Universidad de las Ciencias Informáticas

FACULTAD 3

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

Título: Generador de código base de la arquitectura de Cedrux

Autor: Cdte. Roberto Aguilar Hidalgo

Tutores: Ing. Yusnier Matos Arias

Ing. Joisel Pérez Pérez

Cotutor: Ing. Yanay Hernández Sosa

Junio 2011

Pensamiento

A nuestros jóvenes"

"Se puede adquirir conocimientos y conciencia a lo largo de la vida, pero jamás en ninguna otra época de su existencia una persona volverá a tener la pureza y el desinterés con que, siendo joven, se enfrenta a la vida."

Fidel Castro Ruz

Declaración de autoría

Declaro que soy el único autor del trabajo “Herramienta para le generación de código base de la arquitectura de Cedrux” y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firma la presente a los ____ días del mes de _____ del año _____.

_____

Roberto Aguilar Hidalgo

Autor

_____ _____

Ing. Yusnier Matos Arias

Tutor

Ing. Joisel Pérez Pérez

Tutor

_____

Ing. Yanay Hernández Sosa

Cotutor

Agradecimientos

A mis padres por haberme traído al mundo, por apoyarme incondicionalmente ante todas las decisiones importantes que he tomado en mi vida, por educarme y formarme como una persona de bien, por sacrificarse para que nunca me faltara nada. A mami le agradezco todo el cariño y el amor que siempre me has brindado, por guiarme en todo momento haciéndome entender lo que es correcto y lo que no. A papi por inculcarme su carácter, por ser mi ejemplo y mi modelo a seguir ante la vida, por enseñarme a no rendirme por dura y difícil que fuera la situación. Ustedes son mi vida

A mis hermanos que han sido mi complemento en la vida. A mi hermano mayor Keilen por cuidar siempre de mí, por enseñarme cosas útiles para la vida, por haberme dado la alegría de ser tío del mejor niño del mundo. A mi hermanito Ricardo que es mi orgullo y nunca me ha defraudado en nada.

A mi novia María Elena Caro por ser tan especial y por brindarme todos los días su apoyo, por ayudarme en los momentos más difíciles dándome la fuerza para seguir adelante.

A mis tutores por el esfuerzo y la guía que me proporcionaron, sin la cual no podría haber realizado este trabajo.

A mis amigos y compañeros, en especial a Pedro Noguera y a Jorge Luis Agüero, más que amigos han sido hermanos para mí. A Manuel Alejandro Castellanos, Damián Falcón, a los integrantes del antiguo 4106(año 2006) y a los integrantes del 3504(año 2011).

A todos los que de una forma u otra aportaron su granito de arena para la realización de este trabajo.

A todos muchas Gracias.

Roberto Aguilar Hidalgo.

Dedicatoria

Este trabajo va dedicado especialmente a mis padres, por el amor y la educación que siempre me han brindado.

Resumen

El país avanza con amplios pasos dentro del proceso de utilización ordenada y masiva de las tecnologías de la información y las comunicaciones, siendo este un factor decisivo para el desarrollo de las empresas, economía y de la sociedad cubana; proceso que busca satisfacer las necesidades de información y conocimiento de todas las personas y esferas de la sociedad. La necesidad de mejorar y agilizar los procesos de desarrollo del software está impulsando a la adopción de herramientas que intervienen en la construcción de estos en cualquier parte del ciclo de vida, ya sea en el análisis, el diseño, la implementación o las pruebas. Con el propósito de agilizar el proceso de construcción del los componentes en los módulos del programa ERP-Cuba el presente trabajo comprende el Diseño y la Implementación de la Herramienta para la generación de código base de la arquitectura del Cedrux.

Palabras claves:

Componentes, Herramientas Case, Interfaces, Puertos

Tabla de contenido

Introducción

La Informática, y como parte de esta, la producción de software han alcanzado, en la actualidad, un elevado auge e importancia a nivel mundial. Su desarrollo crece de forma vertiginosa y con ello la demanda de mejores sistemas en menos tiempo y costo. En el país también se han notado avances en este sentido, ya que la vinculación de todas sus ramas: económicas, políticas y sociales con el mundo Informático son de primordial interés para el Estado Cubano. No solo por los beneficios que trae desde el punto de vista del desarrollo de sistemas para el uso interno, sino también con el objetivo de introducirse en el mercado a escala mundial aprovechando su perspectiva económica.

Para lograr incursionar en el mercado mundial se necesitan software de gran calidad y de funcionalidad probada, pero también es necesario ganar en rapidez y eficiencia a la hora de desarrollarlos, pues hoy en día la producción de software va en ascenso, lo cual pondría en peligro la utilidad del producto que se está creando o la pérdida de mercado, pues pueden aparecer otros que desempeñen las mismas funcionalidades dejando por detrás estos softwares y sin posibilidades de promocionar el trabajo realizado en el mundo.

Un ejemplo que denota la madurez alcanzada en la industria del software en el país es la tarea que lleva a cabo la Universidad de las Ciencias Informáticas (UCI) en conjunto con el Ministerio de Finanzas y Precios de crear un Sistema de Planificación de Recursos Empresariales (ERP: Enterprise Resource Planning) acorde a las necesidades reales de las empresas y basado en el sistema económico nacional. El sistema está basado en los principios de la independencia tecnológica y debe vencer las barreras de software que existen para migrar hacia el Software Libre para aprovechar las nuevas posibilidades que este brinda.

La agilización en el desarrollo de este producto es un aspecto fundamental por la importancia que el mismo representa para el control y desarrollo de la economía del país. En la isla la mayoría de las empresas para llevar el control de sus recursos emplean softwares extranjeros que en su gran mayoría son privativos y no cumplen con las particularidades de la economía cubana, y el uso de estos provoca gastos innecesarios en licencias anuales y soporte. Por lo que producir este sistema con mayor agilidad representaría un importante ahorro de capital para la economía.

Actualmente en el Cedrux (Nombre comercial del ERP-Cuba) se realiza de manera ineficiente la creación de la estructura básica de la arquitectura de los módulos del sistema. Una parte de esta arquitectura es generada automáticamente, como las clases del **modelo** y el acceso a datos**,** en cambio las clases controladoras y las clases de servicios se crean manualmente. Esto crea una carga de trabajo extra para los desarrolladores lo que provoca la dilación del proceso de desarrollo de los componentes que llevan estos módulos y la liberación final del producto como tal.

Hoy en día existen herramientas destinadas a acelerar el proceso de desarrollo del software, las cuales intervienen en cualquier parte del ciclo de vida de este automatizando una parte del proceso, pero estas en su mayoría son privativas y no cumplen con las particularidades para ajustarse a la arquitectura base que presentan los módulos del Sistema Integral de Gestión Cedrux.

Por lo tanto se precisa optimizar la gestión de la arquitectura base del Sistema Cedrux garantizando la agilidad en el proceso de construcción de los módulos de este y disminuyendo en tiempo la implementación de los componentes de los mismos.

A partir de la situación problemática planteada, se define como **problema a resolver** ¿Cómo agilizar el desarrollo de los componentes en los proyectos del programa ERP-Cuba de manera que se reduzca el tiempo de implementación de los mismos?

Como **objeto de estudio** se define la generación de código en sistemas de software y como **campo de acción** se tiene la generación de código en la arquitectura de sistemas de gestión.

Se plantea como **objetivo general** para solucionar el problema: Desarrollar una herramienta que permita generar código base de la arquitectura de los sistemas desarrollados en el ERP-Cuba a partir de la modelación de la misma, de tal manera que contribuya a disminuir el tiempo de implementación de los componentes.

Para el cumplimiento del objetivo propuesto, se proponen los siguientes **objetivos específicos**:

1. Fundamentar la investigación, mediante la elaboración del Marco Teórico.
2. Desarrollar la herramienta para la generación de código base de la arquitectura.
3. Validar la herramienta implementada.

Para realizar con éxito el presente trabajo y cumplir con el objetivo propuesto, han sido definidas las siguientes **tareas investigativas**:

1- Realización del análisis de sistemas informáticos vinculados a la generación de código, los logros y limitaciones en los enfoques existentes.

2- Análisis de los proceso de generación de código en las herramientas dedicadas a generar código web.

3- Investigación de los patrones de diseño, estándares de codificación, herramientas y tecnologías definidos por la Línea de Arquitectura.

4- Diseño e implementación de las clases de la herramienta

5- Realización de pruebas para la evaluación de la solución

Se plantea como **idea a defender** que: Si se desarrolla y aplica una herramienta capaz de generar el código base de la arquitectura de integración del Cedrux, se reducirá el tiempo en el proceso de construcción de los componentes del sistema.

En cumplimiento a las tareas antes mencionadas, se pusieron en práctica los siguientes **métodos de investigación**:

Métodos teóricos:

1- **Analítico**: Posibilitando procesar toda la información enfocada hacia la investigación de los procesos de generación de código.(37)

2- **Sintético:** Permitiendo organizar y simplificar el análisis de todo el volumen de datos a recopilar en fracciones más factibles.(37)

Métodos empíricos:

1- **Experimento**: Favoreciendo el desarrollo de pruebas para la verificación de las funcionalidades implementadas, con el fin de detectar errores y comprobar el correcto funcionamiento del negocio.(37)

2- **Observación**: Para percibir y planificar como quedaría concebido el sistema. (37)

Como **posible resultado** se espera obtener una herramienta para la generación de código base de la arquitectura de los sistemas desarrollados en los proyectos del programa ERP-Cuba.

Capítulo 1: Fundamentación teórica.

1.1 Introducción

El presente capítulo estará centrado en la realización de un profundo análisis para obtener información actualizada de algunos sistemas generadores de código que han sido implementados para facilitar el trabajo de los desarrolladores y a partir de ahí conocer qué aspectos deben ser tomados en cuenta para que el sistema esté al nivel que se exige hoy en día. Se valorarán las tendencias actuales en el contexto informático partiendo de que continuamente surgen aplicaciones novedosas con mejoras que se van imponiendo en la industria del software. Será explicado el Modelo de desarrollo elaborado por el equipo de arquitectura del proyecto a partir del cual estará orientada la solución. Se realizará un análisis de la Arquitectura definida para el Sistema y se especificarán las tecnologías y herramientas de desarrollo a utilizar durante la implementación para facilitar el cumplimento de los requisitos tanto técnicos como funcionales.

1.2 Conceptos básicos asociados al dominio del problema.

Herramienta Case: 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. (1)

1.3 Sistemas (CASE) o generadores de código vinculados al campo de acción.

A continuación, serán descritas y caracterizadas algunas aplicaciones informáticas que se encargan de la generación de código web, específicamente en el lenguaje php; se valorará en cada caso si es factible su empleo o no en el marco de trabajo del programa ERP-Cuba, y si son viables para dar solución al problema planteado anteriormente.

1.3.1 PhP Maker.

Herramienta que se encarga de generar de forma rápida un conjunto de scripts PHP a partir de una base de datos. Con ella se pueden crear sitios webs completos con varias funcionalidades incluidas para el cliente facilitando el trabajo de este.

Es compatible con varias bases de datos para realizar sus funcionalidades entre las que se encuentran PostgresSQL, MySQL, Microsoft Access, Microsoft SQL Server, y en su última versión Php Maker 8 con Oracle, aunque se puede

trabajar con todas ellas tiene una mayor compatibilidad con MySQL. Los scripts generados son multiplataforma ya que pueden ser ejecutados en diversos sistemas operativos (Windows/Linux).

Los scripts emplean nombres genéricos posibilitando saber a que tabla se refiere y la función que realiza nombre_tabla+acción+php. Se pueden modificar los scripts permitiendo agregar o quitar nuevas funciones según se necesiten. La herramienta permite la instalación de plug-ins para el mejoramiento de sus tareas. Su instalación es bastante simple y consume pocos recursos cuando se ejecuta. (3)

Observaciones:

Es una herramienta que no es multiplataforma y solo se puede trabajar con ella desde el sistema operativo Windows. Se necesita de una base de datos que contenga toda la información para poder generar el código php. Por ser una aplicación Desktop no se puede emplear pues es imposible ajustarla al marco de trabajo del programa ERP-Cuba que está basado en aplicaciones WEB.

1.3.2 Visual Paradingm.

Visual Paradigm es una herramienta CASE desarrollada por Visual Parading International que utiliza UML como lenguaje de modelaje, es considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. El software que modela del UML le ayuda a construir aplicaciones de calidad más rápido, mejor y en más bajo costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación; entre los lenguajes que genera se encuentran Java, C++, CORBA IDL, PHP, XML y Python. (4)

Principales características:

1. Soporta aplicaciones web.
2. Es un producto de calidad.
3. Fácil de instalar y actualizar.
4. Compatibilidad entre ediciones.
5. Varios idiomas.

Modo de generar el código PHP:

Genera una estructura compuesta por cinco archivos .php, un html, un dll y dos carpetas.

Dentro de la carpeta clases se guardan los archivos php de los diagramas de clases, estos contienen las funcionalidades get y set de sus atributos.

Detecta la visibilidad tanto de los atributos como de las funcionalidades y estas tienen siempre un valor de retorno.

Observaciones:

La estructura de carpetas para la generación del código no es la más apropiada para emplear ya que genera barios archivos y carpetas. El código generado está orientado para trabajar con el frameworks de doctrine específicamente, es decir, para gestionar el acceso a los datos de una base de datos, en algunos ficheros emplea varios lenguajes simultáneamente como xml junto al php. Necesita una licencia para su activación de lo contrario sus componentes no funcionan correctamente o no se puede ejecutar el software. Por ser una aplicación Desktop no se puede emplear pues es imposible ajustarla al marco de trabajo del programa ERP-Cuba que está basado en aplicaciones WEB.

1.3.3 BOUML.

Software desarrollado principalmente para modelar diagramas UML y la generación de código en varios lenguajes de programación entre los que se encuentran PHP, Java, C++, IDL.

Diagramas que se pueden generar:

Diagrama de casos de uso

Diagrama de secuencia.

Diagrama de colaboración.

Diagrama de objeto.

Es una herramienta libre y está patentada bajo una licencia GPL (General Public Licence) por lo que se puede modificar y redistribuir. Fue desarrollada empleando tecnología Java, se le pueden agregar herramientas aunque no estén incluidas en la instalación pero deben de estar desarrolladas en Java o

C++, es multiplataforma por lo que se puede emplear en barios Sistemas Operativos. Además es fácil de instalar y la aplicación en sí consume pocos recursos. (6)

Modo de generar el código PHP:

Presenta etiquetas de comentario lo que evitara la sobre-escritura de las implementaciones una vez que se haya generado el código.

Detecta bien la inicialización de los atributos así como la visibilidad de estos

Se crean las carpetas según el paquete de en donde se encuentre. Esto se hará pero antes se debe de configurar manualmente la localización en donde se desea generar

Se pueden generar instancias de clases

Genera las funcionalidades con su visibilidad.

Observaciones:

En cuanto a la generación de código los valores de retorno de las funcionalidades siempre son nulos, sin importar el tipo de dato que estas devuelvan, en los comentarios que genera no cometa el tipo de dato de los atributos, o la visibilidad, a la hora de generar los métodos no se tienen en cuenta los parámetros que estos puedan tener. Es una herramienta desarrollada en java, por lo que es necesario instalar la máquina virtual de este lenguaje para que pueda ejecutarse. Por ser una aplicación Desktop no se puede emplear pues es imposible ajustarla al marco de trabajo del programa ERP-Cuba que está basado en aplicaciones WEB.

1.3.4 ArgoUML.

Software destinado para el análisis y el diseño de software orientados a objetos y para la generación de código en varios lenguajes de programación entre los que se encuentran PHP, Java, C++y C#.

Diagramas que se pueden generar:

Diagrama de clases.

Diagrama de caso de uso.

Diagrama de secuencia.

Diagrama de colaboración.

Diagrama de actividades

Es una herramienta libre y está patentada bajo una licencia BSD (Berkeley Software Distribution) por lo que se puede modificar, redistribuir e incluso se puede emplear para desarrollar software privativos. Fue desarrollada empleando tecnología Java, es multiplataforma por lo que se puede emplear en barios Sistemas Operativos. Además es fácil de instalar. (9)

Modo de generar el código PHP:

Genera comentarios en los que especifica detalles minuciosos entro los que se destacan el autor, la fecha, entre otros.

Los valores de retorno de las funcionalidades son detectados, es decir, que no retorna nulo.

Detecta la visibilidad de los atributos así como su inicialización.

Presenta etiquetas de comentario para evitar sobre-escribir las implementaciones una vez que se haya generado el código.

Reconoce las dependencias de clases, esto implica que si señalamos alguna dependencia entre las clases que se desean generar, en algún momento se generara un objeto del tipo de la clase que se necesita o se hará una instancia de la misma.

Observaciones:

En cuanto a la generación de código a la hora de crear la clase genera código extra; genera un include para cada archivo donde se aloja la clase, lo que provoca que si está relacionada con varias clases se generaría más código innecesario, entre una funcionalidad y otra se genera un excesivo grupo de cometarios que oscurecen el código, al igual que el ArgoUML a la hora de generar los métodos no se tienen en cuenta los parámetros que estos puedan tener. Es una herramienta desarrollada en java, por lo que es necesario instalar la máquina virtual de este lenguaje para que pueda ejecutarse, la aplicación es muy pasada y requiere de mucha memoria para correr, además presenta una interfaz poco amigable al usuario. Por ser una aplicación Desktop no se puede

emplear pues es imposible ajustarla al marco de trabajo del programa ERP-Cuba que está basado en aplicaciones WEB.

1.4 Valoración del estado del arte.

Después de realizar un estudio a los sistemas generadores de código antes mencionados donde se tuvieron en cuenta no sólo los aspectos fundamentales desde el punto de vista del software sino también del producto, se evidencia la no existencia de un software que responda cabalmente a lo que en realidad se necesita para generar el código base de la arquitectura del Cedrux, presentan diferentes inconvenientes ya sea en la generación de los códigos, por las herramientas sobre las que están soportadas o bien porque no son capaces de adaptarse al marco de trabajo definido para el desarrollo del programa ERP-Cuba, sería entonces factible que el proyecto se desarrollara sobre herramientas y tecnologías que no fueran propietarias, siendo una aplicación web y el trabajo con PHP una vía razonable para desarrollar un sistema de este tipo con el fin de posibilitar la integración con el marco de trabajo Saucex.

1.5 Tendencias y tecnologías actuales.

La industria del software ha mostrado ser una de las áreas más dinámicas y con mayor crecimiento en los últimos años. La evolución hacia a un modelo más racional para los usuarios, con menos costes de licencia, donde se intensifique la prestación de servicios, que reduzca el tiempo de desarrollo e incremente la calidad, viene siendo lo más importante a la hora de desarrollar cualquier tipo de aplicación. A continuación se ofrecerá una valoración sobre algunas de las tendencias y tecnologías que marcan un nivel alto en el mundo del software y que bien contribuye a lo dicho anteriormente:

Enfoque Orientado a Objetos (OO):

El enfoque Orientado a Objetos actualmente se encuentra en una etapa de madurez como paradigma del desarrollo de sistemas de información. Ayuda a explotar el poder expresivo de todos los lenguajes de programación basados en objetos y los orientados a objetos, como Java y PHP 5; apoya la reutilización no solo del software, sino de diseños completos; produce sistemas que están construidos en formas intermedias estables y por ello son más resistentes al cambio en especificaciones y tecnología; y brinda un mecanismo para formalizar el modelo de la realidad.

Aplicaciones basadas en entornos web:

Los productos de software cada vez están más enfocados hacia los conceptos del entorno web y buscan una mejor manera de comunicación vía Internet. Al usar aplicaciones web se puede migrar de sistema operativo o cambiar el hardware libremente sin afectar el funcionamiento de las aplicaciones de servidor y no se requieren complicadas combinaciones de hardware/software. Realizar cambios en el Software es sencillo y sin riesgos de incompatibilidades; solo existe una versión en el servidor que implica que no hay que distribuirla entre los demás computadores.

La utilización de esta tecnología unida al uso del patrón Modelo-Vista-Controlador (MVC) y arquitectura en capas conlleva a reducir costos y complicaciones, y proporciona mayor libertad a la hora de realizar cualquier tipo de cambios; siendo así una excelente opción para que las pequeñas empresas automaticen sus procesos sin invertir demasiado en equipo, desarrollo y capacitación.

Arquitectura cliente-servidor:

La arquitectura cliente/servidor no es más que un sistema distribuido entre múltiples procesadores donde hay clientes que solicitan servicios y servidores que los proporcionan. La mayoría de las aplicaciones que se están desarrollando actualmente en la industria de software utilizan este tipo de arquitectura debido a las funcionalidades que brindan, entre ellas:

- Permite integrar y compartir información entre sistemas diferentes sin necesidad de que todos tengan que utilizar el mismo sistema operativo.
- Posibilita el mantenimiento y el desarrollo rápido de aplicaciones. (12)

La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.

Utilización de Software Libre:

La utilización de tecnologías estándares y herramientas libres posibilitan una mayor independencia de plataformas y ofrecen la posibilidad de mantenimiento futuro de las aplicaciones. Los programas libres existentes (su modificación, integración o corrección), también posibilitan obtener productos de excelente calidad, en tiempos relativamente bajos y, por consiguiente, con menores costos. Y con el uso de licencias libres, el producto terminado debe ser entregado al

cliente con toda la documentación y el código fuente, sin imponer ninguna traba a la futura extensión del mismo, asegurando un trato justo, claro y transparente.

1.6 Modelo de desarrollo

El modelo de desarrollo que se empleará para la confección de la herramienta es el propuesto y usado por el programa ERP-Cuba.

Centrado en la arquitectura:

La arquitectura determina la línea base y los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades en la producción y resuelve las necesidades tecnológicas y de soporte para el desarrollo.

Orientado a componentes:

Las iteraciones son orientadas según la significación arquitectónica de los componentes, los mismos son abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan, el componente es la unidad de medición y ordenamiento de las iteraciones.

Iterativo e incremental:

Las iteraciones son planificadas y coordinadas con el equipo de arquitectura, los clientes y la alta gerencia. Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo de esta manera la evolución incremental del producto.

Ágil y adaptable al cambio:

El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados.

1.7 Arquitectura

Uno de los elementos clave en todo proceso de desarrollo de software es el diseño de la Arquitectura. Básicamente sobre ella se sustentan todas las representaciones de la estructura general de la aplicación a desarrollar. En la medida que sea concebida la arquitectura basada en los principios de cohesión,

utilidad y flexibilidad de los componentes se obtendrá un mejor acabado del producto.

En otras palabras la arquitectura de una aplicación es la vista conceptual de la estructura de esta y donde se establecen los fundamentos básicos para que analistas, diseñadores y programadores trabajen en una línea común y así alcanzar los objetivos del sistema de acuerdo con las necesidades del cliente. La correcta definición de los estilos arquitectónicos a utilizar, patrones y mecanismos de diseño es la esencia de lo dicho anteriormente.

1.7.1 Características de la Arquitectura Base

Para el desarrollo del Sistema Cedrux se decidió adoptar la propuesta de Arquitectura Base definida por la línea de Arquitectura del programa ERP-Cuba conformada por las diferentes vistas y estilos arquitectónicos que serán especificados a continuación:

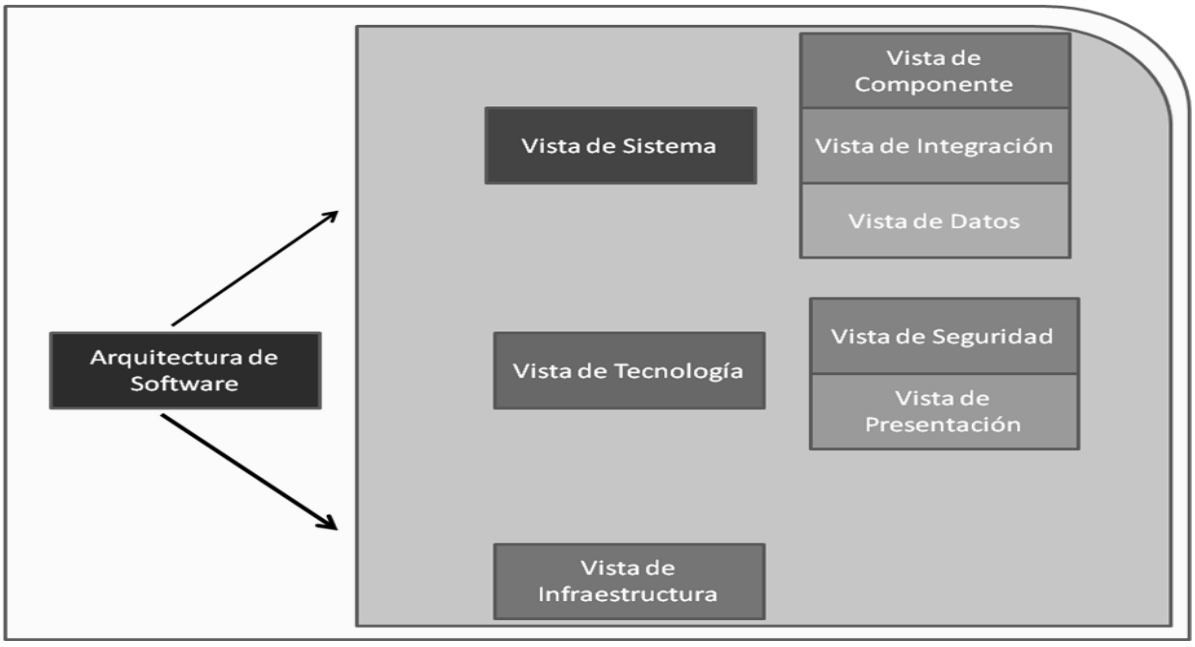


Figura 1 Vistas de la Arquitectura. (14)

Vista de Sistema:

Propone las partes del software: componentes, conectores, las restricciones y las configuraciones de estas partes, se subdivide en tres vistas fundamentales:

- Vista de Componentes: Encargada de las definiciones de los tipos de componentes posibles a definir en el proyecto, de la especificación de sus características, así como de la composición estructural interna de cada uno de estos componentes.

- Vista de Integración: Encargada de los procesos de integración interna y externa, establece las definiciones, estándares, protocolos de comunicación y reglas de intercambio de información.

- Vista de Datos: Encargada de todas las definiciones a nivel de datos, de la integración de los distintos modelos, de los patrones, estándares y definiciones a este nivel.

Vista Tecnológica:

Es la base del software, propicia los elementos necesarios para crear el producto, esta a su vez se subdivide en dos vistas:

- Vista de seguridad: Chequea e implementa todos los aspectos relacionados con el acceso a la aplicación, la modificación, lectura o eliminación de la información, etc.

- Vista de presentación: Encargada de cómo luce el software, cuáles son los colores que lleva la aplicación, cómo son los botones, los vínculos y todos los elementos significativos desde el punto de vista de la presentación.

Vista de Infraestructura:

Es la encargada de determinar la plataforma tecnológica a utilizar en la elaboración del producto, la definición y disponibilidad de los distintos servicios telemáticos necesarios en la confección del mismo, así como del diseño de los distintos escenarios de despliegue posibles. (14)

Particularidades de la arquitectura:

- Arquitectura Basada en Componentes:

Uno de los enfoques en los que actualmente se trabaja es la arquitectura basada en componentes que tiene como objetivo hacer un uso correcto de software reutilizable, para la construcción de aplicaciones mediante el ensamblaje de partes ya existentes.

“Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y de requisitos, que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio” (15)

El equipo de arquitectura del programa ERP-Cuba dentro de la vista del sistema antes presentada incluye la vista de componente como una de sus

subdivisiones; en ese caso propone que todas las funcionalidades levantadas y modeladas en las fases de negocio y requerimientos deben ser expresadas o contenidas en al menos un componente y que las distintas interacciones entre ellos originen funcionalmente la existencia de subsistemas en consecuencia a las dependencias definidas.

- Patrón Modelo-Vista-Controlador (MVC):

El patrón arquitectónico MVC es utilizado para el desarrollo de aplicaciones Web con el fin de separar en tres componentes distintos la interfaz de usuario, la lógica de negocio y los datos persistentes, potenciando la flexibilidad y la adaptabilidad a futuros cambios. Por su parte:

El Modelo: Es la representación de la información que maneja la aplicación. Son los datos puros que puestos en un contexto del sistema son mostrados al usuario por medio del Controlador, proveen de información al usuario o a la aplicación misma.

La Vista: Constituye la representación del modelo en forma gráfica, disponible para la interacción con el usuario. En una aplicación web la "Vista " es la página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.

El Controlador: Se encarga de responder a las solicitudes del usuario desde la Interfaz, manejando los diferentes eventos a través de las funcionalidades necesarias y la información perteneciente al Modelo. (16)

Para el desarrollo del Sistema Integral de Gestión de Entidades Cedrux se decidió trabajar con este patrón, evidenciándose en los framework definidos para cada una de las capas como parte del MVC de cada componente dentro de la aplicación, es decir: para la Vista: Extjs-Framework, el cual es muy utilizado en el desarrollo de aplicaciones Web con tecnología AJAX, para el Controlador: Zend-Framework quien emplea específicamente el estilo Modelo- Vista-Controlador como base de su funcionamiento y para agilizar el acceso a datos en el Modelo se utilizó Doctrine, un potente y completo sistema ORM (Mapeo Objeto Relacional).

De forma general, según lo descrito con anterioridad, en la arquitectura del Programa ERP-Cuba los estilos arquitectónicos que se emplean no se pueden ver de forma independiente sino como un estilo híbrido que comprende numerosas ventajas:

- Desarrollos paralelos: en cada capa.
- Aplicaciones más robustas debido al encapsulamiento.
- Mantenimiento y soporte más sencillo: es más fácil cambiar un componente que modificar íntegramente una aplicación.
- Mayor flexibilidad: se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad.
- Alta escalabilidad: La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. (17)

1.8 Patrones

En términos generales, un patrón es un cúmulo de información que proporciona respuesta a un conjunto de problemas similares en un contexto dado. Los patrones hacen la producción de software más resistente al cambio, establecen parejas problema-solución, ayudan a especificar interfaces, facilitan la reutilización del código y permiten una fácil comprensión debido a la documentación estándar que presentan.

Patrones de diseño:

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Patrones **GRASP**: Patrones de Software para la Asignación General de Responsabilidad. En este caso las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento:

Conocer: Conocer los datos privados encapsulados, conocer los objetos relacionados, conocer las cosas que puede derivar o calcular.

Hacer: Hacer algo él mismo, como crear un objeto o hacer un cálculo, iniciar una acción en otros objetos, controlar y coordinar actividades en otros objetos.

GRASP destaca 5 patrones principales: Experto, Creador, Alta cohesión, Bajo acoplamiento, Controlador.

Experto: Este patrón es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria

para ejecutar la tarea. Refuerza el encapsulamiento y esto redunda en bajo acoplamiento.

Creador: El patrón creador ayuda a identificar quién debe ser el responsable de la instanciación o creación de nuevas clases u objetos. La clase podrá crear la nueva instancia si y sólo si tiene en cuenta al menos uno de los siguientes criterios:

- Tiene la información necesaria.
- Usa directamente las instancias creadas del objeto.
- Almacena o maneja varias instancias de la clase.
- Contiene o agrega la clase.

La visibilidad entre la clase creada y la clase creadora es una de las facilidades que se deriva del uso de patrón y además conduce a un bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización así como mayor claridad.

Alta cohesión: Indica que la información que almacena una clase debe ser coherente, de manera que todos sus métodos tengan un comportamiento bien definido.

- Las Clases se pueden reutilizar con mayor facilidad y flexibilidad.
- Una Clase con baja cohesión hace muchas cosas no relacionadas.
- Una Clase con alta cohesión hace lo que uno podría esperar que hiciera.

Si el sistema fallara por alguna razón es mucho más fácil encontrar responsabilidades si las Clases del sistema son cohesivas.

Bajo acoplamiento: Patrón evaluativo que asigna responsabilidades de modo que se mantenga un engranaje pobre entre las clases y objetos, reduce el impacto de los cambios y aumenta la reutilización.

Controlador: Asignar la responsabilidad a una clase de manejar mensajes correspondientes a eventos en un sistema. Encargada de recibir los datos del usuario y enviarlos a las distintas clases según el método llamado a modo de intermediario entre una determinada interfaz y el algoritmo que la implementa. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación para aumentar la reutilización de código y a la vez tener un mayor control. (18)

Patrones **GOF**:

Cadena de Responsabilidad: La cadena de responsabilidad se encarga de evitar el acoplamiento del remitente de una petición a su receptor, dando a más de un objeto la posibilidad de manejar la petición. (19)

1.9 Lenguajes de modelado y desarrollo

1.9.1 Lenguaje UML

UML

UML (Unifed Modeling Languaje) es el lenguaje para modelación unificado para la especificación, visualización, construcción y documentación de los artefactos de un sistema de software. Prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. (20)

UML también intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos se crea una documentación que cualquier desarrollador con conocimientos de UML será capaz de entender. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otro rama. (20)

1.9.2 Lenguajes de desarrollo

El término lenguaje de programación está dado por un lenguaje que puede ser utilizado para controlar el comportamiento de una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen la estructura, el significado de sus elementos y expresiones. Un lenguaje de programación permite a uno o más programadores especificar de manera precisa sobre qué datos una computadora debe operar, cómo deben ser estos almacenados, transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias.

1.9.2.1 Lenguajes del lado del servidor

Los lenguajes de lado servidor son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un

formato comprensible para él. Estos se ejecutan en el servidor web, justo antes de que se envíe una página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. (21)

PHP

PHP, acrónimo de Hipertext Preprocesor, es un lenguaje de programación interpretado de alto nivel embebido en páginas HTML. Es gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Permite rápidamente a los desarrolladores la generación dinámica de páginas. Es uno de los lenguajes de programación más populares, debido a la gran fluidez y rapidez de sus scripts; siendo realmente fácil de utilizar por ventajas como su gratuidad y seguridad.

Producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se reparen rápidamente. Una de sus características más potentes es su compatibilidad con las bases de datos más comunes, como MySQL, PostgreSQL, mSQL, Oracle, Informix, ODBC, entre otras. También ofrece la integración con varias bibliotecas externas.

PHP es la opción natural para los programadores en máquinas con Linux que ejecutan servidores web con Apache, pero funciona igualmente bien en cualquier otra plataforma de UNIX o de Windows, con el software de Netscape o del web server de Microsoft. PHP también utiliza las sesiones de HTTP, conectividad de Java, expresiones regulares, LDAP, SNMP, IMAP, protocolos de COM (bajo Windows). (22)

1.9.2.2 Lenguajes del lado del cliente

Los lenguajes de lado cliente son aquellos que pueden ser directamente interpretados por el navegador y no necesitan un pre tratamiento, ya que el navegador es quien soporta la carga de procesamiento.

JavaScript

JavaScript es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web. Gran parte de su programación está centrada

en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros.

Permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc. Además pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente.

Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado, es soportado por Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros. (35)(36)

HTML

Lenguaje de Marcas de Hipertexto (HTML, siglas de HyperText Markup Language) es un lenguaje que permite describir hipertextos, textos presentados de forma estructurada y agradable, con enlaces que conducen a otros documentos o fuentes de información relacionadas, y con inserciones multimedia como gráficos y sonido.

Una de sus características esenciales es la universalidad, prácticamente cualquier ordenador, independientemente del sistema operativo, puede leer o interpretar una página web. Se ha convertido en uno de los formatos más populares y predominantes en la construcción de páginas web. (33)

XML

También conocido como un lenguaje universal de marcado para documentos estructurados y datos en la Web, XML constituye un grupo de reglas y convenciones sintácticas que ofrece un formato para la descripción de datos estructurados y que se pueden utilizar para construir grupos de elementos de marcación propios.

XML se desarrolló para proporcionar una flexibilidad y consistencia que no se podían alcanzar con HTML; no solo es un lenguaje de marcado, sino también un metalenguaje cuya particularidad más importante es que no posee etiquetas prefijadas con anterioridad, permitiendo describir otros lenguajes de marcado y definir lenguajes de presentación propios en dependencia del contenido del documento.

La meta fundamental del XML es hacer la cooperación y la interoperabilidad más fáciles entre módulos que pertenecen a diferentes aplicaciones, e incluso a diferentes organizaciones. Permite la definición, transmisión, validación e interpretación de datos; garantizando que los datos estructurados sean uniformes e independientes de aplicaciones o fabricantes, y el intercambio de cualquier tipo de información, sin que ocasione problemas de tipo "contenido" o de tipo "presentación". (34)

1.10 Frameworks.

Un framework, en el desarrollo de software, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

1.10.1 Ext.

ExtJs es una librería Javascript que permite construir aplicaciones complejas en Internet. Incluye componentes UI del alto performance y personalizables, modelo de componentes extensibles, un API fácil de usar y licencias Open source y comerciales.

Su sistema de licenciamiento no contempla la licencia LGPL, o se tiene código 100% GPL o se debe pagar por su licencia de desarrollo.

Es soportado por varios navegadores web como Internet Explorer, FireFox, Safari y Opera. Una de las grandes ventajas de utilizar ExtJs es que nos permite crear aplicaciones complejas. (23)

1.10.2 Zend

Es un framework de alta calidad y de código abierto para el desarrollo de aplicaciones y servicios web con PHP.

Zend Framework brinda facilidades de uso y poderosas funcionalidades. Proporciona soluciones para construir modernos, robustos y seguros sitios web, está diseñado para php5 y posee buenas capacidades de ampliación. Presenta entre otras, las siguientes características:

1. Proporciona un sistema de caché de forma que se puedan almacenar diferentes datos.

2. Proporcionan los componentes que forma la infraestructura del patrón MVC.

3. Proporciona una capa de acceso a base de datos, construida sobre PDO[1] pero ampliándola con diferentes características.

4. Proporciona mecanismos de filtrado y validación de entradas de datos.

5. Permite convertir estructuras de datos PHP a JSON[2] y viceversa, para su utilización en aplicaciones AJAX (especificado en el epígrafe 1.10.1).

6. Proporciona capacidades de búsqueda sobre documentos y contenidos. (26)

Se estará haciendo uso de Zend Framework en su versión 1.9.7.

1.10.3 Doctrine

Doctrine es un potente y completo sistema ORM (object relational mapper) para el desarrollo de aplicaciones PHP 5.2 o superior que utilicen bases de datos. Implementa el patrón ORM para desarrollar el dominio de una aplicación y cuenta con una capa de abstracción para el acceso a bases de datos y un lenguaje de consulta propio que abstrae del gestor que se está utilizando Entre sus funcionalidades permite exportar una base de datos existente a sus clases correspondientes y también convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientado a objeto. Esto les proporciona una alternativa poderosa a diseñadores de SQL manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. (27)

1.11 Tecnologías y herramientas de desarrollo

Para la realización de esta herramienta es necesaria la existencia de un modelo estandarizado de las tecnologías y herramientas a utilizar conjuntamente con sus versiones para la implementación de las capas de presentación, negocio y acceso a datos. De acuerdo con lo planteado anteriormente se determinó emplear:

1.11.1 Tecnología Ajax.

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los

[1] PDO: (en inglés: PHP Data Objects) es una interface de acceso a datos que permite la conexión a diferentes bases de datos utilizando tecnología orientada a objetos. (24)

[2] JSON: es un formato ligero para el intercambio de datos. (25)

usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

Permite realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

AJAX es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting lenguaje) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. (28)

1.11.2 Herramienta CASE.

Visual Paradigm:

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Entre sus principales características podemos encontrar:

1. Soporta aplicaciones Web.
2. Es un producto de calidad.
3. Fácil de instalar y actualizar.
4. Compatibilidad entre ediciones. (6)

Se estará haciendo uso de Visual Paradigm en su versión 6.4.

1.11.3 Herramienta de desarrollo colaborativo

Control de versiones

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones

realizadas. Su principal objetivo es permitir editar de forma colaborativa y compartir información. Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión.

Subversion.TortoiseSVN 1.4.5

Subversion es un software de sistema de control de versiones, software libre, conocido también como SVN. Es un sistema centralizado para compartir información que permite realizar modificaciones atómicas y gestionar archivos y directorios, así como sus cambios a través del tiempo; lo que facilita las tareas administrativas. Su capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración.

TortoiseSVN es el cliente gratuito para el sistema de control de versiones Subversion, con código abierto y software libre bajo la licencia GNU GPL. Está disponible en 28 idiomas diferentes y puede ser usado sin un entorno de desarrollo. (29)

1.11.4 Entorno integrado de desarrollo

Un entorno integrado de desarrollo o IDE (en inglés: Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación que permite de forma cómoda y ágil editar, compilar, ejecutar y depurar programas.

- **NetBeans IDE :**

NetBeans IDE es un entorno integrado de desarrollo (IDE), modular, de base estándar, escrito en el lenguaje de programación Java. Se puede instalar en Windows, MacOS, Linux y Solaris. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general para compilar cualquier tipo de aplicación. Con este IDE se pueden crear aplicaciones que sean soportadas por la plataforma Java como JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby on Rails, Groovy and Grails y C/C++. (38)

Este proyecto cuenta con el apoyo de una gran comunidad de desarrolladores que brindan accesorios (plug-ins) que aumentan la cantidad de las funcionalidades y opciones de la herramienta.

1.11.5 Servidor Apache.

Un servidor de aplicaciones es un software que ayuda al servidor Web a procesar las páginas que contienen scripts o etiquetas del lado del servidor.

Apache 2.0

El Servidor Apache HTTP es un servidor Web de tecnología Open Source sólido, y el más usado por los servidores en todo Internet. Apache es el servidor web hecho por excelencia, su robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Apache es una tecnología gratuita de código fuente abierto, su licencia es una descendiente de la licencias BSD, no es GPL, permitiendo hacer modificaciones en su código fuente.

Es un servidor que corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal. Tiene una alta configurabilidad en la creación y gestión de logs y permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.

Apache es un servidor altamente configurable de diseño modular que trabaja con gran cantidad de lenguajes de script como Perl, PHP y otros, teniendo todo el soporte que se necesita para tener páginas dinámicas. (30)

1.11.6 Sistema gestor de base de datos.

Se denomina Sistema Gestor de Base de Datos (siglas: SGBD) al conjunto de programas que permiten definir, construir y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad.

PostgreSQL:

PostgreSQL es un servidor de base de datos relacional[3], libre. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y joins de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta de software libre PostgreSQL tiene entre otras ventajas las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y algo muy importante es que es multiplataforma. Fue diseñado para ambientes de alto volumen. Escala muy bien al aumentar el número de CPUs y la cantidad de

[3] Base de datos relacional: Modelo utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente basados en el empleo de relaciones entre las tablas. (31)

RAM. Soporta transacciones y desde la versión 7.0, claves ajenas con comprobaciones de integridad referencial. Tiene mejor soporte para vistas procedimientos almacenados en el servidor, transacciones, almacenamiento de objetos de gran tamaño y además tiene ciertas características orientadas a objetos.

Se estará haciendo uso de PostgreSQL en su versión 8.3 ó superior e inferior a 8.4

1.11.7 Navegador

El navegador es una especie de aplicación capaz de interpretar las órdenes recibidas en forma de código HTML fundamentalmente y convertirlas en las páginas que son el resultado de dicha orden. Permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores web.

Mozilla Firefox 3.0

Mozilla Firefox es un navegador web libre desarrollado por la Corporación Mozilla. Considerado uno de los navegadores más usado en la actualidad, es multiplataforma y disponible en varias versiones de Microsoft Windows, Mac OS X y GNU/Linux. Su código fuente es software libre, publicado bajo una triple licencia GPL/LGPL/MPL. Mozilla Firefox es compatible con varios estándares web, incluyendo HTML, XML, XHTML, CSS, JavaScript, DOM e imágenes. (32)

1.12 Propuesta de solución.

Se propone implementar una herramienta mediante la cual se modelen los componentes del sistema y las conexiones entre estos, expresados en interfaces de servicios con sus respectivos métodos que serían los puertos de entrada y salida de cada una de dichas interfaces. Hasta el momento se tiene definido un único protocolo de comunicación: IOC, que ha sido implementado en el propio proyecto para la integración de los componentes.

1.13 Conclusiones del capítulo.

Este capítulo fue esencial para valorar el estado de la generación de código web, específicamente en el lenguaje PHP a partir de un análisis a herramientas CASE vinculadas a la generación de código en este lenguaje, evidenciando la no existencia de un sistema informático capaz de cumplir con los requisitos establecidos en el marco de trabajo del proyecto.

Este capítulo fue básico para la definición de un conjunto de conceptos fundamentales asociados al dominio del problema. Se explicó el modelo de desarrollo a utilizar. Se analizaron las tendencias y tecnologías actuales para el desarrollo de aplicaciones informáticas y por último se realizó una presentación de las tecnologías y herramientas conjuntamente con sus versiones, propuestas por la dirección del proyecto para el desarrollo de la solución.

Capítulo 2: Diseño e implementación.

2.1 Introducción

El presente capítulo tiene como objetivo describir la solución informática de La herramienta para le generación de código base de la arquitectura del Cedrux. Obtener el diseño de la solución arquitectónica a partir de procesos identificados por los analistas, patrones de diseño y definiciones arquitectónicas establecidas en la línea base de la arquitectura del proyecto. Se definen las funcionalidades necesarias; además se estudian los estándares de codificación y se realiza la implementación y descripción de algoritmos no triviales.

2.2 Diseño de la solución

El diseño es el primer paso en la fase de desarrollo de cualquier producto o sistema de ingeniería. Su objetivo es producir un modelo o representación de una entidad que se va a construir posteriormente. (Pressman, 2005)

2.2.1 Descripción de procesos

A continuación se describen algunos procesos fundamentales para el correcto funcionamiento de La herramienta para la generación del código base de la arquitectura del Cedrux.

En la Figura 1 se representa el flujo que se debe seguir para adicionar correctamente un nuevo proyecto. (Ver Tabla 1)

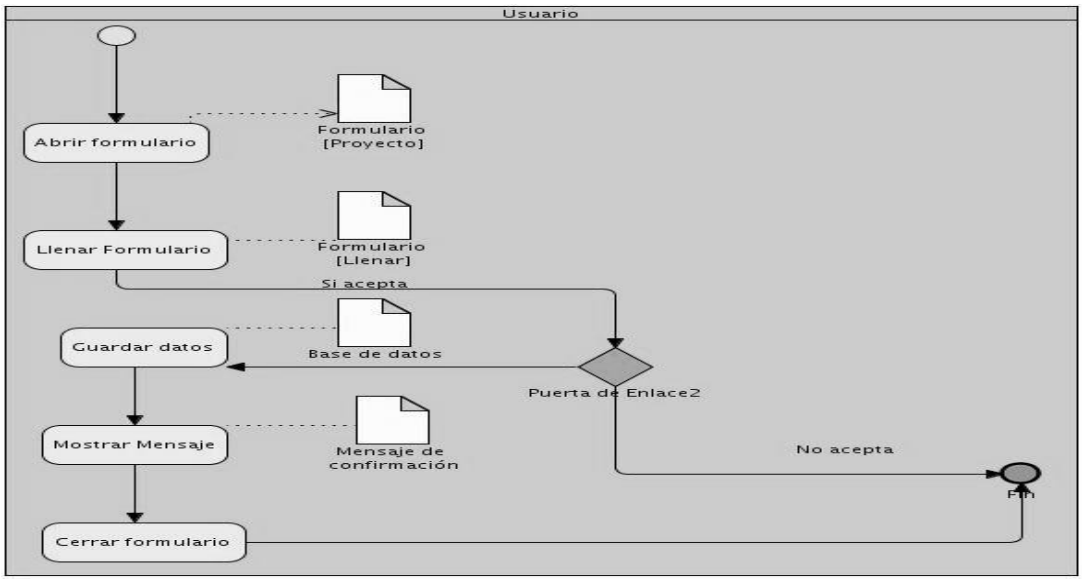


Figura 1: Diagrama del proceso Adicionar proyecto.

En la Figura 2 se representa el flujo que se debe seguir para adicionar un nuevo componente al sistema de forma correcta. (Ver Tabla 2)

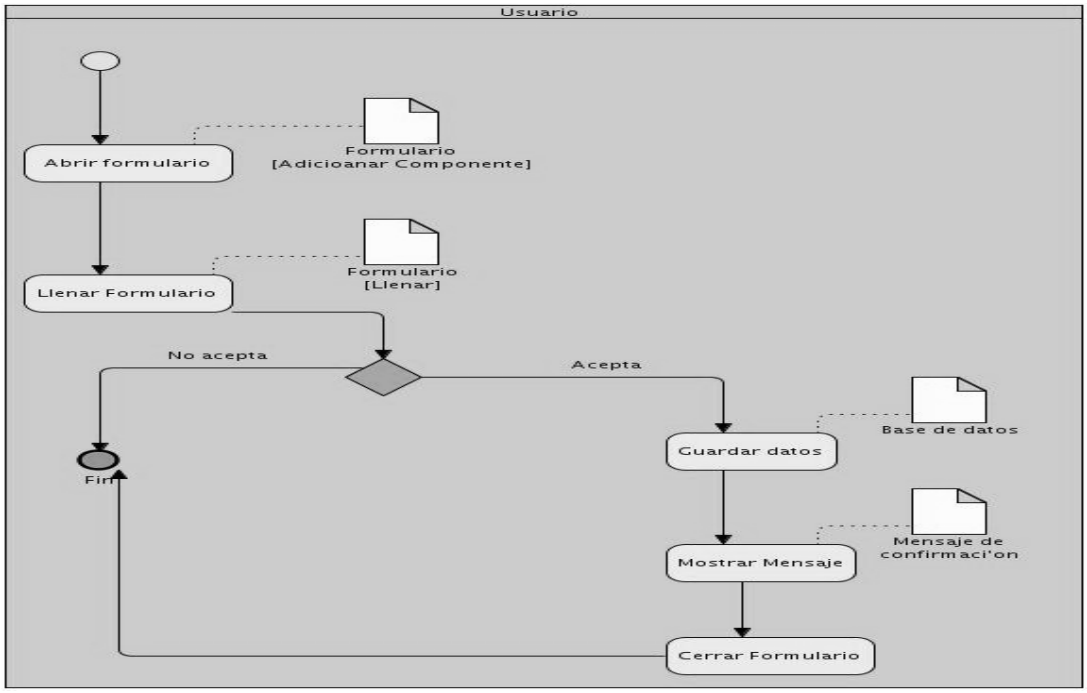


Figura 2: Diagrama del proceso Adicionar componente.

En la Figura 3 se representa el flujo que se debe seguir para modificar un componente ya existente en el sistema de manera correcta. (Ver Tabla 3)

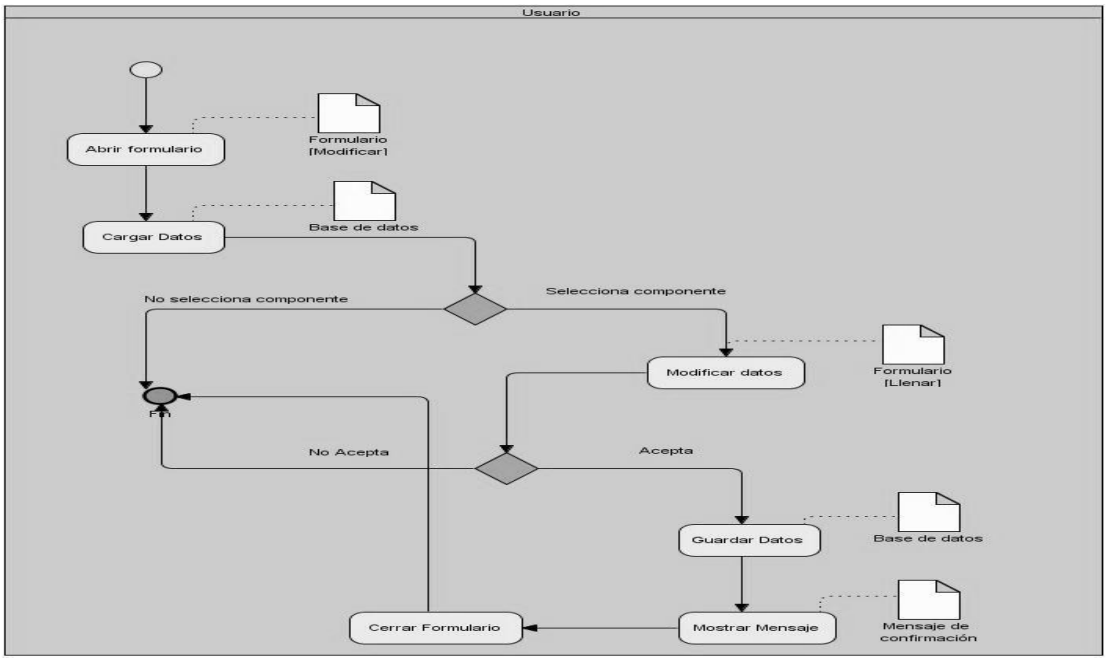


Figura 3: Diagrama del proceso Modificar componente.

En la Figura 2 se representa el flujo que se debe seguir para eliminar un componente ya existente en el sistema de forma correcta. (Ver Tabla 4)

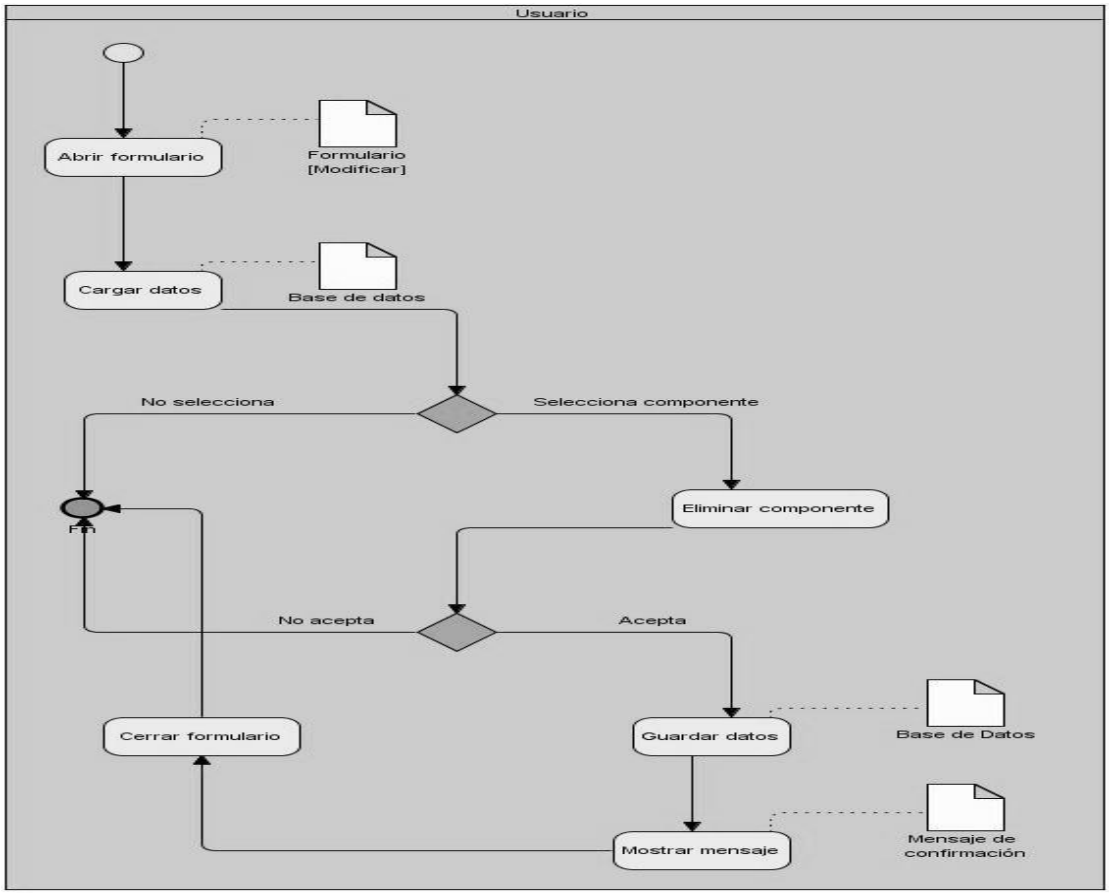


Figura 4: Diagrama del proceso Eliminar componente.

2.2.2 Especificación de procesos

Tabla 1: Proceso Adicionar proyecto.

Objetivo	Realizar un nuevo proyecto para modelar y generar componentes
Evento(s) que lo genera(n)	
Pre condiciones	N/A
Marco legal	N/A.
Clientes internos	N/A
Clientes externos	N/A.
Entradas	Nombre del proyecto
Flujo de eventos	
Flujo básico	

1.	El usuario abre el formulario para agregar un nuevo proyecto
2.	Se llenan los datos correspondientes al nuevo proyecto.
3.	Se guardan los datos introducidos por el usuario
4.	Se muestra el mensaje de confirmación.
5.	Se cierra el formulario
6.	Concluye el proceso.
Pos-condiciones	
1. Se ha creado un nuevo proyecto.	
Salidas : N/A.	
Flujos paralelos	
1. N/A.	
Pos-condiciones	
1. N/A	
Salidas	
1. N/A	
Flujos alternos	
Flujo alternativo: Se cancela la acción de adicionar proyecto	
1. Concluye el proceso	
Pos-condiciones	
1. N/A.	
Salidas	
1. N/A.	
Asuntos pendientes: N/A	

Tabla 2: Proceso Adicionar componente.

Objetivo	Crear un nuevo componente para el proyecto
Evento(s) que lo genera(n)	
Pre condiciones	Se ha creado un proyecto
Marco legal	N/A.
Clientes internos	N/A.
Clientes externos	N/A.
Entradas	Nombre del componente

	Proyecto al que pertenece
Flujo de eventos	
Flujo básico	
1. Se abre el formulario para agregar un nuevo componente	
2. Se llenan los datos correspondientes al nuevo componente.	
3. Se guardan los datos introducidos por el usuario	
4. Se muestra el mensaje de confirmación.	
5. Se cierra el formulario	
6. Concluye el proceso.	
Pos-condiciones	
2. Se ha creado un nuevo proyecto.	
Salidas : N/A.	
Flujos paralelos	
2. N/A.	
Pos-condiciones	
2. N/A	
Salidas	
2. N/A	
Flujos alternos	
Flujo alternativo: Se cancela la acción de adicionar proyecto	
1. Concluye el proceso	
Pos-condiciones	
2. N/A.	
Salidas	
2. N/A.	
Asuntos pendientes: N/A	

Tabla 3: Proceso Modificar componente

Objetivo	Modificar un componente seleccionado
Evento(s) que lo genera(n)	
Pre condiciones	Se ha creado el componente
Marco legal	N/A.

Clientes internos	N/A.
Clientes externos	N/A.
Entradas	Nombre del componente Proyecto al que pertenece
Flujo de eventos	
Flujo básico	
1. Se abre el formulario para modificar un componente	
2. Se cargan los datos de los componentes	
3. Se cambien los datos correspondientes al componente seleccionado.	
4. Se guardan los datos introducidos por el usuario	
5. Se muestra el mensaje de confirmación.	
6. Se cierra el formulario	
7. Concluye el proceso.	
Pos-condiciones	
3. Se ha creado un nuevo proyecto.	
Salidas : N/A.	
Flujos paralelos	
3. N/A.	
Pos-condiciones	
3. N/A	
Salidas	
3. N/A	
Flujos alternos	
Flujo alternativo: No se selecciona componente para modificar	
1. Concluye el proceso	
Flujo alternativo: Se cancela la acción de modificar el componente	
1. Concluye el proceso.	
Pos-condiciones	
3. N/A.	
Salidas	
3. N/A.	
Asuntos pendientes: N/A	

Tabla 4: Proceso Eliminar componente.

Objetivo	Eliminar un componente seleccionado
Evento(s) que lo genera(n)	
Pre condiciones	Se ha creado el componente
Marco legal	N/A.
Clientes internos	N/A.
Clientes externos	N/A.
Entradas	N/A
Flujo de eventos	
Flujo básico	
1. Se abre el formulario para modificar un componente	
2. Se cargan los datos de los componentes	
3. Se elimina el componente seleccionado.	
4. Se guardan los datos introducidos por el usuario	
5. Se muestra el mensaje de confirmación.	
6. Se cierra el formulario	
7. Concluye el proceso.	
Pos-condiciones	
4. Se ha creado un nuevo proyecto.	
Salidas : N/A.	
Flujos paralelos	
4. N/A.	
Pos-condiciones	
4. N/A	
Salidas	
4. N/A	
Flujos alternos	
Flujo alternativo: No se selecciona componente para eliminar	
1. Concluye el proceso	
Flujo alternativo: Se cancela la acción de eliminar el componente	
2. Concluye el proceso.	
Pos-condiciones	

4. N/A.
Salidas
8. N/A.
Asuntos pendientes: N/A

2.2.3 Principales funcionalidades

Los principales procesos identificados y aprobados se agruparon en un conjunto de funcionalidades con una finalidad u objetivo específico:

1- Gestionar componentes
   Adicionar componente

   Modificar componente

   Eliminar componente

2- Gestionar interfaces
   Adicionar interfaces

   Modificar interfaces

   Eliminar interfaces

3- Gestionar puertos
   Adicionar puertos

   Modificar puertos

   Eliminar puertos

4- Crear proyecto
   Modelar proyecto

   Establecer conexión

   Generar proyecto

5- Guardar proyecto

2.3 Patrones de diseño.

Experto: se puso en práctica en todos los componentes, con el uso de clases que poseen responsabilidades específicas a cumplir de acuerdo con la información que manejan, los componentes cuentan con clases controladoras, modelos y

entidades que poseen funciones concretas de acuerdo con la información que gestionan. Además, se modeló una clase entidad por cada tabla de la base de datos, posibilitando el trabajo específico y directo con el experto en la información.

Creador: Es útil contar con el principio general para la asignación de responsabilidades de creación porque permite que el diseño pueda soportar bajo acoplamiento, mayor claridad, encapsulación y reutilización. En la herramienta se evidencia este patrón en las clases controladoras, pues son responsables de crear el objeto de las modelos, y estas a su vez de las entidades.

Controlador: Se utilizan cuando la aplicación es muy extensa, de esta forma, en el módulo en vez de tener un solo controlador y saturarlo, se tienen clases Controllers, que son controladores más pequeños especializados en las funcionalidades de cada componente.

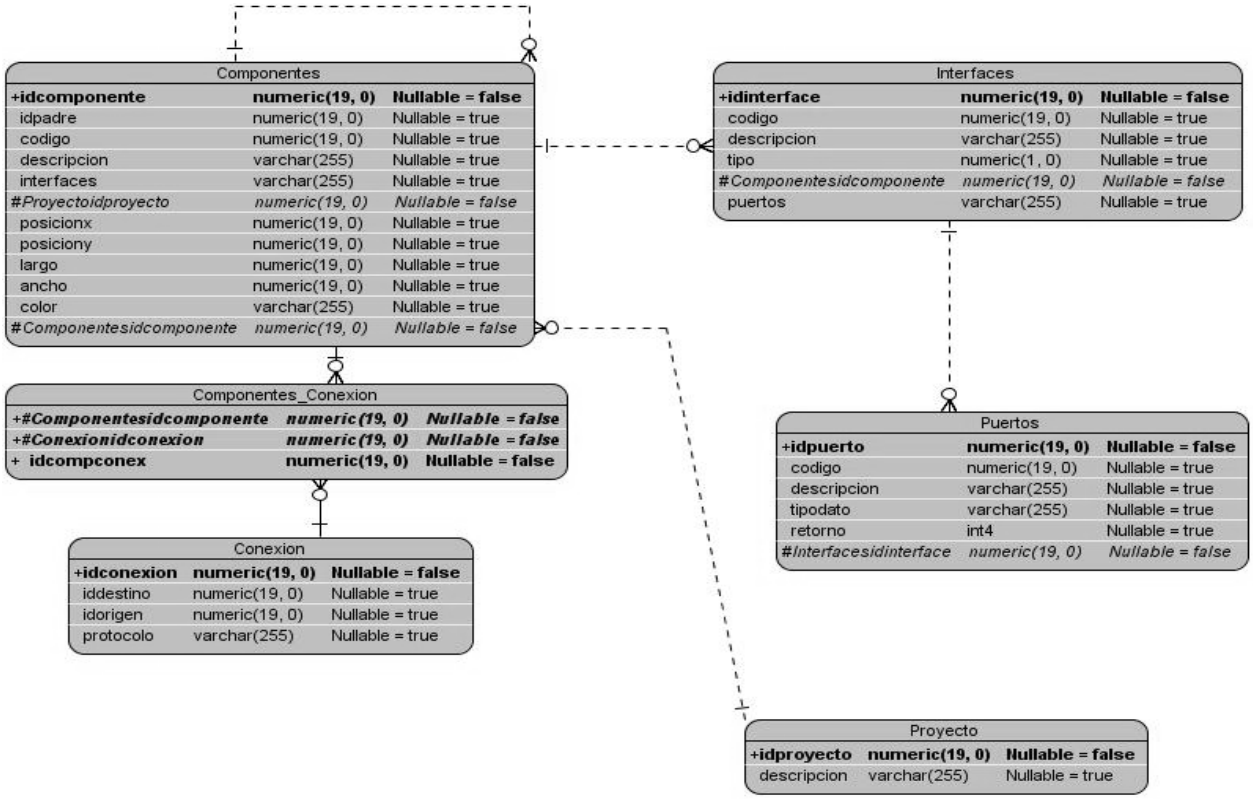
Bajo acoplamiento: La herramienta fue diseñada bajo este principio, porque solo establece las relaciones necesarias para su funcionamiento. La base de datos fue definida de modo que entre las tablas existiera dependencia mínima, haciéndolas más independientes y reutilizables para reducir el impacto de los cambios y acrecentar la oportunidad de una mayor productividad.

Alta cohesión: En la herramienta existe afinidad entre cada clase y los métodos que implementan, estas poseen responsabilidades vinculadas acordes a la información que controlan; y colaboran con otros objetos para compartir el esfuerzo si la tarea es grande, facilitando su mantenimiento y reutilización.

Cadena de responsabilidad: Se manifiesta en el tratamiento que se le dan a las excepciones en cada nivel para mostrarle al usuario la mínima información del posible error ocurrido en el sistema.

2.3.1 Modelo Entidad – Relación

En el proceso de diseño de un sistema de software, la actividad de diseñar una base de datos es un proceso importante e independiente que modela la información persistente del negocio. El modelo de datos propuesto en la solución de La Herramienta para la generación del código base de la arquitectura de Cedrux cuenta con un total de (6) tablas. Para su construcción se tuvo en cuenta la reducción a la mínima expresión de los campos nulos y la persistencia de campos resúmenes para agilizar recuperaciones frecuentes de algunos datos que son complejos de calcular.



2.3.2 Diseño de clases de la herramienta

Los diagramas de clases del diseño describen gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación que contienen:

- clases, asociaciones y atributos
- interfaces, con sus operaciones y constantes
- métodos
- información sobre los tipos de los atributos
- navegabilidad

- dependencias

  Los diagramas de clases del diseño expresan la definición de clases como componentes del software

Diagrama de clases del diseño

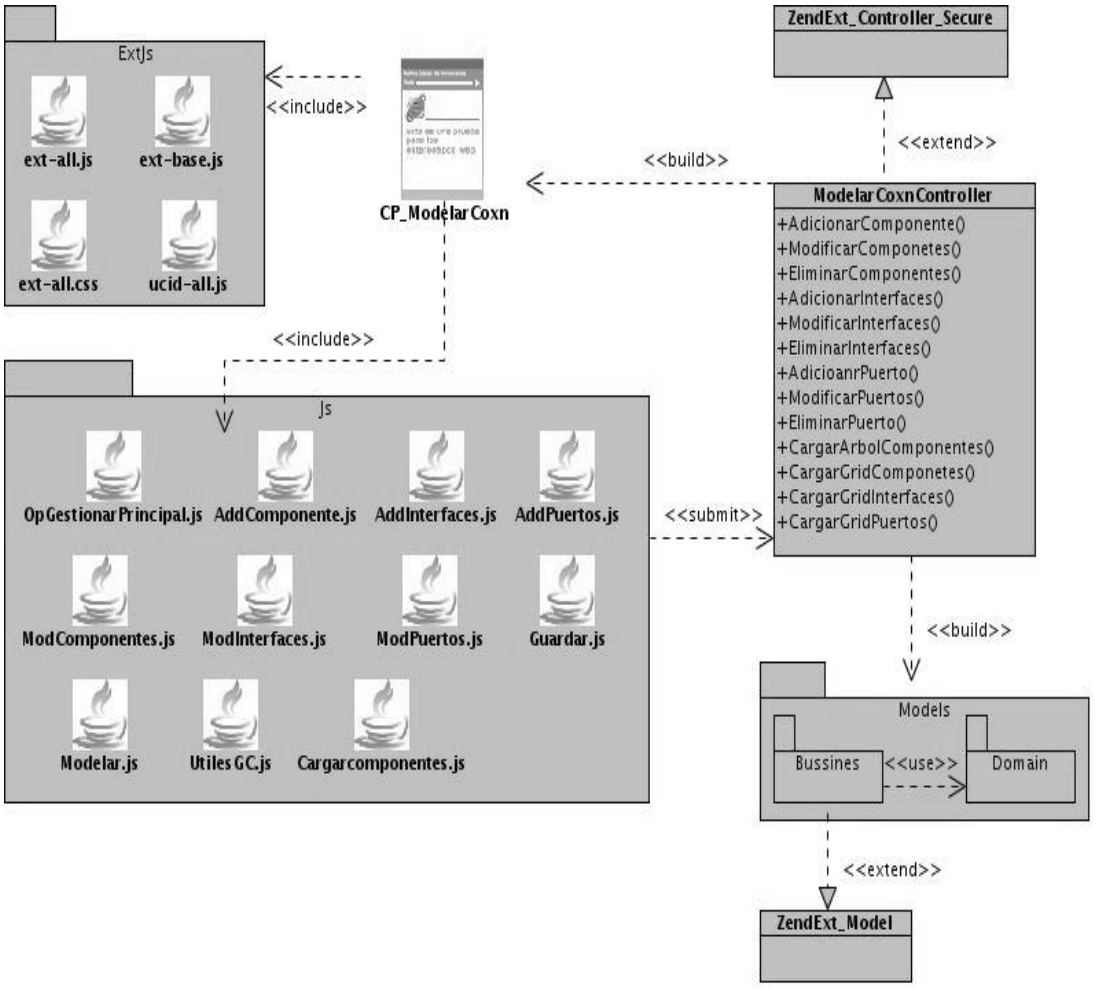


Fig. 5: Diagrama de clases del diseño

Observaciones:

En el diseño realizado se representa con estereotipos web debido a que el software es una aplicación web. Se representan las Clases ModelarCoxn.phtml, que contienen componentes generados en la librería JavaScript ExtJs y son responsables de visualizar a través de los js del paquete, la información necesaria para gestionar las actividades fundamentales de la aplicación. Estas clases a la vez incluyen la controladora ModelarConexionController.php que contienen la implementación de las funcionalidades, y heredan de ZendExt_Controller_Secure para gestionar la seguridad. Se representan también las clases que forman parte de la capa lógica del negocio y de la cual se nutren las clases controladoras. El paquete Model maneja los datos persistentes dentro del componente y contiene el Bussines y el Domain. Hereda de ZendExt_Model para gestionar la seguridad de los datos persistentes ubicados en el Model.

Diagrama de secuencias:

Secuencia con que se ejecuta el proceso de adicionar un nuevo componente

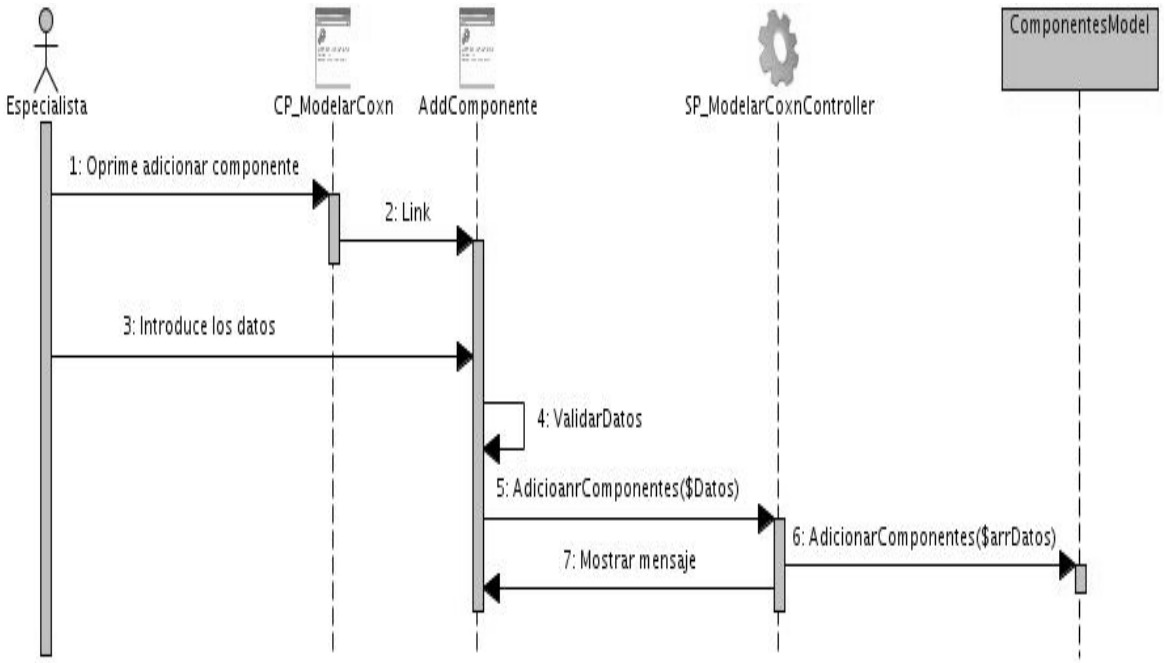


Figura 5: Diagrama de secuencia Adicionar componente

2.4 Implementación

2.4.1 Estructura del marco de trabajo

El gran número de subsistemas, módulos y componentes diseñado para el sistema, la complejidad de los mismos, y la cantidad de información y recursos que gestiona, exige al equipo de arquitectura del proyecto desarrollar y establecer una tecnología o marco de trabajo que permita el desarrollo del sistema informático sobre una arquitectura web, siguiendo el principio de independencia tecnológica. De acuerdo con los principios arquitectónicos definidos por la dirección del proyecto la estructura física del sistema se realizó mediante el empaquetamiento lógico de sus funcionalidades. Dentro de la carpeta raíz de desarrollo del proyecto se encuentran definidas las carpetas apps y web que contienen, respectivamente, la lógica de negocio y vistas de los componentes por módulos referentes a cada subsistemas que se implementan en el proyecto.

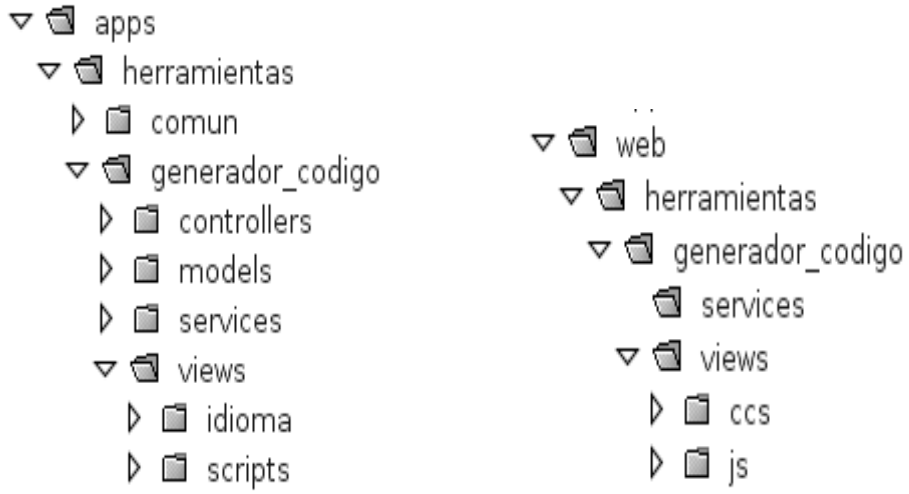


Figura 6: Estructura del marco de trabajo: apps y web

La herramienta cuenta con una estructura como se muestra a continuación

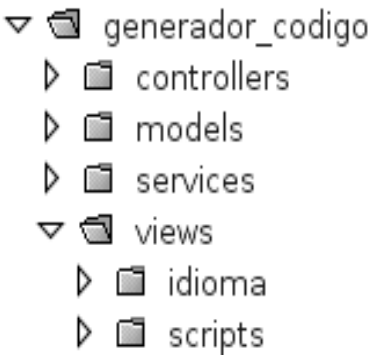


Figura 7: Estructura del marco de trabajo: generador_codigo.

Controllers:

En esta carpeta se almacenarán las clases controladoras encargadas de la gestión de las funcionalidades del componente y el flujo de información entre las vistas y las modelos. Su responsabilidad principal es comunicar las interfaces de usuario con la lógica de negocio de la aplicación.

Models:

En dicha carpeta se programa toda la lógica de negocio del componente en cuestión y se gestiona lo referente a la información física de la base de datos que se archiva en las carpetas bussines y domain.

En la carpeta bussines se generan las clases modelo, se programa toda la lógica de negocio y las acciones de modificación que se van a realizar con determinada entidad de la base de datos, como insertar, actualizar y eliminar.

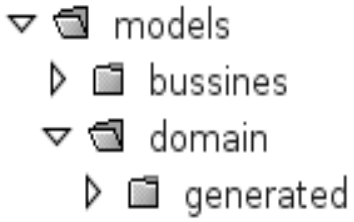


Figura 8: Estructura del marco de trabajo: Models.

La carpeta domain guarda archivos generados por el framework Doctrine, incluye las clases que gestionan las consultas a las tablas de la base de datos. Estas clases heredan de ficheros base definidos como clases php que tienen el mismo nombre de las tablas y se ubican en la subcarpeta generated.

Views:

En esta carpeta se recopilan los ficheros que van a gestionar la capa de presentación, estos ficheros se agrupan en 4 carpetas:

Figura 9: Estructura del marco de trabajo: Views

Idioma:

Contiene ficheros de tipo json que recopilan etiquetas para la gestión los mensajes vinculados a la presentación.

Scripts:

En este directorio se incluyen todas las vistas, para ello se crea una carpeta para cada clase controladora y dentro se incluye la vista o script, archivos de extensión phtml donde se especifica el título de la página que se gestiona y se carga el archivo js que mostrará la presentación.

Css:

En este directorio se encuentran las plantillas y estilos para el diseño del módulo.

Js:

Es la carpeta donde se incluyen las clases JavaScript, al igual que en la carpeta de scripts por cada clase controladora existe una carpeta que tendrá incluido el fichero js correspondiente a dicha clase de control. El fichero js no es más que un documento con la extensión .js donde se escribirá el código correspondiente a la capa de presentación, las interfaces de usuario.

2.4.2 Prototipo de interfaz funcional

En el momento de diseñar los prototipos funcionales de interfaz de usuario, se debe considerar que estos por estar ubicados en la capa de presentación y ser el primer elemento del sistema con el cual va a interactuar el usuario, deben cumplir los principios de usabilidad y funcionalidad definidos en el estándar de interfaz de usuario los cuales serán factores claves para ganar de primer momento la aceptación de usuario. (Ver anexo 7)

2.4.3 Descripción general del funcionamiento de la herramienta

Para la interacción con la herramienta el usuario realiza una petición en el navegador, la cual genera un evento mediante un formulario JS, que envía una notificación a la clase controladora mediante AJAX. El controlador entonces es el encargado de decidir quién va a ser el responsable de obtener los datos de la petición, una clase Bussines si la petición realizada inserta modifica o elimina o una clase Domain si lo que se desea es consultar, esta última es mapeada por el ORM Doctrine y se conecta mediante PDO (Doctrine Record) a la DBO (Capa interna de doctrine basada en PDO nativo) y la DBO es la que se conecta directamente a la base de datos. La clase controladora es la encargada de la lógica propia del negocio, cálculos y procesamientos.

Para la generación de componentes o el usuario selecciona del árbol que contiene las etiquetas de los componentes (Ver anexo 7) cual desea generar. Los datos se envían a la controladora ejecutando el método **generarCodigoAction** (Ver Anexo 8) en el cual se capturan para poder escribir en el directorio donde se guarda la estructura de carpetas del componente junto con sus principales ficheros, esta funcionalidad llama al método recursivo **generarEstructura** (Ver anexo 9), el crea la estructura del componente al igual que la de los componentes que este pueda contener. Para la creación específica de la estructura de carpetas definida por el marco de trabajo para un componente se emplea la funcionalidad **crearcCrpetas**, la cual utiliza las funcionalidades **crearCarpetasWeb** y **crearCarpetasApps** (Ver anexos 10), las cuales crean las vistas y la lógica de negocio del componente. Para la creación de las clases de servicio se especifica le tipo, servicio de entrada

o salida, los servicios de salida se guardan en la carpeta services en la cual se crea un fichero PHP con el nombre de la clase, que dentro contiene todas las funcionalidades definidas por el usuario; a su vez los servicios de entrada se almacenan dentro de la carpeta proxys que contiene la misma estructura que la anterior, la funcionalidad encargada de crear y escribir en estos ficheros es **createServices** (Ver anexo 11). Para establecer la comunicación entre los componentes creados se emplean dos ficheros xml; el (ioc) para escribir la estructura de los servicios que brinda el componente y el (in) para escribir la estructura de los servicios que consume el componente; estos dos ficheros comparten la misma estructura por lo que para su creación se emplearon funcionalidades similares **createIoc** y **createIn** (ver anexo 12).

Para la representación gráfica de los componentes se crea un objeto que contiene los datos del componente seleccionado, dicho objeto se representa a través de una imagen (Ver anexo 13). Para la creación de dicho objeto se emplea la función **addStaticImage**, la cual recibe por parámetro la dirección de la imagen, luego la imagen se introduce dentro de un panel el cual es rendereado para poder ser mostrado en el panel principal del área de modelado.

2.4.4 Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura para facilitar la lectura, comprensión y mantenimiento del código. Debido a la complejidad del sistema Cedrux, el numeroso personal involucrado en él y el alto nivel de integración existente entre sus componentes, el grupo arquitectónico del proyecto definió normas de codificación con el fin de obtener un estándar en la implementación por el equipo de desarrollo que permitiera asegurar la calidad del software, obteniendo un código más legible y reutilizable. A continuación se describen algunos de estos estándares empleados durante la implementación de La herramienta para la generación del código base de la arquitectura del Cedrux.

2.4.4.1 PascalCasing

El estándar PascalCasing establece que los identificadores, nombres de clases, variables, métodos o funciones que están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula. La nomenclatura de las clases de La herramienta para la generación de código base para la arquitectura del Cedrux se realizó sobre la base de este estándar, usando palabras compuestas sugerentes acordes al propósito de la misma.

Nomenclatura de clases según su tipo

- **Controllers**: clases controladoras del negocio.

El nombre de la clase controladora debe estar estructurado por el nombre propio de la misma en mayúsculas seguido por la palabra Controller y heredar siempre de la súper clase del framework ZendExt_Controller_Secure. Ejemplo: (ModelarConexionController)

- **Clases de los modelos**

Business: Clases modelo del negocio.

Las clases modelo tendrán por identificador el nombre de la tabla en la que trabajan seguido por la palabra Model y heredarán de la súper clase del framework Zend_Ext llamada ZendExt_Model. Ejemplo: (ComponetesModel).

Domain: clases entidades del dominio.

Los archivos situados en el domain tienen el mismo nombre de las tablas que representan, definiéndolas como clases php, pueden incluir los prefijos Dat o Nom para diferenciar entre sus usos. Ejemplo: (Componentes).

2.4.4.2 CamelCasing

El estándar CamelCasing es parecido al PascalCasing con la particularidad de que la letra inicial del identificador no comienza con mayúscula. Esta notación se utilizó para el nombre de funciones y atributos.

Nomenclatura de las funciones

El identificativo a emplear para las funciones o métodos se escribe con la primera palabra en minúscula utilizando

La notación CamelCasing y nombres que deduzcan su propósito. Ejemplo (adiciaonarComponetes).

Los denominadores de las acciones de las clases controladoras tienen la peculiaridad de ir seguidos por la palabra “Action”. Ejemplo (adiciaonarComponetesAction).

Nomenclatura de los atributos

Las variables se nombran convenientemente de acuerdo con el estándar CamelCasing. Ejemplo: (idComponete) y (descripción)

2.4.4.3 Notación húngara.

Esta convención, también conocida como notación: REDDICK por el nombre de su creador, se basa en definir prefijos para cada tipo de datos según el ámbito de las variables con el fin de brindar mayor información al nombre de la variable, método o función. La notación húngara fue utilizada para la definición de variables de acuerdo con los siguientes prefijos:

Tabla 5: Prefijos para la creación de variables.

Tipo de datos	Prefijos
Arreglos	arr
Objetos	obj
Enteros	int
Cadena	str
Float	flt
Boolean	bool

Nomenclatura de las variables

El nombre a emplear para las variables se escribe siguiendo la notación CamelCasing comenzando con el prefijo según su tipo de datos. Ejemplo: arrComponentes define un arreglo de componentes.

Nomenclatura de las constantes

La definición de las constantes se realiza utilizando todas las letras en mayúscula. Ejemplo: SUBSISTEMA.

2.4.5 Descripción de clases del componente y tipo.

A continuación se describen las clases y sus operaciones más importantes por componentes agrupándolas de acuerdo con la clasificación siguiente:

Clases Controladoras

Las clases controladoras son responsables de la lógica de negocio, gestionando todo el flujo de datos y operaciones entre la vista y demás clases del negocio.

Clases Modelo

Actúan como intermediarias entre las clases controladoras y las clases entidades. Complementan las funcionalidades de las clases controladoras y realizan las funciones de insertar, modificar y eliminar datos.

Clases Entidad

Las clases entidad modelan los objetos del sistema y comportamiento asociado; frecuentemente representan conceptos y datos persistentes de larga duración. Contienen métodos para la gestión consultas y solicitud de información de la base de datos.

2.4.5.1 Clases de la herramienta

Clase Controladora

Tabla 6: Descripción de la clase ModelarConexionController

ModelarConexionController	
Tipo de calse: Controladora	
Para cada responsabilidad	
Nombre	Descripción
AdicioanrComponentesAction	Adiciona comonentes a un proyecto
CargarArbolComponentesAction	Carga todos los componentes de un proyecto
ModificarComponentesAction	Modifica un compoennete seleccionado
EliminarComponenteAction	Elimina un componente seleccionado
AdicioanrInterfacesAction	Adiciona una interfaz a un componente
CargarInterfacesAction	Carga todas las interfaces de un componente
ModificarInterfacesAction	Modifica una interfaz seleccionada
EliminarInterfazAction	Eliminar una interfaz seleccionada
AdicionarPuertoAction	Adicionar un puerto a una interfaz
CargarPuertosAction	Carga los puertos de una interfaz dada
ModificarPuertosAction	Modifica un puerto seleccionado

EliminarPuertosAction	Elimina un puerto seleccionado

Clase del Modelo

Tabla 7: Descripción de la clase ComponentesModel

ComponentesModel	
Tipo de calse: Modelo	
Para cada responsabilidad	
Nombre	Descripción
InsertarComponente	Inserta un nuevo componente
ActualizarComponente	Modifica un componente dado
EliminarComponente	Elimina un componente dado

2.5 Conclusiones del capítulo.

La descripción del diseño de la solución arquitectónica facilitó la exitosa implementación de la herramienta. Se garantizó además, a partir de la arquitectura propuesta, el acoplamiento con el marco de trabajo del Cedrux. Con la solución propuesta se estimula el desarrollo de nuevas aplicaciones que pueden extender las funcionalidades incluyéndoles nuevos procesos de gestión afines a la modelación y la generación de código; garantizando la flexibilidad de la arquitectura así como la independencia tecnológica del país al utilizar el software libre para su desarrollo.

Capítulo 3: Validación de la solución propuesta.

3.1 Introducción

La dificultad para construir sistemas de software multiplica la probabilidad de que persistan errores aún después de haberse finalizado. Es imposible asegurar que un software se encuentre completamente libre de errores; sin embargo, existen formas y métodos para acercarse lo más posible a un resultado óptimo.

En este capítulo se realiza la validación de la solución propuesta. Se evalúa el diseño empleando métricas de software que proporcionan una medida de la complejidad y calidad del software. Se aplican pruebas con el objetivo de verificar la funcionalidad y estructura del componente.

3.2 Métricas para la evaluación del modelo de diseño propuesto.

Entre las métricas aplicadas durante la evaluación de la solución del diseño del módulo Banco se destacan las siguientes:

3.2.1 Métrica Tamaño Operacional de Clases (TOC)

Tabla 8: Métrica Tamaño operacional de Clase (TOC)

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

Resultados del instrumento de evaluación de la métrica Tamaño Operacional de clase (TOC)

Tabla 9: Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC.

Atributo	Categoría	Criterio
Responsabilidad	Bajo	< =Prom.
	Medio	Entre Prom. y 2* PRom.
	Alto	> 2* Prom.
Complejidad implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	<= Prom.

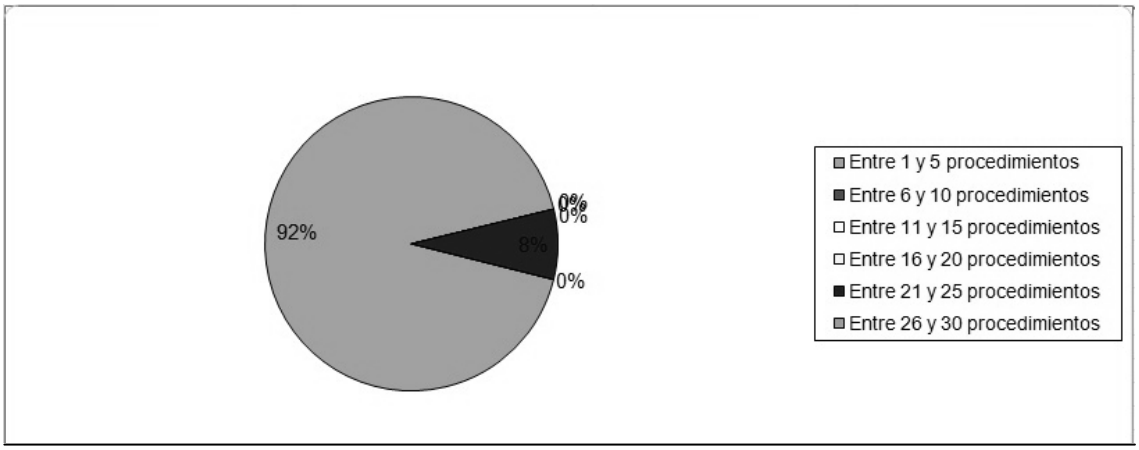


Figura 10: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

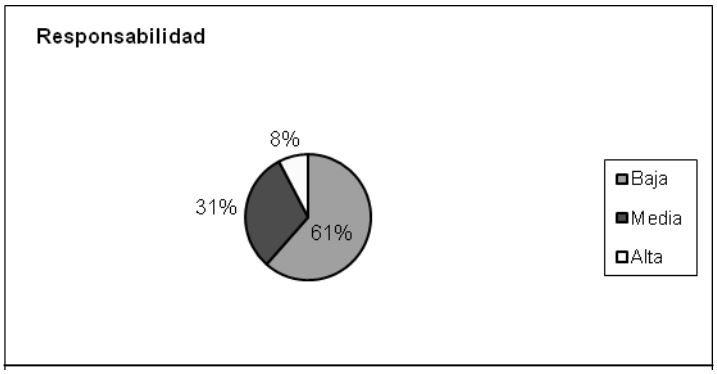


Figura 11: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

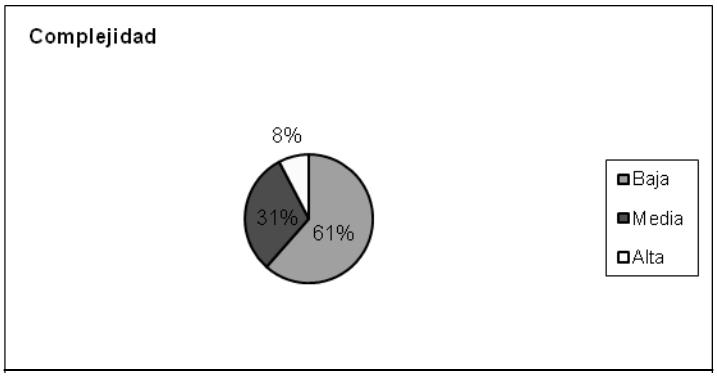


Figura 12: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de implementación.

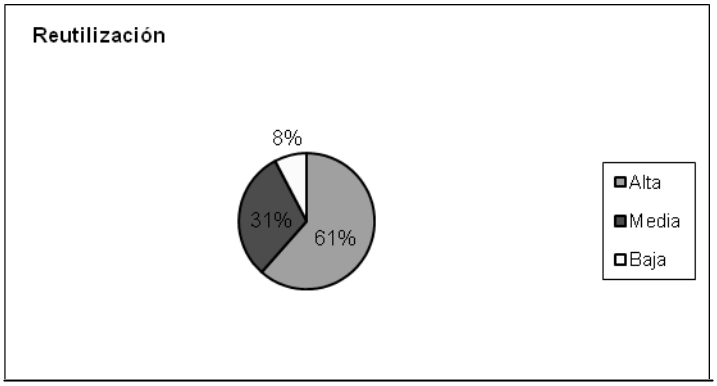


Figura 13: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Los resultados obtenidos durante la evaluación del instrumento de medición de la métrica TOC demuestran que el diseño propuesto para la herramienta se encuentra dentro de los niveles aceptables de calidad, mostrando que la mayoría de las clases el (92%) poseen menos cantidad de operaciones que la media registrada en las mediciones. Para el (92%) de las clases los atributos de calidad fueron evaluados satisfactoriamente confirmando la alta reutilización, baja complejidad y responsabilidad en el diseño.

3.2.2 Métrica Relaciones entre Clases (RC).

Tabla 10: Métrica Relaciones entre Clases (RC).

Relaciones entre Clases (RC)	
Descripción:	Está dada por el número de relaciones de uso de una

	clase con otras.
Atributos que afecta:	**Modo en que lo afecta:**
Acoplamiento	El aumento del RC provoca un aumento del acoplamiento de la clase.
Complejidad del mantenimiento	El aumento del RC provoca un aumento de la complejidad del mantenimiento de la clase.
Reutilización	El aumento del RC provoca una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	El aumento del RC provoca un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Resultados del instrumento de evaluación de la métrica Relaciones entre Clases (RC).

Tabla 11: Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica (RC).

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad Mant.	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.
Cantidad de Pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

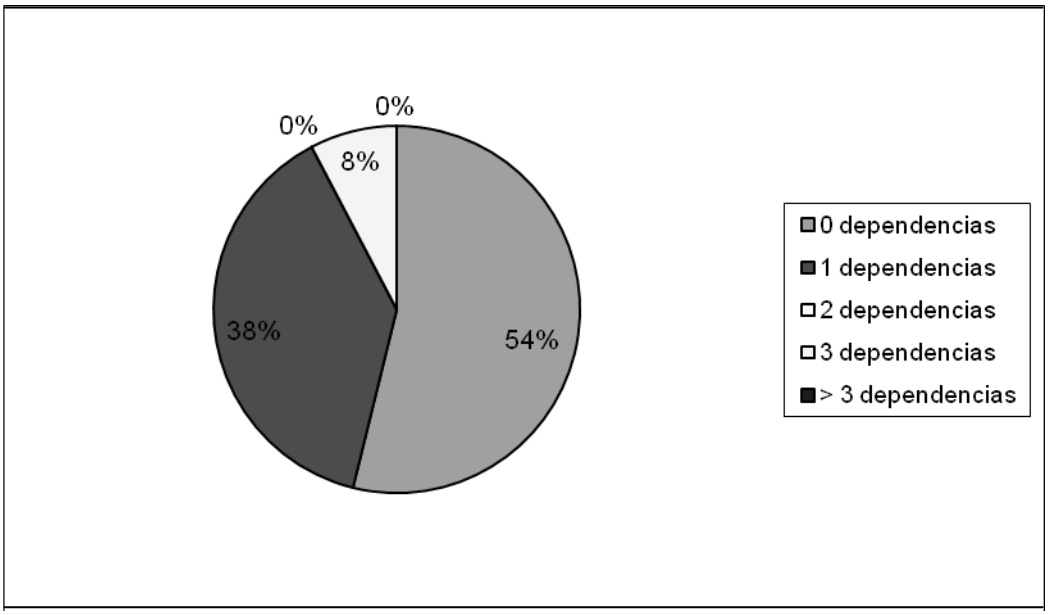


Figura 14: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

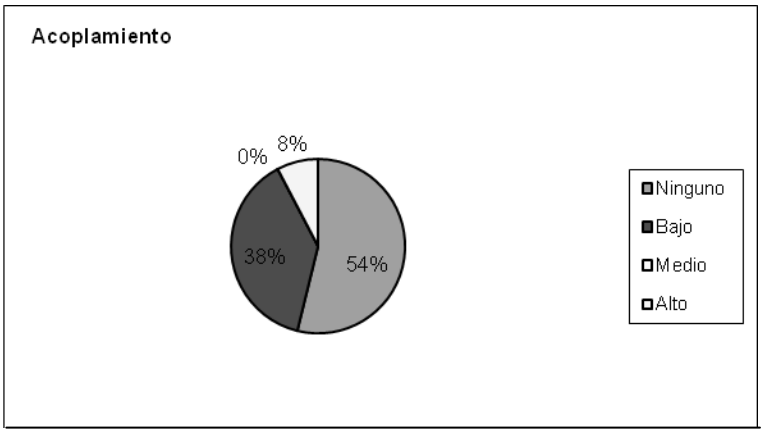


Figura 15: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

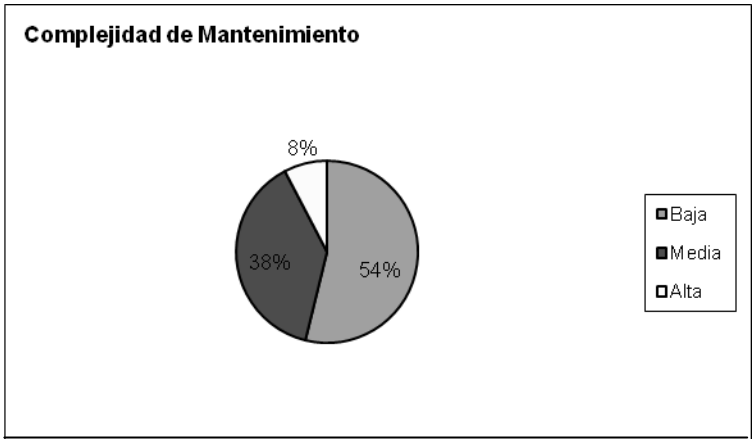


Figura 16: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.

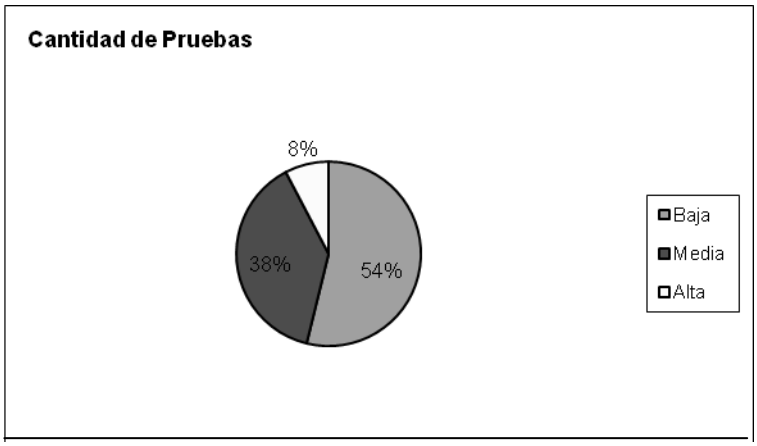


Figura 17: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

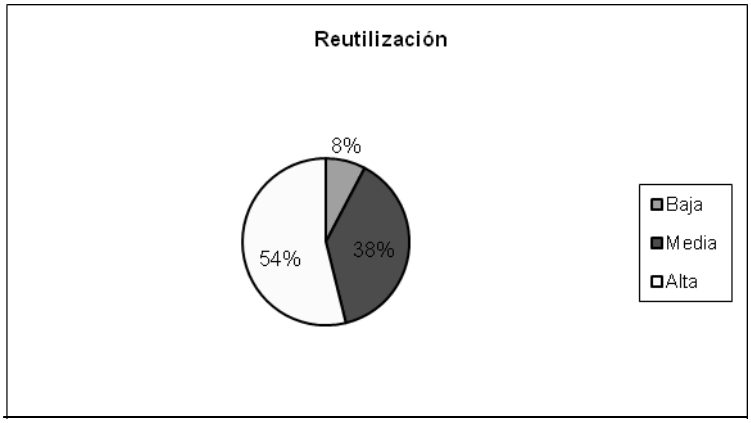


Figura 18: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Los resultados obtenidos durante la evaluación del instrumento de medición de la métrica RC demuestran que el diseño propuesto para la herramienta se encuentra dentro de los niveles aceptables de calidad, mostrando que el (92%) de las clases poseen menos de 3 dependencias. Los atributos de calidad fueron evaluados satisfactoriamente para el (92%) de las clases, confirmando la elevada reutilización, el bajo acoplamiento, complejidad y cantidad de pruebas en el diseño.

Las métricas de software aplicadas posibilitaron estimar la calidad de los atributos internos del producto, demostrando una aceptable calidad de diseño. La solución propuesta contribuirá a la disminución de disturbios durante la implementación de

la herramienta, garantizando la reutilización y agilidad en el proceso de desarrollo de software.

3.3 Métrica para validar el problema

Variables:

TDI – Tiempo de definición de una interfaz.

TDS – Tiempo de definición de un servicio.

TDC – Tiempo de definición de de la estructura base de un componente.

CSI – Cantidad de servicios de una interfaz de servicio.

CIS – Cantidad de interfaces de servicios de un componente.

CC – Cantidad de componentes de la estructura de componentes.

TDEC – Tiempo de definición de la estructura base de todos los componentes.

TDIS – Tiempo de desarrollo de una interfaz de servicio

Métrica:

TDIS = (CSI * TDS) + TDI)

$TDC = \sum^{CIS} TDIS$

$TDEC = \sum^{CC} TDC$

Para su aplicación (Ver Anexo 14)

Valoración de los resultados:

Luego de realizarse una encuesta a un grupo de desarrolladores y arquitectos pertenecientes a diversos proyectos del programa ERP-Cuba y obtener los tiempos promedios de la elaboración de las interfaces de servicios y de los servicios que estas brindan, se evaluaron empleando la métrica antes presentada obteniendo como resultado que se cumple con la variable definida en el problema referente a la disminución del tiempo de implementación de los componentes, demostrando que el empleo de la herramienta para la generación de código base para la arquitectura de Cedrux.

3.4 Niveles de pruebas aplicados.

Durante el desarrollo de la herramienta se definieron pruebas para diferentes objetivos, escenarios o niveles de trabajo. Para aplicarlas, se evalúa el sistema comenzando por los procesos más simples y pequeños, y se avanza progresivamente hasta probar todo el software en su conjunto:

- Pruebas de caja negra: Diseñadas para comprobar el funcionamiento externo de los procesos de la aplicación.
- Pruebas de aceptación: El cliente comprueba que el software funciona según sus expectativas, y se encuentra ejecutando las funciones y tareas para las cuales fue construido.

3.5 Estrategia de pruebas.

Para llevar a cabo el proceso de pruebas a la herramienta se define una estrategia de pruebas con el propósito de garantizar la calidad del software. Se determina aplicar, a partir del método de caja negra la técnica de Partición de equivalencia y luego pruebas de aceptación al software por parte del cliente.

Se utiliza la técnica de Partición de equivalencia para definir casos de prueba que descubran clases de errores, reduciendo el número de casos de prueba a desarrollar para demostrar que las funciones del software son operativas; las entradas se aceptan de forma adecuada y se producen salidas correctas. Los casos de pruebas se diseñarán basados en los procesos y el código, comparando cada funcionalidad implementada con la descrita, para verificar hasta qué punto se cumple con las necesidades del cliente.

Se utilizaran las pruebas de aceptación para determinar el grado de satisfacción del cliente con el producto desarrollado y para detectar posibles errores o incongruencias en la elaboración del software.

3.6 Diseño de Casos de Prueba para caja negra.

Los casos de prueba para caja negra se basan en las diferentes entradas que pueden recibir el software, y sus correspondientes valores de salida; están centrados en realizar pruebas del software a través de la funcionalidad.

Los casos de prueba demuestran que:

- Las funcionalidades son operativas.

- Las entradas se aceptan de forma adecuada produciendo el resultado correcto.
- La integridad de la información externa se mantiene.

Para verificar que la aplicación se comporta según los requerimientos establecidos por el cliente, se diseñan casos de pruebas usando el método de caja negra. A continuación se especifica el caso de prueba para el proceso “Adicionar componente” el cual agregara un nuevo componente a un proyecto dado y poder establecer las relaciones de este con el resto de los componentes de dicho proyecto.

Condiciones de ejecución:

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- Se debe de seleccionar el subsistema Herramientas.
- Se debe de seleccionar la opción Generador de Código Base de la Arquitectura.
- Debe de existir al menos un proyecto creado en el sistema.

Tabla 12: Descripción del caso de prueba para el proceso Adicionar componente.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1:Adicionar Componente	Se brinda la posibilidad de adicionar un nuevo componente, para ello deben de estar creadas las condiciones idóneas para adicionar un nuevo componente, de no ser así, no se podrá realizar. Entre las condiciones que deben cumplirse se encuentran: debe de existir al menos un	EP 1.1: Adicionar un componente cuando existen las condiciones creadas para realizar esta operación correctamente.	– Se presiona el botón **Abrir proyecto.** – **Se presiona el botón Adicionar componente.** – Se introducen los datos del componente. – Se presiona el botón **Aceptar**. – Se muestra el mensaje “Operación

	proyecto creado en el sistema		realizada correctamente". – Se presiona el botón **Aceptar** del mensaje de confirmación emitido por el sistema. – Se cierra la ventana de Adicionar componentes
		EP 1.2: Adicionar un componente dejando campos obligatorios en blanco	– Se presiona el botón **Abrir proyecto.** – **Se presiona el botón Adicionar componente.** – **Se dejan campos en blancos.** – Se presiona el botón **Aceptar**. – Se muestra el mensaje "No se puede realizar la operación ya que existen campos obligatorios en blanco." – Se presiona el botón **Aceptar** del mensaje de confirmación emitido – Se muestra la ventana de Adicionar

			componentes.
		EP 1.3: Aplicar. Cuando existen las condiciones creadas para realizar esta operación correctamente.	– **Se presiona el botón Adicionar componente.** – Se introducen los datos del componente. – Se presiona el botón **Aplicar**. – Se muestra el mensaje “Operación realizada correctamente”. – Se presiona el botón **Aceptar** del mensaje de confirmación emitido por el sistema. – Se muestra la ventana de Adicionar componentes.

		EP 1.4: Aplicar. Dejando campos obligatorios en blanco.	– **Se presiona el botón Adicionar componente.** – **Se dejan campos en blancos.** – Se presiona el botón **Aplicar**. – Se muestra el mensaje “No se puede realizar la operación ya que existen campos obligatorios en blanco.” – Se presiona el botón **Aceptar** del mensaje de confirmación emitido – Se muestra la ventana de Adicionar componentes.
		EP 1.5: Cancelar.	– Se presiona el botón **Cancelar.** – Se cierra la ventana Adicionar componentes

Tabla 13: Descripción de la variable del caso de prueba para el proceso Adicionar componente.

No	Nombre de campo	Tipo	Válido	Inválido
1	Nombre	Campo de texto	Combinación de caracteres	NA

Tabla 14: Datos de prueba del caso de prueba para el proceso Adicionar componente.

Id del escenario	Escenario	Nombre	Respuesta del sistema	Resultado de la prueba
EP 1.1	Adicionar un componente introduciendo datos válidos en el campo nombre.	V(Contabilidad) V(Tecnología)	El sistema confirma que se realizó correctamente la operación emitiendo un mensaje con el texto: “Operación realizada correctamente”.	
EP 1.2	Adicionar un componente dejando campos obligatorios en blanco.	NA	El sistema muestra el mensaje: “No se puede realizar la operación ya que existen campos obligatorios en blanco.”.	
EP 1.3	Cancelar.	NA	Se cancela la operación y no se adiciona un nuevo componente.	

3.7 Conclusiones del capítulo.

Durante el desarrollo de este capítulo se ejecutaron métricas de diseño para caracterizar numéricamente los distintos aspectos del desarrollo y se realizaron pruebas que permitieron evaluar todos los elementos del software. Fueron validadas las funcionalidades implementadas, a través de diferentes pruebas de caja negra, mostrando cómo respondían adecuadamente las principales funcionalidades y garantizando la satisfacción plena de las necesidades reales de los usuarios y demandas del cliente. A partir de los resultados de las métricas y las pruebas aplicadas al sistema, se obtuvo un código de mayor calidad,

funcionalmente probado, y se evaluaron satisfactoriamente los atributos relacionados con el desarrollo del software.

Conclusiones Generales

Una vez terminado el presente trabajo de diploma se puede concluir que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos, para esto:

- Se analizaron ventajas y deficiencias de sistemas informáticos vinculados a la generación de código web; evidenciándose de esta manera la no existencia de una solución informática capaz de cumplir con las necesidades existentes para la generación de código base para la arquitectura del Cedrux.
- Se realizó el diseño y la implementación de la Herramienta para la generación de código base de la arquitectura del Cedrux, con el objetivo de erradicar los problemas en los plazos de entrega y evitar la carga por re-trabajo a los desarrolladores.
- Se evaluó la viabilidad de la herramienta a través de pruebas de software efectuadas para el nivel de unidad, las cuales arrojaron resultados favorables posibilitando dar cumplimiento a las funcionalidades previstas para la misma.

Esta propuesta exhibe valor técnico, donde se destaca la incorporación de principios por los que se mide la factibilidad de un diseño de software, ejemplo: la utilización de patrones que posibilita la reutilización, garantizando la sostenibilidad y mantenimiento del sistema.

Recomendaciones

Al concluir el presente trabajo de diploma, considerando cumplidos los objetivos trazados en el mismo, se recomienda:

- Continuar realizando pruebas de calidad a la herramienta.
- Identificar nuevas funcionalidades a desarrollar en versiones posteriores de la herramienta.
- De manera general refinar la solución propuesta a partir de sugerencias y otras recomendaciones.

Bibliografía

1. **Informática, Instituto nacional de Estadistica e.** *Herramientas Case.* s.l. : Talleres de la Oficina de Impresiones de la Oficina Técnica de Difusión Estadística y Tecnología Informática del Instituto Nacional de Estadística e Informática (INEI), 1999.

2. [En línea] 17 de 07 de 2010. http://www.todoroms.com/multi-e-world-tech-php-maker-v7-1-incl-keygen-lz0.

3. hkvstore. [En línea] http://www.hkvstore.com/phpmaker/.

4. [En línea] http://www.visual-paradigm.com.

5. [En línea] http://www.visual-paradigm.com./product/vpuml/tutorials/phpdoctrine.jsp.

6. [En línea] http://bouml.free.fr/doc/index.html.

7. phpexperto. [En línea] http://phpexperto.blogspot.com/2008/12/generando-cdigo-php-desde-uml.html.

8. [En línea] 2009. http://www.ctr.unican.es/asignaturas/procodis_3_II/Doc/SeminarioHerramientas.pdf.

9. [En línea] http://argouml.tigris.org/.

10. [En línea] http://www.methodsandtools.com/tools/tools.php?argouml.

11. [En línea] http://argouml-stats.tigris.org/nonav/documentation-es/manual-0.30/ch03s03.html.

12. desarrolloweb. [En línea] http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.

13. **Vega, Yanet.** *Definición del ciclo de vida del proyecto.* Habana : s.n., 2009.

14. **Fernández, Osmar Leyet.** *Documento Línea Base de proyecto-CEDRUX-1.0.* 2010.

15. **Lidia Fuentes, osé M. Troya y Antonio Vallecillo.** *Desarrollo de Software Basado en Componentes.* Málaga, Spain : s.n., 2007.

16. [En línea] http://es.debugmodeon.com/articulo/el-patron-mvc.

17. [En línea] http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf.

18. El mundo informatico. [En línea] http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/.

19. **Prieto, Félix. 2009.** Patrones de diseño. 2009.

20. [En línea] http://www.programacion.com/tutorial/uml/1/.

21. [En línea] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

22. [En línea] http://www.maestrosdelweb.com/editorial/phpintro/.

23. [En línea] http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo.

24. [En línea] http://php.net/manual/es/book.pdo.php.

25. [En línea] http://www.json.org.

26. [En línea] http://framework.zend.com/manual/en/.

27. [En línea] http://www.doctrine-project.org.

28. [En línea] http://www.abartiateam.com/desarrollo-web/200602_uso-de-la-tecnologia-ajax-en-el-desarrollo-web.

29. [En línea] http://picandocodigo.net/downloads/docs/subversion-presentacion-01.pdf.

30. [En línea] http://linux.ciberaula.com/articulo/linux_apache_intro.

31. UptoDown. [En línea] http://postgresql.uptodown.com/ .

32. Mozilla Europe. [En línea] http://www.mozilla-europe.org/es/ .

33. Desarrolloweb. [En línea] http://www.desarrolloweb.com/manuales/21/.

34. msdn. [En línea] http://msdn.microsoft.com/es-es/library/aa292164(v=vs.71).aspx.

35. [En línea] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.

36. [En línea] http://www.desarrolloweb.com/articulos/25.php.

37. **Dr. Rolando Alfredo Hernandez León, Dra Sayda Coello Gónzalez.** *El Paradigma Cuantitativo de la Investigación Científica.* Ciudad de la Habana : Editorial Universitaria (EDUNIV), 2002.

38. Netbeans. *Netbeans.* [En línea] http://netbeans.org/.

Glosario de términos

Actividad: Define un conjunto de tareas, funciones y definiciones, agrupadas bajo un mismo contexto, teniendo en cuenta la relación entre estas funciones, el lugar y el personal que las realiza, basados además en los principios del control interno.

Algoritmo: Es un conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea o resolver un problema.

Base de datos: Es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

CSS (Cascading Style Sheets): Es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). Las hojas de estilo en cascada permiten separar la estructura de un documento de su presentación.

Componente: Fragmento de un sistema de software que puede ser ensamblado con otros fragmentos para formar piezas más grandes o aplicaciones completas

DOM (Document Object Model): Es un conjunto de utilidades específicamente diseñadas para manipular documentos XML o XHTML y HTML. Técnicamente, es una API de funciones que se pueden utilizar para manipular las páginas XHTML de forma rápida y eficiente.

Fichero: Es una manera particular de codificar información para almacenarla en un archivo informático.

Interfaces: Clases de servicios encargadas de almacenar las principales funcionalidades para establecer la comunicación con otros componentes.

Open Source: código fuente disponible públicamente.

ORM (Object Relational Mapping): Es una técnica de programación para convertir datos entre el lenguaje de programación orientado a objetos utilizado y el sistema de base de datos relacional. El mapeo objeto-relacional posibilita el uso de las

características propias de la orientación a objetos (básicamente herencia y polimorfismo)

Protocolo PDO: Permite la transmisión de datos en tiempo real y con identificadores de alta prioridad hacia/desde el bus evitando sobrecargarlo con información redundante.

Puertos: Funcionalidades de las clases servicios.

SOA (Service Oriented Architecture): Arquitectura de software que propone la integración y el desarrollo de aplicaciones construidas sobre los servicios y la composición de servicios

Anexos

Anexo 1: Diccionario de datos.

Tabla 15: Tabla Componentes del modelo de datos

Nombre tabla	Componentes	
Descripción	En esta tabla se guarda el árbol de componentes	
Atributo	Tipo	Descripción
idcomponente	Numeric	Identificador principal de la tabla
descripción	Varchar	Campo para almacenar el nombre del componente
idpadre	Numeric	Identificador del padre al cual pertenece el componente
posiciónx	Numeric	Posición que ocupa el componente en el eje de las x
posicióny	Numeric	Posición que ocupa el componente en el eje de las y
largo	Numeric	Longitud del componente
ancho	Numeric	Ancho del componente
color	Varchar	Color del componente
proyectoidproyecto	Numeric	Identificador del proyecto al cual pertenece el componente

Tabla relacionada	Tipo de relación
Interfaces	Uno a mucho
Proyecto	Uno a mucho
Conexion	Muchos a muchos

Tabla 16: Tabla Interfaces del modelo de datos

Nombre tabla	Interfaces	
Descripción	En esta tabla se guardan las interfaces asignadas a los componentes	
Atributo	Tipo	Descripción
idinterface	Numeric	Identificador principal de la tabla
descripción	Varchar	Campo para almacenar el nombre de la interfaz
tipo	Numeric	Campo para asignar un tipo de interfaz
componenteidcomponente	Numeric	Identificador del componente al cual pertenece la interface
Tabla relacionada		Tipo de relación
Componente		Uno a mucho
Puertos		Uno a mucho

Tabla 17: Tabla Interfaces del modelo de datos

Nombre tabla	Puertos

Descripción	En esta tabla se guardan los puertos asignados a las interfaces	
Atributo	Tipo	Descripción
idpuerto	Numeric	Identificador principal de la tabla
descripción	Varchar	Campo para almacenar el nombre del puerto
tipodato	Numeric	Parámetros que recibe el puerto
interfaceidinterface	Numeric	Identificador de la interfaz a la cual pertenece el puerto
Tabla relacionada		Tipo de relación
Interfaces		Uno a mucho
Puertos		Uno a mucho

Tabla 18: Tabla Proyecto del modelo de datos

Nombre tabla	Proyecto	
Descripción	En esta tabla se Guardan los proyectos existentes	
Atributo	Tipo	Descripción
idproyecto	Numeric	Identificador principal de la tabla
descripción	Varchar	Campo para almacenar el nombre del proyecto
Tabla relacionada		Tipo de relación
Componente		Uno a mucho

Tabla 19: Tabla Conexión del modelo de datos

Nombre tabla	Conexión	
Descripción	En esta tabla se guardan las conexiones que existen entre los componentes	
Atributo	Tipo	Descripción
idconexion	Numeric	Identificador principal de la tabla
iddestino	Numeric	Identificador del componente que recibe servicios
idorigen	Numeric	Identificador del componente que brinda servicios
protocolo	Varchar	Protocolo a través del que se establecen las conexiones
Tabla relacionada		Tipo de relación
Componente		Muchos a muchos

Anexo 2: Diagrama de secuencia Modificar componente.

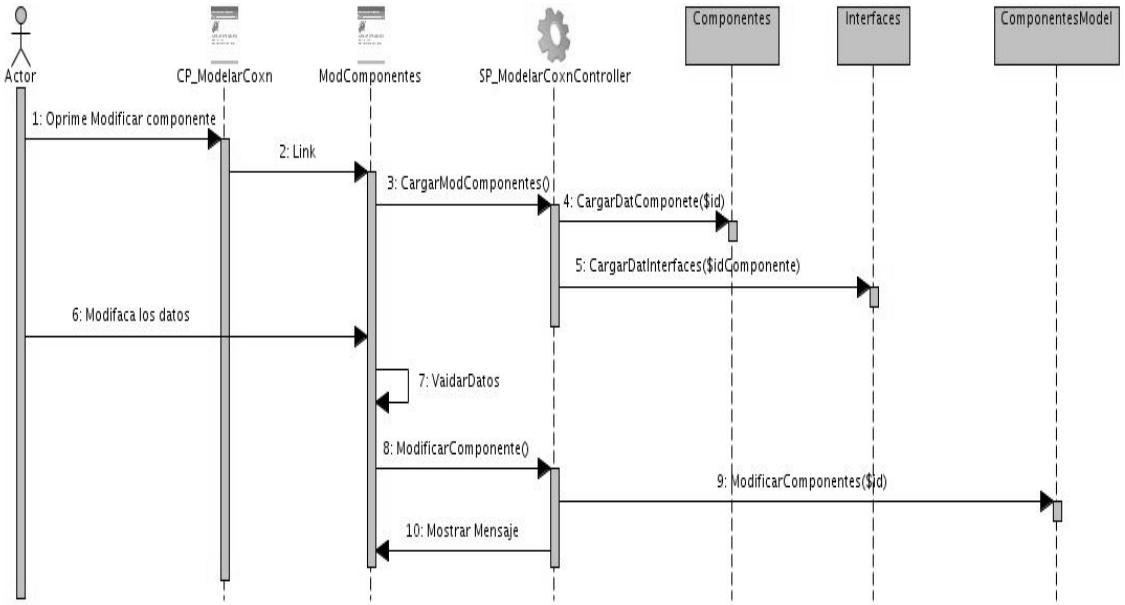


Figura 19: Diagrama de secuencia del diseño Modificar componente

Anexo 3: **Diagrama de secuencia Eliminar componente**

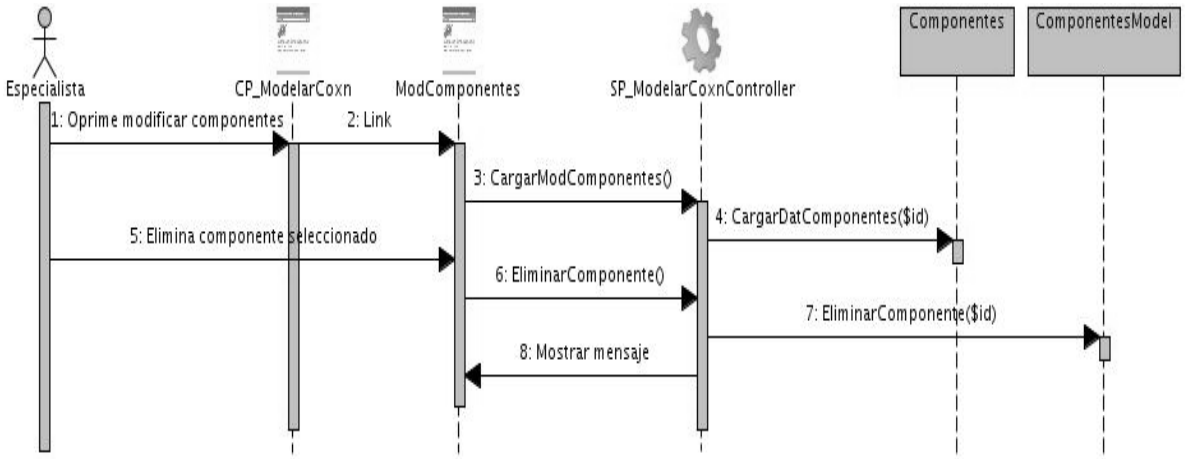


Figura 20: Diagrama de secuencia del diseño Eliminar componente.

Anexo 4: Diagrama de despliegue

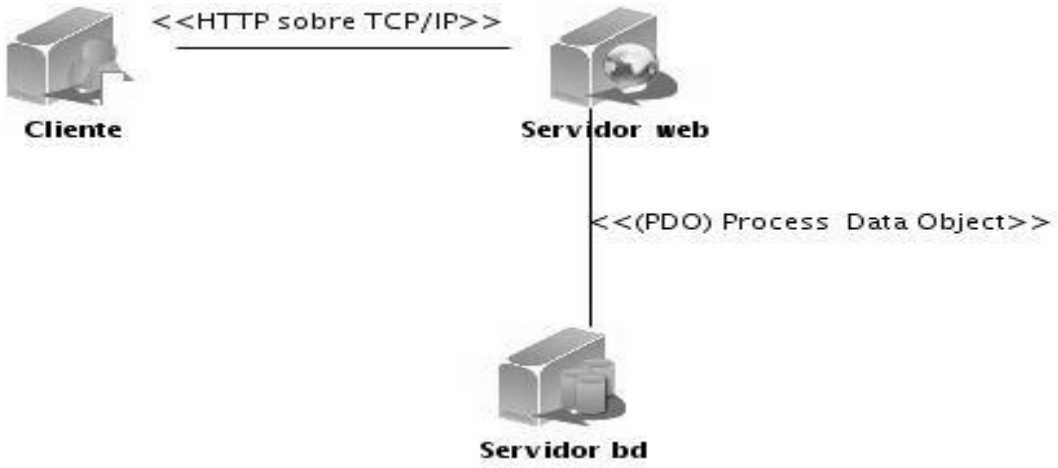


Figura 21: Diagrama de despliegue

Anexo 5: Instrumento de medición de la métrica Tamaño operacional de clases (TOC).

Tabla 20: Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Complejidad de Implementación y Reutilización).

No	Subsistema	Clases	Cantidad de Métodos	Responsabilidad	Complejidad	Reutilización
1	Generador de Código	ModelarConxController	23	Alta	Alta	Baja
2		ComponentesConexionmodel	0	Baja	Baja	Alta
3		ComponentesModel	3	Media	Media	Media
4		ConexionModel	0	Baja	Baja	Alta
5		InterfacesModel	3	Media	Media	Media
6		ParametrosModel	3	Media	Media	Media
7		ProyectoModel	1	Baja	Baja	Alta
8		PuertosModel	3	Media	Media	Media
9		Componentes	2	Baja	Baja	Alta
10		ComponentesConexion	0	Baja	Baja	Alta
11		Conexion	0	Baja	Baja	Alta
12		Interfaces	2	Baja	Baja	Alta
13		Parametros	1	Baja	Baja	Alta
14		Proyecto	1	Alta	Alta	Baja
15		Puertos	2	Alta	Alta	Baja