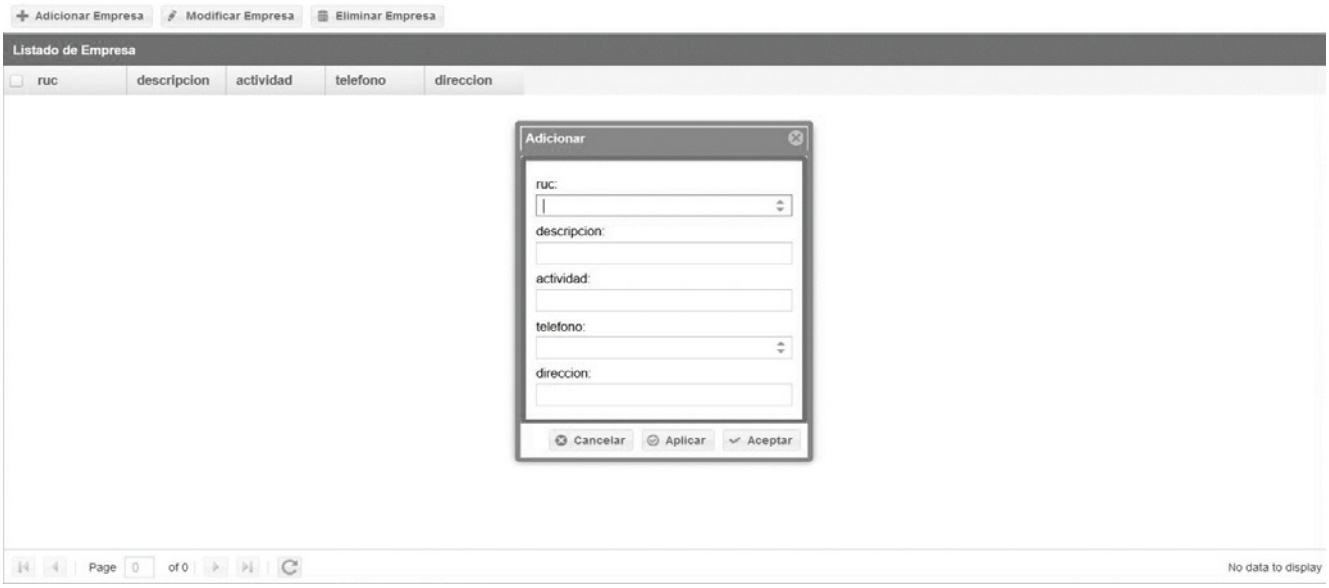

Fig. 2.3 Interfaces desarrolladas por el generador de código.

Estas interfaces serán guardadas en la estructura de carpetas que requiere el marco de trabajo EXT JS en la versión 4 o superior (estructura MVC), donde la carpeta raíz será el nombre que se le haya asignado a la entidad y se desglosará de la siguiente forma: un archivo principal “app.js” y cuatro carpetas: “controller”, “model”, “store” y “view”, en las cuales se guardarán los controladores, modelos, almacenes y vistas respectivamente; como se observa en la fig. A.7.

2.4 Modelo Conceptual

Una vez analizada la propuesta de solución, se determinó como modelo conceptual a utilizar, el mostrado a continuación en la fig. 2.4.

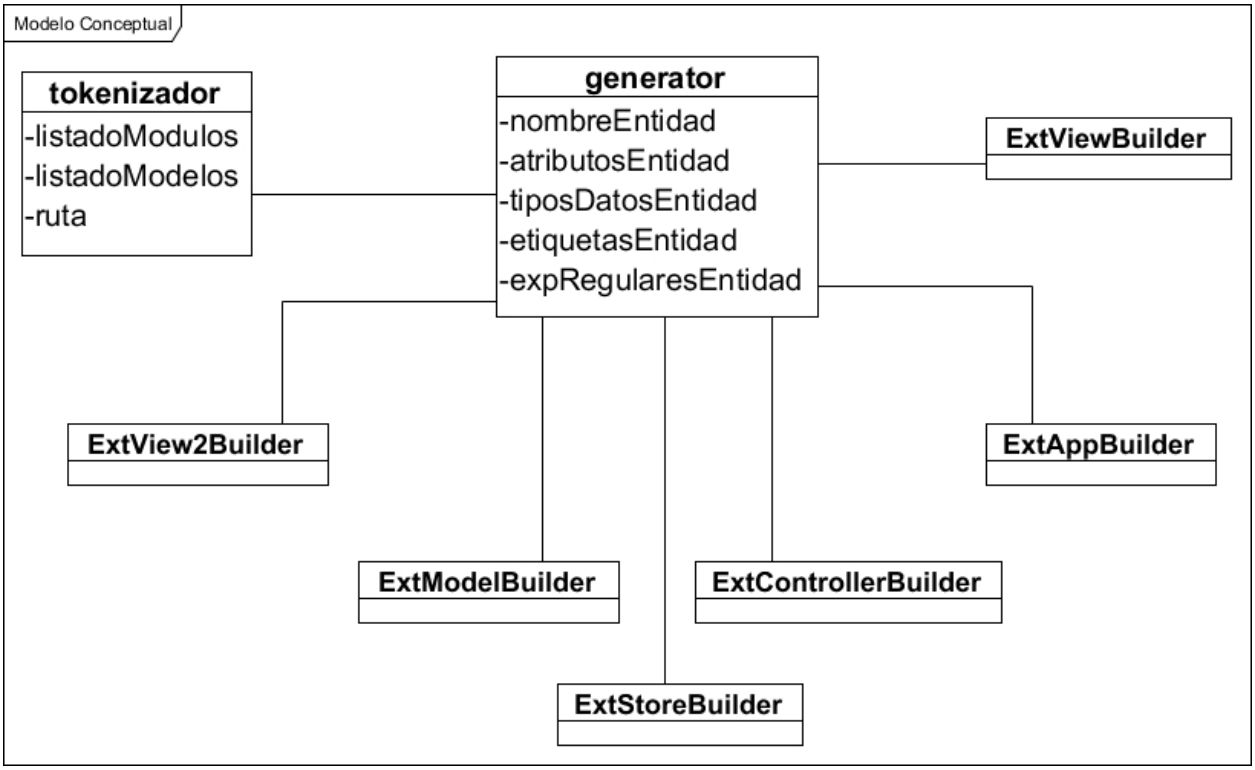

Fig. 2.4 Modelo conceptual del generador de código.

Tokenizador: se encarga de transformar la entidad en tokens y luego convertirla en datos entendibles al usuario; además de buscar los componentes y entidades del sistema.

Generator: a partir de los datos que se le envían, manda a ejecutar la generación del código para el CRUD en EXT JS 4.

ExtModelBuilder: permite implementar la plantilla ExtModelBuilder.js.twig

ExtViewBuilder: permite implementar la plantilla ExtViewBuilder.js.twig

ExtAppBuilder: permite implementar la plantilla ExtAppBuilder.js.twig

ExtControllerBuilder: permite implementar la plantilla ExtControllerBuilder.js.twig

ExtStoreBuilder: permite implementar la plantilla ExtStoreBuilder.js.twig

ExtView2Builder: permite implementar la plantilla ExtView2Builder.js.twig

2.5 Requisitos del software

Como técnicas para el proceso de captura de requisitos del sistema fueron empleadas la entrevista y la tormenta de ideas, donde los implicados fueron:

Analista del departamento de Componente la Ing. Katia Sarial Preval.

Los estudiantes, analista y desarrollador, involucrados en el desarrollo de la herramienta.

2.5.1 Requisitos funcionales

A continuación se representan los requisitos funcionales del sistema expresados en lenguaje natural. Los mismos serán identificados con las siglas RF-más el número del requisito (Ej. RF-1.), clasificados según su prioridad en alta, media o baja.

Identificador	Nombre	Prioridad	Descripción
RF-1	Buscar los componentes	Alta	Permite al desarrollador buscar a partir del directorio raíz donde se encuentra el proyecto, todos los componentes del sistema.
RF-2	Listar los componentes	Alta	Permite visualizar todos los componentes en una vista, mediante una lista desplegable.
RF-3	Buscar las entidades del componente	Alta	Permite al desarrollador buscar las entidades asociadas a cada componente.
RF-4	Listar las entidades	Alta	Permite visualizar las entidades de cada componente en una vista, mediante una lista desplegable.
RF-5	Descomponer la entidad	Alta	Permite al desarrollador

	en tokens		descomponer en tokens cada una de las entidades.
Identificador	**Nombre**	**Prioridad**	**Descripción**
RF-6	Obtener datos	Alta	Permite al desarrollador obtener los datos a partir de los tokens generados anteriormente.
RF-7	Generar CRUD	Alta	Permite al desarrollador generar el código fuente en EXT JS 4 de un CRUD a partir de las configuraciones y parámetros (atributos y tipos de datos) entrados por el usuario, así como también visualizar el código a través de la estructura de carpeta que se crea.
RF-8	Visualizar interfaces del CRUD	Media	Permite al desarrollador visualizar las interfaces del CRUD desarrollado en EXT JS 4.
RF-9	Renderizar a la vista principal	Baja	Permite al usuario volver a repetir el proceso, dándole la opción de retornar a la vista principal.

Tabla 2.1 Descripción de los requisitos funcionales.

2.5.2 Validación de requisitos

Para la validación de estos requisitos fueron empleadas las técnicas de:

- **Prototipos**: Algunas propuestas se basan en obtener de la definición de requisitos prototipos

que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario (Olsina, y otros, 1999). Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final (Pressman, 2006).

- **Revisión técnica formal**: La revisión de las especificaciones de los requerimientos del proceso de generación de código fuente fue realizada con la analista principal del proyecto. Con la utilización de esta técnica se valida que no existan errores en el contenido o malas interpretaciones, información incompleta, inconsistencias y que los requisitos no sean contradictorios, imposibles o inalcanzables (Pressman, 2006), dando como resultado que fueran aprobados los que estaban descritos de forma correcta, clara y consistente.
- **Generación de casos de prueba (test de requisitos)**: Fueron definidos y diseñados casos de pruebas para cada requerimiento especificado, con el objetivo de verificar el cumplimiento de los mismos. La utilización de esta técnica ofreció los siguientes resultados: fueron identificados los posibles escenarios de los requisitos, así como los juegos de datos de los campos determinados en estos, se validaron y aprobaron los que estaban bien enunciados, descritos y consistentes (Pressman, 2006).

2.5.3 Historias de usuario

De cada uno de estos requisitos funcionales identificados fueron diseñadas sus historias de usuario. A continuación se muestra en la tabla 2.2, el ejemplo de la historia de usuario (HU) del requisito “Buscar los componentes”, el resto de ellas serán mostradas en los anexos.

Número: 1.1	**Nombre del requisito:** Buscar los componentes
Programador: Efraín Francisco Ruiz Zamora	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Medio	**Tiempo Real:** 8 horas

Descripción: Permite al desarrollador buscar a partir del directorio raíz donde se encuentra el proyecto, todos los componentes que contienen al menos una entidad.
Observaciones: N/A
Prototipo de interfaz: N/A

Tabla 2.2 HU del requisito “Buscar los componentes”.

2.5.4 Requisitos no funcionales

A continuación se representan los requisitos no funcionales del sistema expresados en lenguaje natural. Los mismos serán identificados con las siglas RNF-más el número del requisito.

Identificador	Clasificación	Requisito
RNF-1	Software	El sistema se desarrollará con tecnología PHP versión 5.4.3 Se utilizará un servidor Web Apache 2.0. En las computadoras de los clientes solo se requiere de un navegador.
RNF-2	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-3	Hardware	Para el cliente los requerimientos mínimos deben ser una computadora Pentium III a 1.0 GHz con 512 Mb de memoria RAM, además de una tarjeta de red.
RNF-4	Interfaz	La interfaz de la aplicación a desarrollar debe ser sencilla para reducir el tiempo de capacitación de los usuarios.
RNF-5	Portabilidad	El sistema funcionará sobre las plataformas Windows y Linux.

Tabla 2.3 Descripción de los requisitos no funcionales.

2.6 Arquitectura del software

La arquitectura orientada a componentes (SCA por sus siglas en inglés) es una especificación que define cómo crear y ensamblar los diversos componentes de negocio como componentes modulares, con la finalidad de incrementar la flexibilidad y facilidad de mantenimiento de los sistemas de información (Pérez, 2012).

2.6.1 Principios Fundamentales

- **Reusable**: los componentes son usualmente diseñados para ser utilizados en escenarios diferentes por diferentes aplicaciones, sin embargo, algunos componentes pueden ser diseñados para tareas específicas.
- **Sin contexto específico**: los componentes son diseñados para operar en diferentes ambientes y contextos. Información específica como el estado de los datos deben ser pasadas al componente en vez de incluirlos o permitir al componente acceder a ellos.
- **Extensible**: un componente puede ser extendido desde otro existente para crear un nuevo comportamiento.
- **Encapsulado**: los componentes exponen interfaces que permiten al programa usar su funcionalidad. Sin revelar detalles internos, detalles del proceso o estado.
- **Independiente**: los componentes están diseñados para tener una dependencia mínima de otros componentes. Por lo tanto los componentes pueden ser instalados en el ambiente adecuado sin afectar otros componentes o sistemas (Hong, y otros, 2004).

2.6.2 Beneficios

- **Facilidad de instalación**: cuando una nueva versión esté disponible, se podrá reemplazar la versión existente sin impacto en otros componentes o el sistema como un todo.
- **Costos reducidos**: el uso de componentes de terceros permite distribuir el costo del desarrollo y del mantenimiento.
- **Facilidad de desarrollo**: los componentes implementan una interfaz bien definida para proveer la funcionalidad determinada, permitiendo el desarrollo sin impactar otras partes del sistema.

- **Reusable**: el uso de componentes reutilizables, significa que puedan ser usados para distribuir el desarrollo y el mantenimiento entre múltiples aplicaciones y sistemas.
- **Mitigación de complejidad técnica**: los componentes mitigan la complejidad por medio del uso de contenedores de componentes y sus servicios (Hong, y otros, 2004).

Dada las facilidades antes expuesta de la arquitectura y dado que el marco de trabajo Symfony 2 es orientado a componentes, se decidió utilizar la arquitectura orientada a componentes para el desarrollo de este trabajo.

2.7 Modelo del diseño

Una vez seleccionada la arquitectura y a través del análisis de la propuesta de solución, se determinó que el diagrama de clases que podía darle respuesta a la problemática planteada sería el que se muestra a continuación en la fig. 2.5.

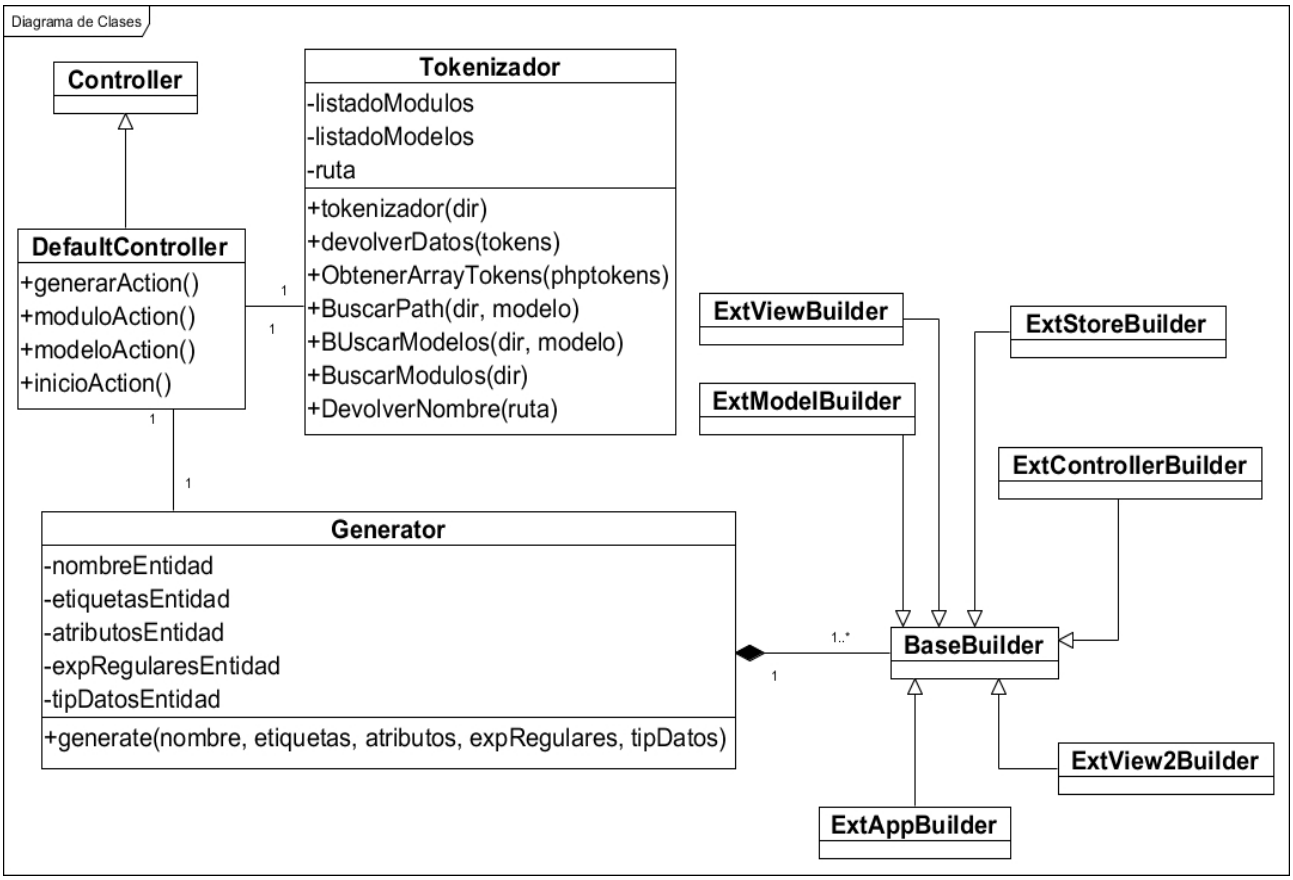

Fig. 2.5 Diagrama de clases del generador de código.

2.8 Patrones de Diseño

2.8.1 GRASP

En diseño orientado a objetos, GRASP es el acrónimo de "General Responsibility Assignment Software Patterns", en español “Patrones de Software de Asignación de Responsabilidades Generales”. Aunque se considera que más que patrones propiamente dichos, son una serie de "Buenas Prácticas" de aplicación recomendable en el diseño de software (Larman, 2003).

2.8.1.1 Experto en información

El patrón experto en información es el principio básico de asignación de responsabilidades. Este patrón indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce la información necesaria para crearlo (Larman, 2003).

2.8.1.2 Creador

El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que:

- Tiene la información necesaria para realizar la creación del objeto, o
- Usa directamente las instancias creadas del objeto, o
- Almacena o maneja varias instancias de la clase (Larman, 2003).

2.8.1.3 Controlador

El patrón controlador sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado (Larman, 2003).

2.8.1.4 Alta cohesión

Expresa que la información que almacena una clase debe de ser coherente y está en la mayor medida de lo posible relacionada con la clase (Larman, 2003).

2.8.1.5 Bajo acoplamiento

Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el

resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases (Larman, 2003).

2.9 Métricas para validar el diseño

Lorenz y Kidd dividen las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones para una clase individual, y promedian los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones en la jerarquía de clases. Las métricas para valores internos de clase, examinan la cohesión y asuntos relacionados con el código, y las métricas orientadas a valores externos examinan el acoplamiento y la reutilización (Pressman, 2006).

Del conjunto de métricas planteadas por Lorenz y Kidd, las que se utilizaron para validar el diseño propuesto son las siguientes:

- **Tamaño operacional de la clase (TOC).**
- **Relaciones entre clases (RC).**

Atributos de calidad que se abarcan:

- **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.
- **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.

2.9.1 Métrica Tamaño Operacional de Clase (TOC)

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.

Tabla 2.4 Métricas para el tamaño operacional de la clase.

2.9.1.1 Criterio de evaluación de la métrica TOC

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

Tabla 2.5 Criterio de evaluación de las métricas TOC.

2.9.1.2 Instrumento de evaluación de la métrica TOC.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
DefaultController	4	Alta	Alta	Baja
Tokenizador	8	Alta	Alta	Baja
Generator	2	Media	Media	Media
ExtViewBuilder	0	Baja	Baja	Alta
ExtModelBuilder	0	Baja	Baja	Alta
ExtStoreBuilder	0	Baja	Baja	Alta
ExtControllerBuilder	0	Baja	Baja	Alta
ExtView2Builder	0	Baja	Baja	Alta
ExtAppBuilder	0	Baja	Baja	Alta

Tabla 2.6 Instrumento de evaluación de las métricas TOC.

$$Promedio = \frac{\sum_{i=1}^{n} CantProced_i}{n}$$

$CantProced_i$ → cantidad de procedimientos de la clase i

n → total de clases

Promedio = 1.56

2.9.1.3 Resultados de la aplicación de la métrica TOC

En el gráfico de la fig. 2.6 se puede observar el resultado de la evaluación de los atributos de la métrica TOC.

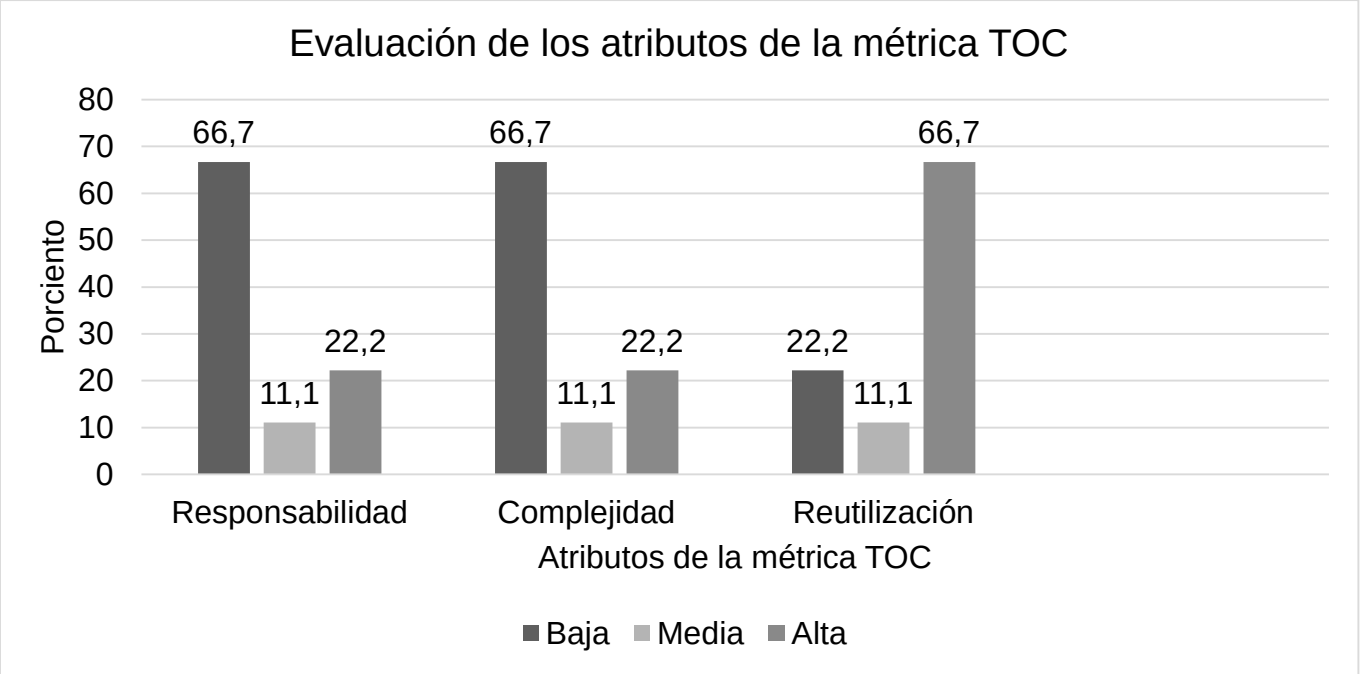

Fig. 2.6 Evaluación de los atributos de la métrica TOC.

Haciendo 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 del generador de código tiene una buena calidad; teniendo en cuenta que se obtuvieron resultados satisfactorios en los diferentes atributos de calidad medidos, estos resultados van unidos al hecho de que el 89% de las 9 clases medidas (8 clases) tienen 4 o menos operaciones. Por lo que se puede afirmar que se obtuvo una solución eficiente con altos niveles de reutilización (66,7%) lo que garantiza 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 (66,7%) y complejidad de implementación (66,7%), pues se le asignaron correctamente las responsabilidades a las clases involucradas en la solución.

2.9.2 Métrica Relación entre Clases (RC)

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.

Tabla 2.7 Métricas de relación entre clases (RC).

2.9.2.1 Criterio de evaluación de la métrica RC

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
	Baja	<=Promedio

Cantidad de pruebas	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Atributo	**Categoría**	**Criterio**
Reutilización	Baja	> 2* Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	<=Promedio

Tabla 2.8 Criterios de evaluación de las métricas RC.

2.9.2.2 Instrumento de evaluación de la métrica RC.

Clase	Cantidad de relaciones de uso	Acoplamiento	Complejidad del mantenimiento	Cantidad de pruebas	Reutilización
DefaultController	2	Medio	Alta	Alta	Baja
Tokenizador	0	Ninguno	Baja	Baja	Alta
Generator	6	Alto	Alta	Alta	Baja
ExtViewBuilder	0	Ninguno	Baja	Baja	Alta
ExtModelBuilder	0	Ninguno	Baja	Baja	Alta
ExtStoreBuilder	0	Ninguno	Baja	Baja	Alta
ExtControllerBuilder	0	Ninguno	Baja	Baja	Alta
ExtView2Builder	0	Ninguno	Baja	Baja	Alta
ExtAppBuilder	0	Ninguno	Baja	Baja	Alta

Tabla 2.9 Instrumento de evaluación de las métricas RC.

$$Promedio = \frac{\sum_{i=1}^{n} CantRelUso_i}{n}$$

$CantRelUso_i$ → cantidad de relaciones de uso de la clase i

n → total de clases

Promedio = 0.89

2.9.2.3 Resultados de la aplicación de la métrica RC

En el gráfico de la fig. 2.7 se puede observar el resultado de la evaluación de los atributos de la métrica RC.

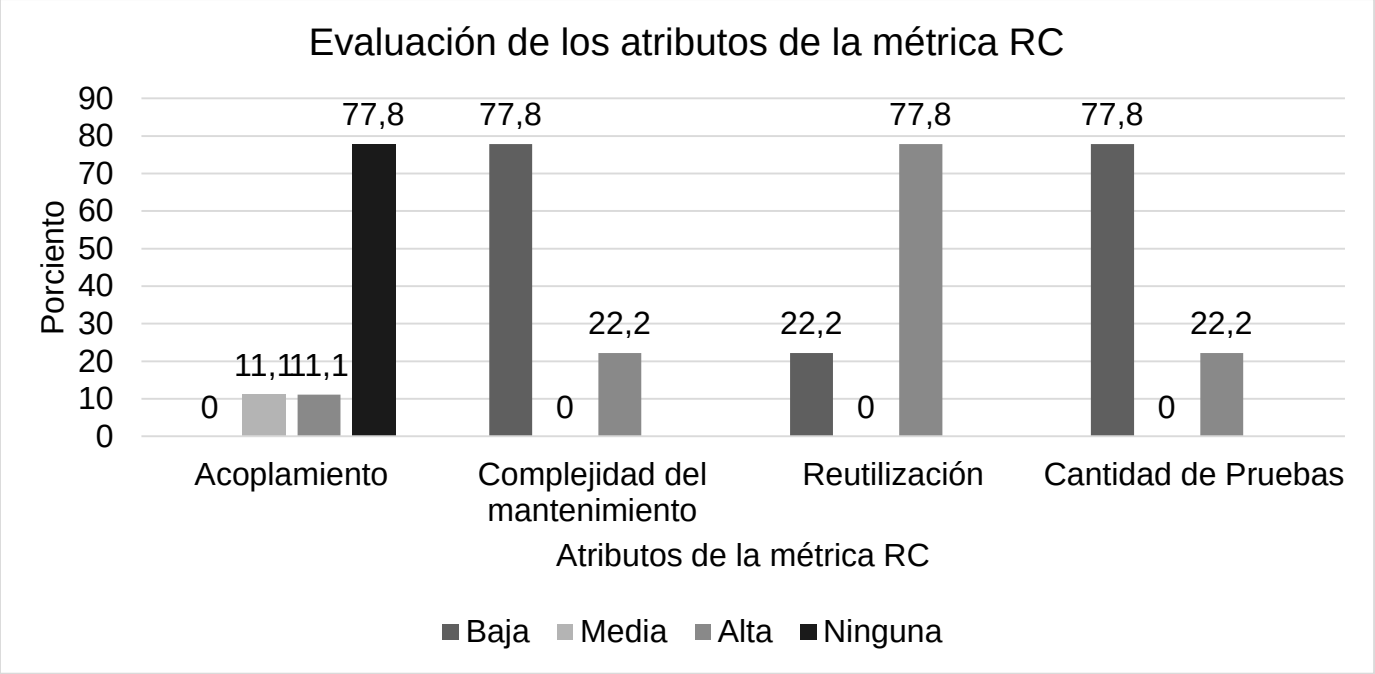

Fig. 2.7 Evaluación de los atributos de la métrica RC.

Haciendo 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 del generador de código tiene una buena calidad teniendo en cuenta que el 88,9% de las clases incluidas en el sistema posee 2 o menos dependencias de otras clases. Por lo que al ocurrir un cambio de alguna de las clases, la afectación en las restantes sea de poca importancia. El 77,8% de las clases poseen acoplamiento ninguno, 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 77,8% de las clases con índices bajos, 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 77,8% de baja cantidad de pruebas fomentando así un alto índice de reutilización con un 77,8%.

2.9.3 Matriz de inferencia de indicadores de calidad

La matriz inferencia de indicadores de calidad o matriz de cubrimiento, es una representación estructurada de los atributos de calidad y métricas utilizadas para evaluar la calidad del diseño propuesto. Esta matriz permite conocer si los resultados obtenidos de las relaciones atributo/métrica es positivo o no, llevando estos resultados a una escala numérica donde, si los resultados son positivos se le asigna el valor de 1, si son negativos toma valor 0 y si no existe relación es considerada como nula y es representada con un guion simple (-). Luego se puede obtener un resultado general para cada atributo promediando todas sus relaciones no nulas.

2.9.3.1 Resultados de la evaluación de la relación atributo/métrica.

Atributos\Métricas	TOC	RC	Promedio
Responsabilidad	1	(-)	1
Complejidad de implementación	1	(-)	1
Reutilización	1	1	1
Acoplamiento	(-)	1	1
Complejidad de Mantenimiento	(-)	1	1
Cantidad de Pruebas	(-)	1	1

Tabla 2.10 Resultados de la evaluación de la relación atributo/métrica.

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica TOC y RC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que en la matriz de cubrimiento todos los atributos de calidad dieron 1 como resultado.

2.10 Validación de entrada y salida del componente

La entrada del generador de código es una entidad generada por Doctrine 2.0, que es un proceso que se realiza de forma automática; pero para evitar errores que pudiera presentar la misma o modificaciones por parte de terceros, luego de descomponerla en tokens mediante el componente desarrollado, se realiza un análisis léxico y sintáctico para comprobar que tiene la estructura correcta y no presenta inconformidades en los datos.

La salida de este componente es el código fuente de las interfaces de un CRUD desarrollado en EXT JS 4; para la validación de la salida, el sistema muestra la estructura de carpetas obtenida una vez concluido el proceso de generación, así como el código de cada fichero generado, para que el usuario pueda verificar que no existieron errores en el proceso; además le da la opción al usuario de comprobar que todo está correcto mediante la visualización de las interfaces obtenidas, en caso de que este no domine tanto el marco de trabajo EXT JS como para encontrar inconformidades mediante la comprobación del código, ya que en caso de que existan errores durante el proceso, no se visualizan las interfaces.

2.11 Conclusiones del capítulo

En este capítulo se han abordado aspectos relacionados con la propuesta de solución, la cual es, implementar una herramienta que genere código JavaScript para construir interfaces gráficas en Ext JS 4 de un CRUD para aplicaciones web de gestión, con el objetivo de disminuir el tiempo de desarrollo de estas interfaces en aquellos marcos de trabajo que utilicen Doctrine 2.0 para generar sus entidades; también se identificaron los requerimientos del sistema, obteniéndose así una serie de requisitos funcionales y no funcionales a tener en cuenta para el desarrollo de la herramienta y se presentaron algunas evidencias de los artefactos que requiere la metodología como: historias de usuarios, el modelo conceptual y el diagrama de clases correspondiente; además de explicar cómo se podrá comprobar que no existieron errores tanto en las entrada como las salidas del componente y validar el diseño de las clases presentado mediante las métricas del diseño: TOC y RC.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

En el presente capítulo se describe los tipos de pruebas que se le van a realizar a la aplicación, para comprobar su correcto funcionamiento, así como los estándares de codificación que presenta la herramienta desarrollada.

3.2 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 utilizarán 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).

3.2.1 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 empleará notación PascalCasing. Ejemplo: Generador.

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

DefaultController.

3.2.2 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 empleará 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: generarAction.

3.2.3 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 empleará notación CamelCasing. Ejemplo: $entidad.

3.2.4 Normas de comentariado

Se debe comentar todo lo que se haga dentro del desarrollo, establecer las pautas que conlleven a lograr un código más legible y reutilizable y así se pueda aumentar su mantenimiento a lo largo del tiempo.

3.2.5 Estilo del código

En la implementación, al escribir las sentencias en php, se utilizarán los tabs del lenguaje como se muestra en la fig. 3.1.

```
45      <?php
46
47       //codigo
48
49      ?>
50
```

Fig. 3.1 Estilo del código.

Sangría o indexado: la política de sangría a utilizar en la implementación es por tab. Como se muestra en la fig. 3.2.

```
126     public function resultadoAction(){
127         if (isset($_POST['recomenzar'])) {
128             $tokenizador = new Tokenizador();
129             $dirweb = str_replace("web/app_dev.php", "src", $_SERVER['SCRIPT_FILENAME']);
130             $listadoModulos = $tokenizador->buscarModulos($dirweb);
131 
132             return $this->render('GeneradorEXTJSBundle:Default:principal.html.twig', array(
133                 'listadoModulos' => $listadoModulos
134             ));
135         }
136         if (isset($_POST['ver_interfaces'])) {
137             $nombreEntidad = $this->getRequest()->get('entidad');
138                     return $this->render('GeneradorEXTJSBundle:Default:Final.html.twig', array(
139                 'nombreEntidad' => $nombreEntidad
140             ));
141         }
142     }
```

Fig. 3.2 Estilo del código: sangría o indexado.

Brazas o llaves: en la declaración de clases o interfaces, métodos, bloques y switch, la apertura de llaves se hace en la misma línea. Un ejemplo donde se puede observar esto es el que se muestra a continuación en la fig. 3.3.

```
13  class Generator {
14 
15      public function generate($nombreEntidad, $etiquetasEntidad, $atributosEntidad, $expRegulatesEntidad, $datosEntidad,$ancho,$alto){...}
86 
87      public function ObtenerEntidad($string) {
88          $string = strrev($string);
89          $pos = strpos($string, 'y');
90          $string = substr($string, 0, $pos);
91          $string = strrev($string);
92          return $string;
93      }
94  }
95 
```

Fig. 3.3 Estilo del código: brazas o llaves.

3.3 Pruebas unitarias

Al desarrollar un nuevo software o sistema de información, la primera etapa de pruebas a considerar es la etapa de pruebas unitarias. En la cual se encuentran presentes las pruebas de caja negra y de caja blanca, en la primera de ellas se realiza un análisis de los datos de entrada y de salida, y en la segunda se analiza el proceso interno del sistema para evaluar las inconsistencias que pueda estar presentando el sistema, para así llegar a una corrección de los mismos y proseguir con la nueva fase del proceso de desarrollo del sistema. Podemos destacar que las pruebas unitarias son una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado (Rodriguez, 2006).

El objetivo fundamental de las pruebas unitarias es asegurar el correcto funcionamiento de las interfaces, o flujo de datos entre componentes de manera tal que a la hora de realizar una unificación de los diferentes componentes que conforman el sistema en general, exista una congruencia que favorezca el desarrollo de la aplicación que se quiere realizar. Las pruebas unitarias aseguran que un único componente de la aplicación produce una salida correcta para una determinada entrada. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular (Potencier, y otros, 2010).

3.3.1 Pruebas de caja blanca

Las pruebas de caja blanca garantizan que se ejerciten por lo menos una vez todos los caminos independientes del código, así como la ejecución de todos los bucles en sus límites operacionales y todas las decisiones lógicas en las vertientes verdaderas y falsas (Pressman, 2003).

Para aplicar las pruebas de caja blanca se empleó la técnica del camino básico. Esta permitió obtener una medida de la complejidad lógica para el diseño de los casos de prueba de caja blanca y usar dicha medida como guía para la definición de un conjunto básico de caminos de ejecución. Se tomó como ejemplo el método “devolverNombre”, perteneciente a la clase “Tokenizador” como base para realizar el procedimiento anteriormente descrito. A continuación en la fig. 3.4 se muestra este método.

```
public function devolverNombre($string)
    {
        $string = strrev($string);
        if ($_SERVER['SCRIPT_FILENAME'][0] == '/') {
            $pos = strpos($string, "/");
            $string = substr($string, $pos + 1);
            $pos = strpos($string, "/");
            $string = substr($string, 0, $pos);
            $string = strrev($string);

        } else {
            $pos = strpos($string, "\\");
            $string = substr($string, $pos + 1);
            $pos = strpos($string, "\\");
            $string = substr($string, 0, $pos);
            $string = strrev($string);
        }
        return $string;
    }
```

Fig. 3.4 Método utilizado para la técnica del camino básico.

Para aplicar la técnica del camino básico se realizaron una serie de pasos que a continuación se describen:

1. A partir del diseño o del código fuente, dibujar el grafo de flujo asociado, el cual está compuesto por los siguientes elementos:

- Nodos: son círculos que representan una o más sentencias procedimentales.
- Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.
- Regiones: son las áreas delimitadas por aristas y nodos.

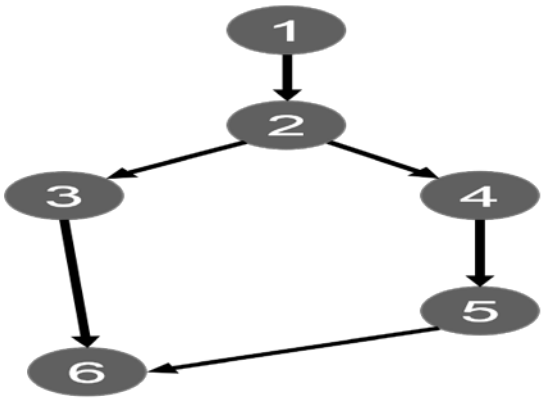

Fig. 3.5 Grafo de flujo asociado al camino básico.

2. Se calcula la complejidad ciclomática del grafo, la cual se calcula mediante tres formas:

- El número de regiones corresponde a la complejidad ciclomática. V(G) = 2
- La complejidad ciclomática, V(G), de una gráfica de flujo, G, se define como V(G)= E-N+2 donde E es el número de aristas, y N, el número de nodos de la gráfica de flujo.

  V (G)=E-N+2= 6-6+2 =2

- La complejidad ciclomática, V(G), de una gráfica de flujo G también se define como V(G)=P+1, donde P es el número de nodos predicados incluidos en la gráfica de flujo G.

  V (G) = P+1 = 1+1 = 2

3. Se determina un conjunto básico de caminos independientes, el valor de V(G) da el número de caminos linealmente independientes de la estructura de control del programa, por lo que se definen los siguientes 2 caminos:

Camino básico#1: 1-2-4-5-6

Camino básico#2: 1-2-3-6

4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico, cada camino independiente es un caso de prueba a realizar. En este caso se obtuvieron 2 caminos básicos, por tanto se hace necesario la confección de igual número de casos de prueba, para aplicar las pruebas a este método (Pressman, 2006).

A continuación se brinda un ejemplo de uno de ellos:

Caso de prueba: Camino básico #2	
Entrada	Ejecutar la aplicación en el sistema Linux, pasándole al método la ruta completa para llegar hasta la entidad.
Resultados esperados	Se devuelve el nombre que posee la entidad en el sistema.
Condiciones	$_SERVER['SCRIPT_FILENAME'][0] == '/'

Tabla 3.1 Caso de prueba asociado al camino básico #2.

Una vez aplicadas las pruebas de caja blanca mediante la técnica del camino básico al método "devolverNombre", se obtuvieron resultados satisfactorios.

Otra prueba de caja blanca realizada a la herramienta fue el empleo del marco de trabajo PHPUnit, que es una herramienta para la ejecución de pruebas al desarrollo de programas/sistemas desarrollados específicamente en PHP. Se trata de un miembro de la familia xUnit y proporciona un marco que realiza de manera fácil la escritura y la ejecución de las pruebas, así como el análisis de los resultados. Las pruebas son fáciles de aprender, de escribir y de ejecutar (Bergmann, 2001).

Los pasos básicos para escribir las pruebas en PHPUnit son los siguientes:

- Las pruebas de una clase Class van dentro de una clase ClassTest.
- ClassTest hereda (la mayor parte del tiempo) de PHPUnit_Framework_TestCase.
- Las pruebas son métodos públicos que no esperan parámetros y son nombrados test*.
- Dentro de los métodos de pruebas, la afirmación de métodos tales como assertEquals() se utilizan para afirmar que el valor real coincide con un valor esperado.

En la fig. 3.6 se expone un fragmento del código de la prueba realizada sobre la herramienta que dará una idea general de cómo utilizar PHPUnit para el desarrollo de este tipo de pruebas.

```
class TokenizadorTest extends \PHPUnit_Framework_TestCase
{
    public function testBuscarModulos()
    {
        $tokenizador = new Tokenizador();
        $resultados = $tokenizador->BuscarModulos('C:/wamp/www/RepoSymfony/Boson/src');
        $modulos = array(
            'ADTBundle',
            'GeneradorEXTJSBundle',
            'SeguridadBundle',
            'TrazasBundle'
        );
        $this->assertEquals($modulos, $resultados);
    }
```

Fig. 3.6 Fragmento de prueba realizada a la clase Tokenizador.

Al realizar las pruebas con la herramienta PHPUnit la respuesta del sistema teniendo en cuenta el consumo de tiempo y de recursos y la ejecución de los métodos fue la mostrada en la fig. 3.7 y fig. 3.8 respectivamente.

```
Time: 3.08 minutes, Memory: 12.00Mb

OK (5 tests, 5 assertions)
```

Fig. 3.7 Consumo de tiempo y recursos.

```
ok 1 - UCI\Boson\GeneradorEXTJSBundle\Tests\TokenizadorTest::testBuscarModulos
ok 2 - UCI\Boson\GeneradorEXTJSBundle\Tests\TokenizadorTest::testBuscarModelos
ok 3 - UCI\Boson\GeneradorEXTJSBundle\Tests\TokenizadorTest::testBuscarPath
ok 4 - UCI\Boson\GeneradorEXTJSBundle\Tests\TokenizadorTest::testTokenizador
ok 5 - UCI\Boson\GeneradorEXTJSBundle\Tests\TokenizadorTest::testDevolverNombre
1..5
```

Fig. 3.8 Ejecución de los métodos.

3.3.2 Pruebas de caja negra

En las pruebas de caja negra o de funcionamiento, se decide no tener en cuenta el funcionamiento interno de un sistema y solo se analizan sus entradas y salidas. Se aplica tanto como estrategia de testeo, fijándose más en el exterior (usuario) o en la conexión entre diferentes sistemas (interfaz), que como necesidad cuando no es accesible o no es practico estudiar el funcionamiento interno del sistema en análisis. Se requiere menos habilidad técnica, menos tiempo y menos herramientas. Por ende, menos costo. Pero solo te permite detectar errores y fallos pero no te acerca a la solución de éstos (Pressman, 2006). Mediante los casos de prueba de caja negra se puede comprobar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta.

- La integridad de la información externa se mantiene (Pressman, 2006) (Espinosa, y otros, 2008).

Dentro de la prueba de caja negra se incluyen las técnicas de pruebas que serán descritas a continuación:

- Partición de Equivalencia: Divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Análisis de Valores Límites: Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Grafos de Causa-Efecto: Permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones (Pressman, 2006).

En este trabajo fue seleccionada la técnica de partición de equivalencia, diseñando casos de prueba para cada uno de los requisitos funcionales de la aplicación.

3.3.2.1 Diseños de caso de prueba

A continuación se muestra el diseño de caso de prueba del requisito “Generar CRUD”, el resto serán mostrados en los anexos:

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Entrar las etiquetas de los atributos	Campo de texto en el que sólo se puede introducir letras y que responde al nombre con el que se va a mostrar el atributo en la columna del CRUD	V	N/A	Se entran las etiquetas de los atributos que se quieran mostrar con otro nombre
		Usuario		

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.2 Entrar las etiquetas introduciendo errores en los datos	Campo de texto en el que sólo se puede introducir letras y que responde al nombre con el que se va a mostrar el atributo en la columna del CRUD	I User*54	El sistema no permite introducir caracteres inválidos	Intentar escribir caracteres que no sean letras en el campo de texto que corresponde a la etiqueta
EC 1.3 Entrar la expresión regular en el campo de texto	Campo de texto que es mostrado al seleccionar la opción "otra" de la lista desplegable que responde a la expresión regular	V /^[a-z]$/	El sistema muestra el mensaje de error "La expresión regular debe ser de la forma: /^expresión$/."	Seleccionar la opción "otra" de la lista desplegable que corresponde a la expresión regular, escribir una expresión que no sea de la forma correcta en el campo de texto que se muestra y presionar el botón Siguiente

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.4 Dejar el campo de texto expresión regular en blanco	Campo de texto que es mostrado al seleccionar la opción "otra" de la lista desplegable que responde a la expresión regular	N/A	El sistema muestra el mensaje de error "Debe escribir alguna expresión regular en el campo de texto."	Seleccionar la opción "otra" de la lista desplegable que corresponde a la expresión regular, no introducir ningún valor en el campo de texto que se muestra y presionar el botón Siguiente
EC 1.5 Presionar el botón Atrás	Botón que se encuentra en la parte inferior izquierda	N/A	El sistema regresa a la interfaz anterior	Presionar el botón Atrás
EC 1.6 Presionar el botón Siguiente	Botón que se encuentra en la parte inferior derecha	N/A	El sistema genera el CRUD en EXT JS 4 y avanza a la próxima interfaz	Presionar el botón Siguiente

Tabla 3.2 Diseño de caso de prueba del requisito “Generar CRUD”.

3.4 Pruebas de aceptación

Estos 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. De hecho, 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).

Con el propósito de evaluar la herramienta desarrollada por parte de los usuarios finales, le fueron aplicadas pruebas de aceptación por la especialista del departamento Ing. Claudia Bravo Batista, obteniendo los resultados que se muestran en la fig. 3.9.

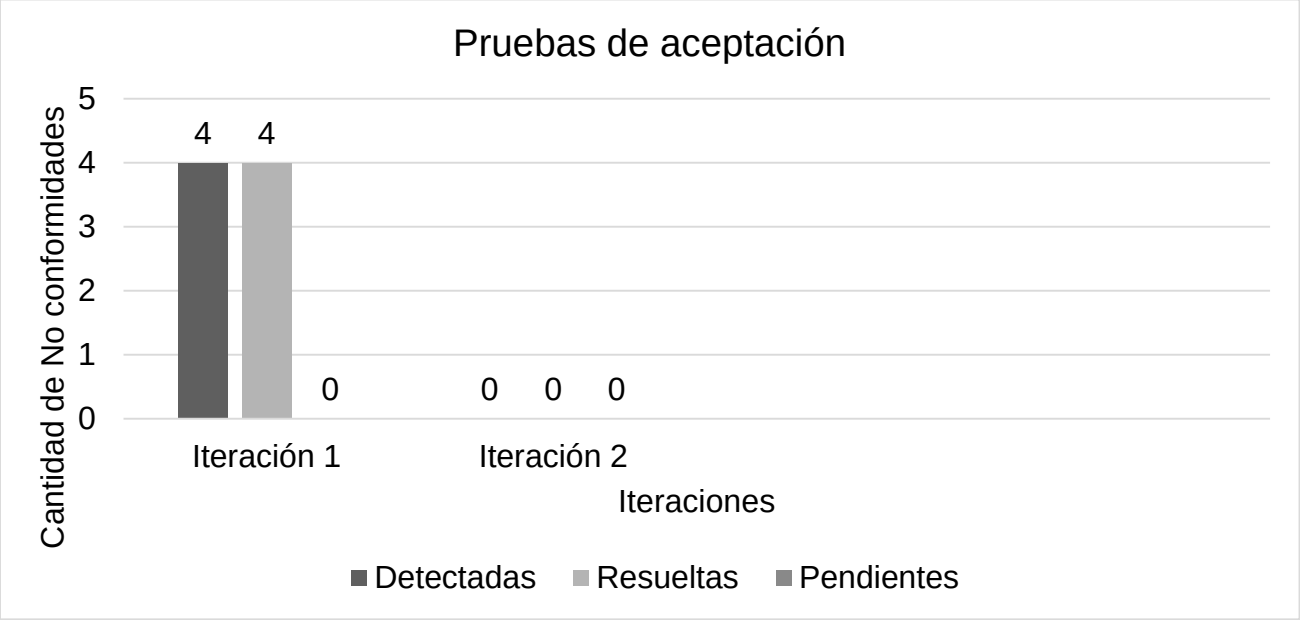

Fig. 3.9 Resultados de la prueba de aceptación por iteraciones.

3.5 Validación de la investigación

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.

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 va a realizar un pre-experimento, mediante el diseño de pre-prueba y post-prueba con un solo grupo; para comprobar cómo se comporta el tiempo de desarrollo de las interfaces gráficas, analizando un antes (sin el uso de la herramienta) y un después (con el apoyo de la herramienta).

Teniendo en cuenta para la pre-prueba los resultados obtenidos mediante la encuesta realizada en el centro CEIGE (los tiempos promedios que se demoran los desarrolladores, que trabajan con el marco de trabajo EXT JS 4, en la creación de las interfaces de un CRUD).

La post-prueba fue aplicada a la misma muestra de 30 desarrolladores, a la que se le realizó la encuesta; esta prueba consistió en el uso del componente desarrollado, con el objetivo de medir el tiempo que se demoraban en desarrollar las interfaces con el uso de la herramienta, tomando como entrada del mismo una de las entidades del sistema seleccionada aleatoriamente. El tiempo fue medido en el proceso completo de creación de las interfaces con el componente (dígase desde la selección del componente del sistema hasta la visualización de las interfaces).

Como resultado del pre-experimento se obtuvieron los siguientes resultados:

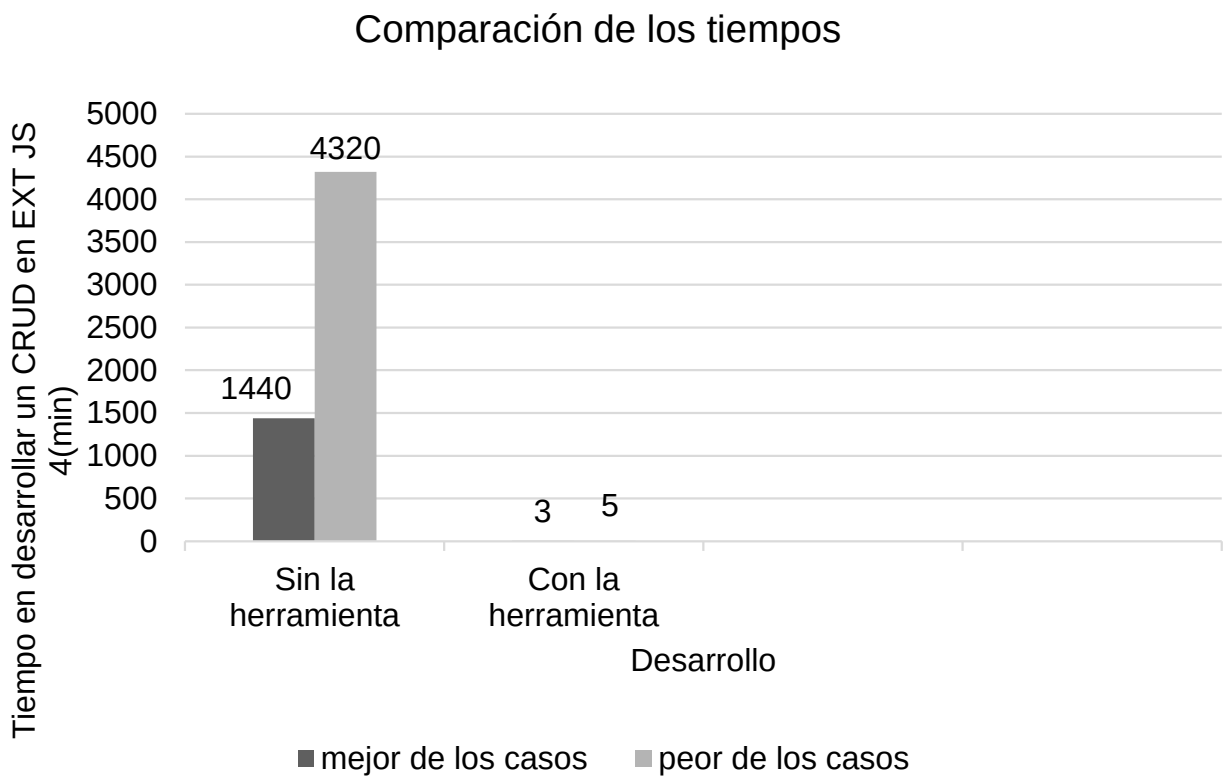

Fig. 3.10 Resultados del pre-experimento realizado.

3.6 Conclusiones del capítulo

En el desarrollo del capítulo se aplicaron pruebas de caja negra y caja blanca al software, que sirvieron de base para la evaluación del sistema propuesto.

Las pruebas de caja blanca aplicadas al software, mediante la técnica del camino básico y el marco de trabajo PHPUnit devolvieron resultados exitosos, evaluando de positiva la calidad del producto.

Mediante las pruebas de caja negra se pudo comprobar que los requisitos de software funcionan correctamente; y que el sistema responde en cada momento según las especificaciones establecidas.

Con la realización del pre-experimento se pudo demostrar como el tiempo de desarrollo de las interfaces disminuyó marcadamente con el uso del componente desarrollado.

CONCLUSIONES GENERALES

Al culminar la investigación se le dio cumplimiento a los objetivos planteados, alcanzando los resultados propuestos, una herramienta capaz de generar el código fuente de un CRUD en EXT JS 4, disminuyendo el tiempo de desarrollo de estas interfaces gráficas.

- Se analizaron los fundamentos teóricos y las principales aplicaciones vinculadas al campo de acción, tanto a nivel nacional como internacional, demostrando la necesidad del nuevo sistema, ya que ninguna de las herramientas estudiadas era factible utilizarla.
- Se diseñó la nueva aplicación en correspondencia con lo que plantea la metodología variación AUP-UCI, que es la que propone la universidad para el desarrollo de software.
- Se seleccionaron las tecnologías y herramientas más adecuadas que pudieran darle solución a la problemática planteada. Empleando PHP v5.4.3 como lenguaje de programación, Apache v2.4 como servidor web, PhpStorm v8.0 como entorno integrado de desarrollo, Visual Paradigm v8.0 como herramienta de modelado; además de utilizar Symfony v2.3 como marco de trabajo y EXT JS v4.2.1 como lenguaje del código de salida.
- Las métricas orientadas al diseño de clases aplicadas al componente, devolvieron valores satisfactorios, validando que se hizo un correcto diseño de la solución propuesta.
- El componente desarrollado es un generador de código fuente de tipo pasivo, implementado a partir de la técnica de plantillas; que garantiza que si cambia la tecnología o el lenguaje de salida, se puedan modificar las plantillas, para que el componente siga siendo reutilizable.
- El componente utiliza como entrada una entidad generada por Doctrine 2.0, de la cual obtiene los atributos y tipos de datos, mediante la descomposición en tokens, para con estos obtener como salida el código fuente de las interfaces de un CRUD desarrollado en EXT JS 4.
- Las pruebas de caja blanca y caja negra, así como las pruebas de aceptación realizadas a la aplicación, devolvieron resultados satisfactorios, validando el correcto funcionamiento de la misma.
- Mediante el pre-experimento se pudo comprobar como el tiempo de desarrollo de las interfaces gráficas disminuyó notablemente con el uso de las herramientas.

RECOMENDACIONES

- Permitirle al usuario poder modificar desde la aplicación el código generado.
- A partir del componente darle la opción al usuario de generar el código del lado del servidor (PHP) para gestionar las entidades, en dependencia del marco de trabajo que estén utilizando.
- Poder personalizar las interfaces del CRUD en EXT JS desde el componente (color, tamaño de las columnas, tipo de letra, tamaño de las letras, etc.).
- Mejorar la apariencia del componente desarrollado.

BIBLIOGRAFÍA

Bergmann Sebastian PHPUnit [En línea]. - 2001. - 01 de 05 de 2015. - https://phpunit.de/.

Abon Cepeda Leevan Generación de código en la programación Web avanzada [Informe] / Universidad de las Ciencias Informáticas. - Ciudad de la Habana : [s.n.], 2008.

Beebom Beebom [En línea] // 15 Best Free PHP Frameworks of 2015. - 2015. - http://beebom.com/2015/02/best-free-php-frameworks.

Boudreau Tim [y otros] Netbeans: The Definitive guide [Libro]. - Estados Unidos de América : O'Reilly & Associates, 2003.

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

Chen Zhixiong y Marx Delia Experiences with Eclipse IDE in programming courses [Libro]. - USA : Consortium for Computing Sciences in Colleges, 2005. - Vol. 21.

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

CodeSmith Tabla de precios del CodeSmith. [En línea]. - 2010. - https://www.codesmithtools.com/store/cart.aspx..

CodeSmith The best [En línea] // Net template base source code generator tools. - 2008. - http://community.codesmithtools.com/blogs/announcements/archive/2008/05/28/codesmith-templatesnow-available-on-google-code.aspx..

Correa Lozano Pablo Ramiro Análisis comparativo de los frameworks Adobe Flex, Java Rich Faces y EXT JS para el desarrollo de aplicaciones enriquecidas en internet (RIA). [Informe] / Facultad de Ingeniería de Sistemas ; Escuela Politécnica Nacional. - Quito. Ecuador : [s.n.], 2010.

Dondo Agustín Por qué elegir PHP [Informe] / Programación en castellano. - 2006.

Durán Jorge Somos Binarios [En línea] // Generadores de codigo : agilizando el desarrollo. - 12 de abril de 2014 . - http://www.somosbinarios.es/generadores-de-codigo/.

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

Espinosa Dania, Gonzalez Susana y Cutiño Durán Estrategia para la aplicación de Pruebas de Caja Blanca y Caja Negra al proyecto Registros y Notarías. [Informe] / Universidad de las Ciencias Informáticas. - La Habana,San Antonio : [s.n.], 2008.

Estrada Israel Productores de Multimedia [En línea]. - 2012. - http://www.productoresmultimedia.net/software/netbeans-ide-ideal-para-programadores/.

Falkner Jayson , Galbraith Ben y Irani Romin Desarrollo Web con JSP [Libro]. - España : ANAYA MULTIMEDIA, 2002.

FayerWayer FayerWayer [En línea] // Apache sigue siendo el rey entre los grandes sitios. - 2009. - https://www.fayerwayer.com/2009/04/apache-sigue-siendo-el-rey-entre-los-grandes-sitios/.

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

Gallego Vázquez José Antonio “Desarrollo WEB con PHP y MYSQL” [Libro]. - [s.l.] : ANAYA MULTIMEDIA, 2003. - ISBN84-415-1525-5.

Golding David Beginning CakePHP from novice to professional [Libro]. - New York, USA : Apress, 2008.

González Duque Raúl Python para todos [Libro]. - España : [s.n.], 2011.

Groner Loiane Sencha Architect App Development [Libro]. - Birmingham : Packt Publishing Ltd, 2013.

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. - La Habana. Cuba : [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.

Herrington J Code Generation In Action. [Libro]. - [s.l.] : Manning, 2006.

Hong MEI [y otros] ABC: An Architecture Based, Component Oriented Approach to Software Development [Publicación periódica]. - Beijing, China : Institute of Software, School of Electronics Engineering and Computer Science, Peking University, 2004.

Informática Curso de Compiladores y generadores de Código [En línea]. - 2004. - http://www.mailxmail.com/curso/informatica/generadores/capitulo2.htm..

Informer Technologies Software Informer [En línea] // Ext Designer 1.2. - 2010. - http://ext-designer.software.informer.com/1.2/.

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

López Jorge Instalación y configuración del servidor Lighttpd [Publicación periódica]. - [s.l.] : Todo linux: la revista mensual para entusiastas de GNU/LINUX, 2007. - 84.

Molina Moreno Pedro Juan Especificación de interfaz de usuario: De los requisitos a la generación automática de código. [Informe] / Universidad de Valencia. - 2003.

Montero Ayala Ramón “Fundamentos de programación en XML” [Libro]. - [s.l.] : McGraw Hill, 2001. - ISBN84-481-2894-x.

Olsina L. y Rossi G. Towards Web-site Quantitative Evaluation: defining Quality Characteristics and Attributes [Conferencia] // Proceedings of IV Int'l WebNet Conference, World Conference on tbe WWW and Internet. - Hawaü, US : [s.n.], 1999. - Vol. 1. - págs. 834-839.

Osorio Leyva Alexander Herramientas para automatización en la Generación de Código Fuente en el proceso de desarrollo de Software. [Informe] / Facultad 3 ; Universidad de las Ciencias Informáticas. - La Habana. Cuba : [s.n.], 2008.

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

Padilla Armando Beginning Zend Framework [Libro]. - New York, USA : Apress, 2009.

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

Pérez Marc http://blog.avanttic.com [En línea]. - 17 de Febrero de 2012. - 2015. - http://blog.avanttic.com/2012/02/17/arquitectura-orientada-a-componentes-sca-el-enfoque-de-oracle/.

Potencier Fabien y Zaninotto François Symfony 1.1, la guía definitiva [Libro]. - 2010.

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

Pressman Roger S. Ingeniería del Software. Un enfoque práctico. Sexta Edición [Libro]. - [s.l.] : Mc Graw Hill, 2003. - 970-10-5473-3.

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

Rodriguez Jorge Pruebas unitarias [Informe]. - 2006.

Rúa Suárez Alejandro Generación de Código: Estandarización y eficiencia en el desarrollo de software para su negocio [Informe]. - Julio, 2010.

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

Serrano Pérez Jorge Programación con ASP .NET [Libro]. - España : ANAYA MULTIMEDIA, 2002.

ServidoresAdmin ServidoresAdmin.com [En línea] // Nginx. - febrero de 2015. - http://www.servidoresadmin.com/nginx/.

sitepoint sitepoint [En línea] // Best PHP IDE in 2014 – Survey Results. - 2014. - http://www.sitepoint.com/best-php-ide-2014-survey-results/.

Sosa Marín Ricardo “JASCOG 1.0”. Herramienta para la generación de código JavaScript para la librería ExtJS. [Informe] / Universidad de las Ciencias Informáticas. - La Habana. Cuba : [s.n.], 2009.

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. - http://urn.kb.se/resolve?urn=urn:nbn:se:hig:diva-12010..

symfony.es Symfony.es [En línea] // ¿Qué es Symfony?. - 2007. - http://symfony.es/pagina/que-es-symfony/.

tufuncion tufuncion [En línea] // El framework más popular de Ajax. - 2008. - http://www.tufuncion.com/mejor-framework-ajax.

Upton David CodeIgniter for Rapid PHP Application Development [Libro]. - Birmingham, UK : Olton, 2007.

Windows Server Windows Server [En línea] // Información general del servidor web (IIS). - febrero de 2012. - https://technet.microsoft.com/es-es/library/hh831725.aspx.

ANEXOS

Anexo 1 Historia de usuario del requisito “Listar los componentes”

Número: 1.2	**Nombre del requisito:** Listar los componentes.
Programador: Efraín Francisco Ruiz Zamora	**Iteración Asignada: 1**
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Medio	**Tiempo Real:** 8 horas
Descripción: Permite visualizar todos los componentes en una vista, mediante una lista desplegable.	
Observaciones: N/A	
Prototipo de interfaz: 	

Tabla A.1 HU del requisito “Listar los componentes”.

Anexo 10 Encuesta aplicada a los profesionales del centro CEIGE que trabajan con el marco de trabajo EXT JS, para determinar el tiempo promedio que se demoraban en crear las interfaces de un CRUD en EXT JS 4.

Encuesta sobre uso del marco de trabajo EXT JS 4 en los centros de desarrollo

Esta encuesta está dirigida a los desarrolladores del CEIGE que utilicen como marco de trabajo EXT JS 4. Su objetivo es determinar el tiempo promedio que demoran en diseñar la interfaz de un CRUD (Create, Read, Update, Delete) empleando este marco de trabajo.

1) ¿Usted ha desarrollado alguna interfaz con EXT JS 4?
   a) ___sí
   b) ___no
2) ¿Ha desarrollado en esa(s) interfaz(es) algún CRUD?
   a) ___sí
   b) ___no
3) ¿Utiliza alguna herramienta para generar interfaces en EXT JS 4?
   a) ___sí ¿Cuál (es)?_____
   b) ___no
4) ¿Qué tiempo le demora crear el CRUD?
   a) ___< 30 min
   b) ___1 a 3 horas
   c) ___1 a 3 días
   d) ___otro ¿Cuál?_____

Fig. A.1**1 Encuesta aplicada en el centro CEIGE.**

Anexo 11 Gráfica comparativa de los lenguajes de programación del lado del servidor más usados.

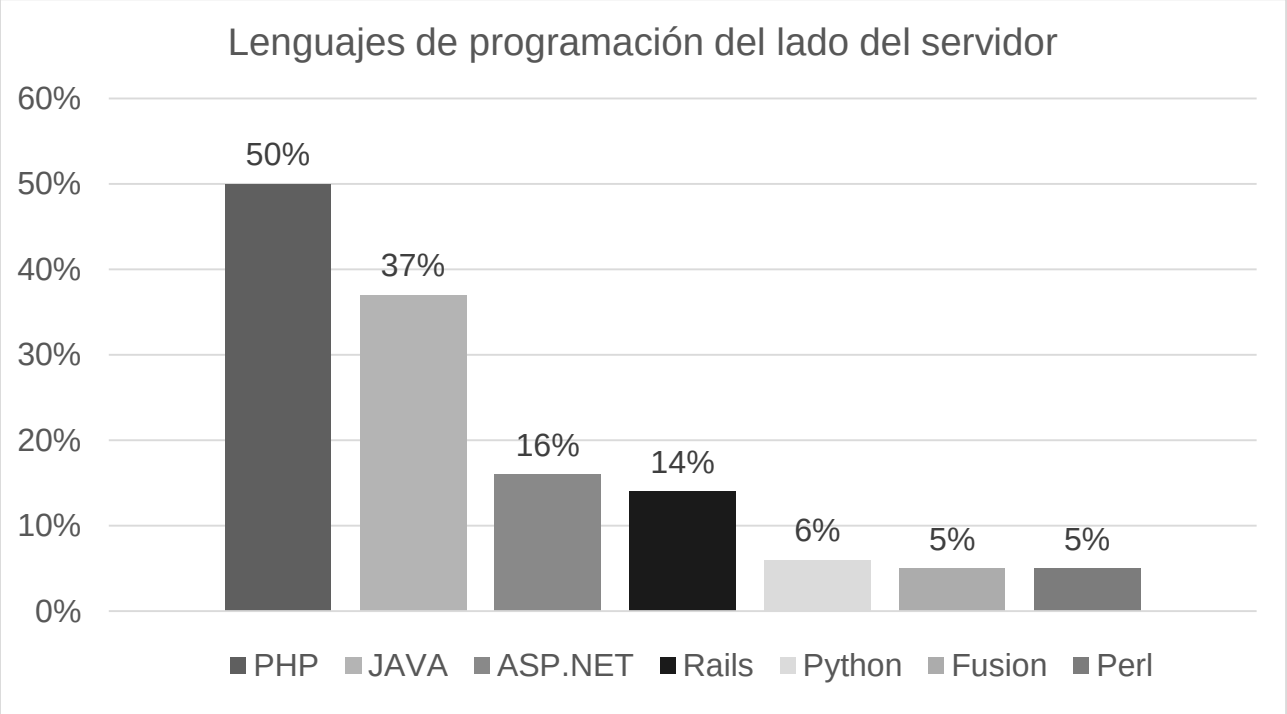

Fig. A.2 Lenguajes de programación, del lado del servidor, más usados. (tufuncion, 2008)

Anexo 12 Gráfica comparativa de los servidores web más usados.

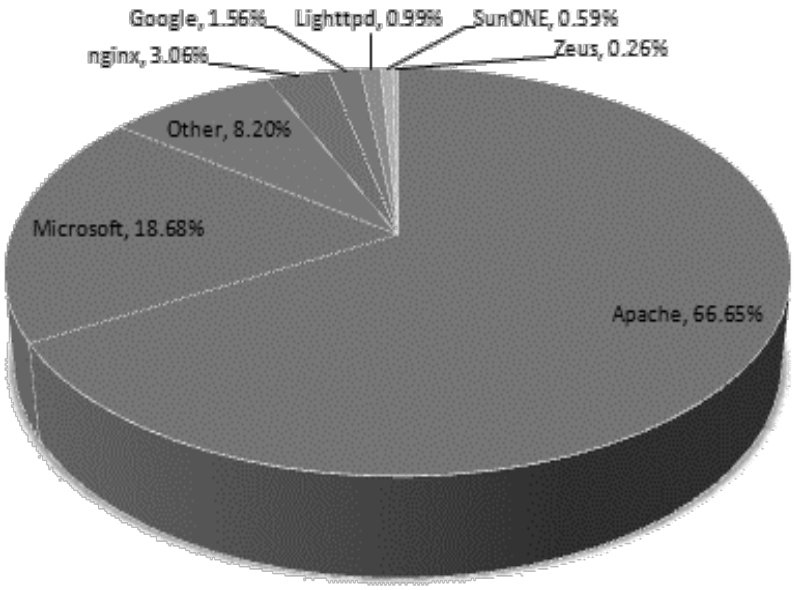

Fig. A.3 Servidores web más usados. (FayerWayer, 2009)

Anexo 13 Gráfica comparativa de los entornos integrados de desarrollo más usados.

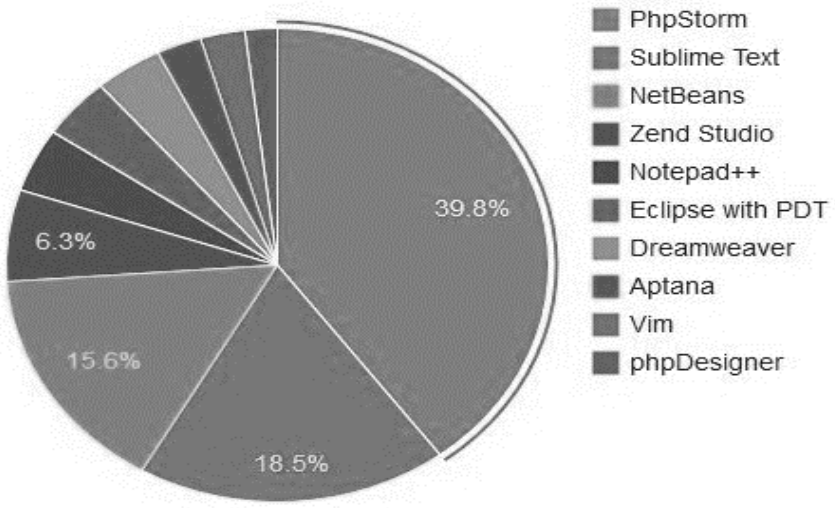

Fig. A.4 Entornos Integrados de Desarrollo (IDEs) más usados. (sitepoint, 2014)

Anexo 14 Gráfica comparativa de los marcos de trabajo más usados.

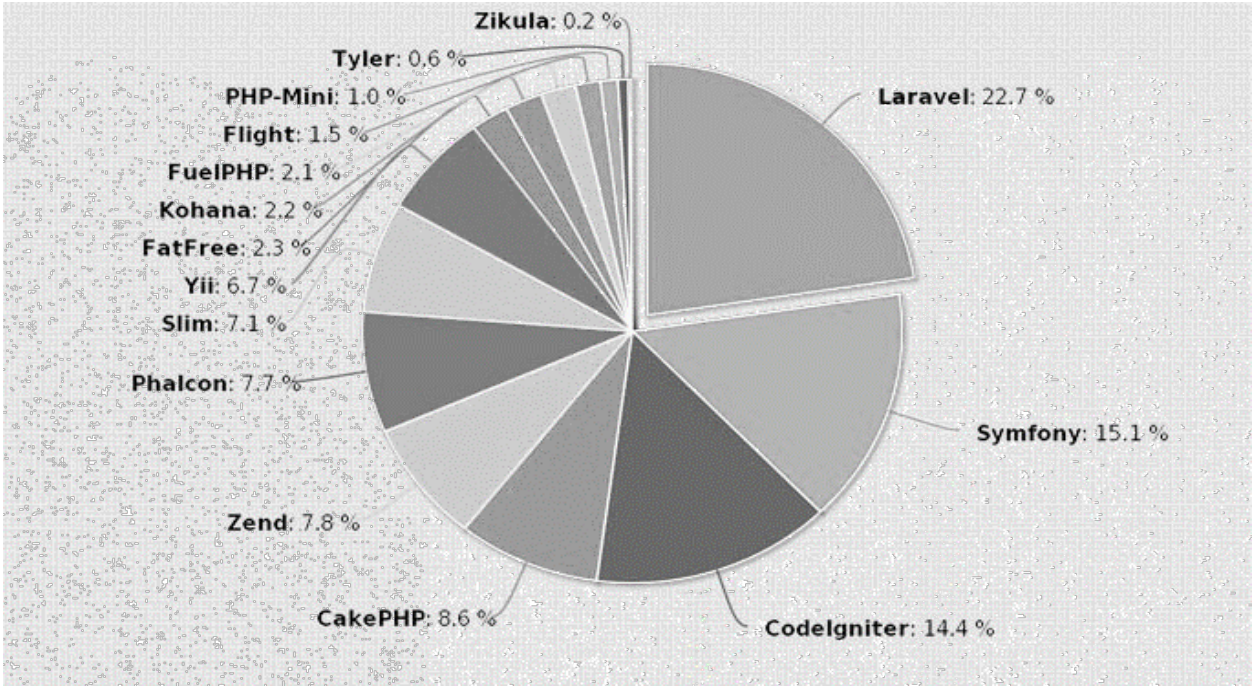

Fig. A.5 Marcos de trabajo más utilizados para el desarrollo web. (Beebom, 2015)

Anexo 15 Tabla del uso de EXT JS en la página principal de empresas reconocidas a nivel mundial.

Empresa	Aplicación	URL
CISCO Systems	Usada como librería de JavaScript en la página principal de CISCO	http://www.cisco.com/
Pixar Animation Studios P I X A R	Usada como librería de JavaScript en la página principal de Pixar	http://www.pixar.com
T-Mobile T · ·Mobile·	Usada como librería de JavaScript en la página principal de T-Mobile	http://www.t-mobile.com/
CANON Canon	Usada como librería de JavaScript en la página principal de T-Mobile	http://www.canon.ca/english/

Fig. A.6 Principales empresas que utilizan EXT JS. (Correa Lozano, 2010)

Anexo 16 Estructura de carpetas que genera la herramienta atendiendo al nombre que se le haya asignado a la entidad.

Fig. A.7 Estructura de carpetas que genera el componente.