Universidad de las Ciencias Informáticas
Facultad 3

Herramienta para la toma de decisiones de la vista arquitectónica de sistema del CedruX

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Dayaris Rodríguez Pupo

Tutor:

Ing. Didier Roque Ginebra

Co-tutor:

Ing. Larisa González Álvarez

La Habana
2012

Declaración de Autoría

Declaro que soy el único autor del trabajo: Herramienta para la toma de decisiones de la vista arquitectónica de sistema del CedruX y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste, firmo la presente a los ____ días del mes de _____ del año 2012.

_____

Dayaris Rodríguez Pupo

Autor

_____ _____

Ing. Didier Roque Ginebra

Tutor

Ing. Larisa González Álvarez

Co-tutor

Datos de Contacto

Tutor: Ing. Didier Roque Ginebra.
Graduado de Ingeniero en Ciencias informáticas en el año 2008. Actualmente ocupa el cargo de Arquitecto de sistema en el equipo de arquitectura del departamento de desarrollo de productos del proyecto CedruX.

Co-tutor: Ing. Larisa González Álvarez
Graduada de Ingeniería en Ciencias Informáticas en el año 2008 con título de oro. Profesor Instructor. Tres años de experiencia en el desarrollo de software de gestión.

Agradecimientos

A mi familia por el esfuerzo y la confianza…

A Rosalia, Yeni y Dairy por apoyarme y brindarme su amistad en un mundo nuevo donde no conocía a nadie.

A Laura por sus consejos y su ayuda.

A Irina y Liu por compartir estos últimos años de amistad a pesar de nuestras diferencias de carácter.

A “D” por su ayuda incondicional, aguantar mis malcriadeces y llenar mi vida de momentos felices.

Agradezco de manera general a todos los que me han ayudado de una forma u otra a lo largo de estos 5 años en la universidad.

Gracias por contar con ustedes…

Dedicatoria

A mi mamá: por su dedicación, esfuerzo y educación. Por sacarnos adelante cuando todo se puso gris, estoy orgullosa de ti.

A mi papá donde quiera que esté…

A mi hermana: porque también ha sido parte de mi formación y educación.

A mi tío Anier por ser como un padre y contribuir en mi educación.

A mis abuelos por cuidarme y estar ahí incondicionalmente.

Para todos ustedes es éste logro…

Resumen

El Sistema Integral de Gestión CedruX cuenta con un total de 126 componentes documentados en 13 artefactos asociados a los subsistemas en los que están ubicados. Estos artefactos son utilizados para recopilar información que permita generar la matriz de integración, la cual relaciona varios componentes mediante los servicios que brindan y/o consumen representados en las intercepciones. Se maneja haciendo uso de las hojas de cálculo de MS Excel[1] que permite a través de fórmulas realizar el cálculo de los indicadores Tamaño, Complejidad y Criticidad de cada componente.

Por tanto se hace difícil para los arquitectos del sistema mantener actualizada la matriz para tomar decisiones sobre la dependencia, prioridad en el desarrollo y soporte de los componentes.

Se necesita desarrollar una herramienta que facilite la operabilidad[2] de la matriz de gestión de la integración de CedruX para ayudar en el proceso de toma de decisiones.

Palabras Clave

Matriz de integración, operabilidad, toma de decisiones.

[1] MS Excel: Microsoft Excel es una aplicación para manejar hojas de cálculo.
[2] “Capacidad del producto del software para permitirle al usuario operarlo y controlarlo” (ISO/IEC, Abril 2005).

Índice

Índice de Figuras

Índice de Tablas

Introducción

Desde el surgimiento de la industria del software la arquitectura ha jugado un papel fundamental en el desarrollo del mismo. Abarca un conjunto de decisiones entorno a la organización de un sistema que orientan significativamente su diseño y evolución.

La arquitectura de sistema proporciona una visión general del software a construir, define la taxonomía de empaquetamiento de los componentes, el estilo y los patrones arquitectónicos a utilizar en los diferentes escenarios identificados y las soluciones arquitectónicas no tecnológicas orientadas a la gestión de la integración y reutilización.

En Julio de 2006 en Cuba se decidió desarrollar un Sistema Integral de Gestión denominado CedruX.

CedruX se desarrolla por componentes, aprovechando las diversas ventajas que provee, donde primero es necesario desarrollarlos y luego ensamblarlos o integrarlos para que conformen el sistema. Como parte de las principales ventajas se encuentran la posibilidad de reutilizar componentes que han sido probados y utilizados en otros contextos; el desarrollo de un sistema mediante el ensamblado de varios que pueden ser desarrollados por diferentes equipos de forma independiente, entre otras.

En CedruX la integración se realiza entre componentes de un mismo subsistema y entre subsistemas distintos, estableciéndose las definiciones, estándares, protocolos de comunicación y reglas de intercambio de información.

CedruX en su primer alcance fue liberado por el Centro Nacional de Calidad de Software (CALISOFT) con un total de 126 componentes documentados que se gestionan en trece artefactos asociados a los subsistemas en los que están ubicados.

Estos artefactos son definidos (CIG-Complejidad_Criticidad, CIG-Especificación_Componente, CIG-Matriz_Integración) por un equipo de arquitectos que permiten recopilar información para generar la matriz de integración a partir de las relaciones mediante servicios de los componentes del sistema.

La matriz de integración relaciona varios componentes mediante los servicios que brindan y/o consumen representados en las intercepciones. Se maneja haciendo uso de las hojas de cálculo de MS Excel que permite a través de fórmulas realizar el cálculo de los indicadores Tamaño, Complejidad y Criticidad de cada componente que son fundamentales para el trabajo de los arquitectos.

Por tanto se hace difícil mantener actualizada la matriz para tomar decisiones sobre la dependencia, prioridad de desarrollo y/o soporte de componentes.

Esta situación conlleva a plantear el siguiente problema científico: La operabilidad de la matriz de gestión de la integración de CedruX afecta el proceso de toma de decisiones de los arquitectos de software del sistema.

Objeto de estudio: La operabilidad de la matriz de integración.

Objetivo de la investigación: Desarrollar una herramienta que facilite la operabilidad de la matriz de gestión de la integración de CedruX para ayudar en el proceso de toma de decisiones de los arquitectos de software del sistema.

Objetivos específicos:

- ✓ Determinar el marco teórico de la investigación que permita conocer las principales tendencias en cuanto a la operabilidad de la matriz de gestión de la integración y al proceso de toma de decisiones.
- ✓ Diseñar una herramienta que facilite la operabilidad de la matriz de gestión de la integración de CedruX para la toma de decisiones.
- ✓ Desarrollar la solución informática para la operabilidad de la matriz de gestión de la integración de CedruX para la toma de decisiones.
- ✓ Validar la solución mediante las pruebas de caja blanca, métricas del diseño TOC[3] y RC[4].

Campo de Acción: La operabilidad de la matriz de integración de CedruX.

*Para la investigación se plantea la siguiente **hipótesis**:* El uso de una herramienta para la operabilidad de la matriz de gestión de la integración de CedruX debe facilitar la toma de decisiones de los arquitectos del sistema.

Métodos empleados en la investigación:

Métodos teóricos:

- ✓ Análisis y síntesis: Utilizado para diagnosticar las dificultades existentes en la operabilidad de la matriz de gestión de integración en el proceso de toma de decisiones de los arquitectos de software en el sistema.

[3] TOC: Tamaño Operacional de Clase.
[4] RC: Relaciones entre Clases.

- ✓ Histórico - Lógico: Utilizado para analizar la historia y evolución de la matriz de gestión de integración en el proceso de toma de decisiones de los arquitectos de software en el sistema.
- ✓ Modelación: Utilizado para representar la propuesta de solución en diversos diagramas y proporcionar una visión general del sistema a construir.

Métodos empíricos:

- ✓ La Entrevista: Utilizado para recopilar información y obtener los requisitos que debe tener la aplicación.

Resultados esperados

Obtención de una herramienta que facilite la operabilidad de la matriz de gestión de la integración de CedruX y ayude en el proceso de toma de decisiones de los arquitectos de software del sistema.

Estructura del trabajo

Capítulo 1: Se realiza un estudio del estado del arte y se analizan algunos temas como: Arquitectura de sistema, Integración de sistemas, Operabilidad de matriz de integración y Toma de decisiones. Además se hace una breve descripción de los lenguajes de programación, las herramientas y la metodología usada para el desarrollo de la herramienta propuesta.

Capítulo 2: Se realiza una descripción del sistema propuesto, los artefactos de la metodología seleccionada, la arquitectura, seguridad y la forma de despliegue de la aplicación. Se detalla la construcción de la propuesta de solución: Diseño de la Base de Datos, Diagrama de clases del diseño, Estándar de codificación, etc.

Capítulo 3: Se valida la propuesta de solución mediante las pruebas de caja blanca, las métricas de diseño TOC y RC.

Capítulo 1: Fundamentación Teórica

Introducción al capítulo

En este capítulo se realiza una recopilación y estudio de la información actualizada asociada a la investigación: Se abordan las definiciones de arquitectura e integración de sistemas, operabilidad de matriz de integración y su papel en el proceso de toma de decisiones de los arquitectos de software. Se valoran las tendencias de la utilización de tecnologías y herramientas libres.

Además se analizan los posibles lenguajes de programación y herramientas de desarrollo a emplear en la propuesta de solución. Se selecciona la metodología que guía el desarrollo del producto en toda su trayectoria.

Elementos del negocio a informatizar

En el presente acápite se abordarán los principales elementos del negocio a informatizar, enfocados en la arquitectura de sistema: cómo está desarrollado CedruX, cómo se lleva a cabo el empaquetamiento y la integración de los componentes. Se estudiará además la matriz de integración y las decisiones que se pueden realizar a partir de ella.

Arquitectura de sistema

“La arquitectura de sistema define la taxonomía de empaquetamiento de los componentes abstraídos, el estilo arquitectónico a utilizar en los diferentes escenarios arquitectónicos identificados, los patrones arquitectónicos a aplicar en el diseño de los componentes, las soluciones arquitectónicas no tecnológicas orientadas a la gestión de la integración y reutilización, además de quedar definidas las soluciones orientadas a aumentar la reutilización, disminuir el acoplamiento y elevar la cohesión del diseño arquitectónico a construir”. (Lage, et al., 2009)

“La arquitectura de sistema proporciona una visión global del sistema a construir. Describe la estructura y la organización de los componentes del software, sus propiedades y las conexiones entre ellos. Los componentes del software incluyen módulos de programas y varias representaciones de datos que son manipulados por el programa”. (Pressman, 2010)

En ambas definiciones se coincide que la arquitectura de sistema define la estructura de empaquetamiento de los componentes del software; aunque la primera se adapta mejor

a las características del sistema CedruX: está desarrollado en componentes y son integrados para conformar subsistemas. Por ello en la presente investigación se adopta como arquitectura de sistema la primera definición.

Formas de empaquetar sistemas

Según (CEIGE, 2011) en CedruX se decide adoptar para su desarrollo el estilo arquitectónico orientado a componentes y un modelo de desarrollo basado en componentes; de forma tal que el desarrollo del sistema se logra mediante subsistemas y componentes.

“Un componente de software es un fragmento de un sistema de software que puede ser ensamblado con otros fragmentos para formar piezas más grandes o aplicaciones completas” (SEI, 1998).

Según (CEIGE, 2011) se le denomina subsistema al conjunto de componentes que responden a un grupo de funcionalidades, es decir, un grupo de interacciones entre ellos, respondiendo a las distintas integraciones y dependencias originadas en el negocio.

Integración de sistemas

En (Larsson, 2005) se expresa que la integración representa el proceso desarrollado cuando las partes son combinadas para formar otras más complejas y finalmente en productos completos. Los elementos críticos en la integración incluyen la descripción y la gestión de interfaces, la secuencia en la cual los componentes son integrados y la comunicación entre los diferentes involucrados. Incluso se reconoce que también se incluyen requerimientos y propiedades del sistema que no pueden ser comprobados desde un nivel de componente, pero que deben ser comprobados a nivel de sistema.

CedruX en su primer alcance fue liberado por el Centro Nacional de Calidad de Software (CALISOFT) con un total de 126 componentes asociados a los trece subsistemas en los que están ubicados. (Sanchez, 2011)

En CedruX la integración se realiza entre componentes de un mismo subsistema y entre subsistemas distintos, estableciéndose las definiciones, estándares, protocolos de comunicación y reglas de intercambio de información.

Operabilidad de la matriz de integración

En la (ISO/IEC, 2005) se conceptualiza la “operabilidad” como capacidad del producto del software para permitirle al usuario operarlo y controlarlo.

“La matriz de integración contiene todos los componentes definidos en el subsistema, de forma matricial y en las intercepciones se especifican los servicios que consume el componente en la vertical del horizontal” (CEIGE, 2011). Su objetivo fundamental es tener registradas las relaciones entre componentes mediante servicios y saber mediante cuáles servicios específicos.

La matriz de integración es un artefacto que se genera en el sistema CedruX para propiciar a los arquitectos una visión de los componentes y las relaciones existentes entre los servicios que brinda y/o consume, ya sea a nivel de subsistema (internos) o producto (externos). En la misma se registra el código del servicio al cual se hace referencia, esto garantiza por ejemplo, que si se quiere independizar un producto con N componentes la matriz facilita la información sobre cuales se deben tomar de conjunto con el que quiero independizar.

En CedruX la matriz de integración es realizada por los arquitectos de sistema de forma manual en las hojas de cálculo de MS Excel. La Figura 1 muestra un ejemplo de la matriz de integración del subsistema Capital Humano con solo 2 de sus componentes.

Consumen / Componentes	Persona	Trabajador
Persona		chpersona01,chtrabajador01
Trabajador	chpersona01	

Figura 1: Matriz de integración del subsistema Capital Humano

Toma de decisiones

“Una decisión es una opción desde múltiples alternativas, normalmente hecha con un grado justo de racionalidad” (Vercellis, 2009). Antes de tomar una decisión se debe calcular o valorar cuál será el posible resultado al escoger una alternativa.

“La toma de decisiones es el proceso de identificación de un problema u oportunidad y la selección de una alternativa de acción entre varias existentes, es una actividad diligente clave en todo tipo de organización” (Schein, 2006). Este proceso indica que un problema o situación es valorado y considerado detenidamente para elegir el mejor camino a seguir según las diferentes alternativas y operaciones.

En el desarrollo de software la toma de decisiones es de vital importancia pues de ellas dependerá el éxito o fracaso de una organización. Deben ser tomadas con rapidez, de

forma oportuna y fundamentada en información concreta, que permita tomar aquellas más eficientes y efectivas.

En CedruX parten de un análisis de la matriz de integración y del cálculo de los indicadores: Tamaño, Complejidad y Criticidad de los componentes, para la toma de decisiones sobre la dependencia, prioridad de desarrollo y/o soporte.

En la investigación titulada “Propuesta metodológica para la obtención de los componentes de software en los proyectos del sistema CedruX” de (Leyet Fernández, 2011) se plantea que el **Tamaño** del componente viene dado por la siguiente fórmula:

$$\mathrm{TC} = \frac{\mathrm{Kr} * 2 + \mathrm{Krd} * 2 + \mathrm{Ks}}{100}$$

Donde:

TC: Tamaño del componente.

Kr: Cantidad de requisitos funcionales.

Krd: Cantidad de restricciones del diseño.

Ks: Cantidad de servicios que brinda el componente.

De igual manera se define el cálculo de la **Complejidad** del componente como:

$$\mathrm{CoC} = \frac{\mathrm{Cr} * 2 + \mathrm{Crd} * 2 + \mathrm{TC}}{100}$$

Donde:

CoC: Complejidad del Componente.

Cr: Complejidad promedio de los requisitos funcionales.

Crd: Complejidad promedio de las restricciones del diseño.

TC: Tamaño del componente.

Por otro lado se define el cálculo de **Criticidad** del componente como:

$$\mathrm{CrC} = \frac{\mathrm{Kcd} * 2 + \sum \mathrm{KcdD}}{100}$$

Donde:

CrC: Criticidad del componente.

Kcd: Cantidad de componentes dependientes directos.

KcdD: Cantidad de componentes dependientes de los dependientes directos del componente.

En la mencionada investigación se afirma que el cálculo de los indicadores Tamaño y Complejidad facilitan la toma de decisiones con respecto a la implementación de varios componentes. En el caso del indicador Criticidad su cálculo va a influir en las decisiones de soporte para la gestión de cambios.

Software libre y aplicaciones web

"La utilización de software, tecnologías y herramientas libres ha garantizado el ahorro de dinero en la adquisición de licencias, la eliminación de barreras presupuestarias y la independencia tecnológica. La entrega final del producto se realiza al cliente con la posibilidad de mostrar el código fuente, lo que facilita conocer mejor la estructura y agiliza el mantenimiento y corrección del mismo". (Pierra Fuentes, 2011)

Esta gama de ventajas garantiza que el usuario no dependa del autor del software y otorga la libertad sobre el mismo.

Tendencias y tecnologías actuales

En el presente acápite se realizará un estudio de diferentes lenguajes de programación, marcos de trabajo y sistemas gestores de base de datos con el objetivo de seleccionar los adecuados para el desarrollo del sistema propuesto a construir.

Lenguajes

La palabra "lenguaje es el empleo de notaciones, señales y vocales para expresar ideas, comunicarse, y establecer relaciones entre los seres humanos. Un lenguaje no sólo consta de 'palabras', sino también de su pronunciación y los métodos para combinar las palabras en frases y oraciones; los lenguajes se forman mediante combinaciones de palabras definidas en un diccionario terminológico previamente establecido" (Departamento de Programación, 2008).

Lenguajes de programación

"Un Lenguaje de Programación es un conjunto de reglas, notaciones, símbolos y/o caracteres que permiten a un programador poder expresar el procesamiento de datos y sus estructuras en la computadora. Cada lenguaje posee sus propias sintaxis. También se puede decir que un programa es un conjunto de órdenes o instrucciones que resuelven un problema específico basado en un Lenguaje de Programación" (Departamento de Programación, 2008).

Un lenguaje de programación es un idioma artificial diseñado para que las computadoras puedan interpretar las órdenes dadas por el hombre. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Lenguajes de programación del lado del Servidor

PHP: "Es un lenguaje de programación interpretado, diseñado para la creación de páginas web dinámicas. Es gratuito, independiente de la plataforma y puede ser usado en cualquier sistema operativo. Posee una gran librería de funciones y documentación. Es de fácil aprendizaje debido a su gran semejanza a otros lenguajes. Presenta alta compatibilidad con los gestores de bases de datos más comunes como PostgreSQL, Oracle y MySQL." (The PHP Group, 2012)

Según (Manes, 2010) algunas de las características que posee PHP son las siguientes:

- ✓ Es un lenguaje multiplataforma.
- ✓ Es completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos (BD[5]).
- ✓ El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML[6] al navegador.
- ✓ Tiene la capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- ✓ Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- ✓ Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- ✓ Permite aplicar técnicas de programación orientada a objetos.
- ✓ No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- ✓ Tiene manejo de excepciones (desde PHP5).

Java: "Es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su

[5] BD: base de datos
[6] HTML (HyperText Markup Language): Lenguaje de Marcas de HiperTexto.

sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.” (Oracle Corporation, 2010) En abril del 2009 Oracle adquirió Sun Microsystems haciéndose propietario del lenguaje Java.

Según (Oracle Corporation, 2010) algunas características de **Java** se muestran a continuación:

- ✓ Es un lenguaje compilado, generando ficheros de clases compilados que son interpretadas por la máquina virtual de Java. Siendo la máquina virtual de java la que mantiene el control sobre las clases que se estén ejecutando.
- ✓ Es un lenguaje multiplataforma: El mismo código java que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual java.
- ✓ Es un lenguaje seguro: La máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.
- ✓ Gracias al API[7] de java se puede ampliar el lenguaje para que sea capaz de, por ejemplo, comunicarse con equipos mediante red, acceder a bases de datos, crear páginas HTML dinámicas y crear aplicaciones visuales al estilo Windows.

Luego del previo estudio realizado se decidió emplear como lenguaje de programación del lado del Servidor PHP en su versión 5.3.8.

Aunque se tuvo en cuenta el lenguaje Java en el estudio antes realizado, se descartó debido a que tiene varias limitantes para el país. Una de ellas y la más importante es la cláusula de la licencia de uso de la máquina virtual de Java que dispone que a esta tecnología sean aplicables las leyes vigentes en los Estados Unidos de América. Este elemento podría comprometer la solución resultante de la presente investigación.

En cambio PHP es multiplataforma, gratuito, con un alto y eficiente rendimiento. Además posee un conjunto muy amplio de librerías para ser utilizadas en diferentes tares relacionadas con la web, se puede conectar con diversas bases de datos, parsear XML[8], enviar correo electrónico y generar archivos PDFs y XLS.

Lenguajes de programación del lado del Cliente

JavaScript: Es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un

[7] API: Interfaz de Programación de Aplicaciones
[8] XML (eXtensible Markup Language): Lenguaje de Marcas Extensible.

lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Su uso se basa fundamentalmente en la creación de efectos en las páginas y la definición de interactividades con el usuario. (Departamento de Programación, 2008)

"JavaScript es un lenguaje con muchas posibilidades que pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente. Soporta el Modelo de Objetos de Documento (DOM[9], Document Object Model)." (Instituto Tecnológico de Veracruz, 2009)

VScript (Visual Basic Script): "Es un lenguaje de programación de scripts del lado del cliente, sólo compatible con Internet Explorer. Es por ello que se usa poco. Está basado en Visual Basic de Microsoft. Tanto su sintaxis y modo de operación es una versión reducida del primero. El modo de funcionamiento de Visual Basic Script para construir efectos especiales en páginas web es muy similar al utilizado en JavaScript y los recursos a los que se puede acceder también son los mismos: el navegador." (Instituto Tecnológico de Veracruz, 2009)

En la presente investigación se emplea como lenguaje del lado del cliente JavaScript (versión 1.8).

Este lenguaje es totalmente integrado con HTML y CSS[10], además de ser estable y con buen rendimiento; fácil de utilizar cuando se emplea con marcos de trabajo o librerías como ExtJS. Es la base para la utilización de AJAX[11] y garantizar la creación de interfaces enriquecida para realizar aplicaciones ágiles e intuitivas.

La exclusión de VScript está centrada en ser un lenguaje desarrollado por la Corporación Microsoft por y para sus tecnologías; de aquí que solo sea aplicable al navegador insignia de esa corporación y sus incompatibilidades con el resto de los navegadores web.

Marcos de trabajo

Un marco de trabajo o framework "define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar" (Rodríguez, 2011).

[9] "Es el conjunto de objetos predefinidos que nos permite acceder a todos los elementos de una página y a ciertas características específicas del navegador." (Instituto Tecnológico de Veracruz, 2009)
[10] CSS: Cascading Style Sheets (Hojas de estilo en cascada)
[11] AJAX: Asynchronous JavaScript And XML (JavaScript asíncrono y XML)

Un marco de trabajo facilita el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes y encapsula operaciones complejas en instrucciones sencillas; proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener.

Teniendo en cuenta que se escogió el lenguaje PHP se va a realizar un estudio de algunos marcos de trabajo y otros que puedan servir para el desarrollo de la aplicación propuesta.

Sauxe

"Sauxe es un framework que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo" (Baryolo, et al., 2008).

"Sauxe utiliza en la capa de acceso a datos el Lenguaje de Consulta de Datos (DQL) que implementa Doctrine. La documentación de éste tiene todas las características necesarias para ser funcional en casi cualquier proyecto" (Baryolo, et al., 2008). También "utiliza ExtJS en la capa de presentación, por la gran gama de componentes que se pueden reutilizar para agilizar el proceso de desarrollo y mostrarle al usuario una interfaz más amigable y funcional" (Baryolo, et al., 2008).

"Sauxe emplea 8 componentes de ZendFramework que conforman un potente y extensible Marco de Trabajo de aplicaciones web. A partir de la extensión de algunos componentes de ZendFramework surge ZendExt, desarrollado por el Departamento de Tecnología perteneciente al centro CEIGE y el UCID[12], con el objetivo de crear un Marco de Trabajo extensible y configurable centrando el desarrollo de las aplicaciones, en la lógica del negocio, en las interfaces de usuario, alejando a los programadores de los detalles arquitectónicos, con soporte para entornos multientidad y para una arquitectura de sistema orientada a componentes" (Baryolo, et al., 2008).

Las razones para usar Sauxe según (Durán, 2012) y (López, 2012) son:

1) *Escalable: "*La arquitectura de Sauxe permite que las aplicaciones que lo emplean crezcan de manera indefinida sin afectar el rendimiento de las mismas." (Durán, 2012)

[12] Unidad de Compatibilización, Integración y Desarrollo.

2) *Probado*: “Sauxe está siendo empleado por varios productos de CEIGE[13] e incluso otros centros de la universidad. Algunos ejemplos son: CedruX, Inventario, Mantenimiento, SIGE y otros.” (Durán, 2012)
3) *Soporte:* En Sauxe “el soporte es llevado a cabo por los miembros[14] del Departamento de Tecnología del CEIGE y se realiza a demanda de los clientes”. (Durán, 2012)
4) *Licencia*: “Sauxe es distribuido con la licencia GNU/GPL[15] v3.” (López, 2012)
5) *Seguro*: “Sauxe emplea el sistema integral de seguridad Acaxia, desarrollado por el Departamento de Tecnología del CEIGE, el cual cubre los aspectos fundamentales de la seguridad (autenticación, acceso y auditoria).” (Durán, 2012)
6) *Calidad*: “Sauxe fue liberado por CALISOFT”. (Durán, 2012)
7) *Internacionalización:* “Sauxe cuenta con una solución de internacionalización que permite a los desarrolladores (que se acojan a los lineamientos arquitectónicos dictados por el Departamento de Tecnología del CEIGE) traducir sus aplicaciones a cualquier idioma.” (Durán, 2012)

Symfony

“Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web” (Potencier, 2008).

“Symfony está desarrollado completamente con PHP 5 y es multiplataforma. Este marco de trabajo es compatible con diversos gestores de bases de datos como: MySQL, PostgreSQL, Oracle y SQL Server de Microsoft” (Potencier, 2008).

Las **características de Symfony** según (Potencier, 2008) son las siguientes:

- ✓ Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.
- ✓ Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y Unix estándares).

[13] CEIGE: Centro de Informatización de la Gestión de Entidades.
[14] Compuesto por estudiantes y profesionales.
[15] GPL: Licencia Pública General

- ✓ Independiente del sistema gestor de bases de datos.
- ✓ Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- ✓ Sigue las mejores prácticas y patrones de diseño para la web.
- ✓ Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

Las **razones para usar Symfony** según (Potencier, 2011) son las siguientes:

1) *Escalable:* Symfony es infinitamente escalable si se disponen de los recursos necesarios. Ejemplo de esto es el caso de Yahoo! que utiliza Symfony para programar aplicaciones con 200 millones de usuarios.
2) *Probado*: Symfony ha sido probado con éxito durante años en varias aplicaciones gigantescas (Yahoo! Answers, Dailymotion, delicious) y en otros miles de sitios pequeños y medianos.
3) *Soporte:* Symfony sigue una política de tipo LTS (long term support), por la que las versiones estables se mantienen durante 3 años sin cambios pero con una continua corrección de errores.
4) *Licencia*: Symfony se publica bajo licencia MIT[16], con la que puedes desarrollar aplicaciones web comerciales, gratuitas y/o de software libre.
5) *Seguro*: Symfony permite controlar hasta el último acceso a la información e incluye por defecto protección contra ataques XSS[17] y CSRF[18].
6) *Documentado:* Symfony es el framework mejor documentado, ya que ha publicado cinco libros gratuitos de calidad y siempre actualizados. Además, toda la documentación está traducida al español.
7) *Calidad*: Su código fuente incluye más de 9.000 pruebas unitarias y funcionales.
8) *Internacionalización:* Symfony incluye todas las herramientas necesarias para traducir fácilmente las aplicaciones.

ExtJS

"ExtJS es una librería JavaScript que permite construir aplicaciones complejas, además de flexibilizar el manejo de componentes de la página como el DOM, Peticiones AJAX, DHTML, tiene la gran funcionalidad de crear interfaces de usuario bastante funcionales" (Pérez, 2011).

[16] MIT: Licencia de Software desarrollada originalmente por el Instituto Tecnológico de Massachusetts y de ahí sus siglas en Ingles.
[17] XSS (Cross Site Scripting): Vulnerabilidad que afecta servidores, páginas y usuarios que navegan por la web.
[18] CSRF (Cross Site Request Forgery): Tipo de exploit malicioso en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía.

Ventajas de ExtJS según (Pérez, 2011):

- ✓ Permite crear aplicaciones complejas utilizando componentes predefinidos.
- ✓ Evita el problema de tener que validar el código para que funcione bien en cada uno de los navegadores (Firefox, IE, Safari, Opera etc.).
- ✓ El funcionamiento de las ventanas flotantes lo pone por encima de cualquier otro.
- ✓ Relación entre Cliente - Servidor balanceado: Se distribuye la carga de procesamiento, permitiendo que el servidor pueda atender más clientes al mismo tiempo.
- ✓ Eficiencia de la red: Disminuye el tráfico en la red pues las aplicaciones cuentan con la posibilidad de elegir que datos desea trasmitir al servidor y viceversa.

Dado el estudio realizado a los marcos de trabajo anteriores, se decidió utilizar Symfony (versión 1.4.11) y ExtJS (versión 3.4) para el desarrollo de la aplicación propuesta.

Se descartó Sauxe para llevar a cabo el sistema propuesto debido a la poca documentación con la que cuenta.

Por otro lado Symfony implementa una serie de patrones de diseño y arquitectónico que hacen su arquitectura robusta. Permite que las aplicaciones crezcan sin causar problemas de rendimiento. Garantiza la protección contra ataques XSS y CSRF; además, cuenta con diversos plugins para manejar los elementos de la seguridad basados en usuario, contraseña y el modelo de control de acceso basado en roles. Posee una amplia documentación y una contínua corrección de errores.

ExtJS es una librería JavaScript que permite crear aplicaciones complejas utilizando componentes predefinidos. Garantiza su funcionamiento en navegadores como Mozilla Firefox, Opera, Internet Explorer, Google Chrome y Safari. Para un mejor rendimiento distribuye la carga de procesamiento, permitiendo que el servidor atienda otras peticiones.

Sistemas de gestión de bases de datos

“Los Sistemas de Gestión de Bases de Datos (SGBD) son aplicaciones que permiten a los usuarios definir, crear y mantener la base de datos y proporciona un acceso controlado a la misma. Los SGBD es la aplicación que interactúa con los usuarios de los programas de aplicación y la base de datos.” (Gil, et al., febrero 2005)

Los objetivos de un SGBD según (Gil, et al., febrero 2005) son:

- ✓ Definir la base de datos mediante el Lenguaje de Definición de Datos, el cual permite especificar la estructura, tipo de datos y las restricciones sobre los datos, almacenándolo todo en la base de datos.
- ✓ Permitir la inserción, eliminación, actualización y consulta de los datos mediante el Lenguaje de Manejo de Datos.
- ✓ Proporcionar acceso controlado a la base de datos.
- ✓ Gestionar la estructura física de los datos y su almacenamiento, proporcionando eficiencia en las operaciones de la base de datos y el acceso al medio de almacenamiento.
- ✓ Eliminar la redundancia de datos, establecer una mínima duplicidad en los datos y minimizar el espacio en disco utilizado.
- ✓ Permitir una fácil administración de los datos.

Teniendo en cuenta el marco de trabajo PHP seleccionado para el desarrollo de la solución propuesta, se realizará un estudio de los gestores de base de datos para los cuales Symfony tiene soporte.

MySQL

"MySQL es un gestor de base de datos sencillo de usar e increíblemente rápido. También es uno de los motores de base de datos más usados en Internet, la principal razón de esto es que es gratis para aplicaciones no comerciales" (Oracle Corporation, 2011). En abril del 2009 Oracle adquirió Sun adueñándose de MySQL.

Las **ventajas de MySQL** según (Oracle Corporation, 2011)**:**

- ✓ Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento.
- ✓ Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- ✓ Facilidad de configuración e instalación.
- ✓ Soporta gran variedad de Sistemas Operativos.
- ✓ Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.
- ✓ Conectividad y seguridad.

PostgreSQL

"PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD[19], que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales y posee más de 15 años de desarrollo activo" (PostgreSQL , 1996 - 2012).

Algunas de las **ventajas de PostgreSQL** según (PostgreSQL , 1996 - 2012) son las siguientes:

- ✓ Aumenta la velocidad de respuesta al aumentar el tamaño de la base de datos.
- ✓ No hay costo asociado a la licencia de software. Esto permite un negocio más rentable, flexibilidad y desarrollo sin costos adicionales de licenciamiento.
- ✓ Extensible: El código fuente está disponible de forma gratuita, para que quien necesite extender o personalizar el programa pueda hacerlo sin costes.
- ✓ Es multiplataforma.
- ✓ Diseñado para ambientes de alto volumen: Utilizando una estrategia de almacenamiento de filas llamada MVCC[20], consigue mejor respuesta en grandes volúmenes. Además, MVCC permite a los accesos de solo lectura continuar leyendo datos consistentes durante la actualización de registros, permitiendo copias de seguridad en caliente.
- ✓ Herramientas gráficas de diseño y administración de bases de datos.
- ✓ Puede operar sobre distintas plataformas, incluyendo Linux, Windows, Unix, Solaris, etc.
- ✓ Buen sistema de seguridad mediante la gestión de usuarios, grupos de usuarios y contraseñas.
- ✓ Gran capacidad de almacenamiento.
- ✓ Buena escalabilidad ya que es capaz de ajustarse al número de CPU y a la cantidad de memoria disponible de forma óptima, soportando una mayor cantidad de peticiones simultáneas a la base de datos de forma correcta.

Para la propuesta de solución se decidió utilizar como SGBD PostgreSQL en su versión 9.2.1.

Se descartó MySQL debido a su absorción por Oracle y sus limitantes con la licencia. En cambio PostgreSQL posee licencia BSD, además de contar con un rendimiento excelente y está diseñado para entornos con altos volúmenes de tráfico/transacciones.

[19] BSD: Berkeley Software Distribution (Distribución de Software Berkeley)
[20] MVCC (Multiversion Concurrency Control): El Control de Concurrencia Multiversión es un método utilizado por los SGBD para proporcionar acceso simultáneo a la base de datos.

Tecnologías y herramientas de desarrollo

En el presente acápite se realizará un estudio de diferentes herramientas CASE para el modelado de la propuesta a desarrollar y entornos integrado de desarrollo para llevar a cabo la implementación.

Herramientas CASE

"Se puede definir a las Herramientas CASE (Ingeniería de Software Asistida por Computación o Computer Aided Software Engineering por sus siglas en inglés) como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software" (INEI, 1999).

Las herramientas CASE pueden proveer muchos beneficios en todas las etapas del proceso de desarrollo del software; algunas de ellas según (INEI, 1999) son:

- ✓ Verificar el uso de todos los elementos en el sistema diseñado.
- ✓ Automatizar el dibujo de diagramas.
- ✓ Ayudar en la documentación del sistema.
- ✓ Ayudar en la creación de relaciones en la base de datos.
- ✓ Generar estructuras de código.

Visual Paradigm

"Visual Paradigm es una herramienta CASE que propicia un conjunto de ayuda para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación" (Morejón, 2010).

Algunas de las **características de Visual Paradigm** según (Morejón, 2010) son:

- ✓ Software libre y disponibilidad en múltiples plataformas (Windows, Linux).
- ✓ Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- ✓ Capacidades de ingeniería directa e inversa.
- ✓ Licencia: gratuita y comercial.
- ✓ Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- ✓ Modelado colaborativo con CVS y Subversión (control de versiones).
- ✓ Diagramas de flujo de datos.
- ✓ Soporte ORM - Generación de objetos Java desde la base de datos.

- ✓ Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- ✓ Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- ✓ Importación y exportación de ficheros XMI.

Rational Rose

"Rational Rose es una herramienta CASE desarrollada por Rational Corporation basada en el Lenguaje Unificado de Modelación (UML) que permite crear los diagramas que se van generando durante el proceso de ingeniería en el desarrollo del software" (Alarcón, 2011).

Las **características Principales de Rational Rose** según (Alarcón, 2011) son las siguientes:

- ✓ Admite como notaciones: UML, OMT (Object Modeling Technique) y Booch.
- ✓ Permite desarrollo multiusuario.
- ✓ Genera documentación del sistema.
- ✓ Disponible en múltiples plataformas.

Para realizar los diversos diagramas generados durante el desarrollo de la solución propuesta se emplea la herramienta CASE Visual Paradigm para UML en su versión 8.0.

Se descartó el uso de Rational Rose debido al costo de su licencia, además no es multiplataforma.

Visual Paradigm para UML es una herramienta multiplataforma, fácil de utilizar que brinda múltiples servicios. Además de facilitar el desarrollo de los distintos diagramas y exportarlos como imágenes, posee la capacidad de ingeniería directa e inversa.

Entorno Integrado de Desarrollo

"Un Entorno Integrado de Desarrollo (en inglés Integrated Development Environment o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Un IDE consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica" (Laffita, 2011).

NetBeans

"NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. NetBeans IDE es un producto libre y gratuito sin restricciones de uso" (Oracle Corporation, 2010).

Algunas **características del NetBeans** según (Cerda, 2009) son las siguientes:

- ✓ Un IDE multilenguaje completo y modular
  - Soporte para Java SE, Java EE, Java ME.
  - Gran cantidad de módulos de terceros (plugins).
  - Desarrollo intuitivo drag-and-drop.
  - Debugger, Profiler, Refactoring, Completa código.
- ✓ Gratis y Open Source
  - Open Source desde Junio de 2000.
  - Gran comunidad de usuarios y desarrolladores.
- ✓ Una plataforma para construir aplicaciones
  - Aplicaciones completas para el cliente.
  - Crea ventanas, menús, barras de herramientas y acciones fácilmente.

Geany

"Es un editor de Texto ligero basado en Scintilla con características básicas de entorno de desarrollo integrado (IDE). Está disponible para distintos sistemas operativos, como GNU/Linux, Mac OS X, BSD, Solaris y Microsoft Windows. Es distribuido como software libre bajo la Licencia Pública General de GNU." (Enrico Tröger, 2011)

Geany "tiene soporte para muchos lenguajes de programación distintos, como C, C++, Java, JavaScript, PHP, HTML, CSS, Python, Perl, Ruby, Fortran, Pascal y Haskell." (Enrico Tröger, 2011). "Es una aplicación gratuita y de código abierto que permitirá crear y editar código fácilmente en un entorno sencillo. Protegido bajo Licencia Pública General de GNU resulta una buena variante para los amantes de la programación y para las pequeñas y medianas empresa que no cuentan con capital para asumir la compra de grandes paquetes para programación". (Torres, 2011)

Características de Geany según (Torres, 2011)

- ✓ Autocompletado
- ✓ Soporte multidocumento
- ✓ Soporte de proyectos
- ✓ Coloreado de sintaxis
- ✓ Emulador de terminal incrustado.

Algunas de las utilidades más conocidas según (Torres, 2011) *son:*

- ✓ Compatible con la mayoría de lenguajes
- ✓ Varios paneles para acceder mejor a los datos

- ✓ Herramientas para compilar
- ✓ Buscador integrado

Según (Torres, 2011) *lo más sorprendente de Geany es que permite:*

- ✓ Posibilidad de compilar y ejecutar directamente desde el entorno (en todos los lenguajes orientados a esta labor). Aunque parezca increíble, es capaz de llamar al compilador y luego ejecutar el programa compilado directamente a través de una consola que se te integra en el programa.
- ✓ Descomposición y representación de las clases y estructuras del código; Geany lo interpreta y en la barra izquierda muestra las estructuras y clases que aparecen en él.
- ✓ Posibilidad de ampliar funcionalidad mediante complementos.

En la presente investigación se usa para llevar a cabo la solución propuesta el IDE Geany en su versión 0.16.

Se descarta el uso del NetBeans debido a su dependencia con la máquina virtual de Java y sus limitantes con respecto a la licencia.

Este IDE de desarrollo es multiplataforma, posee licencia GPL, cuenta con completamiento de código, un entorno amigable y bien organizado.

Servidor de aplicaciones

"Un servidor de aplicaciones web es un software que provee la infraestructura necesaria para las aplicaciones web y gestiona el procesamiento de páginas que contienen scripts o etiquetas" (Driggs Vélez, 2011).

Apache 2.0: "Es un servidor web HTTP de código abierto, multiplataforma, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web.

La arquitectura del servidor Apache es muy modular y consta de una sección core y diversos módulos que aportan muchas de la funcionalidades que podrían considerarse básicas para un servidor web". (The Apache Software Foundation, 2011)

Breve descripción de la metodología de ingeniería usada

En el presente acápite se realizará un estudio de varias metodologías de desarrollo de software, pero primero se debe analizar si utilizar Ágil o Formal.

¿Metodología Ágil o Formal?

"Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo." (Pérez, y otros)

Para desarrollar un software se puede utilizar una metodología Ágil o Formal; en la Figura 2 se muestra la gráfica que proponen Barry Boehm y Richard Turner para ayudar en el proceso de selección.

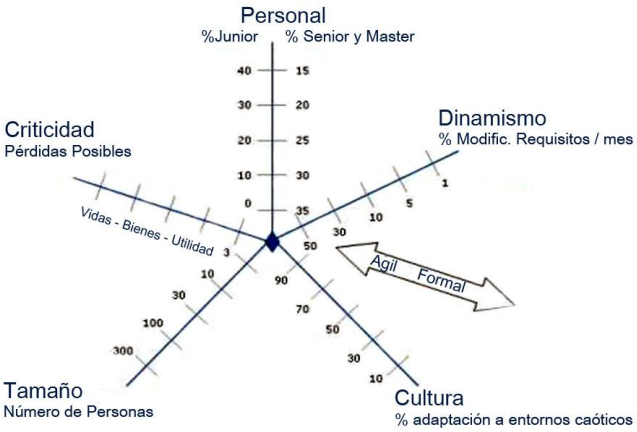


Figura 2: Gráfico de selección de metodología Ágil o Formal
Fuente: (Barry Boehm & Richard Turner)

Tabla 1: Criterios para la selección apropiada de la metodología.

Factor	Discriminadores ágiles	Discriminadores formales
Tamaño	Dependencia y escalabilidad limitada por el porcentaje alto de conocimiento tácito. Apropiado para equipos y productos pequeños.	Escalabilidad y conocimiento explícito. Apropiado para productos y equipos grandes. Duro de mantener en pequeños proyectos.
Criticidad	La simplicidad en la documentación y el diseño dificulta los planes de pruebas. No aconsejado para sistemas con niveles de criticidad altos (IEEE 1012)	Rigor de requisitos y diseño adecuados para procesos de pruebas, verificación y validación. Duros de gestionar en proyectos de escasa criticidad
Dinamismo	"Re-factorizar" desde un diseño básico hasta el producto final es un método ideal para entornos dinámicos e in-novadores, pero muy caro por el "re-trabajo" para entornos estables o conocidos.	En sistemas estables y conocidos, partir de requisitos completos y diseños detallados permite trazar y seguir un plan completo y "hacerlo bien a la primera".
Personal	Los métodos de trabajo ágiles requieren una masa crítica de técnicos con niveles de experiencia medios-altos, capaces de comprender y adaptar los métodos y las técnicas empleadas.	Aunque es aconsejable contar con personas expertas en las fases de definición del proyecto, luego pueden ejecutarse con menor masa crítica de expertos.
Cultura	Más apropiado para culturas de	Más apropiado en culturas en las que las

	“empowerment” responsabilidad y horquilla de decisión y libertad personal.	personas se sienten seguras con un marco de tareas y responsabilidades bien definido.

Luego de analizar el gráfico de selección de metodología Ágil o Formal propuesto por Barry Boehm y Richard Turner, se decidió usar para la propuesta de solución de la presente investigación una metodología Ágil; para ello se realizará un estudio de algunas metodologías de ese tipo.

Scrum

“Scrum es una metodología ágil que según muchos especialistas es óptima para equipos de trabajo de hasta 8 personas, aunque hay empresas que la han utilizado con éxito en equipos más grandes.” (Palacio, 2008)

Es una metodología que no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto. Su objetivo principal es elevar al máximo la productividad de un equipo de desarrollo.

“Sin embargo, más que una metodología de desarrollo software, es una forma de autogestión de los equipos de programadores; son los que deciden cómo hacer sus tareas y cuánto van a tardar en realizarlas. Ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro. Igualmente permite seguir de forma clara el avance de las tareas a realizar, de forma se puedan ver día a día cómo progresa el trabajo.

Scrum es una metodología ágil que no responde a ninguna moda, sino a una necesidad realmente demandada en el desarrollo del software. No es ni la mejor metodología ni la única pero sí es la que está empujando muy fuerte por la facilidad de implantación y por su agilidad en cuanto a cambios y lo que propiamente aporta en comparación con otras. Por un lado, Scrum evita la burocracia y la generación documental: No significa que no se deba o no se pueda documentar sino a lo que se refiere es que no se exige documentar nada para iniciar un proyecto, algo que en otras metodologías es impensable. La idea principal es la de ponerse a trabajar prácticamente desde el primer momento y empezar a sacar frutos de ese trabajo para que el cliente vaya viendo los avances y se quede satisfecho con lo que se está haciendo y cómo se está haciendo.” (Zambrano, 2008).

Características de Scrum según (Itzcoalt, 2007)

- ✓ Es una metodología de desarrollo ágil.
- ✓ Está pensada para equipos de desarrollos pequeños (no más de 8 personas).
- ✓ Permite la entrega de un producto funcional al finalizar cada Sprint.

- ✓ Da la posibilidad de ajustar la funcionalidad en base a la necesidad de negocio del cliente.
- ✓ Permite hacer una visualización del proyecto diaria.
- ✓ Tiene un alcance acotado y viable.
- ✓ Se aplica en equipos integrados y comprometidos con el proyecto y que se auto administran.

Funcionamiento de Scrum según (Itzcoalt, 2007)

Antes de iniciar cada iteración, el equipo revisa las tareas pendientes y selecciona la parte que entregará como un incremento de funcionalidad al finalizar la iteración (Sprint).

El equipo debe revisar los requisitos, considerar la tecnología a utilizar, evaluar su conocimiento y en forma colectiva determinar la forma en la que implementara la funcionalidad.

Roles de Scrum según (Itzcoalt, 2007)

- ✓ Product Owner (Propietario del Producto): Representa a todos los interesados en el producto final; marca las prioridades del producto y lleva el control de las estimaciones.
- ✓ Scrum Master (Maestro del Scrum): Se encarga de la incorporación de Scrum en la cultura de la organización, asegura el cumplimiento de los roles y responsabilidades, así como de la formación y entrenamiento en el proceso.
- ✓ Scrum Team (Equipo de Scrum): Debe transformar las tareas del Sprint Backlog en un incremento de funcionalidad en el software: desarrollar el producto con calidad, auto-gestionado, auto-organizado, multi-funcional y no mayor a ocho personas.

Artefactos de Scrum según (Itzcoalt, 2007)

Scrum define una pequeña cantidad de artefactos para el seguimiento del proyecto y control de las actividades asociadas al Sprint.

- ✓ Product Backlog: Listado con los requisitos del sistema
    - o Mantenido y priorizado por el Product Owner.
    - o Documento dinámico que incorpora constantemente las necesidades del sistema.
    - o Se mantiene durante todo el ciclo de vida.

- ✓ Sprint Backlog: Lista de tareas (realistas) extraídas del Product Backlog que serán convertidas en un incremento de funcionalidad. (Es recomendable que las tareas tengan una duración entre 4 y 16 hrs, en caso de tareas mayores deben intentar descomponerse en sub-tareas de ese rango de tiempo).

Ventajas de Scrum según (Itzcoalt, 2007)

- ✓ Entrega de un producto funcional al finalizar cada Sprint.
- ✓ Posibilidad de ajustar la funcionalidad en base a la necesidad de negocio del cliente
- ✓ Visualización del proyecto día a día.
- ✓ Alcance acotado y viable.
- ✓ Equipos integrados y comprometidos con el proyecto.

Extreme Programing (XP)

"La Programación Extrema es una metodología Ágil de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado." (Escribano, 2007)

Roles de XP según (Escribano, 2007)

- ✓ Programador: Produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y los restantes miembros del equipo. Todo el código de producción lo escriben dos personas frente al ordenador, con un sólo ratón y un sólo teclado.
- ✓ Cliente: Escribe historias de usuario para generar el código y le asigna prioridad a las mismas, así decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
- ✓ Encargado de pruebas (Tester): Ayuda al cliente a elaborar las historias de usuario y prueba cada iteración (historia).
- ✓ Encargado de seguimiento (Tracker): proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.
- ✓ Entrenador (Coach): Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

- ✓ Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto.
- ✓ Gestor (Big boss): Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

Artefactos de XP según (Escribano, 2007)

- ✓ Historia del usuario.
- ✓ Tarjetas CRC[21].
- ✓ Tareas de Ingeniería.

Ventajas del uso de XP según (Escribano, 2007)

- ✓ Programación organizada.
- ✓ Menor taza de errores.
- ✓ Satisfacción del programador.

Teniendo en cuenta las metodologías ágiles estudiadas y dadas las características de la solución informática a desarrollar, en la presente investigación se emplea Scrum.

Se descartó el uso de XP ya que esta metodología establece que la programación debe realizarse en estrictamente en dúo y genera más artefactos que Scrum.

Scrum no establece límites para los programadores y solo genera 2 artefactos: el Product backlog y el Sprint backlog.

[21] CRC: Clase, Responsabilidad y Colaboración

Conclusiones del Capítulo

En el presente capítulo se estudiaron los conceptos asociados al problema de la investigación. También se realizó un análisis de diversos lenguajes, marcos de trabajo, Sistemas de Gestión de Bases de Datos, herramientas CASE, Entorno Integrado de Desarrollo, y metodologías ágiles para definir la adecuada en el desarrollo de la solución propuesta.

Luego de un estudio se decidió utilizar como lenguaje de programación del lado del servidor PHP (versión 5.3.8) y del lado del cliente JavaScript (versión 1.8). Para facilitar la programación y organización de la aplicación se emplea el uso del marco de trabajo Symfony (versión 1.4.11) y para llevar a cabo la implementación el IDE Geany (versión 0.16). Para modelar y crear un abstracción de la solución se utiliza la herramienta CASE Visual Paradigm (versión 8.0). Como SGBD y aprovechando las diversas características que lo distingue se utiliza PostgreSQL (versión 9.1.2) y el servidor web Apache (versión 2.2.21). Todos los lenguajes y herramientas fueron seleccionados para el desarrollo web enfocados en la petición del cliente.

Para guiar el proceso de desarrollo de la propuesta de solución de decidió utilizar la metodología ágil Scrum.

Capítulo 2: Propuesta de solución

Introducción al Capítulo

En el presente capítulo se hace una descripción de la propuesta de solución basada en la descripción de los procesos de negocio; se detalla la arquitectura del sistema y se explican los principales artefactos de la metodología empleada. Se especifican los aportes prácticos, vías de solución de la problemática existente y se describe el modelo de datos. Se realiza la construcción de la propuesta de solución desarrollando el Product Backlog, la planificación de las Iteraciones o Sprint, los diagramas de clases de diseño, se valoran los principios de diseño y el estándar de codificación.

Descripción del sistema propuesto

En el presente acápite se describirán los elementos fundamentales para el desarrollo del sistema propuesto, se mostrarán los requisitos funcionales a implementar, los patrones arquitectónicos y de diseño que se emplean, además de definir los elementos de la seguridad para acceder a la aplicación.

Product Backlog

En el Product Backlog se enumeran los requisitos del sistema o del producto en correspondencia con todas las tareas, funcionalidades o requerimientos a realizar. El Product Owner es responsable del contenido, priorización y disponibilidad del Product Backlog.

El Product Backlog nunca se acaba y es usado en la planificación del proyecto convirtiéndose en una estimación inicial y real de los requisitos en cuanto a tiempo (días).

El Product Backlog se desarrolla paralelamente a medida que el producto y el ambiente en el cual se trabaja evoluciona; es dinámico y maneja constantemente los cambios para identificar que necesita el producto para ser: apropiado, competitivo, y útil.

Tabla 2: Requisitos que conforman el Product Backlog

Id	Nombre del Requisito	Requisitos asociadas	Estimación en días
1	Importar XML	-	5
2	Gestionar Producto	Adicionar Producto	3
3		Modificar Producto	1
4		Eliminar Producto	1
5		Listar Producto	1
6		Buscar Producto	1

7	Gestionar componente	Adicionar componente	3
8		Modificar componente	1
9		Eliminar componente	1
10		Buscar componente	1
11		Listar componente	1
12		Calcular Criticidad	2
13		Calcular Complejidad	2
14		Calcular Tamaño	2
15		Obtener reporte de mayor Criticidad	1
16		Obtener reporte de menor Criticidad	1
17		Obtener reporte de mayor Complejidad	1
18		Obtener reporte de menor Complejidad	1
19		Identificar servicios que no se están usando	3
20		Listar servicios de un componente	2
21	Gestionar Requisito	Adicionar Requisito	3
22		Modificar Requisito	1
23		Eliminar Requisito	1
24		Listar Requisito	1
25		Buscar Requisito	1
26	Gestionar Tipo De Componente	Adicionar Tipo de Componente	3
27		Modificar Tipo de Componente	1
28		Eliminar Tipo de Componente	1
29		Listar Tipo de Componente	1
30		Buscar Tipo de Componente	1
31	Gestionar servicio	Adicionar servicio	3
32		Modificar servicio	1
33		Eliminar servicio	1
34		Mostrar descripción de servicio	1
35		Buscar servicio	1
36		Listar servicios	1
37	Gestionar Atributo	Adicionar Atributo	3
38		Modificar Atributo	1
39		Eliminar Atributo	1
40		Listar Atributo	1
41		Buscar Atributo	1
42	Gestionar Tipo de Dato	Adicionar Tipo de Dato	3
43		Modificar Tipo de Dato	1
44		Eliminar Tipo de Dato	1
45		Listar Tipo de Dato	1
46		Buscar Tipo de Dato	1
47	Imprimir información	-	5
48	Exportar XML	-	5

Los requisitos fueron validados mediantes la técnica de prototipado la cual consiste en contruir una maqueta del futuro sistema propuesto a partir de los requisitos obtenidos en el Product Backlog. Esta maqueta fue evaluada y aprobada por el cliente.

En la figura se muestra un ejemplo de propotipo que responde a los requisitos: Adicionar y Modificar Tipo de Dato.

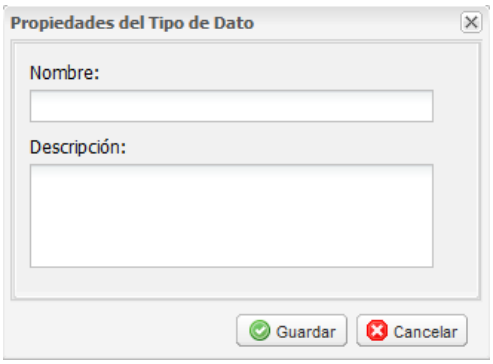


Figura 3: Prototipo de interfaz Adicionar y Modificar Tipo de Dato

Arquitectura

Una de las ventajas de utilizar un marco de trabajo es que estos ya traen incluidas algunas prácticas y criterios que sirven como referencia a la hora de resolver problemas específicos. A su vez, cuentan con la informatización de patrones que proporcionan una estructura al código fuente, forzando al desarrollador a trabajar de forma legible y organizada.

El marco de trabajo Symfony implementa una serie de patrones de diseño y arquitectónicos que hacen su arquitectura sea suficientemente robusta y a la vez flexible como para adaptarse a los casos más complejos.

Patrón arquitectónico implementado por Symfony

Modelo Vista Controlador

"MVC[22] es un patrón de arquitectura de software encargado de separar la lógica de negocio de la interfaz del usuario y es el más utilizado en aplicaciones Web, ya que facilita la funcionalidad, mantenibilidad y escalabilidad del sistema, de forma simple, sencilla y a la vez que permite 'no mezclar lenguajes de programación en el mismo código." (Bahit, 2011)

El MVC divide las aplicaciones en tres niveles de abstracción:

1. El Modelo: define la lógica de negocio. Es el encargado de acceder de forma directa a los datos actuando como 'intermediario' con la base de datos. Symfony guarda todas las clases y archivos relacionados con el modelo en el directorio lib/model/.
2. La Vista: es lo que utilizan los usuarios para interactuar con la aplicación. En la propuesta de solución se emplea ExtJS para el uso de interfaces enriquecidas, ubicándose los archivos correspondiente con la vista en el directorio /web/js/, correspondiende al código JavaScript.

[22] MVC: Modelo Vista Controlador

3. El Controlador: es el intermediario entre la vista y el modelo. Es quien controla las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista para que ésta, lo presente al usuario.

La implementación que realiza Symfony de la arquitectura MVC incluye varias clases como son:

- ✓ *sfController*: Es la clase del controlador y se encarga de decodificar la petición y transferirla a la acción correspondiente.
- ✓ *sfRequest*: Guarda todos los elementos que integran la petición (parámetros, cookies, cabeceras, etc.).
- ✓ *sfResponse*: Posee las cabeceras de la respuesta y los contenidos. El contenido de este objeto se convierte en la respuesta HTML que se remite al usuario.

Patrones de diseño implementados por Symfony

Patrones GRASP[23]

Experto: Es uno de los patrones que más se utiliza en Symfony con la inclusión de la librería Doctrine para mapear la Base de Datos. Symfony utiliza esta librería para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades.

Creador: En la clase Actions se encuentran las acciones definidas para el sistema y se ejecutan en cada una de ellas. En dichas acciones se crean los objetos de las clases que representan las entidades, lo que evidencia que la clase Actions es "creador" de dichas entidades.

Alta Cohesión: Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase Actions, la cual está formada por varias funcionalidades que están estrechamente relacionadas, siendo la misma la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades.

Bajo Acoplamiento: La clase Actions hereda únicamente de sfActions para alcanzar un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las

[23] GRASP (General Responsibility Assignment Software Patterns): Patrones Generales de Asignación de Responsabilidad de Software

de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.

Controlador: Todas las peticiones web son manipuladas por un solo controlador frontal (sfActions), que es el único punto de entrada de toda la aplicación en un entorno determinado. Este patrón se evidencia en las clases 'actions'.

Patrones GoF

Singleton: Es un patrón creacional que garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. A través de la clase sfContext::getInstance(), se obtiene una referencia a todos los objetos del núcleo del marco de trabajo.

Seguridad

La seguridad es uno de los aspectos más importantes a tener en cuenta en el desarrollo de aplicaciones web, es la garantía de la confidencialidad [24], integridad [25] y disponibilidad [26] de la información; para ellos de se estableció el mecanismo de autenticación al sistema basado en usuario y contraseña y el modelo de control de acceso basado en roles utilizando el plugin *sfDoctrineGuardPlugin*.

"El plugin *sfDoctrineGuardPlugin* basa su funcionamiento en 3 aspectos fundamentales: Autorización, Seguridad y Autenticación, es decir, los usuarios tendrán que iniciar una sesión (Autenticación) para acceder a ciertas áreas (Seguridad). Diferentes usuarios pueden tener diferentes privilegios (Autorización) garantizando un sistema para diversos tipos de usuarios." (Eguiluz, 2012)

El *sfDoctrineGuardPlugin* brinda la posibilidad de administrar los privilegios del sistema en la parte administrativa de la aplicación (Autorización).

Con los procedimientos antes analizados se garantiza que sólo un usuario correctamente identificado pueda acceder al sistema, asegurando así la confidencialidad e integridad de los datos; estos usuarios solo tendrán acceso a la información precisa que por el rol establecido en el sistema puede controlar.

[24] Confidencialidad: es el acceso a la información únicamente por personas que cuenten con la debida autorización.
[25] Integridad: consiste en mantener con exactitud la información sin ser manipulada o alterada por personas no autorizadas.
[26] Disponibilidad: es el acceso a la información por personas autorizadas en el momento que lo requieran.

Sprint Backlog

"El Sprint Backlog es la lista que descompone las funcionalidades del product backlog en las tareas necesarias para construir un incremento: una parte completa y operativa del producto." (Palacio, 2008)

"En el Sprint Backlog se asigna a cada tarea la persona que la va a llevar a cabo, y se indica el tiempo de trabajo que se estima, aún falta para terminarla. Es útil porque descompone el proyecto en tareas de tamaño adecuado para determinar el avance a diario; e identificar riesgos y problemas sin necesidad de procesos complejos de gestión." (Palacio, 2008)

A continuación se reflejan las actividades de cada Sprint llevado a cabo para implementar la propuesta de solución.

Tabla 3: Sprint Backlog 1 "Definición de la arquitectura base"

Sprint 1	Inicio: 09/01/2012		
Nombre de la tarea	**Tareas asociadas**	**Estimación en días**	**Tiempo Real en días**
Arquitectura base	Crear el proyecto	1	0.5
	Crear la aplicación	0.5	0.5
	Configurar la conexión con la BD	1	0.5
	Crear el diseño de la BD	2	1
	Crear el schema del diseño de la BD	2	4
	Generar el sql a partir del schema	2	3
	Cargar datos iniciales (fixtures)	1	0.5
	Incluir ficheros de librería ExtJS	0.5	0.5

En el Sprint Backlog 1 (Tabla 3) se muestran las actividades realizadas para crear la arquitecutra base del sistema. En esta iteración se creó el proyecto al cual se le denominó "BejucoX", la aplicación se llamó "frontend". Se configuró la conexión con la BD mediante el gestor de bases de datos PostgreSQL.

Una vez culminadas las configuraciones iniciales se realizó el diseño de la base de datos (DER[27]), se creó el esquema y se generaron las instrucciones SQL. Para comprobar el correcto funcionamiento y diseño de la BD se cargaron datos iniciales mediante los archivos ubicados en el directorio data/fixtures.

Para la utilización de la librería ExtJS se hizo necesario la configuración de los ficheros ubicados en los directorio: web/js/extjs.

[27] DER: Diagrama Entidad Relación

Tabla 4: Sprint Backlog 2 "Gestionar Producto"

Sprint 2 Inicio: 27/01/2012			
Nombre del Requisito	**Requisitos y tareas asociados**	**Estimación**	**Real**
Gestionar Producto	Generar módulo "Producto"	0.5	0.5
	Crear interfaz para "Adicionar Producto"	0.5	0.5
	Implementar acción "Adicionar Producto"	1	0.5
	Crear botón "Modificar Producto"	0.5	0.5
	Crear botón "Eliminar Producto"	0.5	0.5
	Implementar acción "Eliminar Producto"	1	0.5
	Crear interfaz para "Listar Producto"	1	0.5
	Implementar acción para "Listar Producto"	1	0.5
	Crear interfaz para "Buscar Producto"	1	0.5
	Implementar acción "Buscar Producto"	1	0.5

En el Sprint Backlog 2 (ver Tabla 4) se le dió cumplimiento a los requisitos contenidos en el "Gestionar Producto" del Product Backlog; además de una serie de tareas englobadas en este aspecto, para lograr la solucion esperada.

Luego de culminada la iteración ya se puede Adicionar, Modificar, Eliminar, Listar y Buscar un Produto correctamente.

Tabla 5: Sprint Backlog 3 "Gestionar Tipo de Componente"

Sprint 3 Inicio: 15/03/2012			
Nombre del Requisito	**Requisitos asociados**	**Estimación**	**Real**
Gestionar Tipo de Componente	Generar módulo "Tipo de Componente"	0.5	0.5
	Crear interfaz "Adicionar Tipo de Componente"	1	0.5
	Implementar acción "Adicionar"	1	1
	Crear botón "Modificar Tipo de Componente"	0.5	0.5
	Crear botón "Eliminar Tipo de Componente"	0.5	0.5
	Implementar acción "Eliminar"	1	0.5
	Crear interfaz "Listar Tipo de Componente"	1	0.5
	Implementar acción "Listar"	1	0.5
	Crear interfaz "Buscar Tipo de Componente"	0.5	0.5
	Implementar acción "Buscar"	1	0.5

El Sprint Backlog 3 (ver Tabla 5) contiene todas las acciones relacionadas con el requisito "Gestionar Tipo de Componente" al cual se le asignó una prioridad alta en el orden de realización, teniendo en cuenta que hay módulos que tienen una dependencia funcional de él.

Tabla 6: Sprint Backlog 4 "Gestionar Tipo de Dato"

Sprint 4	Inicio: 22/03/2012		
Nombre del Requisito	**Requisitos asociados**	**Estimación**	**Real**
Gestionar Tipo de Dato	Generar módulo "Tipo de Dato"	0.5	0.5
	Crear interfaz "Adicionar Tipo de Dato"	1	0.5
	Implementar acción "Adicionar Tipo de Dato"	1	0.5
	Crear botón "Modificar Tipo de Dato"	0.5	0.5
	Crear botón "Eliminar Tipo de Dato"	0.5	0.5
	Implementar acción "Eliminar Tipo de Dato"	1	0.5
	Crear interfaz "Listar Tipo de Dato"	1	0.5
	Implementar acción "Listar Tipo de Dato"	1	0.5
	Crear interfaz "Buscar Tipo de Dato"	1	0.5
	Implementar acción "Buscar Tipo de Dato"	1	0.5

En el Sprint Backlog 4 se le dio solución a todos los requisitos asociados al Gestionar "Tipo de Dato". A esta iteración también se le asignó una prioridad alta en su desarrollo, debido a que hay módulos que tienen una dependencia funcional de él.

Tabla 7: Sprint Backlog 5 "Gestionar Atributo"

Sprint 5	Inicio: 29/03/2012		
Nombre del Requisito	**Requisitos asociados**	**Estimación**	**Real**
Gestionar Atributo	Generar módulo "Atributo"	0.5	0.5
	Crear interfaz "Adicionar Atributo"	1	0.5
	Implementar acción "Adicionar Atributo"	1	0.5
	Crear botón "Modificar Atributo"	0.5	0.5
	Crear botón "Eliminar Atributo"	0.5	0.5
	Implementar acción "Eliminar Atributo"	1	0.5
	Crear interfaz "Listar Atributo"	1	0.5
	Implementar acción "Listar Atributo"	1	0.5
	Crear interfaz "Buscar Atributo"	0.5	0.5
	Implementar acción "Buscar Atributo"	1	0.5

En la interación 5 (ver Tabla 7) se realizan todas las tareas necesarias para el correcto funcionamiento del requisito Gestionar Atributo (Adicionar, Modificar, Eliminar, Listar, Buscar).

Tabla 8: Sprint Backlog 6 "Gestionar Componente I"

Sprint 6	Inicio: 29/03/2012		
Nombre del Requisito	**Requisitos asociados**	**Estimación**	**Real**
Gestionar Componente	Generar módulo "Componente"	0.5	0.5
	Crear interfaz "Adicionar Componente"	1	0.5

	Implementar acción “Adicionar Componente”	1	0.5
	Crear botón “Modificar Componente”	0.5	0.5
	Crear botón “Eliminar Componente”	0.5	0.5
	Implementar acción “Eliminar Componente”	1	0.5
	Crear interfaz “Listar Componente”	1	0.5
	Implementar acción “Listar Componente”	1	0.5
	Crear interfaz “Buscar Componente”	0.5	0.5
	Implementar acción “Buscar Componente”	1	0.5

El Sprint Backlog 6 (ver Tabla 8) le da solución al 35.71% de las funcionalidades del módulo “Componente”; se le dio prioridad alta en su desarrollo debido a la dependencia funcional que tiene con otros módulos.

Tabla 9: Sprint Backlog 7 “Gestionar Requisito”

Sprint 7	**Inicio: 06/04/2012**		
Nombre del Requisito	**Requisitos asociados**	**Estimación**	**Real**
Gestionar Requisito	Generar módulo “Requisito”	0.5	0.5
	Crear interfaz “Adicionar Requisito”	1	0.5
	Implementar acción “Adicionar Requisito”	1	0.5
	Crear botón “Modificar Requisito”	0.5	0.5
	Crear botón “Eliminar Requisito”	0.5	0.5
	Implementar acción “Eliminar Requisito”	1	0.5
	Crear interfaz “Listar Requisito”	1	0.5
	Implementar acción “Listar Requisito”	1	0.5
	Crear interfaz “Buscar Requisito”	0.5	0.5
	Implementar acción “Buscar Requisito”	1	0.5

En la iteración anterior (ver Tabla 9) se le da solución a los requisitos Adicionar, Modificar, Eliminar, Listar y Buscar Requisito. Este módulo tiene una fuerte dependecia funcional con el módulo “Componente”; por tanto para la creación de un requisito es imprescindible que exista un componente al cual asignarle dicho elemento.

Tabla 10: Sprint Backlog 8 “Gestionar Servicio”

Sprint 8	**Inicio: 09/04/2012**		
Nombre del Requisito	**Requisitos asociados**	**Estimación**	**Real**
Gestionar Servicio	Generar módulo “Servicio”	0.5	0.5
	Crear interfaz “Adicionar Servicio”	1	0.5
	Implementar acción “Adicionar Servicio”	1	0.5
	Crear botón “Modificar Servicio”	0.5	0.5
	Crear botón “Eliminar Servicio”	0.5	0.5
	Implementar acción “Eliminar Servicio”	1	0.5

	Crear interfaz “Listar Servicio”	1	0.5
	Implementar acción “Listar Servicio”	1	0.5
	Crear interfaz “Buscar Servicio”	0.5	0.5
	Implementar acción “Buscar Servicio”	1	0.5

Con el desarrollo de las tareas del Sprint Backlog 8 (ver Tabla 10) ya se puede Adicionar, Modificar, Eliminar, Listar y Buscar un Requisito que esté asociado a un componente antes insertado en el sistema.

Tabla 11: Sprint Backlog 9 “Importar y Exportar XML”

Sprint 9	**Inicio: 16/04/2012**		
Nombre del Requisito	**Requisitos asociados**	**Estimación**	**Real**
Importar y Exportar XML	Crear interfaz “Cargar XML”	1	0.5
	Implementar acción “Cargar XML”	1	1
	Crear interfaz “Exportar XML”	1	0.5
	Implementar acción “Exportar XML”	1	1

Luego de culminado el Sprint Backlog 9 (ver Tabla 11) ya se puede Importar XML con el objetivo de extraer su información y agilizar el trabajo de los arquitectos del sistema. De igual manera luego culminada esta iteración se puede Exportar XML para el almacenamiento de la Información existente en la aplicación.

Tabla 12: Sprint Backlog 10 “Cálculos de indicadores”

Sprint 10	**Inicio: 20/04/2012**		
Nombre del Requisito	**Requisitos asociados**	**Estimación**	**Real**
Cálculos	Crear interfaz para “Calcular Criticidad”	1	0.5
	Implementar acción “Calcular Criticidad”	1	0.5
	Crear interfaz para “Calcular Complejidad”	1	0.5
	Implementar acción “Calcular Complejidad”	1	0.5
	Crear interfaz para “Calcular Tamaño”	1	0.5
	Implementar acción “Calcular Tamaño”	1	0.5

Finalizado el Sprint Backlog 10 (ver Tabla 12) ya se puede realizar el cálculo de los indicadores Tamaño, Complejidad y Criticidad que son imprescindibles para la toma de decisiones de los arquitectos.

Tabla 13: Sprint Backlog 11 “Reportes”

Sprint 11	**Inicio: 25/04/2012**		
Nombre del Requisito	**Requisitos asociados**	**Estimación**	**Real**
Reportes	Crear interfaz para “Obtener reporte de mayor Criticidad”	1	0.5
	Crear interfaz para “Obtener reporte de menor Criticidad”	1	0.5

	Implementar acción para obtener “Mayor Criticidad”	1	0.5
	Implementar acción para obtener “Menor Criticidad”	1	0.5
	Crear interfaz para “Obtener reporte de mayor Complejidad”	1	0.5
	Crear interfaz para “Obtener reporte de menos Complejidad”	1	0.5
	Implementar acción para obtener “Mayor Complejidad”	1	0.5
	Implementar acción para obtener “Menor Complejidad”	1	0.5

Finalizado el Sprint Backlog 12 (ver Tabla 13) ya se pueden obtener reportes de Mayor y Menor de los indicadores Complejidad y Criticidad.

Despliegue

“El diagrama de despliegue es un tipo de diagrama del Lenguajes Unificado de Modelado (UML) que se utiliza para representar las relaciones físicas entres los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes de software (procesos y objetos que se ejecutan en ellos).

Un Diagrama de Despliegue modela la arquitectura y describe la configuración del sistema para su ejecución en un ambiente del mundo real; para llevar a cabo el despliegue se deben tomar decisiones sobre los parámetros de la configuración, funcionamiento, asignación de recursos, distribución y concurrencia” (Cuellar, 2012).

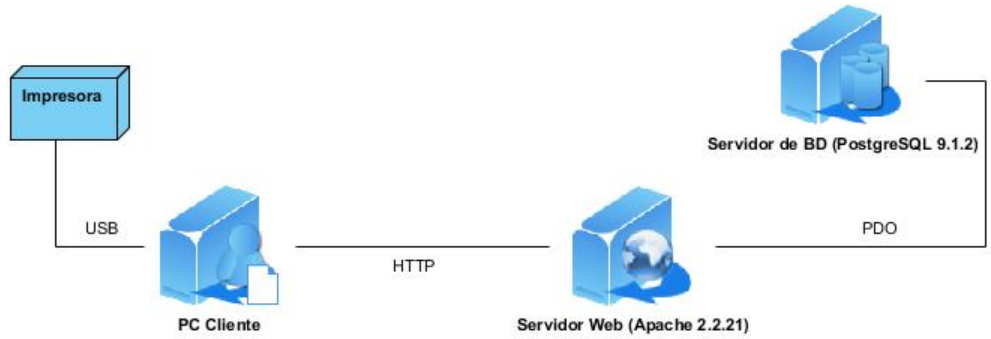


Figura 4: Diagrama de Despliegue

Aportes prácticos y vías de solución

Entre los elementos que conforman el Product Backlog se encuentran los requisitos Importar y Exportar XML.

“XML es un Lenguaje de Marcas Extensible parecido a HTML porque la estructura de ambos esta formada por etiquetas que se abren y se cierran” (Hernández, 2012).

Mediante un archivo XML se puede almacenar información, como es el caso del utilizado en el sistema CedruX para gestionar la integración de sus subsistemas y componentes.

El objetivo del sistema propuesto es permitir a los arquitectos del sistema gestionar la información sobre la integración de los subsistema a los que pertenecen; sin embargo como un valor añadido se incluye la funcionalidad de importar un archivo XML que contenga dicha información, facilitando así la carga de los datos al sistema.

El uso de PHP5 facilita la lectura e interpretación de archivos XML con la introducción de la extención SimpleXML. Esto proporciona un juego de herramientas para generar un Objeto que pueda ser procesado a partir de una llamada a la función simplexml_load_file.

Construcción de la propuesta de solución

En este acápite se describe como se llevó a cabo la construcción del sistema propuesto, los diagramas de clases de diseño correspondiente a cada uno de los módulos generados, el diseño de la base de datos capaz de dar cumplimientos con los requisitos, además de los estándares de codificación y principios del diseño que se tuvieron en cuenta para la implementación.

Diagrama de Clases de Diseño

“Un diagrama de clases del diseño con extensiones web describe la arquitectura y las relaciones entre cada una de las clases del sistema desarrollado. Es la vista más simple y detallada de representar la propuesta de solución” (Navarro, 2009).

Cada diagrama va a estar compuesto por una página cliente[28] que contiene un formulario[29] el cual se encarga de recoger los datos que el usuario ingrese y se los envía al controlador frontal que hace función de página servidora[30].

El controlador (Controller) captura el evento y hace la llamada al modelo (Model).

El modelo se va a encargar de interactuar con la base de datos y retornar la información solicitada al controlador para que se la envíe a la vista.

La vista procesa la información recibida y se la muestra al usuario.

[28] Página cliente: <<Client Page>>
[29] Formulario: <<Form>>
[30] Página servidora: <<Server Page>>

A continuación se muestran los diagramas de clases del diseño asociados a cada uno de los módulos creados en el sistema propuesto.

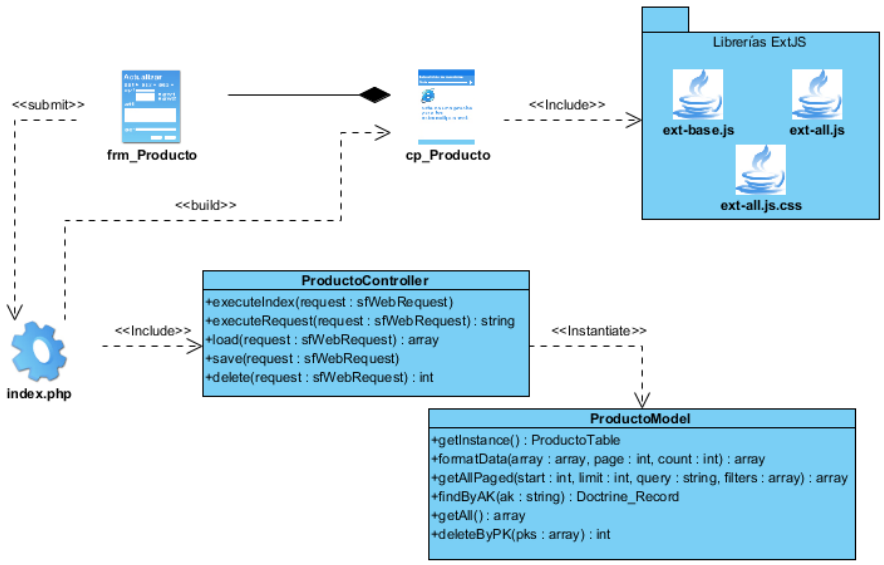


Figura 5: Diagrama de clases del diseño del módulo Producto

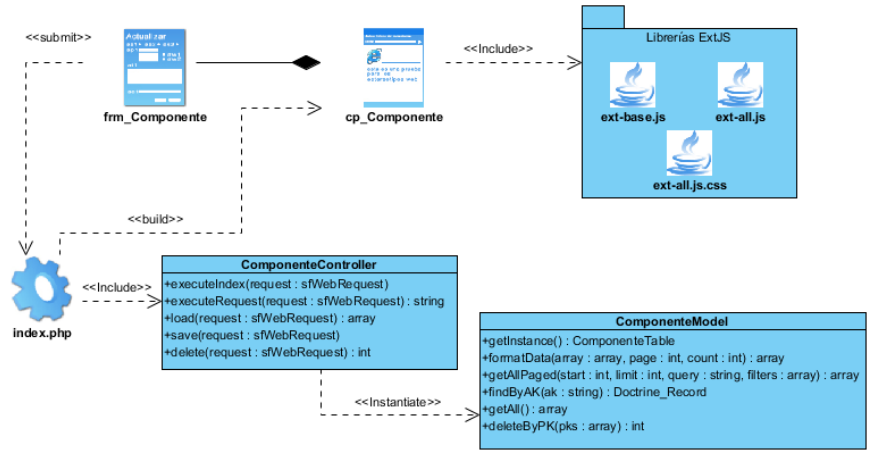


Figura 6: Diagrama de clases del diseño del módulo Componente

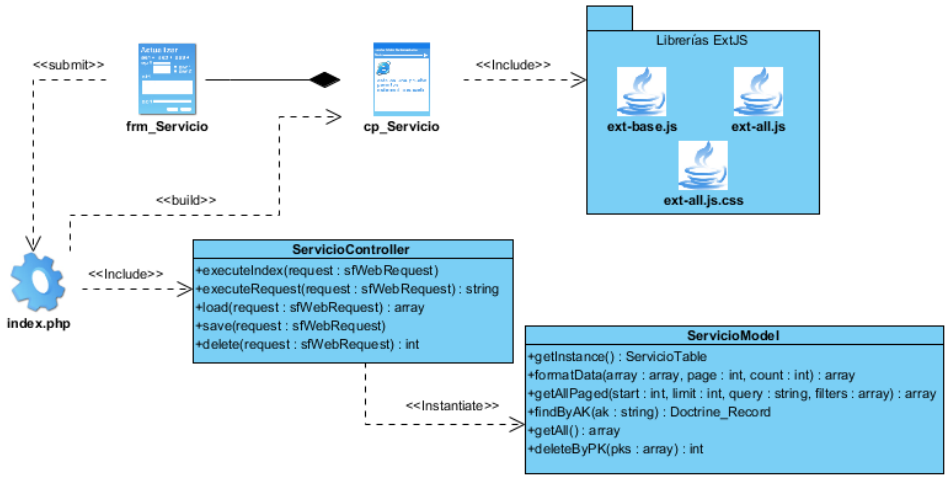


Figura 7: Diagrama de clases del diseño del módulo Servicio

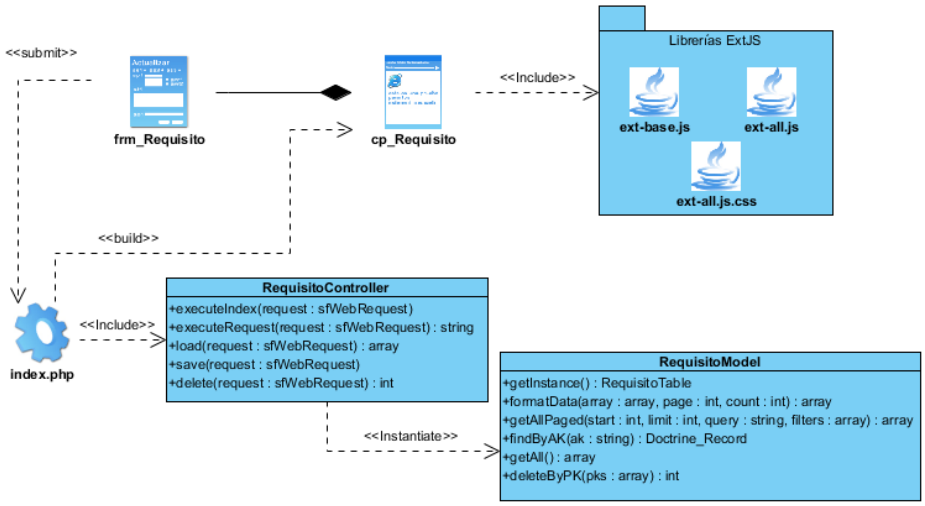


Figura 8: Diagrama de clases del diseño del módulo Requisito

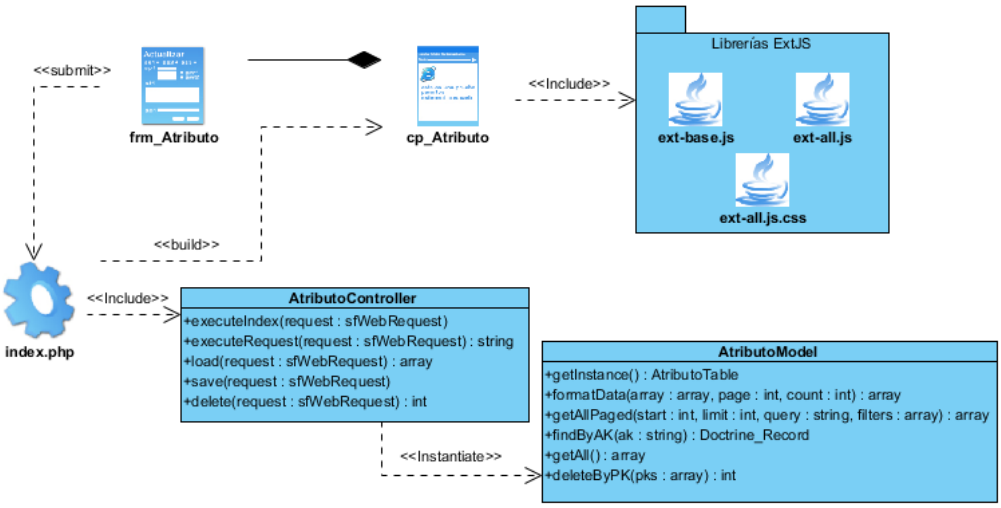


Figura 9: Diagrama de clases del diseño del módulo Atributo

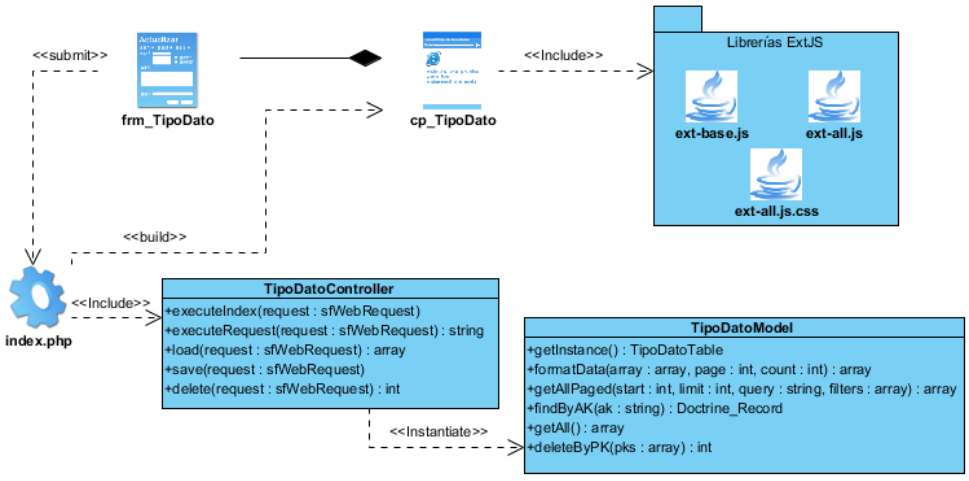


Figura 10: Diagrama de clases del diseño del módulo TipoDato

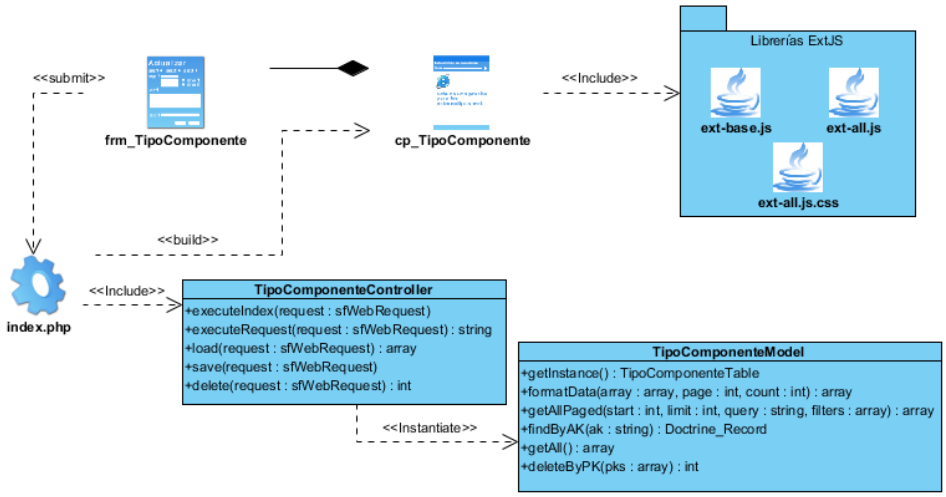


Figura 11: Diagrama de clases del diseño del módulo TipoComponente

Diseño de la base de datos

Uno de los aspectos fundamentales para realizar un sistema informático consiste en construir un esquema relacional enfocado en evitar posibles errores de:

- ✓ *Redundancia:* "Se llama así a los datos que se repiten continua e innecesariamente por las tablas de las bases de datos." (Sánchez, 2011)
- ✓ *Ambigüedades:* "Datos que no clarifican suficientemente el registro al que representan. Es decir, los datos de cada registro podrían referirse a más de un registro o incluso puede ser imposible saber a qué ejemplar exactamente se están refiriendo." (Sánchez, 2011)
- ✓ *Pérdida de restricciones de integridad*: "Normalmente debido a dependencias funcionales." (Sánchez, 2011)
- ✓ *Anomalías en operaciones de modificación de datos*: "Sucede cuando al insertar un sólo elemento haya que repetir tuplas en una tabla para variar unos pocos datos. O que eliminar un elemento suponga eliminar varias tuplas." (Sánchez, 2011)

Cuando se detectan algunos de los problemas mencionados anteriormente se pueden resolver usando las reglas de normalización.

"La normalización de bases de datos es una técnica que puede ayudarle a evitar la aparición de anomalías en los datos así como otras cuestiones relacionadas con la administración de datos. Esta técnica consiste en transformar una tabla en varias fases: primera forma normal, segunda forma normal, tercera forma normal y otras. El objetivo consiste:" según (Gilfillan, 2010):

- ✓ Eliminar las redundancias de datos (y por tanto utilizar menos espacio).

- ✓ Facilitar la tarea de realizar cambios en los datos y evitar las anomalías al hacerlo.
- ✓ Facilitar la implementación de los requisitos de integridad referencial.
- ✓ Generar una estructura fácilmente comprensible muy parecida a la situación que representan los datos y que permita su crecimiento.

Luego de analizar el diseño de la base de datos que se muestra en la Figura 12 se determinó que se encuentra en la *tercera forma normal*.

Primera forma normal (1FN): si y sólo si todos los atributos de la tabla únicamente pueden tomar valores simples[31] y no incluye ningún grupo repetitivo.

Segunda forma normal (2FN): si y sólo si está en 1FN y no incluye dependencias parciales (cuando un atributo depende exclusivamente de parte de una clave primaria).

Tercera forma normal (3FN): si y sólo si está en 2FN y no contiene dependencias transitivas (ver **¡Error! No se encuentra el origen de la referencia.**), es decir, cuando n atributo que no sea clave depende de una clave primaria a través de otro atributo que no sea clave.

Sólo se analizó hasta este nivel de normalización porque se considera que es suficiente para cubrir los requisitos y evitar los posibles errores mencionados anteriormente. A continuación de refleja el diseño de la base de datos para la propuesta de solución.

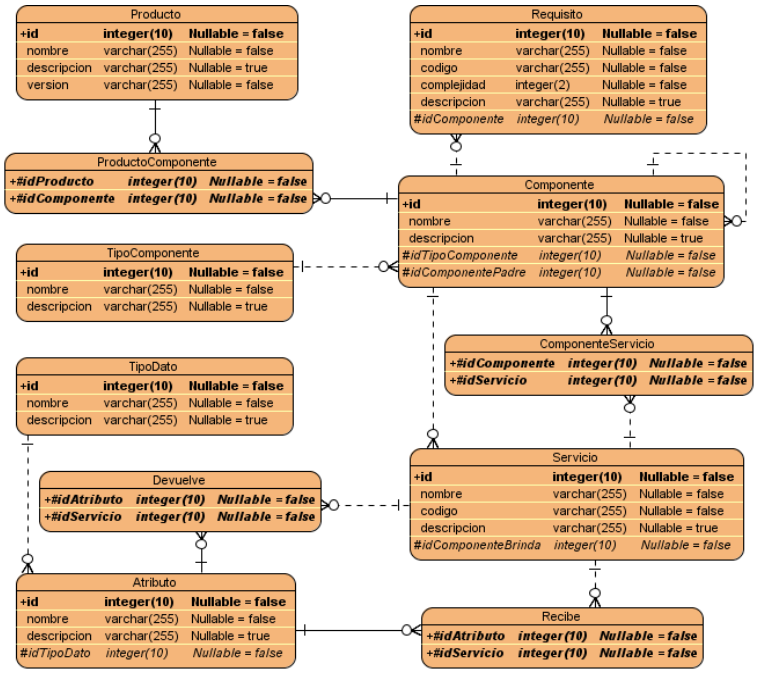


Figura 12: Modelo físico de datos – Diagrama Entidad Relación

[31] Un atributo *simple o atómico* es un atributo no divisible, es decir, que no contiene otros atributos.

Principios del diseño de la aplicación

El diseño universal se basa en “conseguir productos y entornos que sean utilizables por todas las personas sin tener que recurrir a las adaptaciones o a diseños especializados, que a la vez no suponga una inversión extra elevada. El diseño universal intenta favorecer a todas las personas sin tener en cuenta su edad o su grado de habilidad, tanto física como intelectual” (Lidwell, y otros, 2008).

A continuación se abordan los 7 principios del diseño universal según (Lidwell, y otros, 2008).

Principio 1: Igualdad de uso

Según (Lidwell, y otros, 2008) el diseño debe ser fácil de usar y adecuado para todas las personas independientemente de sus capacidades y habilidades. Además debe cumplir con las siguentes pautas: Proporcionar las mismas formas de uso para todos: idénticas cuando sea posible y equivalentes cuando no. Evitar segregar [32] o estigmatizar[33] a cualquier usuario. Todos los usuarios deben contar con las mismas garantías de privacidad y seguridad. El diseño debe ser agradable para todos.

Principio 2: Flexibilidad

Según (Lidwell, y otros, 2008) el diseño debe permitir adecuarse a un amplio rango de preferencias y habilidades individuales. Debe cumplir además con las pautas: Ofrecer las posibilidades de elección en los métodos de uso. Que pueda accederse y usarse tanto con la mano derecha como con la izquierda. Facilite al usuario la exactitud y precisión.

Principio 3: Simple e intuitivo

Según (Lidwell, y otros, 2008) el diseño debe ser fácil de entender independientemente de la experiencia, los conocimientos, las habilidades o el nivel de concentración del usuario. Debe cumplir con las siguientes pautas: Eliminar la complejidad innecesaria. Que sea consistente con las expectativas e intuición del usuario. Que se acomode a un amplio rango de alfabetización y habilidades lingüísticas. Que dispense la información de manera consistente con su importancia. Proporcione avisos eficaces y métodos de respuesta durante y tras la finalización de la tarea.

Principio 4: Información perceptible

Según (Lidwell, y otros, 2008) el diseño debe comunicar de manera eficaz la información necesaria para el usuario, sin importar las condiciones ambientales o las capacidades sensoriales del mismo. Debe proporcionar las siguientes pautas: Usar

[32] Segregar: Separar o apartar algo de otra u otras cosas.
[33] Estigmatizar: Afrentar, infamar.

diferentes modos para presentar de manera redundante la información esencial (gráfica, verbal o táctil). Proporcionar un adecuado contraste entre la información esencial y la adicional. Que amplíe la legibilidad de la información esencial.

Principio 5: Tolerancia a errores

Según (Lidwell, y otros, 2008) el diseño debe minimizar los riesgos y las consecuencias adversas de acciones involuntarias o accidentales. Debe cumplir con las siguientes pautas: Disponer los elementos para minimizar los riesgos y errores: los elementos más usados y más accesibles; y los elementos peligrosos eliminados, aislados o tapados. Proporcionar advertencias sobre peligros, errores y características para controlar las fallas.

Principio 6: Escaso esfuerzo físico

Según (Lidwell, y otros, 2008) el diseño debe permitir ser usado eficaz y confortablemente con el mínimo de esfuerzo o fatiga. Además de proveer las siguientes pautas: Permitir al usuario mantener una posición neutral de su cuerpo. Utilizar de manera razonable las fuerzas de accionamiento o funcionamiento. Minimizar las acciones repetitivas y el esfuerzo físico continuado.

Principio 7: Dimensiones apropiadas

Según (Lidwell, y otros, 2008) debe proporcionarse el tamaño y espacio apropiado para el acceso, alcance, manipulación y uso, sin importar el tamaño de cuerpo del usuario, su postura o su movilidad. Ademas debe cumplir con las siguientes pautas: Proporcionar una línea de visión clara de los elementos importantes para cualquier usuario sentado o de pie. El alcance de cualquier componente sea confortable para cualquier usuario sentado o de pie. Garantizar el espacio necesario para el uso de ayudas técnicas o de asistencia personal.

Estándar de codificación

“Un estándar de codificación comprende todos los aspectos de generación de código que se hacen necesarios establecer al comenzar un proyecto de software. Estos estándares sirven para llegar a un desarrollo más limpio y entendible dentro del proyecto.” (López Morales, y otros, 2010)

Cabecera del archivo

“Siempre es importante que todos los archivos .php inicien con una cabecera específica que indique información de la versión, el autor de los últimos cambios, etc.” (López, 2012)

Ejemplo:

```
/**
 * producto actions.
 *
 * @package    BejucoX
 * @subpackage producto
 * @author     Dayaris Rodriguez Pupo
 */
```

Figura 13: Ejemplo de cabecera del archivo productoActions

Comentarios de las funciones

"Todas las funciones deben tener un comentario antes de su declaración explicando que hacen. Ningún programador debería tener que analizar el código de una función para conocer su utilidad." (López Morales, y otros, 2010)

Ejemplo:

```
// Esta función elimina un/unos productos por llave primaria
    public function delete(sfWebRequest $request) {
        $pks = json_decode(stripslashes($request->getParameter('ids')));
        return Doctrine::getTable('Producto')->deleteByPK($pks);
    }

}
```

Figura 14: Ejemplo de comentario en las funciones

Clases

"Las clases deben ser colocadas en un archivo .php donde se escribirá el código correspondiente a la clase. El nombre del archivo será el mismo de la clase y siempre empezará en mayúscula. En lo posible se debe procurar que los nombres de clase tengan una sola palabra.

Las clases siguen las mismas reglas de las funciones, por tanto, debe colocarse un comentario antes de la declaración de la clase explicando su utilidad." (López Morales, y otros, 2010)

Estructura de control

"Estas incluyen: if, for, while, switch, etc. Aquí esta el ejemplo de la postura "if"." (López Morales, y otros, 2010)

Ejemplo:

```
<?php
        if ((condition1) || (condition2))
        {
                action1;
        }
        else
                if ((condition3) && (condition4))
        {
                action2;
        }
        else
        {
                defaultaction;
        }
?>
```

Llamadas de Función

"Las funciones deberán de ser llamadas sin espacios entre el nombre de la función, el abrir paréntesis y el primer parámetro; espacios entre comas y cada parámetro, y sin espacios entre el último parámetro, el cierra paréntesis y punto y coma." (Lopez, 2012)

Ejemplo:

```
<?php
        $var = foo ($bar, $baz, $ quux);
?>
```

Inclusión de código

"En cualquier parte que se esté incondicionalmente incluyendo un archivo de una clase, se debe usar required_once.

En cualquier parte donde se esté condicionalmente incluyendo un archivo de clase se debe usar include_once.

Cualquiera de esas opciones se asegurará que el archivo de la clase sea incluido únicamente una vez. Ellas comparten la misma lista de archivo así que no hay necesidad de preocuparse por mezclarlas, un archivo incluido con required_once no será nuevamente incluido por include_once." (López Morales, y otros, 2010)

Convención de nombres de variables

"Estas deberán nombrarse con un prefijo de tres letras el cual define el tipo de dato de la misma. Los nombres deber ser descriptivos y concisos, no usar ni grande frases ni pequeñas abreviaturas para las variables.

Todos los nombres deben estar en minúsculas. En caso de usar más de una palabra, esta será separada por un signo de underscore '_'." (López Morales, y otros, 2010)

Ejemplo:

$txt_descripcion: Campo de tipo text.

Espacios entre signos

"Si se tienen signos binarios se deben poner espacios a ambos lados y en caso de signos unario los espacios deben ir a uno de sus lados." (López Morales, y otros, 2010)

Ejemplo incorrecto:

```
        $a=0;
        for($i=5;i<=$j;$i++)
```

Ejemplo correcto:

```
        $a = 0;
        for ($i = 5; i <= $j; $i++)
```

Cadenas de texto entre comillas

"PHP tiene dos formas de poner cadenas de texto (string), con comillas simples y con comillas dobles. La diferencia es que cuando se usa comillas dobles y se coloca dentro

del texto un nombre de variable, el compilador lo interpretará y reemplazará por su valor. Por ésta razón siempre se ha de usar comillas simples a menos que se necesite hacer la interpolación de variables que permiten las dobles." (López Morales, y otros, 2010)

Instrucción "switch"

Cuando se usa este tipo de instrucción es aconsejable utilizar el siguiente estilo:

```
switch ($request->getParameter('component')) {
    case 'combo':
        $rows = ProductoTable::getInstance()->getAll($query);
        break;
    case 'grid':
        $start = $request->getParameter('start');
        $limit = $request->getParameter('limit');
        $filter = json_decode(stripslashes($request->getParameter('filter')));

        $rows = ProductoTable::getInstance()->getAllPaged($start, $limit, $query, $filter);
        break;
    default:
        break;
}
```

Figura 15: Instrucción switch del método load () correspondiente a la clase productoActions.

Números dentro del código

"En ocasiones se ponen números especiales dentro del código para situaciones especiales. Es fundamental que no se haga de esa forma. Si se necesita poner un número especial primero se debe convertir en constante y entonces impleméntalo." (López Morales, y otros, 2010)

Ejemplo:

```
define ('articulos_portada', 10);
for ($i = 0; $i < articulos_portada; $i++)
{
}
```

Guardar archivos PHP

"Los archivos con código PHP deben de ser guardados en formato ASCII utilizando la codificación ISO-8859-1. El formato con esta codificación es en el que se guardan los archivos de texto plano (.txt).

La razón de este estándar es que determinados editores HTML (en especial Dreamweaver), agregan códigos de carácter extraño de salto de línea (como si se tratara de un archivo binario) y esto puede ocasionar que el interprete de PHP encuentre problemas a la hora de leer el script." (López Morales, y otros, 2010)

Conclusiones del Capítulo

En el presente capítulo se realizó el diseño de la herramienta propuesta para facilitar la operabilidad de la matriz de integración de CedruX. Para ello se recopilaron los requisitos a cumplir por el sistema (ubicados en el Product Backlog) y diseñar la BD.

A medida que se fue desarrollando la solución propuesta se realizaron iteraciones (Sprint Backlog) para dar cumplimiento a los requerimientos antes mencionados teniendo presente los principios del diseño.

Se tuvieron en cuenta los aspectos de Autorización, Seguridad y Autenticación que brinda el plugin sfDoctrineGuardPlugin y con la utilización de la función simplexml_load_file se solucionó el requisito “Cargar XML”.

Se realizaron los diagramas de clases del diseño bajo el principio del patrón arquitectónico MVC.

Capítulo 3: Validación de la propuesta

Introducción al Capítulo

En el presente capítulo se realizan las validaciones de software para el diseño y código desarrollado.

El diseño se validará a través de las métricas Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC) para evaluar la calidad del sistema propuesto de acuerdo a los atributos (Responsabilidad, Complejidad de implementación, Reutilización, Acoplamiento, Complejidad del mantenimiento y la Cantidad de pruebas realizadas).

Para garantizar la calidad del código y los requisitos del producto se realizarán pruebas de caja blanca.

Validaciones de Software

Las validaciones de software consisten en los procesos que permiten evaluar y revelar la calidad de un producto software. “Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa probando el comportamiento del mismo.” (López, 2011)

“Las pruebas de software se integran dentro de las diferentes fases del ciclo de vida del producto dentro de la Ingeniería de software; para determinar el nivel de calidad se deben efectuar unas pruebas que permitan comprobar el grado de cumplimiento con respecto a las especificaciones iniciales del sistema.” (López, 2011)

“El proceso de prueba es clave a la hora de detectar errores o fallas y conceptos como estabilidad, escalabilidad, eficiencia y seguridad se relacionan a la calidad de un producto bien desarrollado.” (López, 2011)

Validación del diseño

Para validar la calidad del diseño propuesto se hizo necesario aplicar diversas métricas las cuales permiten medir de forma cuantitativa los atributos internos del sistema construido.

“Las métricas se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de software. Inclina sus objetivos a mejorar la comprensión de la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo.” (Vega, 2012)

Atributos de calidad

Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

- ✓ **Responsabilidad:** Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ✓ **Complejidad de implementación:** Consiste en el grado de dificultad que tiene implementado un diseño de clases determinado.
- ✓ **Reutilización:** Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- ✓ **Acoplamiento:** Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de *Reutilización*.
- ✓ **Complejidad del mantenimiento:** Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costos y la planificación del proyecto.
- ✓ **Cantidad de pruebas:** Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

Métrica Tamaño Operacional de Clase (TOC)

El TOC está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad (Vega, 2012):

- ✓ **Responsabilidad:** Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
- ✓ **Complejidad de implementación:** Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- ✓ **Reutilización:** Un aumento del TOC implica una disminución del grado de reutilización de la clase.

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

Figura 16: Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC
Fuente: (Romero, 2010)

Tabla 14: Afectaciones en el diseño según la métrica TOC

Tamaño Operacional de Clase (TOC)	
Atributos	**Afectación**
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.

Fuente: (Driggs Vélez, 2011)

A continuación se muestran las clases del sistema evaluadas en los atributos de calidad Responsabilidad (RESP), Complejidad de implementación (CI) y Reutilización (REUT) (ver Tabla 15).

Tabla 15: Evaluación de las clases del sistema mediante la métrica TOC

No	CLASE	# Procedimientos	RESP	CI	REUT
1	productoActions	5	Baja	Baja	Alta
2	ProductoTable	6	Media	Media	Media
3	componenteActions	5	Baja	Baja	Alta
4	ComponenteTable	6	Media	Media	Media
5	requisitoActions	5	Baja	Baja	Alta
6	RequisitoTable	6	Media	Media	Media
7	servicioActions	5	Baja	Baja	Alta
8	ServicioTable	6	Media	Media	Media
9	atributoActions	5	Baja	Baja	Alta
10	AtributoTable	6	Media	Media	Media
11	tipoDatoActions	5	Baja	Baja	Alta
12	TipoDatoTable	6	Media	Media	Media
13	tipoComponenteActions	5	Baja	Baja	Alta
14	TipoComponenteTable	6	Media	Media	Media

15	ComponenteServicioTable	4	Baja	Baja	Alta
16	ProductoComponenteTable	4	Baja	Baja	Alta
17	RecibeTable	4	Baja	Baja	Alta

A continuación se muestra una gráfica con agrupaciones por intervalos de la cantidad de clases según la cantidad de procedimientos (ver Figura 17).

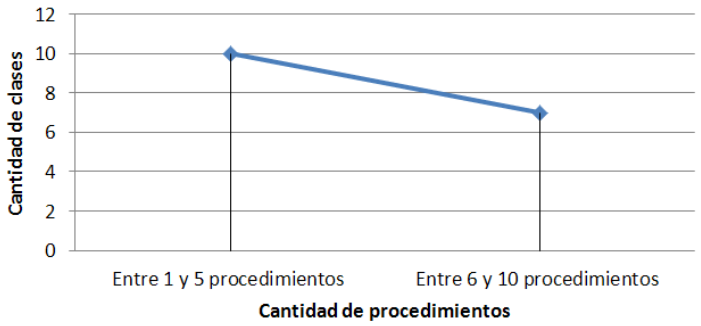


Figura 17: Representación de la cantidad de clases agrupadas en intervalos según la cantidad de procedimientos.

El sistema cuenta con 10 clases que se encuentran entre 4 y 5 procedimientos y 7 clases con 6 procedimientos cada una.

En la Figura 18 se representan los resultados obtenidos al evaluar el diseño de las clases en el atributo Responsabilidad.

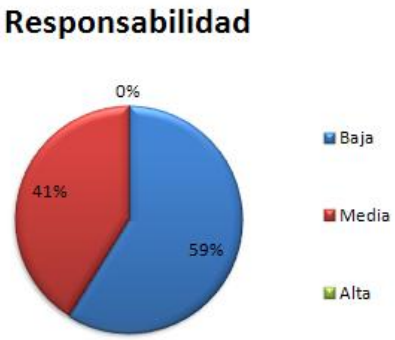


Figura 18: Representación en por ciento (%) de los resultados obtenidos al evaluar las clases en el atributo Responsabilidad.

Se obtuvo que el 41% de las clases poseen un valor Medio y el 59% el valor Bajo, implicando que solo le fue asignada una leve responsabilidad a cada clase.

En la Figura 19 se muestran los resultados obtenidos luego de evaluar el atributo Complejidad de implementación en las clases del sistema.

Los valores obtenidos al evaluar las clases fueron 41% Media y 59% Baja considerándose que la complejidad de implementación se encuentra de manera apropiada.

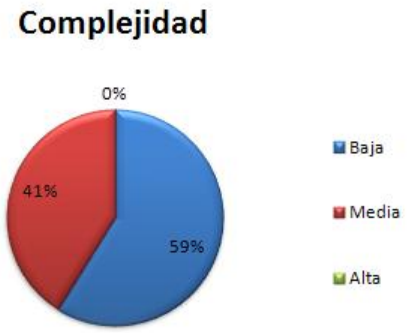


Figura 19: Representación en por ciento (%) de los resultados obtenidos al evaluar las clases en el atributo Complejidad de implementación.

En la Figura 20 se muestran los resultados obtenidos al evaluar las clases en el atributo Reutilización.

Figura 20: Representación en por ciento (%) de los resultados obtenidos al evaluar las clases en el atributo Reutilización.

Como resultado de la evaluación anterior se obtuvo que solo el 41% de las clases poseen una Reutilización Media y el 59% Alta, indicando que la mayor parte de las mismas puede ser reutilizada.

Relaciones entre clases (RC)

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad (Vega, 2012):

- ✓ **Acoplamiento:** Un aumento del RC implica un aumento del Acoplamiento de la clase.
- ✓ **Complejidad de mantenimiento:** Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
- ✓ **Cantidad de pruebas:** Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.
- ✓ **Reutilización:** Un aumento del RC implica una disminución en el grado de reutilización de la clase.

Atributo	Categoría	Criterio
Acoplamiento	Baja	1
	Media	2
	Alta	>2
Complejidad del mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Figura 21: Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica RC

Fuente: (Romero, 2010)

Tabla 16: Afectaciones en el diseño según la métrica RC

Relaciones entre Clases (RC)	
Atributos	**Afectación**
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.

Fuente: (Driggs Vélez, 2011)

A continuación se muestran las clases del sistema evaluadas en los atributos de calidad Acoplamiento (A), Complejidad del mantenimiento (CM), Cantidad de pruebas (CP) y Reutilización (REUT) (ver Tabla 17).

Tabla 17: Evaluación de las clases del sistema mediante la métrica RC

No	CLASE	# Relaciones	A	CM	REUT	CP
1	productoActions	6	Alto	Alta	Baja	Alta
2	ProductoTable	0	Ninguno	Baja	Alta	Baja
3	componenteActions	8	Alto	Alta	Baja	Alta
4	ComponenteTable	0	Ninguno	Baja	Alta	Baja
5	requisitoActions	6	Alto	Alta	Baja	Alta
6	RequisitoTable	0	Ninguno	Baja	Alta	Baja
7	servicioActions	8	Alto	Alta	Baja	Alta
8	ServicioTable	0	Ninguno	Baja	Alta	Baja
9	atributoActions	5	Alto	Media	Media	Media
10	AtributoTable	0	Ninguno	Baja	Alta	Baja

11	tipoDatoActions	5	Alto	Media	Media	Media
12	TipoDatoTable	0	Ninguno	Baja	Alta	Baja
13	tipoComponenteActions	5	Alto	Media	Media	Media
14	TipoComponenteTable	0	Ninguno	Baja	Alta	Baja
15	ComponenteServicioTable	0	Ninguno	Baja	Alta	Baja
16	ProductoComponenteTable	0	Ninguno	Baja	Alta	Baja
17	RecibeTable	0	Ninguno	Baja	Alta	Baja

A continuación se muestra una gráfica con agrupaciones por intervalos de la cantidad de clases según las dependencias entre ellas (ver Figura 22).

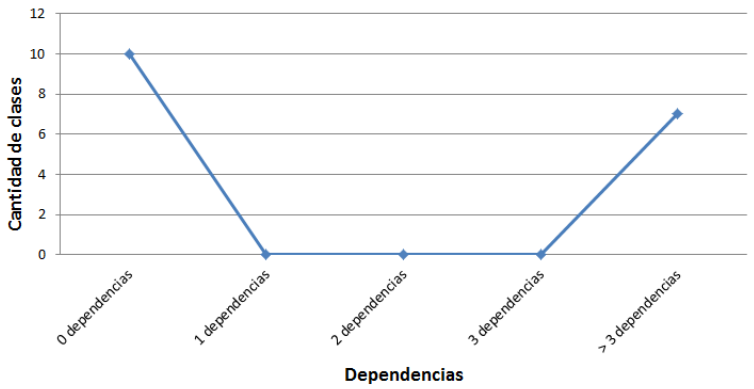


Figura 22: Representación de la cantidad de clases agrupadas en intervalos según las dependencias entre ellas.

El sistema cuenta con 10 clases que poseen una dependencia 0 y las restantes 7 clases con una dependencia superior a 3.

En la Figura 23 se muestran los valores en por ciento obtenidos al evaluar el diseño de las clases en el atributo Acoplamiento.

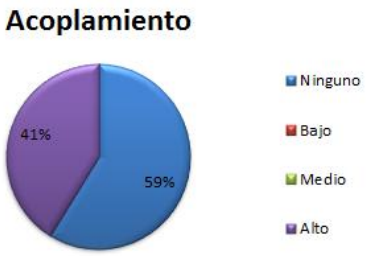


Figura 23: Representación en por ciento (%) de los resultados obtenidos al evaluar las clases en el atributo Acoplamiento.

Luego de la evaluación anterior se obtuvo que el 41% de las clases poseen un valor alto y el 59% no posee ningún acoplamiento.

En la Figura 24 se muestran los resultados obtenidos luego de evaluar las clases en el atributo Complejidad de Mantenimiento.

Los valores obtenidos fueron: 23% Alta, 18% Media y 59% Baja Complejidad de Mantenimiento de las clases.

Figura 24: Representación en por ciento (%) de los resultados obtenidos al evaluar las clases en el atributo Complejidad de Mantenimiento.

En la Figura 25 se muestran los resultados obtenidos al evaluar las clases el atributo Cantidad de Pruebas.

Figura 25: Representación en por ciento (%) de los resultados obtenidos al evaluar las clases en el atributo Cantidad de Pruebas.

En los resultados anteriores se puede observar que un 23% de las clases cuentan con un valor Alto, el 18% Media y el 59% Baja; implicando una disminución en el número de pruebas a realizar para probar una determinada clase.

En la Figura 26 se muestran los resultados obtenidos al evaluar las clases en el atributo Reutilización.

Figura 26: Representación en por ciento (%) de los resultados obtenidos al evaluar las clases en el atributo Reutilización.

Como resultado de la evaluación anterior se obtuvo que 23% de las clases poseen una Reutilización Baja, el 18% Media y el 59% Alta, indicando que solo una pequeña parte de las mismas puede ser reutilizada.

Pruebas de Caja Blanca o Estructurales

“La prueba de caja blanca del software se basa en un examen cercano al detalle procedimental. Se prueban las rutas lógicas del software y la colaboración entre los componentes, al proporcionar casos de prueba que ejerciten conjuntos específicos de condiciones, bucles o ambos.” (Pressman, 2010)

Según (Pressman, 2010) las pruebas de caja blanca intentan garantizar que:

- ✓ Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- ✓ Se utilizan las decisiones en su parte verdadera y en su parte falsa.
- ✓ Se ejecuten todos los bucles en sus límites.
- ✓ Se utilizan todas las estructuras de datos internas.

Entre las diferentes técnicas de Caja Blanca que existen se decidió utilizar en la presente investigación la prueba de la **ruta básica**.

“El método de la ruta básica permite que el diseñador de casos de prueba obtenga una medida de complejidad lógica de un diseño procedimental y que use esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de pruebas derivados para ejercitar el conjunto básico deben garantizar que se ejecuta cada instrucción del programa por lo menos una vez durante la prueba.” (Pressman, 2010)

Antes de comenzar con el método de la ruta básica se debe presentar una notación simple para la representación del flujo de control, llamado *gráfica de flujo*.

La gráfica describe un flujo de control lógico empleando la notación que se muestra en la Figura 27.

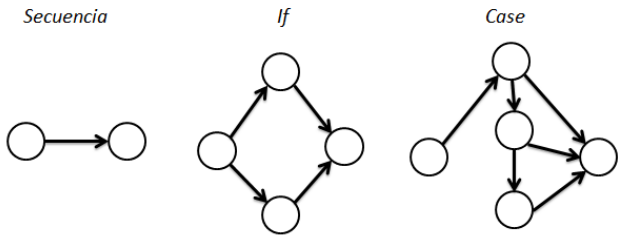


Figura 27: Gráfica de flujo para las instrucciones Secuenciales, If y Case.

Cada gráfico de flujo va a estar compuestos por 3 elementos fundamentales (Ver Figura 28):

- ✓ Nodo: representa una o más sentencias procedimentales.

- ✓ Aristas o enlaces: Representan el flujo de control. Una arista debe terminar en un nodo, aunque el nodo no represente ninguna instrucción procedimental.
- ✓ Regiones: Son las áreas que limitan aristas y nodos. Cuando se cuentan las regiones se incluyen las áreas ubicadas fuera de la gráfica.

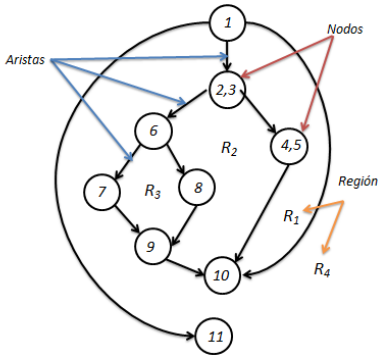


Figura 28: Representación de nodos, aristas y regiones

Como parte de la ruta básica es necesario identificar las rutas independientes.

"Una ruta independiente es cualquier ruta del programa que ingresa por lo menos un conjunto de instrucciones de procesamiento o una nueva condición. Cuando se explica desde el punto de vista de una gráfica de flujo, una ruta independiente debe recorrer por lo menos una arista que no se haya recorrido antes." (Pressman, 2010)

Para saber la cantidad de rutas independientes que se deben buscar es necesario calcular la complejidad ciclomática.

"La complejidad ciclomática es una métrica de software que proporciona una medida cuantitativa de la complejidad lógica de un programa. Cuando se emplea en el contexto del método de prueba de la ruta básica, el valor calculado mediante la complejidad ciclomática define el número de rutas independientes en el conjunto básico de un programa y proporciona un límite superior para el número de pruebas que deben aplicarse para asegurar que todas las instrucciones se hayan ejecutado por lo menos una vez." (Pressman, 2010)

La complejidad ciclomática se basa en la teoría gráfica y se calcula se tres maneras (Pressman, 2010):

1) El número de regiones corresponde a la complejidad ciclomática.
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.

3) 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 predicado[34] incluidos en la gráfica de flujo G.

"El método de prueba de la ruta básica se aplica a un diseño procedimental o al código fuente, para ello se deben realizar los casos de pruebas que forzaran la ejecución de cada ruta en el conjunto básico. Es necesario seleccionar los datos de manera tal que se establezca apropiadamente las condiciones de los nodos predicados a medida que se prueba cada ruta. Cada caso de prueba se ejecuta y compara con los resultados esperados. Una vez completados todos los casos, la persona que aplica la prueba puede estar segura de que todas las instrucciones del programa se han ejecutado por lo menos una vez." (Pressman, 2010)

Resultados de las pruebas

Se realizaron un total de 146 pruebas (correspondientes a 17 clases) de caja blanca mediante la técnica ruta básica, de ellas 56 (38%) se llevaron a cabo en las clases actions (7 clases) y 90 (62%) en las Table (10 clases).

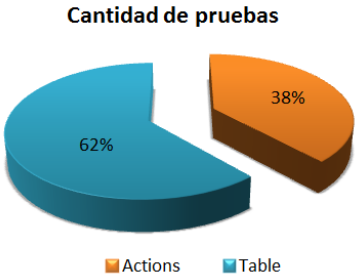


Figura 29: Representación en por ciento (%) de la cantidad de pruebas de caja blanca realizadas a los procedimientos.

A continuación se mostrarán los resultados obtenidos correspondientes a la clase ProductoActions en específico al método save () encargado de guardar y modificar un Producto.

```
public function save(sfWebRequest $request) {
1   $producto = array();
    if ($request->getParameter('id') != '') 2
      3 $producto = Doctrine::getTable('Producto')
                ->find($request->getParameter('id'));

4   if ($producto == array()) {
        $producto = new Producto(); 5
    }
    $producto->setNombre($request->getParameter('nombre'));
    $producto->setVersion($request->getParameter('version'));
    $producto->setDescripcion($request->getParameter('descripcion'));
6
    $jks = json_decode(stripslashes($request->getParameter('componentes')));
    $producto->link('Componentes', $jks);

    $producto->save();
}
```

Figura 30: Método para guardar y modificar un Producto.

[34] Nodos predicados: Es cada nodo que contiene una condición. Se caracteriza porque de él emanan dos o más aristas.

Tomando como referencia la Figura 30 se construyó la gráfica de flujo correspondiente.

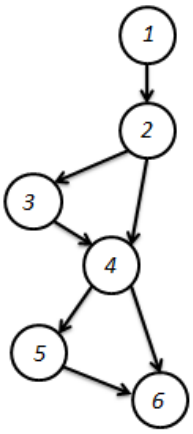


Figura 31: Gráfico de flujo correspondiente al método save ().

Para saber cuántas rutas independientes se van a buscar se debe calcular primero la complejidad ciclomática según se describió anteriormente.

1) Número de regiones encontradas = 3.
2) $V(G) = E - N + 2$
   V (G) = 7 – 6 + 2
   V (G) = 3
3) $V(G) = P + 1$
   V (G) = 2 + 1
   V (G) = 3

Las rutas independientes a probar son solo 2 (son los únicos resultados que se pueden obtener del método analizado):

Primera ruta independiente: 1, 2, 4, 5, 6.

Segunda ruta independiente: 1, 2, 3, 4, 6.

A continuación se describen los casos de pruebas a realizar para comprobar que los resultados obtenidos sean iguales a los esperados y así evaluar el funcionamiento del método save ().

En los casos de pruebas se medirán los siguientes campos:

- ✓ Descripción: Se describe brevemente lo que debe hacer el método al ejecutar la ruta seleccionada para probar.
- ✓ Precondición: Se especifica que debe existir para que se ejecute correctamente el método. Su valor puede ser N/A que significa que para esa ruta no aplica ese campo.

- ✓ Entrada: Se muestran los parámetros que recibe por parámetro éste procedimiento.
- ✓ Respuesta del sistema: se muestran los resultados obtenidos luego de ejecutar el método.
- ✓ Resultados esperados: Se describe el resultado que se espera que devuelva el procedimiento ejecutado.

Tabla 18: Caso de prueba para la ruta independiente 1, 2, 4, 5, 6

Caso de prueba para la ruta 1, 2, 4, 5, 6	
Descripción	El sistema debe adicionar un Producto con los valores que se le pasa por parámetro.
Precondición	N/A
Entrada	$request
Respuesta del sistema	Se mostró la ventana de información “El producto se ha guardado satisfactoriamente” y se adicionó el producto.
Resultados esperados	Se debe adicionar un nuevo producto en el sistema.

Tabla 19: Caso de prueba para la ruta independiente 1, 2, 3, 4, 6

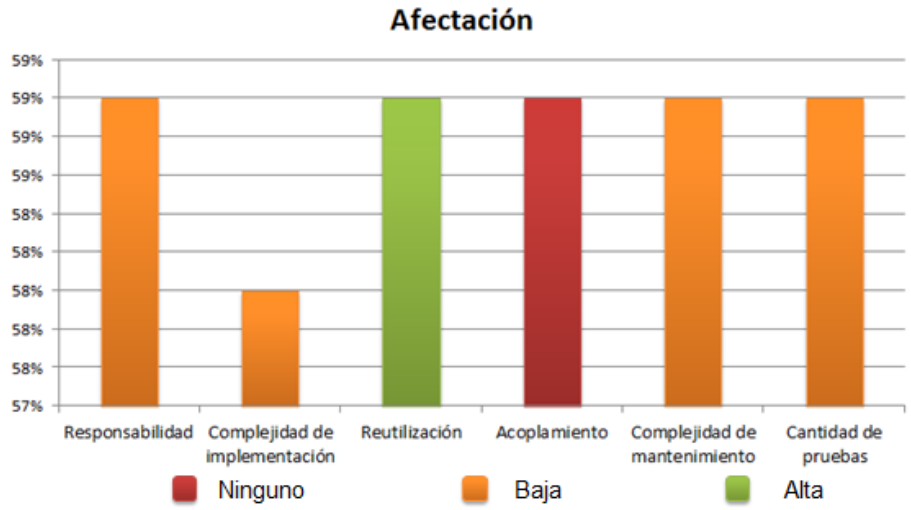
Caso de prueba para la ruta 1, 2, 3, 4, 6	
Descripción	El sistema debe modificar un Producto ya existente en el sistema con los nuevos valores que se le pasa por parámetro.
Precondición	Debe existir el producto previamente creado.
Entrada	$request
Respuesta del sistema	Se mostró la ventana de información “El producto se ha guardado satisfactoriamente” y se modificó el producto existente.
Resultados esperados	Se debe modificar un producto ya existente en el sistema.

Conclusiones del capítulo

En el presente capítulo se realizó la validación de la propuesta de solución fragmentada en dos partes fundamentales: las pruebas de caja blanca para comprobar el correcto funcionamiento de los procedimientos implementados, la validación del diseño mediantes las métricas TOC y RC.

Los resultados obtenidos mostraron una visión general del éxito en el diseño e implementación del producto propuesto.

A continuación se muestra una figura con los por cientos representativos obtenidos al evaluar los atributos de calidad establecidos por las métricas TOC y RC.



De los resultados obtenidos sólo se considera como crítico el atributo “Acoplamiento” con un 59% de las clases evaluadas de Ninguno.

Luego de realizar las pruebas de caja blanca se obtuvo que cada uno de los procedimientos implementados en las 17 clases se ejecutan al menos una vez.

Conclusiones Generales

El estudio realizado en el marco teórico de la investigación permitió conocer los principales elementos referentes a la operabilidad de la matriz de integración y su papel en la toma de decisiones de los arquitectos del sistema con respecto a la implementación y/o soporte de algún componente determinado.

El diseño e implementación de un sistema que facilite la operabilidad de la matriz de integración de CedruX permitió el ahorra de tiempo y recursos en el proceso de construcción de la misma. Considerándose que la solución propuesta ayuda de manera constructiva en el proceso de toma de decisiones de los arquitectos del sistema.

La validación del sistema propuesto permitió conocer que tan bien se había llevado a cabo el diseño y la implementación del producto.

Con la herramienta obtenida se logró mejorar la operabilidad de gestión de la matriz de integración, se redujo la cantidad de artefactos generados, el tiempo de lectura de ficheros para identificar los servicios brindados y consumidos por cada componente. Además permitió minimizar el tiempo para calcular los indicadores de tamaño, complejidad, criticidad y el tiempo para generar la matriz de integración. Con la variación de estos indicadores los arquitectos de sistema podrán tomar decisiones sobre la dependencia, prioridad de desarrollo y soporte de los componentes de manera más rápida que utilizando los artefactos que se manejaban hasta hoy.

Recomendaciones

Teniendo en cuenta los resultados obtenidos con la presente investigación se recomienda:

1) El despliegue del sistema propuesto para el uso por los arquitectos del sistema.
2) Agregar la funcionalidad que permita generar gráficos de los indicadores: Tamaño, Complejidad y Criticidad de los componentes para ser utilizado como recurso visual en el proceso de toma de decisiones.

Bibliografía

Gil, Fidel , Albrigo , Javier y Do Rosario, Javier . febrero 2005. *Sistemas de Gestión de Base de Datos.* Departamento de Computación, Facultad Experimental de Ciencias y Tecnología. Valencia : Universidad de Carabobo, febrero 2005.

Adobe Systems Incorporated. 2012. Adobe. *Adobe Dreamweaver.* [En línea] Adobe, 2012. [Citado el: 20 de Enero de 2012.] http://www.adobe.com/es/products/dreamweaver.html.

Alarcón, Willian Miguel Viltres. 2011. EcuRed. *Rational Rose Enterprise Edition.* [En línea] 2011. [Citado el: 12 de Enero de 2012.] http://www.ecured.cu/index.php/Rational_Rose_Enterprise_Edition.

Bahit, Eugenia. 2011. *El paradigma de la Programación Orientada a Objetos en PHP y el patrón de arquitectura de Software MVC.* 2011. págs. 35, 36, 37, 38.

Barry Boehm & Richard Turner. *¿Metodología Ágil o Formal?*

Baryolo, Oiner Gómez y y otros. 2008. *Plantilla Registro de la Propiedad intelectual (Sauxe).* 2008.

CEIGE. 2011. *Arquitectura de Software.* 2011. CIG-LOG-N-QUI-i3401.

—. 2011. *Documento línea base de subsistema.* 2011.

Cerda, Felipe. 2009. *NetBeans: El único IDE que necesitas.* 2009.

Cuellar, Yalexy González. 2012. EcuRed. *Diagrama de despliegue.* [En línea] 05 de Enero de 2012. [Citado el: 18 de Febrero de 2012.] http://www.ecured.cu/index.php/Diagrama_de_despliegue.

del Toro Ríos, Dr. José Carlos, y otros. 2009. *TALLER INTERNACIONAL "LAS TIC EN LA GESTION DE LAS ORGANIZACIONES".* La Habana : Informática 2009, 2009.

Departamento de Programación. 2008. *Programación.* Celaya : Instituto Tecnológico de Celaya , 2008.

Driggs Vélez, Jorge Carlos. 2011. *Diseño e Implementación de la nueva versión del Módulo Banco del Sistema Integral de Gestión Cedrux.* La Habana : s.n., 2011.

DuBois, Paul. 2009. *MySQL Developer's Library.* s.l. : Pearson Education, 2009. ISBN-13: 978-0-672-32938-8.

Durán, Javier Ruiz. 2012. Escalabilidad. *Razones para usar Sauxe.* UCI, CEIGE : Dpto de Tecnología, 22 de Febrero de 2012.

Eguiluz, Javier. 2012. Symfony. *¿Qué es Symfony?* [En línea] 2012. [Citado el: 11 de Noviembre de 2011.] http://www.symfony.es/que-es-symfony/.

EllisLab. 2012. CodeIgniter. [En línea] 2012. [Citado el: 14 de Enero de 2012.] http://codeigniter.com/.

Enrico Tröger, Matthew Brush, Colomban Wendling, Frank Lanitz, Nick Treleaven,Dominic Hopf. 2011. Geany. [En línea] 2011. [Citado el: 20 de Enero de 2012.] http://www.geany.org/.

Escribano, Gerardo Fernández. 2007. *Introducción a Extreme Programming.* 2007.

Garlan, David & Shaw, Mary. 1996. *Software Architecture: Perspectives on an emerging discipline. Upper Saddle Rive, Prentice Hal.* 1996.

Gilfillan, Ian. 2010. *La Biblia de MySQL.* 2010.

Hernández León, Rolando Alfredo y Coello Hernández, Zayda. 2011. *El proceso de la Investigación Científica.* La Habana : Editorial Universitaria, 2011. ISBN 978-959-16-1307-3.

Hernández, Rubén Font. 2012. EcuRed. *XML.* [En línea] 18 de Abril de 2012. [Citado el: 7 de Mayo de 2012.] http://www.ecured.cu/index.php/XML.

INEI. 1999. *Herramientas Case.* 1999. 875-99-OI-OTDETI-INEI.

Instituto Tecnológico de Veracruz. 2009. Lenguajes de programación del lado del cliente. [En línea] 2009. [Citado el: 18 de Diciembre de 2011.] http://prograweb.com.mx/pweb/0202ladoCliente.html.

ISO/IEC, 9126-1. 2005. *ISO/IEC 9126-1: 2005.* s.l. : Oficina Nacional de Normalización, 2005.

Itzcoalt, Alvarez M. 2007. *Desarrollo Ágil con Scrum.* 2007.

Krall, César. 2011. *Notepad++, un útil editor para.* 2011.

Laffita, Alezenny Sablón. 2011. EcuRed. *IDE de Programación.* [En línea] 2011. [Citado el: 14 de Enero de 2012.] http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

Lage, Ing. Cesar, Muro, Ing. Dorisbel y Lazo Ochoa, Ing. René. 2009. *Reporte de la disciplina de la arquitectura del software desde el enfoque del Rol. Una propuesta de epecialización jerárquica centrada en la responsabilidades y competencias.* UCI, Ciudad de la Habana : s.n., 2009.

Language Unified Modeling. 2011. UML Resource Page. [En línea] 2011. http://www.uml.org/.

Larsson, S. 2005. *IMPROVING SOFTWARE PRODUCT INTEGRATION. Department of Computer Science and Engineering.* s.l. : University Mälardalen, 2005.

Leyet Fernández, Osmar. 2011. *Propuesta metodológica para la obtención de los componentes de software en los proyectos del sistema Cedrux.* Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2011.

Lidwell, William, Hiddern, Kritina y Butler, Jill. 2008. *Principios universales de diseño.* s.l. : Blume, 2008.

López Morales, Adrian y Izquierdo Orozco, Julio. 2010. Estándar de codificación de PHP. [En línea] Universidad de Córdoba, Colombia, 2010. [Citado el: 14 de Marzo de 2012.] http://www.aves.edu.co/ovaunicor/?anonimo=explorar&item=recientes&subItem=1&recurso=266#.

López, Cristian Fernandez. 2012. Razones para usar Sauxe. *Licencia.* CEIGE, Asesor de Comercializacion : UCI, 20 de Febrero de 2012.

López, Yuniel Valdés. 2011. EcuRed. *Pruebas de software.* [En línea] 4 de Noviembre de 2011. [Citado el: 20 de Abril de 2012.] http://www.ecured.cu/index.php/Pruebas_de_software.

Luegues, Greidys Jorda. 2007. Wikipedia de Producción. [En línea] UCI, 2007. [Citado el: 24 de Abril de 2012.] http://ucipedia.uci.cu/index.php/Diagrama_de_clases.

Manes, Juan Pablo Pérez. 2010. EcuRed. *PHP.* [En línea] 2010. [Citado el: 20 de Diciembre de 2011.] http://www.ecured.cu/index.php/PHP#Caracter.C3.ADsticas_de_PHP.

Martínez Fajardo, Alexander y Méndez Pérez, Javier. 2010. *Sistema de Gestión de Materiales Audiovisuales.* Universidad de las Ciencias Informáticas. Santiago de Cuba : s.n., 2010.

Martínez, Alejandro y Martínez, Raúl. 2009. *Guía a Rational Unified Process.* Universidad de Castilla la Mancha : s.n., 2009.

Martínez, Relvis González. 2011. EcuRed. *Symfony.* [En línea] 26 de Octubre de 2011. [Citado el: 16 de Noviembre de 2011.] http://www.ecured.cu/index.php/Symfony.

Morejón, Leysmis Campillo. 2010. EcuRed. *Visual Paradigm.* [En línea] 21 de Septiembre de 2010. [Citado el: 9 de Enero de 2012.] http://www.ecured.cu/index.php/Visual_Paradigm.

Navarro, Jose Angel Franco. 2009. *UML en acción. Modelando Aplicaciones Web.* La Habana : CUJAE, 2009.

Oracle Corporation. 2010. Java. [En línea] 2010. [Citado el: 26 de Noviembre de 2011.] http://www.java.com/es/download/faq/whatis_java.xml.

—. 2011. MySQL. [En línea] 2011. [Citado el: 9 de Enero de 2012.] http://www.mysql.com/.

—. 2010. NetBeans. [En línea] 2010. [Citado el: 11 de Enero de 2012.] http://netbeans.org/index_es.html.

Palacio, Juan. 2008. *El modelo Scrum.* 2008.

Pérez Pérez, Joisel. 2011. *Desarrollo de una solución para la obtención de los indicadores financieros en Cedrux.* Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2011.

Pérez, Arián Jesús Pérez. 2011. EcuRed. *Sencha Ext JS.* [En línea] 26 de Octubre de 2011. [Citado el: 16 de Noviembre de 2011.] http://www.ecured.cu/index.php/Sencha_Ext_JS.

Pérez, Dr Pedro Y. Piñero y Vázquez, MsC Maikel Yelandi Leyva. *Metodologías ágiles y formales o robustas.*

Pérez, Javier Eguíluz. 2008. *Introducción a XHTML.* 2008.

Pierra Fuentes, MSc. Allan. 2011. *¿Por qué Migramos?* La Habana : s.n., 2011.

PostgreSQL . 1996 - 2012. PostgreSQL Global Development Group. [En línea] 1996 - 2012. [Citado el: 9 de Enero de 2012.] http://www.postgresql.org/.

Potencier, Fabien & Zaninotto, François. 2008. *Symfony la guía definitiva .* 2008.

Potencier, Fabien. 2011. Symfony. [En línea] 2011. [Citado el: 11 de Noviembre de 2011.] http://symfony.com/.

Pressman, Roger S. 2010. *Ingeniería de Software, un enfoque práctico. Quinta edición.* 2010. ISBN:978-0-07-337597-7.

Python Software Foundation. 2011. Python . [En línea] 2011. [Citado el: 29 de Noviembre de 2011.] http://python.org/.

Rodríguez, Leover Armando González. 2011. *Alternativas para el desarrollo de Aplicaciones Web.* Departamento de la Especialidad, Facultad Regional de Granma de la Universidad de las Ciencias Informáticas. 2011.

Romero, Denis Boizan. 2010. *Sistema para gestionar la actividad científica del Departamento de informática de la universidad de Guantánamo.* Departamento de informática , Universidad de Guantánamo. Guantánamo : s.n., 2010.

Sánchez, Jorge. 2011. Problemas del esquema relacional para BD. [En línea] Slideshare, 2011. [Citado el: 24 de Marzo de 2012.] http://www.slideshare.net/gabos/sistemadegestionbasededatos-jorgesanchez.

Sanchez, Tamara Rodríguez. 2011. Componentes liberados por CALISOFT. DESPROD, CEIGE : UCI, 16 de Noviembre de 2011.

Schein, EH. 2006. *Process consultation.* Cambridge: Addison-Wesley Publishing Company : s.n., 2006.

SEI. 1998. Software Engineering Institute. *Process Area Components.* [En línea] 1998. [Citado el: 16 de Enero de 2012.] http://www.sei.cmu.edu.

Silega Martínez, Nemury. Septiembre 2010. *Guía metodológica para gestionar la integración de componentes en los proyectos del sistema CEDRUX.* Ciudad de la Habana : Universidad de las Ciencias Informáticas, Septiembre 2010.

The Apache Software Foundation. 2011. Apache. *Visión general de las nuevas funcionalidades de Apache 2.0.* [En línea] 2011. [Citado el: 17 de Enero de 2012.] http://httpd.apache.org/docs/2.0/es/new_features_2_0.html.

The PHP Group. 2012. PHP. *Página Oficial.* [En línea] 2012. [Citado el: 11 de Diciembre de 2011.] http://www.php.net/.

Torres, Lester González. 2011. EcuRed. *Geany.* [En línea] 18 de Diciembre de 2011. [Citado el: 20 de Enero de 2012.] http://www.ecured.cu/index.php/Geany.

Vega, Aibett Cabrera. 2012. EcuRed. *Métricas de diseño.* [En línea] 09 de Enero de 2012. [Citado el: 13 de Abril de 2012.] http://www.ecured.cu/index.php/M%C3%A9trica_de_dise%C3%B1o.

Vercellis, Carlo. 2009. *Business Intelligence:Data Mining and Optimization for Decision Making.* Milano, Italia : John Wiley & Sons Ltd, 2009.

Wallnau, Kurt y B, Alan. *The current state of CBSE.* s.l. : IEEE Software.

Zambrano, Donel Vázquez. 2008. *Sistema de Ventas Mayoristas.* La Habana : s.n., 2008.