Universidad de las Ciencias Informáticas

Facultad 6

Título “Desarrollo del módulo Administrador de Reportes del Generador Dinámico de Reportes”

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

Autor: Juan José Hernández Carvajal

Tutores: Ing. Raidel Oceguera Ravelo

Ing. Yasmany Hernández Hernández

Ciudad de la Habana 2011
“Año 53 de la Revolución Cubana”

“Si en el pasado ocuparon un lugar los héroes de la Sierra.
En el futuro ocuparán ese lugar los héroes de la Ciencia.”

Ernesto Che Guevara

Declaración de autoría

Declaro que soy el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas, sus derechos patrimoniales sobre la misma con carácter exclusivo.
Para que así conste firmamos la presente a los _____ días del mes de _____ del año _____.

_____
Juan José Hernández Carvajal.
Firma del Autor

_____
Ing. Raidel Oceguera Ravelo.
Firma del Tutor

_____
Ing. Yasmany Hernández Hernández.
Firma del Tutor

Datos de Contacto

Tutor: Ing. Raidel Oceguera Ravelo
Ingeniero en Ciencias Informáticas
E-mail: roceguera@uci.cu

Tutor: Ing. Yasmany Hernández Hernández
Ingeniero en Ciencias Informáticas
E-mail: yhhdez@uci.cu

Consultante: Ing. Marleysi López Duque
Ingeniero en Ciencias Informáticas
E-mail: mduque@uci.cu

Agradecimientos

A mi abuelo que es mi razón de ser en esta vida, mi ejemplo a seguir en todo momento, que siempre me decía las palabras que necesitaba escuchar en los momentos más difíciles sin importar cuales fueran, siempre está ahí con esa sonrisa esperándome. Gracias por ser el mejor abuelo del mundo.

A mi mamá: Te agradezco que estés siempre conmigo, durante todos estos años de mi vida, apoyándome y ayudándome en cada paso de este largo camino, ya lo logramos es solo un sueño hecho realidad.

A mi papá: Por ser mi ejemplo, y brindarme todo su amor incondicional, por estar siempre ahí cuando lo necesité.

A mi hermano que siempre he tratado de ser un ejemplo para él. Aunque siempre le estoy peleando es para que aprenda y no cometas los errores que he cometido yo y así sea una mejor persona.

A mis otros padres Alberto y María que los quiero y siempre los llevaré en mi corazón igual que a Reinier que es un hermano para mí.

.A mi abuela que aunque ya no está con nosotros estaría muy orgullosa de haber visto este momento.

A mi novia por estar siempre a mi lado y brindarme tu amor, cariño y comprensión en todo momento, eres lo más bello que me ha sucedido en esta vida. Te quiero con locura.

A Yamek que sin ti mi hermano nunca hubiera podido llegar a este día, ya terminamos nos vamos a casa, siempre serás mi hermano como me presentabas, al igual que el gordo de Luis Gabriel, que nos espera en ciego. No importa el lugar que sea nos mantendremos unidos, cuídate, toma poco y no hagas trampas en el dominó que ese no es el ejemplo que te di....

A todas esas otras viejitas mías de allá del barrio que siempre me han dado su amor y cariño, a juanita a conejo, Juana la negra, La gorda, Diana, Elio, Onésimo, Martha, Eva, Liudmila, en fin a todos en el barrio sin excepción de ninguno.

A el nani que aunque con un poco de lentitud siempre sabía darme un consejo y ayudarnos en todo momento.

A Roberto Carlos (Appa) que siempre me peleaba para que estudiara y me acompañaba en las cafeterías en altas horas de la noche. Espero que estés bien y nos vemos en la finca…

Al negrón Osdel que un día voy a poder decir yo estudié con ese campeón. Sigue adelante que no existe nada imposible, solo cosas un poco distantes.

A todos mis amigos aquí en esta universidad, que no me alcanzaría el papel para nombrarlos, en especial a Los cremas que nos mantuvimos desde el inicio y llegamos al final, a todo aquel grupo maravilloso del 6103.

A la pilla y a bombón por soportarme con todos mis resabios de viejo.

Al botaperro (Dino) te quiero so descarado, nunca te olvides de mi yo no lo haré.

A Freddy y mi sobrinito que son muy especiales para mí.

A mis tutores que con su paciencia y profesionalismo hicieron posible la realización de este trabajo.

Al tribunal y al oponente que con sus revisiones hicieron posible el perfeccionamiento de este trabajo de diploma.

Quiero agradecer a todas esas personas que de una forma u otra aportaron su granito de arena durante estos 5 años para que fuera posible alcanzar esta meta.

No importa donde me encuentre siempre los llevaré en mi corazón, los quiero mucho

Juan José Hernández Carvajal

Dedicatoria

A mi abuelo que es lo más grande que tengo en este mundo.

A mi mamá querida que es la luz de mi vida y mi razón de ser, es mi guía y mi apoyo, es quien se ha dedicado la vida entera a educarme y formarme para lograr lo que soy, a ella va dedicado este trabajo porque es merecedora de él y de mucho más.

A mi papá que es mi ejemplo a seguir durante toda la vida. Espero que estés orgulloso de mí. Te quiero mucho…

A mi hermano para que siga adelante en la vida y se supere cada día más.

A Yamek que te graduaste 2 veces en un mismo año, siempre me apoyaste en todos los momentos vividos a lo largo de todos estos años.

A mi novia que es la luz de mi vida, gracias por existir y estar siempre a mi lado, te adoro mi odiosita.

A mis suegros gracias por acogerme y brindarme ese amor y cariño incondicional.

A Olgalidea que siempre estaba preguntándome que cuando terminábamos, ya es todo nuestro y hay un pedacito que te corresponde de ese título. Un beso y te quiero mucho.

A toda mi familia y amigos que de una forma u otra me apoyaron en todo momento, gracias por educarme y hacer de mi una mejor persona cada día.

Juan José Hernández Carvajal

Resumen

En la Universidad de las Ciencias Informáticas (UCI) se desarrollan un número significativo de proyectos en centros de desarrollo de software vinculados a las facultades. El Centro de Tecnología de Gestión de Datos (DATEC) es uno de ellos y cuenta con el proyecto Generador Dinámico de Reporte, este es una herramienta que permite la construcción de reportes y provee una forma transparente al usuario para realizar consultas a la base de datos y obtener información de ella en forma de reportes, haciendo posible sintetizar la información esencial de sus negocios. El presente trabajo de diploma tiene como objetivo el diseño e implementación del módulo Administrador de Reportes del Generador Dinámicos de Reportes V (2.0), que permita soportar procesos de administración y entrega de los reportes de manera dinámica. Con esto se pretende brindar la posibilidad a los usuarios de realizar subscripciones de forma dinámica, para recibir los reportes mediante servicios ftp o correo electrónico, así como la configuración del formato en el que se desean. Para desarrollar el módulo Administrador de Reportes se utilizó como lenguaje de programación PHP, utilizando como apoyo el Framework Symfony. Para el modelado de la solución se utilizó Visual Paradigm y como entorno de desarrollo integrado NetBeans. El módulo implementado se sometió a una serie de pruebas funcionales mediante las cuales se comprobó el correcto funcionamiento del mismo. Todo esto permite que el sistema pueda ser utilizado a nivel empresarial o embebido en aplicaciones personalizadas para cubrir completamente el ciclo de vida de los reportes de forma dinámica e intuitiva, lo cual es de suma importancia pues beneficia directamente el proceso de desarrollo de software en DATEC.

Palabras Clave: diseño, implementación, generador dinámico de reporte, subscripción, ciclo de vida, correo electrónico, ftp, PHP, Framework, Symfony, Visual Paradigm, NetBeans.

Tabla de Contenidos

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

En el entorno competitivo empresarial actual, obtener información valiosa es esencial. Con el surgimiento y desarrollo de nuevas tecnologías, las empresas de hoy en día crean y almacenan una gran cantidad de datos, los cuales provienen de una gran variedad de fuentes como: clientes, proveedores, asociados, consultores, empresas de investigación de mercados. Al incrementarse notablemente el flujo de la información es cada vez más complicado tomar decisiones y así convertir en útil esta información. La tecnología facilita el proceso de recopilación y análisis de los datos manejados.

Varias de estas organizaciones cuentan con diversos sistemas dispersos, en algunos casos cada uno posee sus propias fuentes de datos y mecanismos de representación. Provocando que el mantenimiento de información actualizada sea extremadamente difícil.

Los procesos de descripción, análisis y representación de la información, así como las nuevas tecnologías asociadas a ellos, son un medio para sintetizar la información esencial y así son una fuente transcendente para la toma de decisiones y a su vez contribuir al desempeño tanto individual como colectivo. Ayudando a la construcción positiva de la empresa, en función de obtener utilidades y crear los valores propios de la organización.

Uno de los principales retos de las empresas es proporcionar la información adecuada a las personas en el momento adecuado. Los responsables de utilizar la información periódicamente necesitan acceder, procesar y visualizar datos de negocio que pueden estar distribuidos por toda la organización y/o fuera de ella.

Los reportes facilitan a los responsables observar la marcha del negocio y que de esta forma identificar nuevas oportunidades de negocio o servicios. El carácter determinante de los mismos, estimula a un mayor esfuerzo para garantizar su eficiente obtención.

Generar reportes ante el gran volumen de datos y la naturaleza dispersa de los mismos, muestra gran complejidad cuando se realiza de forma manual, sin embargo, debido a la informatización de este proceso existen en el mundo diferentes herramientas que permiten la generación de reportes, se pudieran citar algunas como*: Active Reports, JasperReport, y CrystalReports*, pero estas solo cubren parte del ciclo de vida de los reportes, sin embargo existen otras como: *Microsoft SQL Server2005 Reporting Services*, que además, de facilitar el diseño y generación de reportes, soporta los procesos

de administración y entrega de los mismos, pero su costo es muy elevado y es una herramienta privativa.

Aprovechando la desafiante ola que figura el desarrollo de software en el mundo, para Cuba la creación de la UCI (Universidad de las Ciencias Informáticas) es quizás el más prometedor de los pasos dados en ese sentido. En dicha institución, se destaca la creación de DATEC (Centro de Tecnologías de Gestión de Datos). Dicho centro, anuncia el inicio de un conjunto de proyectos que tributan al desarrollo de herramientas y servicios informáticos, especializados en el almacenamiento y análisis de datos. Entre las tecnologías en construcción relacionadas al análisis de datos, despunta la suite de herramientas PATDSI (Plataforma de Ayuda para la Toma de Decisiones y Soluciones Integrales). El sistema GDR (Generador Dinámico de Reportes) incluido en dicha plataforma es uno de los sistemas que presenta las siguientes deficiencias.

El Generador Dinámico de Reportes cuenta en la actualidad con un conjunto de módulos que brindan una serie de funcionalidades para dar soporte a todo el ciclo de vida de los reportes. El módulo Administrador de Reportes, como su nombre lo indica, es de gran importancia para el sistema, en su versión actual sus funcionalidades son limitadas y su ampliación está dificultada por la arquitectura actual del sistema, así como las tecnologías involucradas. Entre las funcionalidades más solicitadas por los clientes se encuentran las subscripciones automáticas a reportes, donde los usuarios puedan solicitar los mismos con una determinada recurrencia, permitiendo configurar el envío de una subscripción hacia un servidor de ficheros, además del formato en que se desea el reporte, también se pueden solicitar las subscripciones mediante correo electrónico, permitiendo la integración con sistemas de seguridad, a través de estándares abiertos. En la actualidad el módulo no brinda la posibilidad de realizar estas tareas. Para dar solución a dicha problemática es necesario desarrollar una herramienta que sea capaz de gestionar el proceso administración de reportes de forma dinámica y personalizada.

De acuerdo a la problemática mencionada se determinó el siguiente **problema de la investigación:** ¿cómo administrar los reportes en el Generador Dinámico de Reportes V (2.0)?

Este problema determinó que el **objeto de estudio** de esta investigación sea: proceso de desarrollo de software del módulo Administrador de Reportes del Generador Dinámico de Reportes V (2.0) y se precisa como **campo de acción:** diseño e implementación de software en el módulo Administrador de Reportes.

Para dar solución al problema planteado se define como **objetivo general:** desarrollar el módulo Administrador de Reportes Versión (2.0) que permita la administración de reportes de forma dinámica.

Este objetivo general se desglosa en los siguientes **objetivos específicos:**

- ✓ Diseñar el módulo Administrador de Reportes Versión (2.0).
- ✓ Implementar el módulo diseñado.
- ✓ Realizar las pruebas unitarias que demuestren el correcto funcionamiento del módulo implementado.

Para dar cumplimiento a los objetivos se realizaron esencialmente las siguientes **tareas**:

- ✓ Revisión de las herramientas existentes para la administración de reportes en sistemas generadores de reportes.
- ✓ Investigación de las herramientas para el diseño e implementación de aplicaciones web.
- ✓ Implementación del diseño para dar solución a los requerimientos funcionales.
- ✓ Realización de pruebas unitarias para validar el correcto funcionamiento del módulo.
- ✓ Definición de los requerimientos no funcionales para el correcto funcionamiento del módulo.

El documento está estructurado de la siguiente manera: resumen, introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía y glosario de términos.

Capítulo 1: Fundamentación teórica: Estudio del arte y las herramientas de reportes que existen en el mundo. Se realiza un análisis crítico y valorativo del estado del arte en el tema.

Capítulo 2: Diseño de la Solución: En este capítulo se obtendrán los artefactos generados durante la etapa de diseño, además se presenta una visión del diseño del sistema.

Capítulo 3: Implementación y Pruebas: Se presenta una vista abstracta del funcionamiento del sistema y muestras de las pruebas unitarias.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

El presente capítulo aborda temas relacionados con el estado del arte de las diferentes herramientas que existen en el mundo para la administración de reportes. Además, se realiza un estudio de las metodologías de desarrollo de software, lenguajes de programación, *frameworks* (marcos de trabajo por su traducción al español) y tecnologías que se utilizan para el desarrollo de *software* basado en la web, definidas previamente por el grupo de arquitectura del proyecto Generador Dinámico de Reportes V(2.0).

1.1 Entorno de Desarrollo Integrado *NetBeans*

NetBeans 7.0 es un entorno de desarrollo integrado (Integrated Development Environment IDE por sus siglas en inglés) de código abierto y gratuito para desarrolladores de software. Los IDE son un conjunto de herramientas de programación, estas aplicaciones se pueden dedicar a un solo lenguaje de programación o a varios lenguajes. La misión principal de los IDE es proporcionar un marco de trabajo fácil y amigable. El proyecto *NetBeans* es apoyado por una gran comunidad de desarrolladores y ofrece una amplia documentación y recursos de capacitación. Además, ofrece todas las herramientas necesarias para crear aplicaciones de escritorio, empresariales, web y móviles con el lenguaje *Java*, *JavaFX*, C/C ++ y lenguajes dinámicos como *PHP, JavaScript*, *Groovy* y *Ruby*. *NetBeans* es fácil de instalar y se puede ejecutar tanto en *Windows*, *Linux*, *Mac OS X* como en *Solaris*. Otras de las características novedosas que integra este IDE son:

- ✓ Introducción de JDK 7 (Java Development Kit, Kit de desarrollo de Java, por su traducción al español) de apoyo, incluidas las mejoras del editor (sintaxis, pistas).
- ✓ Asistente simplificado con instalación guiada al controlador JDBC.
- ✓ Edición y despliegue de los procedimientos almacenados.
- ✓ Mejora de la detección de cambios externos. (1)

1.2 Marco de Trabajo *Symfony*

Symfony 1.4.9 es un marco de trabajo que facilita y automatiza una gran cantidad de funcionalidades que resultan comunes a la hora de realizar algunos tipos de software. Las principales funcionalidades que proveen estas herramientas son, entre otras, la implementación de un patrón arquitectónico como Modelo-Vista-Controlador, así como algunos asistentes para el diseño de interfaces de usuario de

mayor calidad en menos tiempo. *Symfony* es un *framework (marco de trabajo, por su traducción al español)* para desarrollar aplicaciones web que fue programado en PHP 5 y está enfocado al desarrollo en el mismo lenguaje de programación. Incorpora una nueva capa por encima de PHP y proporciona herramientas que simplifican el desarrollo de las aplicaciones web complejas. También separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Es compatible con la mayoría de los gestores de bases de datos, entre ellos *MySQL, PostgreSQL, Oracle* y *SQLServer de Microsoft*. Otras de sus características más novedosas son:

- ✓ Fácil de instalar y configurar en la mayoría de plataformas.
- ✓ Independiente del sistema gestor de bases de datos.
- ✓ Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- ✓ Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- ✓ Preparado para aplicaciones empresariales, y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- ✓ Código fácil de leer que incluye comentarios de *phpDocumentor* y que permite un mantenimiento muy sencillo.
- ✓ Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros. (2)

1.3 Lenguaje de programación PHP

PHP 5.3.3 (Hypertext Preprocessor, procesador de hipertexto por su traducción al español) es un idioma artificial concebido para crear programas de computador, permitiendo incorporar algoritmos con precisión. Está formado de un conjunto de símbolos y reglas tanto sintácticas como semánticas que especifican su estructura y el significado de sus elementos y expresiones. Cada lenguaje de programación tiene sus propias características, las cuales lo hacen más potente o más débil para determinada finalidad. El estudio de los mismos garantiza contar con los recursos apropiados para codificar la solución a desarrollar.

PHP es un lenguaje interpretado de propósito general ampliamente utilizado, diseñado especialmente para aumentar e incrementar el dinamismo de las páginas web y puede ser incrustado dentro de

código HTML (HyperText Markup Language, Lenguaje de Marcado de Hipertexto por su traducción al español). Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida al usuario final, por lo que PHP convierte una página estática en una dinámica. Puede ser desplegado en la mayoría de los servidores web y plataformas sin costo alguno. La nueva versión de PHP posee una serie de nuevas características como son:

- ✓ Es un lenguaje multiplataforma.
- ✓ Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- ✓ El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- ✓ Posee una amplia documentación entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- ✓ Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- ✓ Permite aplicar técnicas de programación orientada a objetos.
- ✓ No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución. (3)

1.4 Gestor de Base de Datos *PostgreSQL*

PostgreSQL 9.0.3 es un gestor de bases de datos muy específico dedicado a servir una interfaz entre las bases de datos, los usuarios y las aplicaciones que lo utilizan. Su propósito general es el de manejar de una manera más clara, sencilla y ordenada los conjuntos de datos que se convierten en información relevante para una organización. En esta nueva versión incorpora nuevas características y cuenta con un enfoque que se centra en la administración y vigilancia. Proporcionan una notable mejora en el tiempo de ejecución y hace más sencillo el análisis de datos avanzados. Además, *PostgreSQL* es una herramienta de trabajo muy importante en la comunidad de software libre**,** que ha sido muy utilizada alrededor del mundo, sus nuevas características incorporadas son:

- ✓ Instalación ilimitada.

  Es frecuente que las bases de datos comerciales sean instaladas en más servidores de lo que permite la licencia. Algunos proveedores comerciales consideran a esto la principal fuente de incumplimiento de licencia.

  Esto tiene varias ventajas adicionales:

  - ✓ Modelos de negocios más rentables con instalaciones a gran escala.
  - ✓ No existe la posibilidad de ser auditado para verificar cumplimiento de licencia en ningún momento.
  - ✓ Flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.

- ✓ Mejor soporte que los proveedores comerciales.
- ✓ Ahorros considerables en costos de operación.
- ✓ Multiplataforma.
- ✓ Diseñado para ambientes de alto volumen.
- ✓ *PostgreSQL* usa una estrategia de almacenamiento de filas para conseguir una mejor respuesta en ambientes de grandes volúmenes.
- ✓ Herramientas gráficas de diseño y administración de bases de datos. (4)

Debido al sistema que se va a implementar y las características del mismo de poder ser utilizado a nivel empresarial, se escogió *PostgreSQL* como gestor de base de datos, además de brindar soporte por una comunidad de desarrollo internacional, elemento este que no posee *MySQL* pues *Oracle* no provee servicios de soporte a las versiones libres de este gestor.

1.5 Administrador de Base de Datos *pgAdmin* III

PgAdmin una herramienta de administración del gestor de base de datos *PostgreSQL*. Esta herramienta responde a las necesidades de los usuarios de poder crear simples consultas SQL o

consultas de una alta complejidad de una manera más fácil, debido a que se le ha incorporado una nueva interfaz gráfica la cual resulta de gran ayuda en el momento de gestionar las bases de datos. Es una herramienta de código abierto y es respaldada por una amplia comunidad de desarrolladores que encaminan sus esfuerzos al perfeccionamiento de la herramienta.

- ✓ La aplicación también incluye un editor de resaltado de sintaxis SQL.
- ✓ Interfaz administrativa gráfica.
- ✓ Herramienta de consulta SQL.
- ✓ La conexión al servidor se puede hacer a través de TCP / IP.
- ✓ No se requieren controladores adicionales para comunicarse con el servidor de base de datos. (5)

1.6 Metodología de Desarrollo

Una metodología de desarrollo de software es una guía que muestra paso a paso las tareas y actividades que se deben realizar para obtener un producto informático con una buena calidad. Indica cuál es el personal que debe participar en el desarrollo de las distintas actividades así como el papel que deberán enfrentar. Además, muestra toda la información necesaria para culminar o iniciar alguna actividad que se genere como resultado de los diferentes procesos por los cuales transcurre un software.

OpenUp

Open Unified Process (OpenUP) es un proceso de desarrollo unificado que está basado en *Rational Unified Process* (RUP). Mantiene las mismas características de RUP, pues está dirigido por casos de uso, centrado en la arquitectura y además es iterativo e incremental. El ciclo de vida de un proyecto según la metodología OpenUP se divide en 4 fases fundamentales:

La metodología *OpenUp*, divide en 4 fases el desarrollo del software:

- ✓ **Inicio:** se determina la visión del proyecto.
- ✓ **Elaboración:** el objetivo es determinar la arquitectura óptima.

- ✓ **Construcción:** en esta etapa el objetivo es obtener la capacidad operacional inicial.
- ✓ **Transición**: se obtiene la integración del proyecto. (3)

OpenUP propone 6 flujos de Trabajo:

Requerimientos: en este flujo de trabajo se realizan entrevistas con el cliente para comprender el problema a resolver y se definen los requerimientos.

Análisis y Diseño: se realiza el diseño de los requisitos que serán después implementados.

Implementación: en esta disciplina se realiza la implementación del sistema basándose en el diseño realizado.

Prueba: busca los defectos a lo largo del ciclo de vida.

Gestión del Proyecto: involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

Gestión de Configuración y Cambios: describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización y actualización, control de versiones, etcétera.

Características fundamentales de OpenUp

- ✓ Colaboración para unificar intereses y compartir conocimientos.
- ✓ Evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP.
- ✓ Permite detectar errores tempranos a través de un ciclo iterativo.
- ✓ Enfoque en la articulación de la arquitectura.
- ✓ Desarrollo continuo para obtener realimentación y realizar las mejoras respectivas.

Roles presentes en la metodología OpenUp

- ✓ Gerente de Proyecto
- ✓ Analista
- ✓ Diseñador
- ✓ Arquitecto de software
- ✓ Desarrollador
- ✓ Stakeholder (interesados)
- ✓ Tester (Probador)

De los roles propuestos por la metodología de desarrollo OpenUp, se generaron los artefactos correspondientes al rol de Diseñador, debido a que es el encargado de generar el diseño del sistema y entre sus funciones está:

- ✓ Generar el diseño arquitectónico y diseño detallado del sistema, basándose en los requisitos.
- ✓ Generar prototipos rápidos del sistema (con analistas y programadores) para chequear los requisitos.
- ✓ Generar el documento de diseño arquitectónico del software y mantenerlo actualizado durante el proyecto.

En cada disciplina de la ingeniería, el diseño acompaña el enfoque disciplinado que se utiliza para inventar la solución de un problema, entregando así un camino entre los requisitos y la implementación. En ingeniería de software, el propósito del diseño es la construcción de un sistema que cumpla con los siguientes aspectos:

- ✓ Satisfaga una especificación funcional dada.
- ✓ Cumpla con las limitaciones del medio receptor del sistema.
- ✓ Cumpla requisitos implícitos y explícitos de rendimiento y uso de recursos.

- ✓ Satisfaga criterios de diseño implícitos y explícitos en la forma del artefacto construido.
- ✓ Satisfaga restricciones del mismo proceso de diseño, tal como su duración y costo, o las herramientas disponibles para realizar el diseño.

El propósito del diseñador es el de crear una estructura interna limpia y relativamente simple, también llamada a veces una arquitectura. Un prototipo de diseño es el producto final del proceso de diseño. Así, una de las metas en el diseño de software es derivar una arquitectura del sistema. Esta arquitectura sirve como un marco desde el cual se conducen más actividades de diseño detallado.

Las buenas arquitecturas de software tienden a tener algunos atributos comunes. Entre ellos se pueden mencionar los siguientes.

- ✓ Se encuentran construidos en niveles de abstracción bien definidos, provistos a través de una interfaz controlada y definida en niveles de abstracción inferiores.
- ✓ Existe una separación clara de preocupaciones entre la interfaz y la implementación de cada nivel, haciendo posible cambiar la implementación de un nivel sin violar las suposiciones que hicieron los clientes.
- ✓ La arquitectura es simple, comportamiento común se obtiene a través de abstracciones y mecanismos comunes.

Para evaluar la calidad de la representación del diseño, es necesario establecer criterios técnicos para un buen diseño. A continuación se presenta una lista de criterios que pueden utilizarse.

- ✓ Un diseño debe contener una organización jerárquica que haga un uso inteligente del control entre los elementos del software.
- ✓ Un diseño debe conducir a módulos (esto es, subrutinas o procedimientos), que muestren características funcionales independientes.
- ✓ Un diseño debe considerar interfaces que reduzcan la complejidad de conexiones entre módulos y con el ambiente externo.
- ✓ Un diseño debiese ser construido usando un método repetible, guiado por la información obtenida durante la fase de requisitos de software. (6)

También se generaron los artefactos correspondientes para el rol de Desarrollador. Uno de los principales objetivos que posee este rol, es el de convertir la especificación del sistema en código fuente ejecutable, utilizando uno o más lenguajes de programación, así como herramientas de software de apoyo a la programación. El éxito del desarrollo de software depende grandemente de conocimiento. Este conocimiento no sólo corresponde a habilidades de programación, sino a una percepción y entendimiento de los últimos desarrollos de la industria del software. En los mercados actuales, rápidamente cambiantes y altamente competitivos, se hace necesario conocer los últimos desarrollos, quien da soporte, y como pueden beneficiar al proyecto y a la organización. A través de este conocimiento se genera un camino hacia el éxito futuro.

Uno de los principales objetivos de los programadores durante su trabajo debe ser la de reducir la complejidad del software. Algunos de los beneficios que implican la reducción de la complejidad del programa son:

- ✓ Menor cantidad de problemas de testeo.
- ✓ Aumento de la productividad de los programadores.
- ✓ Aumento de la eficiencia en la manutención del programa.
- ✓ Aumento de la eficiencia en la modificación del programa.
- ✓ Reducir el tiempo de codificación, aumentando la productividad del programador.
- ✓ Disminuir el número de errores que ocurren durante el proceso de desarrollo.
- ✓ Disminuir el esfuerzo de corregir errores en secciones del código que se encuentran deficientes, remplazando secciones cuando se descubren técnicas más confiables, funcionales o eficientes.
- ✓ Disminuir los costos del ciclo de vida del software. (6)

1.7 Artefactos generados

En correspondencia de la metodología definida por el grupo de arquitectura del proyecto y los roles involucrados en el desarrollo de la solución del sistema se generan los artefactos siguientes.

Clases del Diseño

Este artefacto es la encapsulación completa de los atributos, métodos y objetos que existan para cada clase, los cuales comparten las mismas responsabilidades, relaciones, atributos y semánticas. (7)

Paquetes del Diseño

Un paquete de diseño se utiliza para estructurar el modelo de diseño dividiéndolo en componentes más pequeños. No ofrece una interfaz formal, aunque puede exponer algunos de sus contenidos (marcados como 'públicos') que ofrecen comportamiento. Los paquetes de diseño deben utilizarse principalmente como herramienta organizativa del modelo, para agrupar relaciones, ejecuciones de guión de uso, diagramas y otros paquetes. Su contenido es responsabilidad de un único Rol: diseñador. (8)

Realización Subsistemas de Diseño

El artefacto subsistema de diseño es una parte del sistema que encapsula comportamiento, expone un conjunto de interfaces y empaqueta otros elementos del modelo. (8)

Realización de Casos de Uso del Diseño

Describe cómo un caso de uso se lleva a cabo en términos de clases de diseño y sus objetos. Hay una correspondencia directa entre la Realización de un Caso de Uso de Diseño y la Realización de un Caso de Uso de Análisis.

- ✓ Diagrama de Clases: que contiene las clases que participan en el caso de uso, aunque algunas de ellas puedan participar en varios.
- ✓ Diagrama de Secuencia: que muestra una secuencia detallada de interacción entre los objetos de diseño. En algunos casos es posible incluir Subsistemas dentro de los diagramas. Los diagramas de secuencia visualizan el intercambio de mensajes entre objetos.
- ✓ Descripción del Flujo de Eventos de Diseño: descripción textual que explica y completa a los diagramas de colaboración. (3)

Modelo de Implementación

El modelo de implementación representa la composición física de la implementación y describe como se organizan los componentes de acuerdo a las estructuras y los mecanismos de modularización que se disponga en el entorno y lenguaje de programación elegido, y cómo dependen los componentes unos de otros. (9)

1.8 Estándar de codificación

En general, los estándares de codificación son reglas que se siguen para la escritura del código fuente. El mismo, se puede definir a nivel de empresa, a nivel de proyecto o incluso a nivel del propio cliente. La aplicación de un estándar de codificación aporta características siempre deseadas durante la implementación del software:

- ✓ Eleva la claridad del código.
- ✓ Sirve como punto de referencia para los programadores.
- ✓ Mantiene un estilo de programación.
- ✓ Ayuda a mejorar el proceso de codificación, haciéndolo más eficiente.

En particular, el Centro de Tecnologías de Gestión de Datos define un estándar de codificación a nivel de centro, del cual se señalan algunos aspectos:

- ✓ Durante el proceso de desarrollo de software cada entregable es sometido a pruebas de liberación, en las que se evalúan las características y sub-características de calidad definidas por la ISO/IEC 9126. Como estándar de codificación se usará CamelCase. Para el diseño de la Base de Datos se usará un convenio para el nombre de las tablas, así como para las vistas y funciones que sean necesarias implementar.
- ✓ Todos los métodos, nombres de clases y variables se escribirán en estructura *CamelCase*, comenzando siempre en minúscula. (10)

1.9 Patrones de diseño

En la actualidad la sociedad requiere sistemas más complejos y mayores. Los recursos desarrollados son cada vez más escasos y por tanto se hacen imprescindibles los mecanismos de reutilización. Ante

la necesidad de asimilar la reutilización en el desarrollo de software se han popularizado mecanismos como: componentes, objetos distribuidos y patrones de diseño.

Un patrón de diseño describe una estructura recurrente de componentes que se comunican para resolver un problema general de diseño en un contexto particular. Nomina, abstrae e identifica los aspectos clave de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. Identifica las clases e instancias participantes, sus roles y colaboraciones y la distribución de responsabilidades.

En esencia, presentan los siguientes elementos:

- ✓ **Nombre**: permite describir, en una o dos palabras, un problema de diseño junto con sus soluciones y consecuencias.
- ✓ **Problema**: describe cuándo aplicar el patrón. Explica el problema y su contexto. A veces incluye condiciones que deben darse para que tenga sentido la aplicación del patrón.
- ✓ **Solución**: describe los elementos que constituyen el diseño, sus relaciones, responsabilidades y colaboraciones. La solución no describe un diseño o implementación en concreto, sino que es más bien una plantilla que puede aplicarse en muchas situaciones diferentes.
- ✓ **Consecuencias**: son los resultados, así como ventajas e inconvenientes de aplicar el patrón.

Patrones de diseño GRASP (General Responsability Assignment Software Patterns, Patrones de Software para la asignación General de Responsabilidad, por su traducción al español):

- ✓ **Bajo Acoplamiento:** es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las clases. Estas clases no se afectan por los cambios en otros componentes, fáciles de entender por separado y de fácil reutilización.
- ✓ **Alta Cohesión:** plantea que la información que almacena una clase debe ser coherente y estar en la mayor medida relacionada con la clase. Estas clases mejoran la claridad y la facilidad con que se entiende el diseño. Simplifican el mantenimiento y las mejoras en funcionalidad. A menudo genera un bajo acoplamiento, además la ventaja de una gran funcionalidad soporta

una mayor capacidad de reutilización ya que una clase muy cohesiva puede destinarse a un propósito en específico.

- ✓ **Experto:** es el principio básico de asignación de responsabilidades. Indica que la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Mantiene el encapsulamiento, los objetos usan su propia información para llevar a cabo sus tareas, soportando un bajo acoplamiento que favorece el hecho de tener sistemas más robustos y de fácil mantenimiento. Distribuye el comportamiento entre las clases que contienen la información requerida, brindando una alta cohesión.

- ✓ **Creador:** identifica quien debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Se brinda un soporte al bajo acoplamiento.

- ✓ **Controlador:** asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Le ofrece mayor potencial de los componentes reutilizables, garantizando que la empresa o los procesos de dominio sean manejados por la capa de los objetos de dominio y no por la de la interfaz.

- ✓ **Fabricación Pura:** cuando los problemas se complican, se debe construir clases que se encarguen de construir los objetos adecuados en cada momento (factorías).

- ✓ **Polimorfismo:** siempre que se tenga que llevar a cabo una responsabilidad que dependa del tipo, se tiene que hacer uso del polimorfismo, cuando las alternativas o comportamientos relacionados varían según el tipo (clase), asignar la responsabilidad para el comportamiento- utilizando operaciones polimórficas a los tipos para los que varia el comportamiento.

- ✓ **Indirección:** el patrón de indirección nos aporta mejorar el bajo acoplamiento entre dos clases asignando la responsabilidad de la mediación entre ellos a un tercer elemento (clase) intermedio. (11)

Patrones de diseño GOF (Gang of Four "Banda de los Cuatro" por su traducción al español):

- ✓ **Creacionales**: resuelven problemas relativos a la creación de objetos.

Builder (Constructor virtual): abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto.

Factory Method (Método de fabricación): centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crear.

Prototype (Prototipo): crea nuevos objetos clonándolos de una instancia ya existente.

Singleton (Instancia única): garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

- ✓ **Estructurales**: resuelven problemas relativos a la composición de objetos.

Front Controller (Controlador frontal): es un patrón muy utilizado por las aplicaciones web. Describe básicamente cómo construir un sólo punto de acceso para todas las peticiones de una aplicación web. Escucha las peticiones que vienen desde una URL (Uniform Resource Locator, Localizador Uniforme de Recurso, por su traducción al español), y se encarga de llamar al controlador específico, posteriormente llama a la acción deseada del controlador.

Adapter (Adaptador): adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla.

Bridge (Puente): desacopla una abstracción de su implementación.

Composite (Objeto compuesto): permite tratar objetos compuestos como si de un simple se tratase.

Fachada (Fachada): provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.

Flyweight (Peso ligero): reduce la redundancia cuando gran cantidad de objetos poseen idéntica información.

- ✓ **De Comportamiento**: resuelven problemas relativos a la interacción entre objetos.

Observer (Observador): define una dependencia entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.

Command (Orden): específica una forma simple de separar la ejecución de un comando del entorno que generó dicho comando. Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.

Interpreter (Intérprete): dado un lenguaje, define una gramática para dicho lenguaje, así como las herramientas necesarias para interpretarlo.

Iterator (Iterador): permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos.

Strategy (Estrategia): permite disponer de varios métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución. (12)

Los patrones de diseño presentan las siguientes ventajas:

- ✓ Facilitan la localización de los objetos que formarán el sistema.
- ✓ Facilitan la determinación de la granularidad adecuada.
- ✓ Especifican interfaces para las clases.
- ✓ Especifican implementaciones (al menos parciales).
- ✓ Facilitan el aprendizaje y la comunicación entre programadores y diseñadores.
- ✓ Mejoran la calidad del diseño y la implementación.
- ✓ Se pueden considerar como “normas de productividad”. (13)

1.10 Patrones de Arquitectura

Los patrones de arquitectura establecen un esquema fundamental para arquitecturas de software concretas. Definen las propiedades de la estructura global del sistema y tienen un impacto en la arquitectura de cada subsistema, además expresan un diseño de organización estructural para los sistemas de software. Proporcionan un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y lineamientos para organizar la relación entre ellos. La selección de un patrón de arquitectura o la combinación de varios es solo el primer paso cuando se diseña la arquitectura de un sistema software. (3)

Modelo – Vista – Controlador

El patrón Modelo-Vista-Controlador (MVC), es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Su principal finalidad es mejorar la reusabilidad y que las modificaciones en las vistas impacten en menor medida en la lógica de negocio o de datos.

El **Modelo** es el responsable de:

- ✓ Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- ✓ Define las reglas de negocio (la funcionalidad del sistema).
- ✓ Lleva un registro de las vistas y controladores del sistema.
- ✓ Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo.

El **Controlador** es responsable de:

- ✓ Recibir los eventos de entrada.
- ✓ Contiene reglas de gestión de eventos, estas acciones pueden suponer peticiones al modelo o a las vistas.

Las **Vistas** son responsables de:

- ✓ Recibir datos del modelo y la muestra al usuario.
- ✓ Tienen un registro de su controlador asociado.
- ✓ Pueden dar el servicio de actualización para que sea invocado por el controlador o por el modelo cuando es un modelo activo. (14)

Monolítica

Es una arquitectura donde el software se estructura en grupos funcionales muy acoplados.

Pipeline

Consiste en ir transformando un flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior.

Esta arquitectura es muy común en el desarrollo de programas para el intérprete de comandos, ya que se pueden concatenar comandos fácilmente con tuberías.

También es una arquitectura muy natural en el paradigma de programación funcional, ya que equivale a la composición de funciones matemáticas.

Orientada a Servicios de cliente (en inglés **S**ervice **O**riented **A**rchitecture)

Es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio.

Permite la creación de sistemas altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma bien definida de exposición e invocación de servicios (comúnmente pero no exclusivamente servicios web), lo cual facilita la interacción entre diferentes sistemas propios o de terceros.

Dirigida por eventos, (*Event-driven architecturge* o EDA)

Es un patrón de arquitectura de software que promueve la producción, detección, consumo de, y reacción a eventos. (14)

Cliente-Servidor

El patrón de arquitectura Cliente-Servidor se caracteriza por la existencia de un nodo (o más) servidor donde reside el servicio que se expone y varios clientes que consumen dicho servicio, en el sistema este estilo responde al hecho de que se trate de una aplicación web y se concreta con un servidor de bases de datos, un servidor web y varios clientes que acceden al sistema a través de un navegador.

Se utiliza el patrón de arquitectura Cliente-Servidor en el Administrador de Reportes debido a que el administrador es quien inicia solicitudes o peticiones, teniendo por tanto un papel activo en la comunicación. Se esperan y se reciben las respuestas del servidor, por lo general se pueden conectar

a varios servidores a la vez e interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

El **Servidor** posee las siguientes características:

- ✓ Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación.
- ✓ Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
- ✓ Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
- ✓ No es frecuente que interactúen directamente con los usuarios finales. (15)

1.11 Estilos Arquitectónicos

Los estilos arquitectónicos, caracterizan una familia de sistemas que están relacionados por compartir propiedades estructurales y funcionales. Generalmente proveen guía y análisis para crear una clase amplia de arquitecturas en un dominio específico donde los patrones se enfocan en solucionar los problemas más pequeños, más específicos dentro de un estilo dado. Los estilos, en cambio, expresan la arquitectura en el sentido más formal y teórico, constituyendo un tópico esencial, ellos se encuentran en el centro de la arquitectura y constituyen buena parte de su solución.

Tipos de Estilos arquitectónicos

Estilos de Flujo de datos

- ✓ Tuberías y Filtros

Estilos centrados en datos

- ✓ Arquitecturas de Pizarra o repositorio

Estilos de Llamada y Retorno

- ✓ Modelo – Vista – Controlador (MVC)
- ✓ Arquitectura en Capas

- ✓ Arquitectura Orientada a Objetos
- ✓ Arquitectura basada en Componentes

Estilo de Código Móvil

- ✓ Arquitectura de Máquinas Virtuales

Estilos *Peer–To –Peer*

- ✓ Arquitectura basada en Eventos
- ✓ Arquitectura Orientada a Servicios (SOA)

Los estilos se usan combinados; cada capa o componente puede ser internamente de un estilo diferente al de la totalidad; muchos estilos se encuentran ligados a dominios específicos, o a líneas de producto particulares. (3)

1.12 Herramientas de Modelado Visual Paradigm for UML

Visual Paradigm for UML es una herramienta UML 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 UML 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, generar código desde diagramas y generar documentación. Soporta notación UML 2.x, capacidades de ingeniería.

Principales características

- ✓ Soporte de UML versión 2.1.
- ✓ Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- ✓ Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- ✓ Licencia: gratuita y comercial.

- ✓ Producto de calidad.
- ✓ Varios idiomas.
- ✓ Fácil de instalar y actualizar. (15)

1.13 Formato alternativo de envío y recepción de datos

Notación de Objetos de *JavaScript* (*Java Script Object Notation* JSON por sus siglas en inglés) es un formato ligero de intercambio de datos. Está basado en un subconjunto del lenguaje de programación *JavaScript*. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, *Java*, *JavaScript*, *Perl*, *Python*. Es muy sencillo de usar, especialmente como alternativa a XML (Extensible Markup Language, lenguaje de etiquetado extensible, por su traducción al español). Por lo que una ventaja referente a XML es que los datos en JSON ocupan mucho menos que en XML debido a que este último añade bastante sobrecarga a los datos que queremos serializar. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. (16)

1.14 Lenguaje de Etiquetado Extensible

Lenguaje de Etiquetado Extensible, conformado por un conjunto de reglas para definir etiquetas semánticas orientadas a organizar un documento en diferentes partes. XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Lo cual admite que el usuario especifique su propio lenguaje adecuado a las necesidades.

- ✓ Es extensible: después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.
- ✓ El analizador: es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan *errores* y se acelera el desarrollo de aplicaciones.
- ✓ Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Se pueden comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos.

- ✓ Transforma datos en información, pues se le añade un significado concreto y los relaciona a un contexto, con lo cual tiene flexibilidad para estructurar documentos.
- ✓ Las aplicaciones se pueden generar rápidamente y su mantenimiento es sencillo. (17)

Conclusiones del capítulo

Después de realizar un estudio sobre las metodologías y tecnologías actuales en el mundo necesarias para el desarrollo de la herramienta y teniendo en cuenta principalmente las exigencias y necesidades del cliente se decide optar por:

Como metodología de desarrollo se seleccionó *OpenUp*, pues el tiempo de desarrollo del software es relativamente corto y el equipo de trabajo es pequeño. Además, esta metodología mantiene las principales características de ser de código abierto y ser dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Evita la elaboración de documentación innecesaria que requiere la metodología RUP, y permite detectar errores tempranos a través de un ciclo iterativo. Es un proceso que valora los beneficios de la colaboración y de los involucrados con el resultado del proyecto, por encima de formalidades innecesarias.

Como herramientas de desarrollo se ha elegido el IDE NetBeans 7.0, ya que proporcionar un marco de trabajo fácil y amigable, es multiplataforma e incorpora una serie de características como las mejoras añadidas para la instalación del controlador, así como la introducción de JDK 7 de apoyo, incluidas las mejoras del editor (sintaxis, pistas). Además brinda soporte para el *framework Symfony,* el cual es utilizado en el módulo Administrador de Reportes debido a que brinda la posibilidad de facilitar y automatizar una gran cantidad de funcionalidades que resultan comunes en las aplicaciones web. Provee la implementación del patrón de arquitectura Modelo-Vista-Controlador, además de estar enfocado para el desarrollo del lenguaje dinámico de programación PHP, es fácil de instalar y configurar en la mayoría de las plataformas y es independiente del sistema gestor de bases de datos.

CAPÍTULO 2. DISEÑO DE LA SOLUCIÓN

Introducción

En este capítulo se describe el diseño y se hace una especificación de los requerimientos. Se aborda sobre los patrones arquitectónicos y de diseño que están presentes en la herramienta para dar solución al problema. Se realizan los diagramas de clases del diseño, a través de los cuales se muestra la estructura estática del sistema y además se describen las clases más relevantes. Para modelar los aspectos dinámicos del sistema se realizan los diagramas de interacción donde se representan las clases y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellas. Se especifica el diagrama de caso de uso del sistema, la matriz de trazabilidad y la vista lógica.

Objetivos del Diseño

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción, lo cual contribuye al desarrollo de una arquitectura estable y sólida. Este flujo de trabajo es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales.

Sus principales objetivos son:

- ✓ Transformar los requisitos en un diseño del sistema.
- ✓ Adaptar el diseño para que se ajuste al entorno de implementación, con un diseño pensado para el rendimiento. (9)

2.1 Requisitos Funcionales

Los requisitos funcionales son condiciones que el sistema debe efectuar. El módulo Administrador de Reportes del Generador Dinámico de Reportes (V 2.0) debe satisfacer los requisitos funcionales que a continuación se describen.

- ✓ **Copiar reporte:** permite copiar un reporte de una categoría hacia otra categoría permaneciendo el reporte en ambas.
- ✓ **Mover reporte:** permite mover un reporte de una categoría hacia la categoría de reporte deseada permaneciendo el reporte solamente en esta última.
- ✓ **Eliminar reporte:** elimina un reporte determinado de todas las categorías existentes.

- ✓ **Renombrar reporte:** permite cambiar el nombre de un reporte permaneciendo en las categorías donde se encontraba de manera inalterada.
- ✓ **Crear categoría de reporte:** permite crear una nueva categoría de reportes**.**
- ✓ **Editar categoría de reporte:** permite modificar los datos existentes de la categoría.
- ✓ **Eliminar categoría de reporte:** elimina una categoría, así como todas sus subcategorías y reportes contenidos.
- ✓ **Mostrar categorías:** muestra todas las categorías existentes en el sistema.
- ✓ **Crear Subscripción:** permite al usuario crear una subscripción y seleccionar el servicio deseado ya sea por correo electrónico o servicio ftp, así como la frecuencia con la que desea recibir el reporte.
- ✓ **Eliminar Subscripción:** permite eliminar una subscripción.
- ✓ **Modificar acceso a reportes:** concede o elimina el acceso a un reporte a un grupo de usuarios.
- ✓ **Eliminar una plantilla:** permite eliminar una plantilla. (18)

2.2 Requisitos no Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. El Administrador de Reportes es un módulo del producto Generador Dinámico de Reportes (V 2.0) que cuenta con los siguientes requerimientos no funcionales:

Usabilidad

La herramienta propuesta podrá ser usada por cualquier persona autorizada y que posea conocimientos básicos en el manejo de una computadora. Debe ser una herramienta sencilla, manejable y fácil de usar para el usuario.

Fiabilidad

El sistema deberá estar disponible las 24 horas del día. En caso de fallo, pudiera estar fuera de servicio por un período de 72 horas máximo. La precisión y exactitud de las salidas del sistema se corresponden con la calidad y exactitud de la información contenida en las base de datos desde donde se extraigan los reportes. El sistema no es responsable por la falta de veracidad de dicha información. Algunos errores que pueden resultar críticos son:

- ✓ Que salgan de funcionamiento las bases de datos desde donde se extraen los reportes o que no exista conectividad hacia ellas.
- ✓ Que falle el servidor donde se despliegue la solución.

Eficiencia

La eficiencia del sistema depende en gran medida de la velocidad de conexión a la base de datos donde se encuentre, así como del volumen de información contenido en las mismas. Sin embargo el sistema debe mantener tiempos de respuestas en un marco razonable, con tal fin se implementará un mecanismo de paginación de los reportes que permitirá que los mismos sean obtenidos de manera progresiva disminuyendo así el tráfico por la red de la información y las consultas que devuelven grandes cantidades de tuplas. (18)

Restricciones de diseño e implementación

El sistema deberá ser implementado en el lenguaje de programación PHP versión 5.2 o superior. Como marco de trabajo se utilizará *Symfony* en su versión 1.4.9, el cual propone una arquitectura modular en tres capas: modelo, vista y controlador. Unas de las bibliotecas fundamentales en el desarrollo de la herramienta es la de EXT JS versión 3.3.0 la cual es una librería en *Javascript* que permite el diseño de interfaces visuales interactivas usando tecnologías como AJAX y permite crear aplicaciones WEB con la apariencia de aplicaciones de escritorio. Como IDE de desarrollo se utilizará *NetBeans* 7.0 y como gestor de base de datos se utilizará *PostgreSQL* 9.0.3.

Otra librería importante y necesaria en el desarrollo de la herramienta es *JasperReports*, la cual constituye el núcleo del proceso de generación de reportes.

Requerimientos de Software

El servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos

- ✓ Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 10.04 o superior.
- ✓ Paquetes: apache2, php5, libapache2-modphp5, php5-cli, php5-mysql, php5-pgsql, php5-sqlite, php5-xsl, php5-gd.

El servidor donde se instalará la base de datos debe cumplir con los siguientes requisitos

- ✓ Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 10.04 o superior.
- ✓ PostgreSQL versión 9.0.3.
- ✓ PGAdminIII o algún administrador para PostgreSQL.
- ✓ Usuario con privilegios para instalar la base de datos.
- ✓ PostgreSQL debe estar correctamente configurado para aceptar conexiones vía TCP/IP. (18)

Requerimientos de Hardware

El servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos

- ✓ Procesador Intel Pentium 4 1.7 GHz o AMD similar.
- ✓ 1GB de memoria RAM.
- ✓ 40 GB de espacio en disco duro.

El servidor donde se instalará la base de datos debe cumplir con los siguientes requisitos

- ✓ Procesador Intel Pentium 4 1.7 GHz o AMD similar.
- ✓ 1GB de memoria RAM.
- ✓ 40 GB de espacio en disco duro.

La máquina utilizada para acceder a la aplicación debe cumplir con los siguientes requisitos

- ✓ Procesador 1.7 GHz, o superior.

- ✓ 512 MB RAM.
- ✓ 20 GB de espacio en disco duro. (18)

2.3 Diagrama de Casos de Uso del Sistema

El diagrama de casos de uso documenta el comportamiento de un sistema desde el punto de vista del usuario. Representan las funciones que un sistema puede ejecutar.

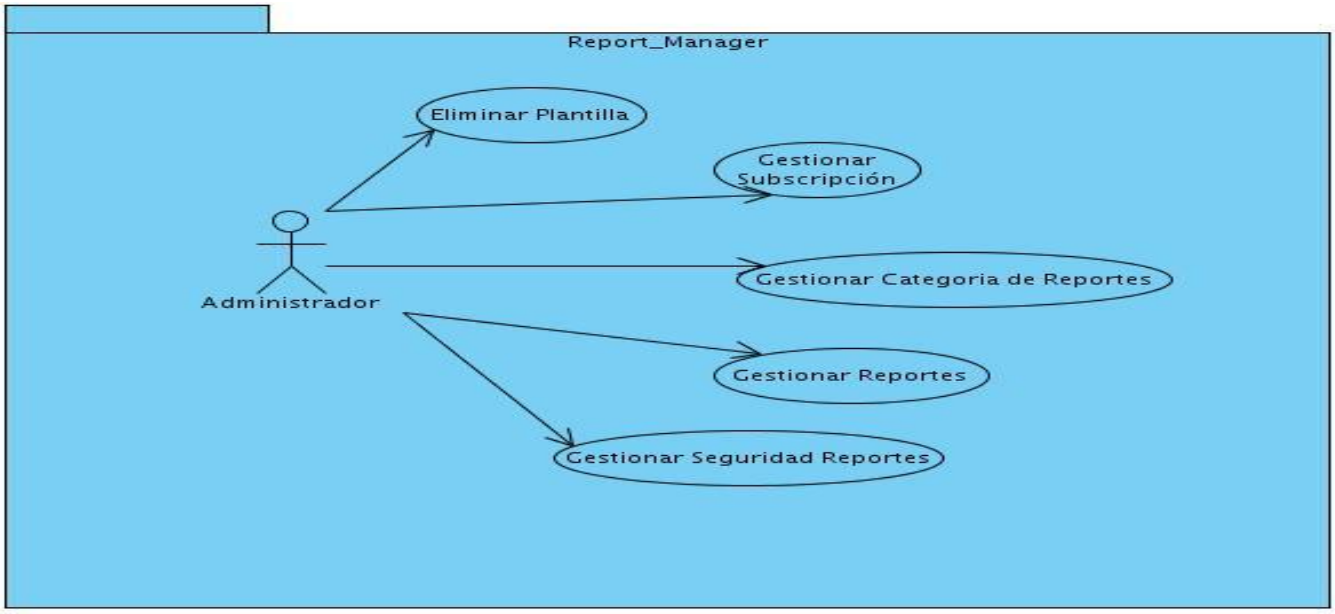

Figura 1: Diagrama Caso de Uso del Sistema

2.4 Descripción de los Casos de Uso

Caso de uso Gestionar Categoría de Reportes: posibilita la adición, eliminación y modificación de categorías que es una forma para organizar los reportes en el sistema.

Caso de uso Gestionar Reportes: permite copiar o mover un reporte previamente creado hacia la categoría deseada. Se puede también eliminar un reporte existente.

Caso de uso Gestionar Seguridad de Reportes: permite establecer permisos a los roles y a los usuarios del sistema creados a determinado reporte.

Caso de uso Eliminar Plantilla: es el encargado de realizar la operación referente a la eliminación de una plantilla.

Caso de uso Gestionar Subscripción: el objetivo es la creación o eliminación de una subscripción a un reporte creado anteriormente. Se puede también modificar una subscripción existente.

- ✓ **Programar_Subscripciones:** permite especificar atributos a las subscripciones como un identificador, fecha de inicio y culminación de la tarea programada.
- ✓ **Programar_Recurrencia:** posibilita configurar la programación de la subscripción bajo determinadas frecuencias, ya sea diariamente, semanalmente, a una hora específica o solo una vez.
- ✓ **Configurar_Archivo:** el caso de uso permite configurar el envío de una subscripción a un servidor de ficheros. Se especifica la dirección de donde se encuentra el servidor, el formato del reporte, las credenciales para acceder y opciones de escritura.
- ✓ **Configurar_Correo:** posibilita configurar el envío de la subscripción por correo electrónico, estableciendo el destinatario, el asunto del correo, las copias y el cuerpo del mismo.

Una vez realizado el análisis de los requerimientos funcionales que el Administrador de Reportes del Generador Dinámico de Reportes V (2.0) debe cumplir. Se determinaron 12 requisitos funcionales los cuales fueron agrupados en 5 casos de uso.

Matriz de Trazabilidad

Tabla 1: Matriz de Trazabilidad

	Gestionar Seguridad	Gestionar Reportes	Gestionar Categorías	Gestionar Subscripción	Eliminar Plantilla
Copiar reporte		X			
Mover reporte		X			
Eliminar reporte		X			
Renombrar reporte		X			
Crear categoría de reporte			X		
Editar categoría de reporte			X		
Eliminar categoría de reporte			X		
Mostrar categorías			X		
Crear Subscripción				X	
Eliminar Subscripción				X	
Modificar acceso a reportes	X				
Eliminar una plantilla					X

2.5 Vista Lógica

Comprende las clases, interfaces y colaboraciones que forman el vocabulario del problema y su solución por lo que cuando se crea una aplicación es importante iniciar con una arquitectura lógica que clarifique los roles de todos los componentes, separe funcionalidades de forma tal que los miembros del equipo trabajen juntos efectivamente. La arquitectura lógica debe tener la estructura necesaria para que sea lo suficientemente flexible en la arquitectura física que se utilizará. (9)

Patrones de diseño implementados en el Generador Dinámico de Reportes

Patrones GRASP:

Bajo Acoplamiento**:** es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las clases. Estas clases no se afectan por los cambios en otros componentes, fáciles de entender por separado y de fácil reutilización.

Por ejemplo las clases del módulo se pueden modificar sin que se afecten directamente las demás clases del modelo ni las clases del controlador, por otra parte las clases del controlador no afectan el modelo al ser modificadas ya que estas son acciones independientes, estas acciones pueden ser reutilizadas desde otros módulos y aplicaciones del proyecto Symfony. El hecho de modificarlas no implica que dejen de funcionar esos módulos siempre que se respeten los métodos que deben recibir y devolver información en el formato Json definido.

Alta Cohesión: en el Administrador de Reportes se asignan responsabilidades, por ejemplo la clase addCategory tiene la responsabilidad de definir las acciones para adicionar categorías y colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a las categorías, es decir, se encuentran estrechamente relacionadas proporcionando que el software sea flexible frente a grandes cambios.

Experto: en el Administrador de Reportes se sigue este patrón al incluir *Propel* como mapeo objeto-relacional *Object Relational Mapping* (ORM por sus siglas en inglés), se generan las clases para la gestión de las entidades con las responsabilidades debidamente asignadas. Esta es precisamente la

solución que propone el patrón Experto, pues cada una de estas clases cuenta con un conjunto de funcionalidades relacionadas directamente con la entidad que representan.

Creador**:** identifica quien debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Se brinda un soporte al bajo acoplamiento.

Controlador**:** asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Le ofrece mayor potencial de los componentes reutilizables, garantizando que la empresa o los procesos de dominio sean manejados por la capa de los objetos de dominio y no por la de la interfaz. Esto es lo que hacen las acciones de Symfony, en el controlador, recibir las peticiones y decidir el flujo a seguir para darles respuesta.

Patrones GOF:

Front Controller (Controlador frontal): es un patrón muy utilizado por las aplicaciones web. Describe básicamente como construir un sólo punto de acceso para todas las peticiones. En el Administrador de Reportes se implementó este patrón debido a que escucha peticiones que vienen desde la URL y se encarga de llamar al controlador específico, esto es sumamente cómodo ya que en el sistema cada posible acción a realizar por el administrador es una clase que se encuentra dentro del paquete controlador.

Observer (Observador): define una dependencia entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.

Command (Orden): específica una forma simple de separar la ejecución de un comando del entorno que generó dicho comando. Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma. En el framework Symfony las acciones son un ejemplo de este patrón.

Los patrones de arquitectura seguidos durante la realización del módulo Administrador de Reportes fueron:

El patrón Cliente-Servidor: se caracteriza por la existencia de un nodo o más, entre los cuales se puede encontrar el servidor donde reside el servicio que se expone y varios clientes que consumen

dicho servicio, característica que posee el Generador de Reportes, por lo cual resulta de gran interés ya que de esta manera responde a que es una aplicación web. De esta manera los clientes acceden a la aplicación mediante un navegador, facilitando a los usuarios finales interactuar con el sistema mediante una interfaz gráfica.

EL patrón Modelo-Vista-Controlador: se caracteriza por separar en capas una aplicación y de esta manera convierte una aplicación en un paquete fácil de mantener, modular y de desarrollo rápido. La modularidad y el diseño independiente permiten a los desarrolladores y diseñadores hacer cambios en alguna parte de la aplicación sin afectar a los demás.

El estilo de arquitectura mantenido en el módulo administrador de reportes es:

Estilo de llamada y retorno

Este estilo encapsula los patrones de arquitectura Modelo Vista-Controlador y Cliente-Servidor por lo que permite al diseñador del software construir una estructura de programa relativamente fácil de modificar y ajustar a escala. Existen 2 subestilos:

• Arquitectura de programa principal: clasifica de programación descompone las funciones en una jerarquía de control donde un programa principal llama a un número de componentes del programa, los cuales pueden también llamar a otros componentes.

• Arquitectura de llamada de procedimiento remoto: los componentes de una arquitectura de programa principal/subprograma, están distribuidos entre varias computadoras en una red.

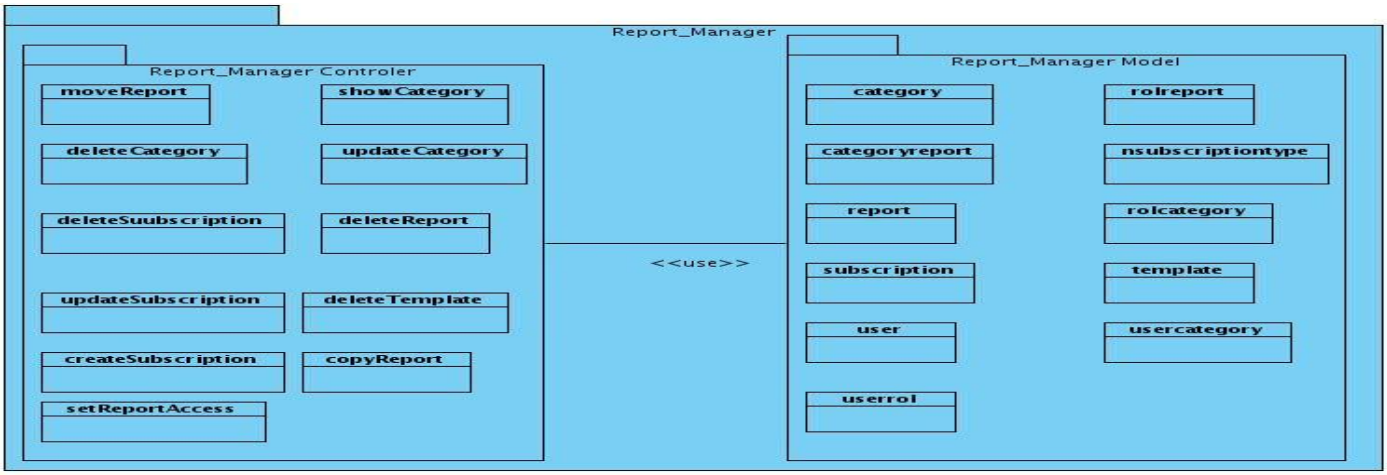

Figura 2: Vista Lógica

2.6 Diagramas de clases

El diagrama de clases del diseño muestra los atributos y métodos de cada clase, representando de forma sencilla la colaboración y las responsabilidades de cada una de ellas entorno al sistema que conforman.

Diagrama de Clases del Diseño del Caso de Uso Gestionar Categoría de Reportes

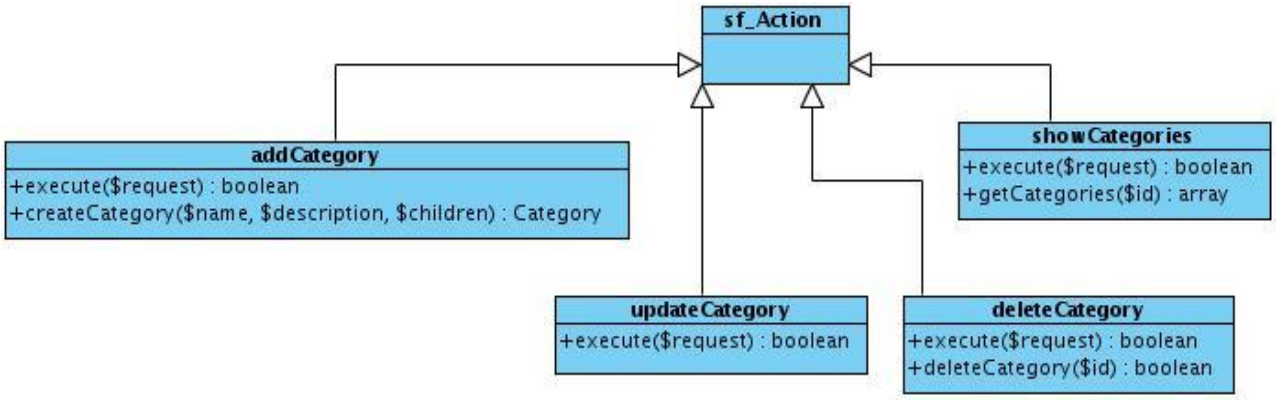

Figura 3: Diagrama del Caso de Uso Gestionar Categoría

Estas clases son las encargadas de gestionar las categorías y se agrupan, en el caso de uso Gestionar Categorías de Reporte el cual se inicia al actor solicitar gestionar las categorías de los reportes. El caso de uso permite crear nueva categoría, editarla, eliminarla y mostrar todas las categorías. Finaliza al realizarse algunas de las operaciones referentes a las categorías.

Diagrama de Clases del Diseño del Caso de Uso Eliminar Plantilla

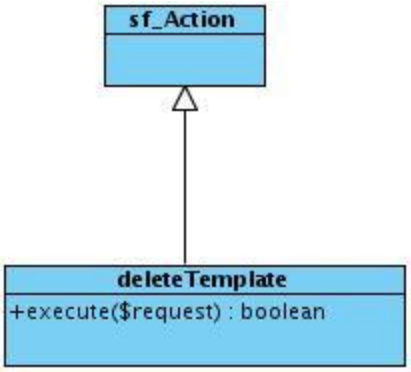

Figura 4: Diagrama del Caso de Uso Eliminar Plantilla

Esta clase es la encargada de eliminar una determinada plantilla, el cual se inicia al actor solicitar a través del caso de uso base eliminar plantilla. El caso de uso únicamente permite eliminar una plantilla Finaliza al eliminar una plantilla existente en el sistema.

Diagrama de Clases del Diseño del Caso de Uso Gestionar Reportes

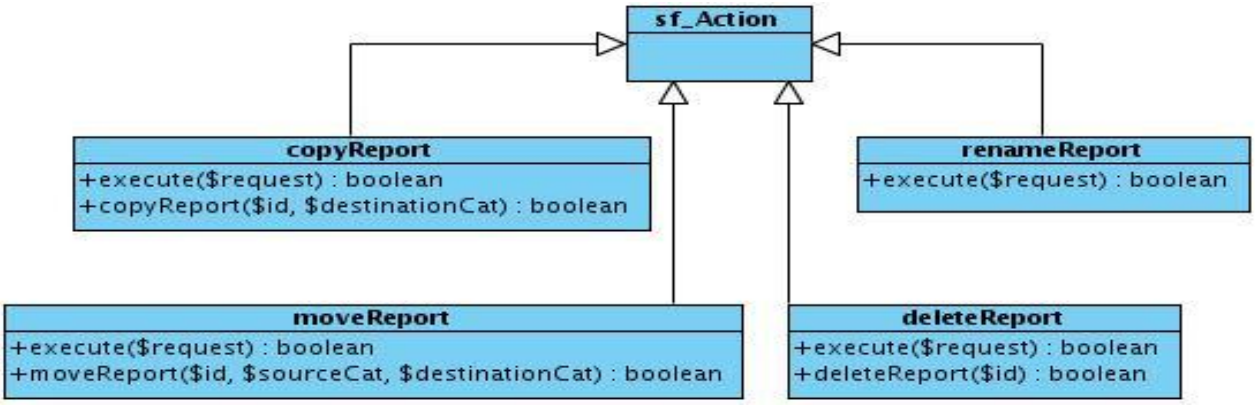

Figura 5: Diagrama del Caso de Uso Gestionar Reportes

Estas clases son las encargadas de gestionar los reportes y están agrupadas dentro del caso de uso Gestionar Reportes. El caso de uso comienza al seleccionar la opción "*Administrador de reportes*", el cual permite la posibilidad de eliminar un reporte, así como mover un reporte de categoría, copiarlo en otra categoría o renombrar un determinado reporte. Culmina al realizar cualquiera de estas operaciones referentes a los reportes.

Diagrama de Clases del Diseño del Caso de Uso Gestionar Subscripción

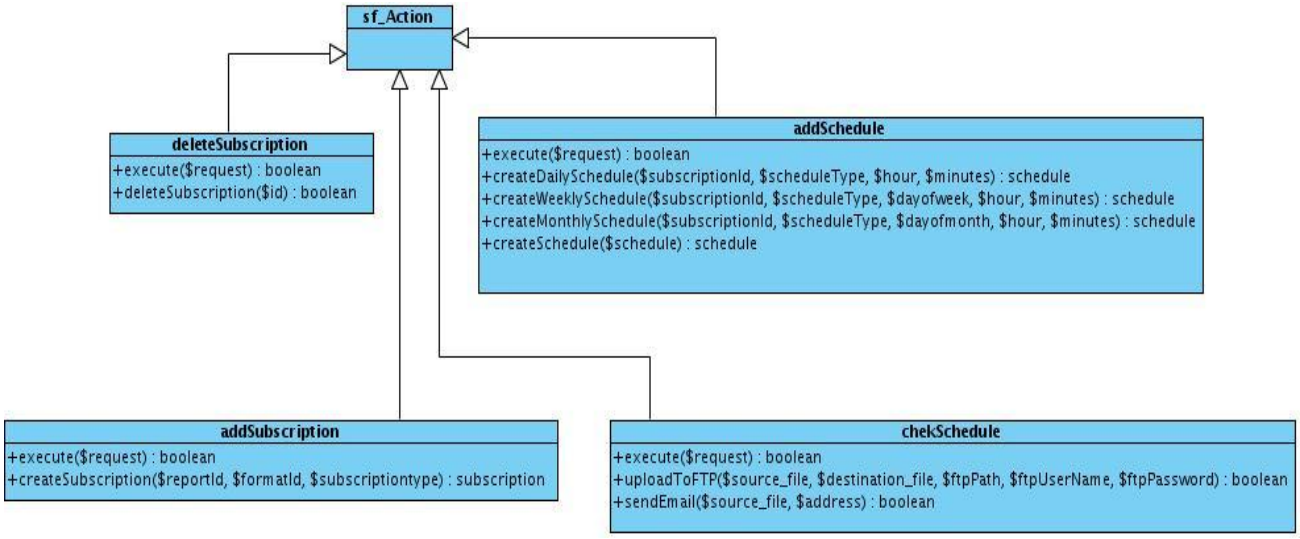

Figura 6: Diagrama del Caso de Uso Gestionar Subscripción

Estas clases son las encargadas de gestionar las subscripciones a los reportes de manera dinámica Se encuentran agrupadas dentro del caso de uso “Gestionar Subscripciones”. El caso de uso comienza al seleccionar la opción “Gestionar Subscripción”, permitiendo adicionar o eliminar una subscripción. Además permite configurar el tipo de servicio mediante el cual se desean recibir los reportes ya sea por ftp o correo electrónico, así como el formato en el cual se desean los reportes. Culmina al terminar de realizar la operación de adicionar o eliminar una subscripción.

Diagrama de Clases del Diseño del Caso de Uso Gestionar Seguridad Reportes

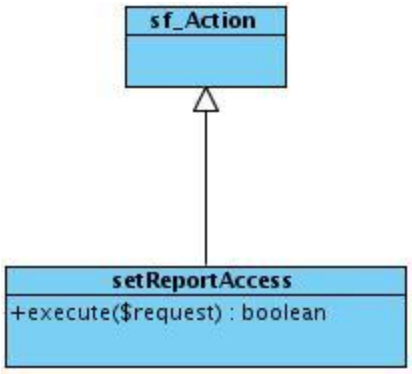

Figura 7: Diagrama del Caso de Uso Gestionar Seguridad de Reporte

Esta clase es la encargada de modificar los permisos de un usuario sobre un determinado reporte, se inicia al actor solicitar a través del caso de uso base “Gestionar Seguridad de Reporte”. El caso de uso únicamente permite modificar permisos sobre los reportes. Finaliza al modificar los permisos de un usuario sobre los reportes existentes en el sistema.

2.7 Descripción de las Principales Clases

Report_Manager: es el módulo encargado de responder las peticiones realizadas al Administrador de Reportes del Generador Dinámico de Reportes (V 2.0).

Tabla 2: Módulo Report_Manager

Nombre: Report_manager	
Tipo de clase: controladora	
Nombre	**Descripción:**
addCategoryAction()	Adiciona una categoría
deleteCategoryAction()	Elimina una categoría
showCategoriesAction()	Muestra todas las categorías

updateCategoryAction()	Actualiza una categoría ya existente
deletePlantillaAction()	Elimina una plantilla
moveReportAction()	Mueve un reporte
renameReportAction()	Renombra el reporte deseado
deleteReportAction()	Elimina un reporte
copyReportAction()	Copia un reporte hacia una categoría
createSubscriptionAction()	Crea una nueva Subscripción
deleteSubscriptionAction()	Elimina una Subscripción
setAccessReportAction()	Otorga o retira permisos
addScheduleAction()	Adiciona una nueva recurrencia
checkScheduleAction()	Chequea cual recurrencia debe ser ejecutada

2.8 Diagramas de interacción del diseño

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema lo que conlleva modelar instancias concretas o prototípicas de clases interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. En el presente trabajo se usan diagramas de secuencia.

El diagrama de secuencia muestra la interacción entre usuarios, sistemas y subsistemas, y hace énfasis en el orden temporal de los mensajes.

Diagramas de Secuencia del Caso de Uso Gestionar Categoría de Reporte

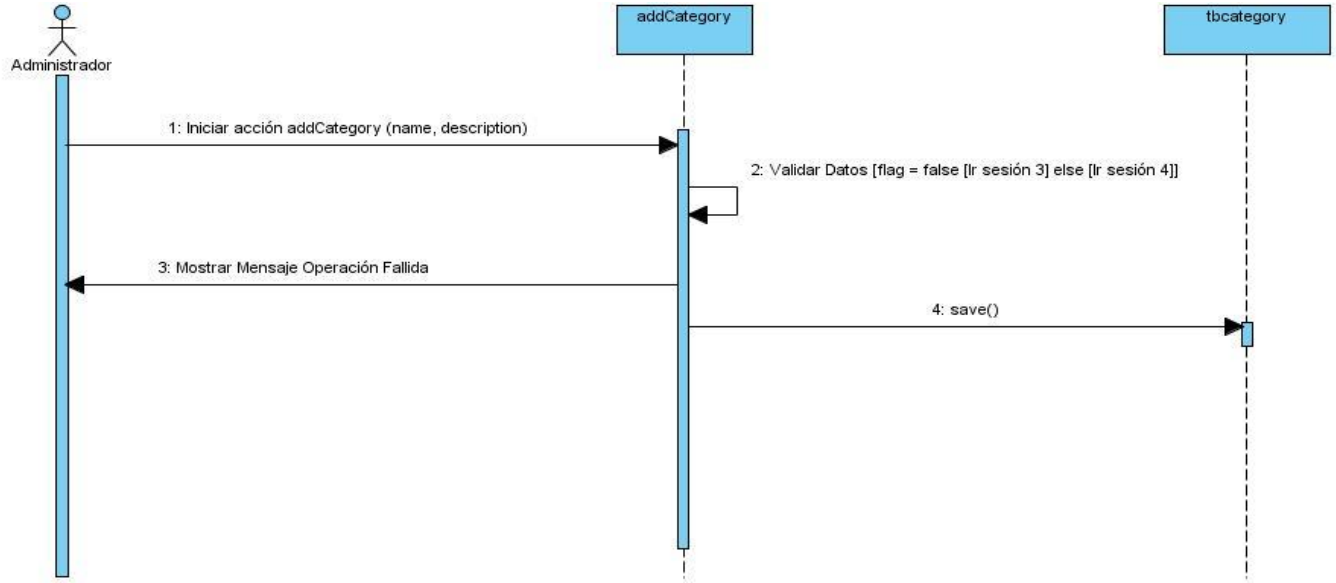

Figura 8: Diagrama de Secuencia Escenario Adicionar Categoría

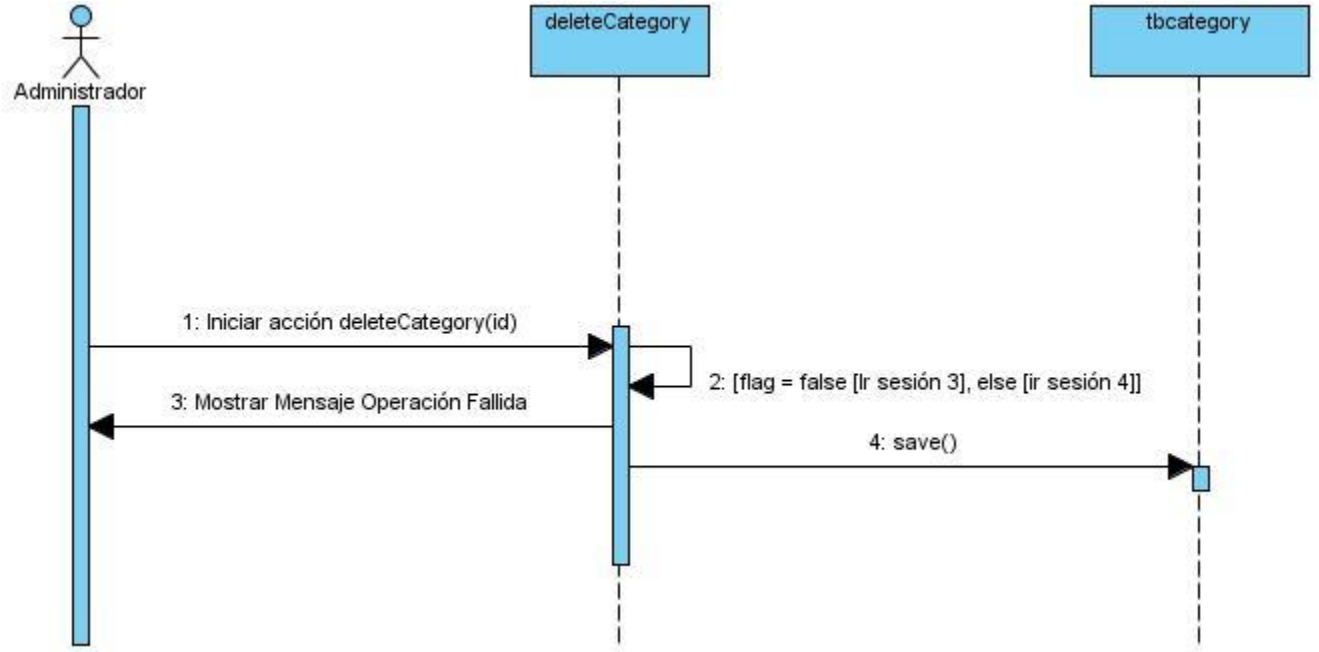

Figura 9: Diagrama de Secuencia Escenario Eliminar Categoría

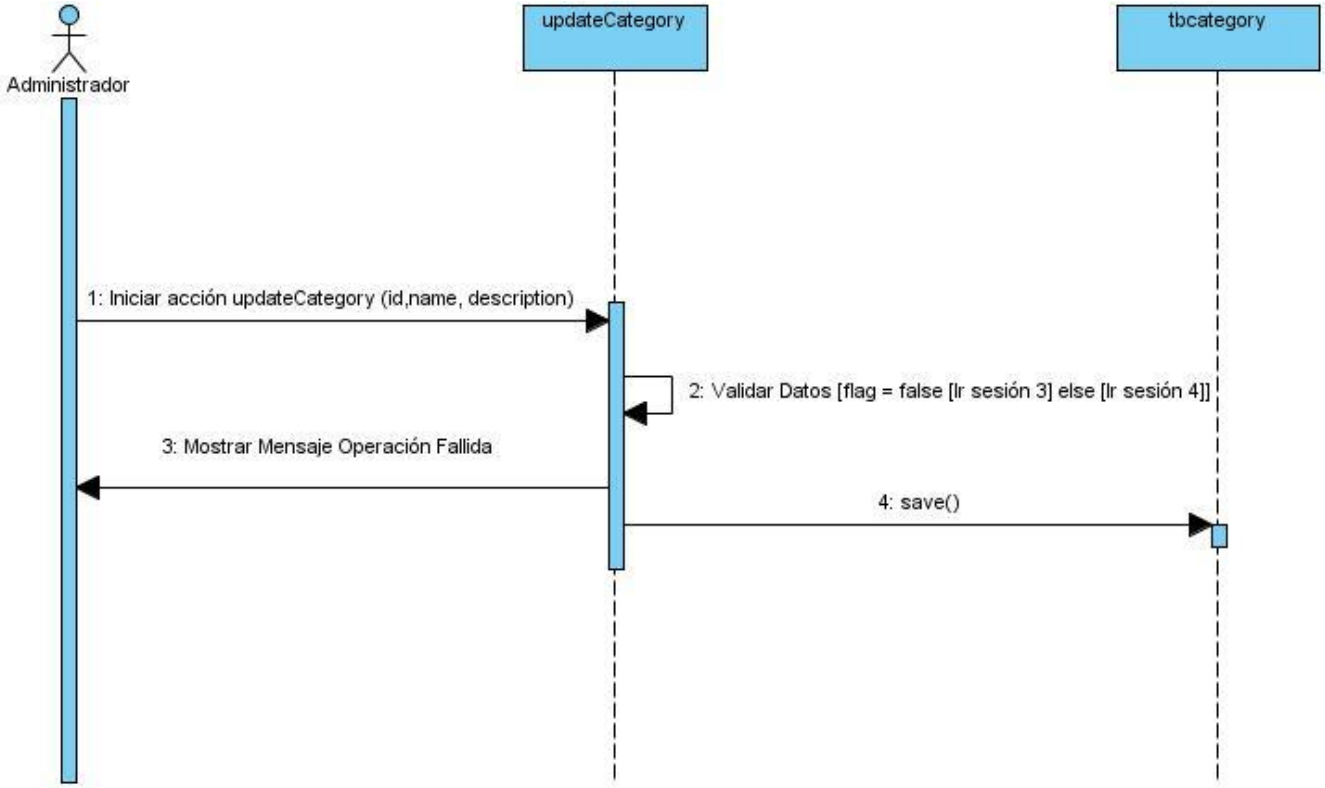

Figura 10: Diagrama de Secuencia Escenario Actualizar Categoría

Diagrama de Secuencia del Caso de Uso Eliminar Plantilla

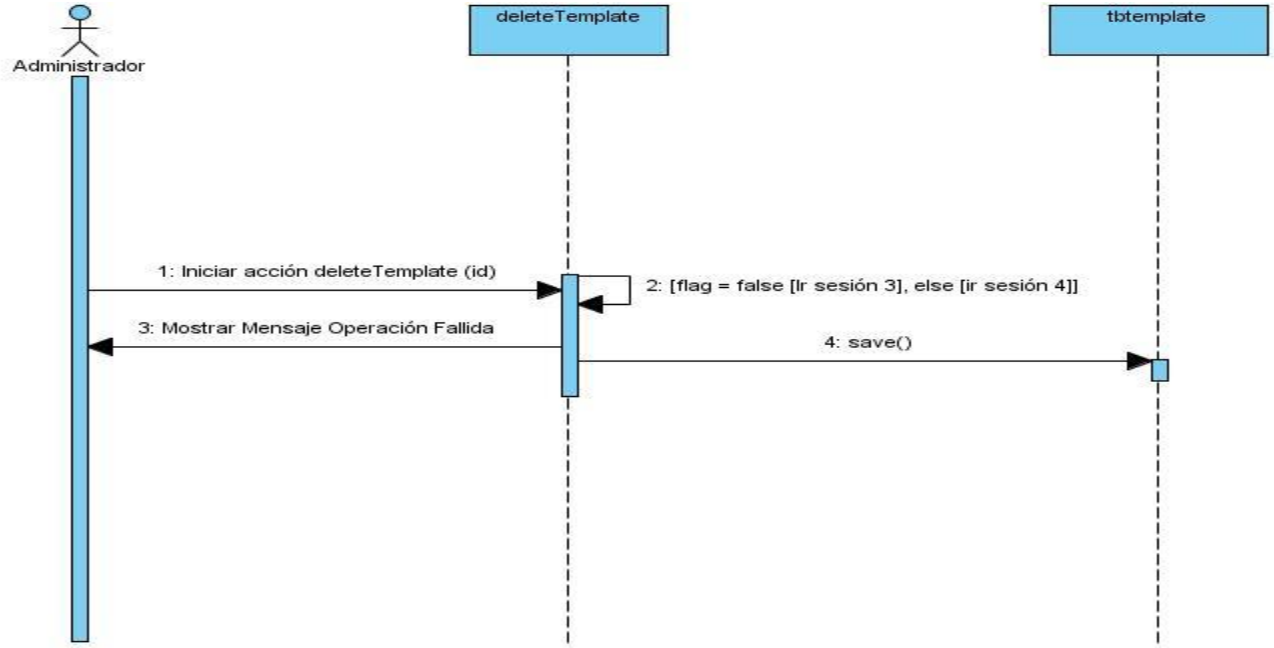

Figura 11: Diagrama de Secuencia Escenario Eliminar Plantilla

Conclusiones del capítulo

En este capítulo se planteó la relación entre los actores y los casos de uso mediante el diagrama de casos de uso del sistema (DCUS), representando además las funcionalidades que ofrece el sistema en lo que se refiere a su interacción externa. Se definieron los requisitos funcionales y no funcionales que el sistema debe poseer y efectuar.

Se determinó como patrón arquitectónico, el patrón Modelo-Vista-Controlador y Cliente-Servidor. Como patrones de diseño para asignar responsabilidades a objetos se usaron los patrones GRASP y GOF, entre ellos el Experto, Creador, Bajo acoplamiento, Alta cohesión. Se representó la estructura estática y dinámica del sistema, a través del diagrama de clases del diseño y de los diagramas de interacción (secuencia) respectivamente. También se describieron las clases más relevantes con el propósito de lograr un mejor entendimiento de las mismas

.

Capítulo 3: IMPLEMENTACIÓN Y PRUEBAS

Introducción

En este capítulo se realiza el diagrama de componentes para estructurar el modelo de implementación y se muestra la ubicación física de los nodos de procesamiento a través del diagrama de despliegue. Se realizan, pruebas de caja blanca para comprobar el correcto funcionamiento del software, se verifica que la entrada de datos se acepte de forma adecuada y que se produzca un resultado correcto.

3.1 Implementación

La implementación es cómo desarrollar y organizar los componentes basándose en las especificaciones del diseño. Es la realización de las especificaciones técnicas o algoritmos como un programa, componente software, u otro sistema de cómputo. Dentro de los principales objetivos que se desea alcanzar en una etapa de implementación se encuentran:

- ✓ Definir la organización del código, en términos de los subsistemas de implementación organizados en capas.
- ✓ Implementar los elementos de diseño en términos de los elementos de implementación (archivos de origen, binarios, programas ejecutables y otros).
- ✓ Probar y desarrollar componentes como unidades.
- ✓ Integrar los resultados producidos por los implementadores individuales, o equipos en un sistema ejecutable. (9)

Diagrama de Componentes

Un diagrama de componentes es un diagrama desarrollado en el lenguaje unificado de modelado (UML). Este diagrama representa como un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes.

Normalmente los diagramas de componentes contienen:

- ✓ Componentes: es una parte física de un sistema (módulo, base de datos, programa ejecutable, etc.). Se puede decir que un componente es la materialización de una o más clases, porque una abstracción con atributos y métodos pueden ser implementados en los componentes.

Los componentes se pueden agrupar en paquetes, además pueden haber entre ellos relaciones de dependencia como:

- ✓ Generalización
- ✓ Asociación
- ✓ Agregación
- ✓ Realización

- ✓ Interfaces: es el lazo de unión entre varios componentes.
- ✓ Paquetes o subsistemas.

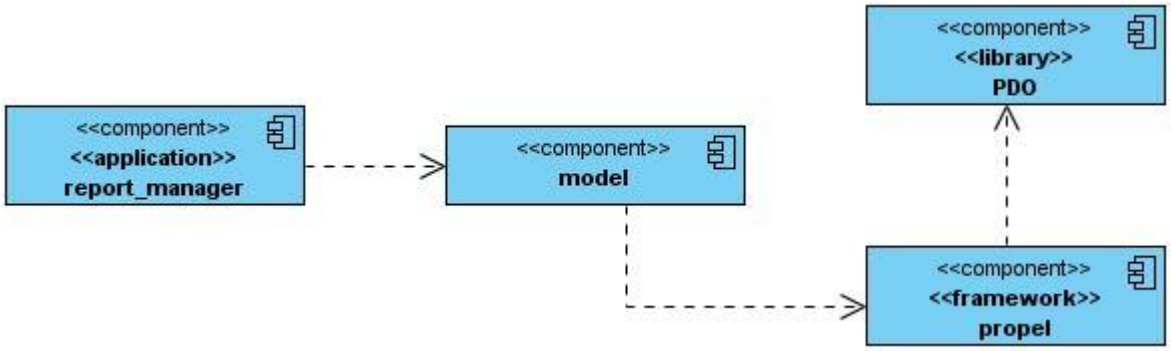

Figura 12: Diagrama de Componentes

Vista de Despliegue

La vista de despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema y muestra la forma en que están ubicados los componentes sobre dichos nodos, formando un grafo de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes, software, objetos y procesos. En general un nodo será una unidad.

Diagrama de Despliegue del Módulo Administrador de Reportes del GDR (V 2.0)

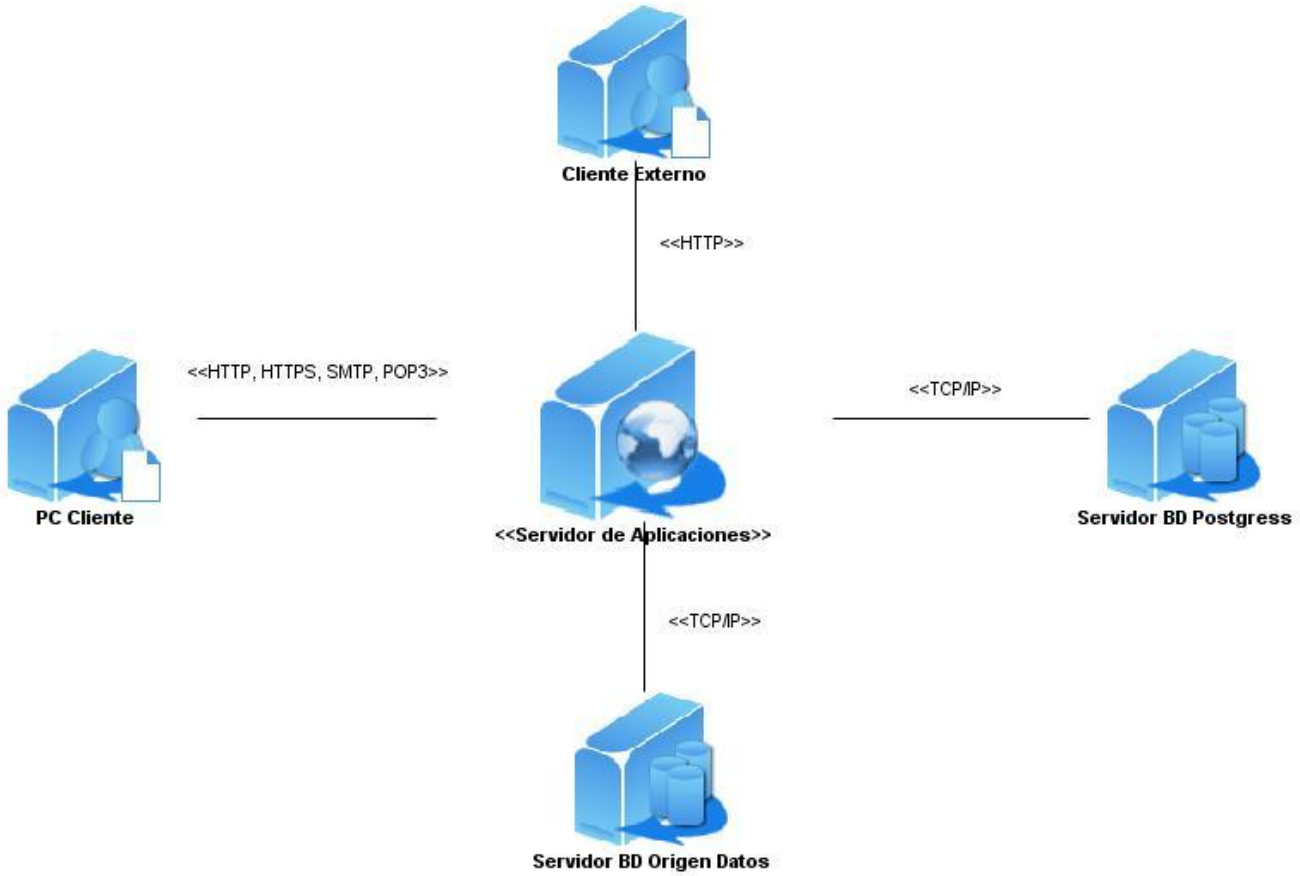

Figura 13: Vista de Despliegue

Implementaciones Relevantes

Entre los requisitos funcionales de relevancia en la nueva versión del Administrador de Reportes del Generador Dinámico de Reportes V (2.0) es la gestión de las subscripciones. Para darle solución a este requisito funcional se requiere una funcionalidad capaz de agregar una nueva subscripción, que permita determinar cuándo es su próxima ejecución de acuerdo a la frecuencia definida por el usuario y modificar el modelo de datos del sistema.

```
        return $this->renderText("true");
    else
        throw new Exception('Error al adicionar la programacion.');
}

public function createDailySchedule($subscriptionId, $scheduleType, $hour, $minutes) {
     // se obtiene la fecha actual con la hora y los minutos introducidos por el usuario
    $d = new DateTime(date("Y-m-d $hour:$minutes"));
    if (($hour < date('G')) || (($hour == date('G')) && ($minutes <= date('i'))))
        $d->add(new DateInterval('P1D'));

    // Creamos el objeto con los atributos
    $schedule = new Schedule();
    $schedule->setIdsubscription($subscriptionId);
    $schedule->setIdscheduletype($scheduleType);
     // se guarda la fecha a ejecutar en el formato '2011-06-06 16:00'
    $schedule->setLastruntime($d->format('Y-m-d G:i'));
    $schedule->setNextruntime('1 DAY');

    return $this->createSchedule($schedule);
}

public function createWeeklySchedule($subscriptionId, $scheduleType, $dayofweek, $hour, $minutes) {
    $d = new DateTime(date("Y-m-d $hour:$minutes"));

    $diaSemanaActual = $d->format('w');

    if ($dayofweek < $diaSemanaActual) {
        $d->add(new DateInterval('P' . (7 - ($diaSemanaActual - $dayofweek)) . 'D'));
    } else if ($dayofweek > $diaSemanaActual) {
        $d->add(new DateInterval('P' . ($dayofweek - $diaSemanaActual) . 'D'));
    } else {
        if (($hour < date('G')) || (($hour == date('G')) && ($minutes <= date('i'))))
            $d->add(new DateInterval('P7D'));
    }

    // Creamos el objeto con los atributos
    $schedule = new Schedule();
    $schedule->setIdsubscription($subscriptionId);
    $schedule->setIdscheduletype($scheduleType);
    $schedule->setLastruntime($d->format('Y-m-d G:i'));
    $schedule->setNextruntime('7 DAY');

    return $this->createSchedule($schedule);
```

3.2 Pruebas

Pruebas de Software

Para contribuir con la calidad del software es recomendable que el producto vaya siendo evaluado a medida que se va construyendo, además se hace necesario realizar durante el desarrollo, un proceso de evaluación y comprobación de los distintos productos o modelos que se van generando, en el que participarán desarrolladores y clientes. Las pruebas de software constituyen un pilar indispensable para evaluar y determinar la calidad de un software.

Tienen como objetivos:

- ✓ Encontrar y documentar los defectos que pueden afectar la calidad del software.

- ✓ Verificar que el software trabaje como fue diseñado.
- ✓ Validar y probar los requisitos que debe cumplir el software.
- ✓ Validar que los requisitos fueron implementados correctamente.

Otros conceptos importantes relacionados con las pruebas de software son:

Validación: es la operación de asegurar que los requerimientos funcionales y no funcionales están correcta y completamente implementados en el producto final.

Verificación: es la operación de establecer las especificaciones y entradas adecuadas a una actividad y la de establecer que las salidas de las actividades son correctas y consistentes con las especificaciones y la entrada.

Pruebas de Desarrollador

Es la prueba diseñada e implementada por el equipo de desarrollo. Tradicionalmente estas pruebas han sido consideradas solo para las pruebas de unidad, aunque en la actualidad en algunos casos pueden ejecutar pruebas de integración. Se recomienda que estas pruebas cubran más que las pruebas de unidad.

Pruebas de Unidad

Se enfocan en un programa o un componente que desempeña una función específica que puede ser probada y que se asegura que funcione tal y como lo define la especificación del programa. Los programadores siempre prueban el código durante el desarrollo, por lo que las pruebas unitarias son realizadas solamente después de que el programador considera que el componente está libre de errores.

Al tratar software Orientado a Objetos (OO) cambia el concepto de unidad. El encapsulamiento dirige la definición de clases y objetos. Esto significa que cada clase e instancia de clase (objeto) empaqueta los atributos (datos) y las operaciones (también conocidas como métodos o servicios) que manipulan estos datos. Por tanto, en vez de módulos individuales, la menor unidad a probar es la clase u objeto encapsulado. Una clase puede contener un cierto número de operaciones, y una operación particular puede existir como parte de un número de clases diferentes. (20)

3.3 Casos de prueba

Un caso de prueba se diseña según las funcionalidades descritas en los casos de usos. El propósito que se persigue con este artefacto es lograr una comprensión común de las condiciones específicas que la solución debe cumplir. Se parte de la descripción de los casos de usos del sistema, como apoyo para las revisiones. Cada planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión, además quedan plasmadas las revisiones realizadas al caso de prueba; así como un registro de todo aquello que no corresponde a la calidad del software.

Partiendo de la descripción de los casos de usos del sistema, como apoyo para las revisiones, se diseñó un caso de prueba asociado a cada caso de uso. Se efectuaron los casos de pruebas a 5 casos de uso críticos, plasmándose en la documentación del proyecto.

Para detallar el caso de uso se utiliza una tabla, donde se desglosa esta funcionalidad en secciones y a su vez estas en escenarios, para hacer más fructífera la ejecución de las pruebas. A continuación se presenta la tabla de secciones a probar en el caso de uso Gestionar Categorías:

Secciones a probar en el Caso de Uso Administrar Categoría de Reportes:

Tabla 3: Sección Administrar Categoría de Reportes

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC1 addCategory	EC 1.1: Adicionar categoría	El administrador decide introducir una nueva categoría, el sistema envía un mensaje indicando que la petición se realizó, terminando así el CU.
SC2 deleteCategory	EC 2.1: Eliminar categoría	El administrador elimina la categoría seleccionada y el sistema envía un mensaje indicando que la petición se ejecutó, terminando así el CU.
SC3 updateCategory	EC 3.1: Actualizar categoría	El administrador decide actualizar elementos de la categoría, el sistema envía un mensaje indicando que la

		petición se realizó, terminando así el CU.
SC4 showCategories	EC 4.1: Mostrar categorías	EL administrador decide ver todas las categorías, el sistema muestra en pantalla todas las categorías existentes, terminando así el CU.

A partir de esta descripción se detallan las variables que se encuentran asociadas al caso de uso.

Tabla 4: Descripción de la variable

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	name	Cadena de Texto	No	Obtiene el nombre de la categoría, está compuesto solo por letras y algunos símbolos de separación que sean estándar, soporta una longitud de hasta 50 caracteres.
2	container	Entero	No	Obtiene el padre o hijo de una determinada categoría. Está compuesto solo por números.
3	description	Cadena de Texto	No	Obtiene la descripción de la categoría, está compuesto solo por letras.
4	id	Entero	No	Identificador de la plantilla, obligatorio, autogenerado por el sistema.

Esta descripción posibilitó que se realizara una matriz de datos, donde se evaluó y probó la validez de cada uno de los datos introducidos en el sistema, específicamente en la sección que se estuvo probando. Utilizando un juego de datos validos e inválidos se identificó el empleo de la técnica de partición de equivalencia.

Matriz de Datos

Tabla 5: Sección addCategory

Escenario	Variables (Enumeradas según descripción de la variable)			Respuesta del Sistema	Resultado de la Prueba	Flujo Central
	1	2	3			
addCategory	V Superior	V 0	V Categoría Máxima	El sistema guarda los nuevos cambios de la Categoría en la BD, enviando un mensaje al usuario.	Satisfactorio	El sistema envía los datos codificados en formato JSON, retornando al cliente la respuesta de confirmación de la ejecución.
	I Superior	V 0	V Categoría Máxima	El sistema devuelve un mensaje tipo Exception debido a que ha ocurrido un error con el nombre de la nueva categoría.	Satisfactorio	

	V Inferior	I 10	V Categoría Máxima	El sistema devuelve un mensaje tipo Exception debido a que ha ocurrido un error con el padre de la nueva categoría.	Satisfactorio	
	V Inferior	V 0	I Categoría Máx1m@	El sistema devuelve un mensaje tipo Exception debido a que ha ocurrido un error con la descripción de la nueva categoría.	Satisfactorio	

Tabla 6: Sección deleteCategory

Escenario	Variables (Enumeradas según descripción de la variable)	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
	4			

deleteCategory	V 10	El sistema elimina la categoría de la BD, enviando un mensaje de confirmación al usuario.	Satisfactorio	El sistema envía los datos codificados en formato JSON, retornando al cliente la respuesta de confirmación de la ejecución.
	I 11	El sistema devuelve un mensaje tipo Exception debido a que ha ocurrido un error con los datos de la categoría.	Satisfactorio	

Tabla 7: Sección updateCategory

Escenario	Variables (Enumeradas según descripción de la variable)				Respuesta del Sistema	Resultado de la Prueba	Flujo Central
	1	2	3	4			
updateCategory	V Medio	V 0	V Categoría Media	V 10	El sistema guarda los nuevos cambios de la Categoría en la BD, enviando un mensaje al usuario.	Satisfactorio	El sistema envía los datos codificados en formato JSON, retornando al cliente la respuesta de confirmación

	V Medios	V 0	V Categoría Media	I 11	El sistema devuelve un mensaje de error con el id de la nueva categoría.	Satisfactorio	de la ejecución.
	I Medio	V 0	V Categoría Media	V 10	El sistema devuelve un mensaje tipo Exception debido a que ha ocurrido un error con el nombre de la nueva categoría.	Satisfactorio	
	V Médicos	I 10	V Categoría Media	V 10	El sistema devuelve un mensaje tipo Exception debido a que ha ocurrido un error con el padre de la nueva categoría.	Satisfactorio	

	V Medio	V 0	I Cat3goría Med1a	V 10	El sistema devuelve un mensaje tipo Exception debido a un error con la descripción de la nueva categoría.	Satisfactorio	

Tabla 8: Sección showCategory

Escenario	Variables (Enumeradas según descripción de la variable)	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
showCategory		El sistema muestra las categorías de la BD.	Satisfactorio	El sistema muestra todas las categorías existentes en la Base de Datos

Después de realizar las pruebas de Caja Negra mediante los Casos de Prueba asociados a cada Caso de Uso, se comprobó el correcto funcionamiento del módulo Administrador de Reportes V (2.0) y la correcta validación de los campos, verificando que las funcionalidades se comporten de acuerdo a los requerimientos realizados por los usuarios. Cada dificultad detectada en el desarrollo del módulo y resueltas a raíz del trabajo de los desarrolladores, fueron recogidas en la planilla de No Conformidades. En la siguiente tabla se muestra un resumen de las dificultades encontradas:

PD: Pendiente **RA:** Resuelta

Tabla 9: Resumen de los resultas de las pruebas aplicadas

Fecha	Versión	Caso de Prueba	Cant. de No Conformidad	Cant. de No Conformidad PD	Cant. de No Conformidad RA
10/05/2011	1.0	CU-Gestionar Reportes	-	-	-
3/06/2011	1.1	CU-Gestionar Reportes	2	-	2

Para más información referente a las dificultades encontradas en el proceso de desarrollo del módulo Administrador de Reportes: ver documentos de “Casos de Pruebas del Administrador de Reportes V (2.0)”.

Mecanismos de Implementación

- ✓ El sistema se desarrolló utilizando una arquitectura cliente-servidor.
- ✓ La parte del servidor se implementó en el lenguaje de programación PHP5, utilizando *Symfony* 1.4.x como *framework* de desarrollo para dicho lenguaje.
- ✓ Se mantuvo el estilo arquitectónico MVC propuesto por *Symfony*.
- ✓ La capa del Modelo (acceso a datos) se generó utilizando *Propel* como *ORM* incluido en la versión de *Symfony* a utilizar.
- ✓ Cada acción de la capa del controlador se escribió en un fichero independiente.
- ✓ Todos los métodos, nombres de clases y variables se escribieron en estructura *CamelCase*, comenzando siempre en minúscula.
- ✓ El acceso a las bases de datos de los orígenes de datos se realizó mediante la capa de abstracción PDO pero de forma unificada haciendo uso de la clase gdrPDO implementada con ese fin.
- ✓ La base de datos del sistema se implementó sobre *PostgreSQL* 9.0.3 y toda la estructura de la misma persistirá en un esquema llamado gdr_sch.
- ✓ No se envió desde el servidor información html ninguna, así como ninguna información específica para modificar el comportamiento de la interfaz de usuario, sólo se enviaron desde el servidor los datos que le atribuyen a este y en formato *JSON*.

Conclusiones del capítulo

En este capítulo se realizó el modelo de implementación con el propósito de mostrar los componentes del sistema y sus relaciones a través del diagrama de componentes. Mediante el diagrama de despliegue se mostró la configuración de los nodos de procesamiento en tiempo de ejecución y los vínculos de comunicación entre ellos. Además, se realizaron los casos de prueba para validar que los requisitos fueron implementados correctamente y se describen los casos de prueba que verifican el correcto funcionamiento del sistema a través de las pruebas de caja negra.

Conclusiones

La correcta especificación de los requerimientos y la realización del diseño del sistema permitieron implementar el módulo Administrador de Reportes V (2.0), utilizando el lenguaje de programación PHP 5.3.3.

Se probó el correcto funcionamiento del módulo a través de las pruebas de caja negra realizadas mediante los casos de prueba.

Como resultado del trabajo realizado se obtuvo el módulo Administrador de Reportes V (2.0), capaz de realizar los procesos de administración y entrega de los reportes de forma dinámica.

Recomendaciones

Luego del estudio realizado que culmina con la implementación del módulo Administrador de Reportes, perteneciente al Generador Dinámico de Reportes (V 2.0) incluido en PATDSI, se listan algunas recomendaciones para la construcción de nuevas versiones de dicho módulo:

- ✓ Realizar la implementación de la interfaz del módulo para una mejor integración con el GDR en su versión actual.
- ✓ Incorporar nuevas funcionalidades que permitan la administración de los reportes.

Bibliografía

- **Linet Lores Sánchez, Diana Monné Roque.** *Trabajo de Diploma: Aplicación de las pruebas de liberación al Sistema Informático de Genética Médica (Junio 2009).* junio 2009.
- [En línea] [Citado el: 10 de 12 de 2010.] econlink.com.a.
- [En línea] [Citado el: 24 de 5 de 2011.] file:///C:/Documents%20and%20Settings/Ary/Escritorio/Documentacion%20Cap%204/Cap%204/Calidad%20de%20Software.htm.
- [En línea] [Citado el: 25 de 11 de 2010.] http://antares.inegi.org.mx/metadatos/metadat1.htm.
- [En línea] [Citado el: 25 de 11 de 2010.] http://www.sinnexus.com/business_intelligence/datawarehouse.aspx.
- [En línea] 2009. [Citado el: 19 de Junio de 2011.] http://msdn.microsoft.com/es-es/library/ms145514.aspx.
- **1.5.0.1, OpenUP Version.** Ayuda de OpenUP Version 1.5.0.1. [En línea]
- **Álvarez Marañón, Gonzalo.** ¿Qué es Java? *Sitio oficial de la IEC.* [En línea] 1999. http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html.
- *Arquitectura de Software Dirigida por Modelos.pdf.*
- **B., Eduardo Abedrapo.** *Principales aspectos del Sistema Nacional de Inversiones de Chile.*
- **Blog, Miguel Matas.** [En línea] 5 de Junio de 2008. [Citado el: 1 de Febrero de 2011.] http://www.miguelmatas.es/blog/2008/05/06/como-definir-un-estandar-de-codificacion-yo-trabajo/.
- **Breidenbach, Craig Walls y Ryan.** *Spring in Action. Second Edition.* s.l. : Greenwick: Manning Publications Co., 2008.
- **Browne, Paul.** O'Relly on Java.com. *The independent Source for Enterprise Java.* [En línea] 2007. http://www.oreillynet.com/onjava/blog/2007/09/red_hat_developer_studio_good.html.
- *Búsqueda binaria y Ordenación.* **Programación, Departamento Docente Central de Técnicas de Programación Asignatura: Introducción a la.** Ciudad de la Habana : s.n., 2010.
- **Cáceres Cruz, Yander y Alea Boffill, Adonys.** *Análisis, Diseño e Implementación del Módulo de Investigaciones Internas del SIIPOL.* Ciudad Habana : s.n., 2009.
- Conferencia #7.Disciplina de Prueba. Ingeniería de Software II. [En línea]
- Conferencia #7.Disciplina de Prueba. Ingeniería de Software II. *Conferencia #7.Disciplina de Prueba. Ingeniería de Software II.* [En línea]
- **Conran, Aaron.** Sencha. [En línea] 3 de Mayo de 2009. [Citado el: 2 de Febrero de 2011.] http://www.sencha.com/blog/ext-js-30-rc11-released.

- ✓ **Corp, IBM.** *Rational Software Architect.* 2006.
- ✓ **Daniele, Marcela.** *Teoría 11: El Arte de Modelar UML.* 2007.
- ✓ *Desarrollo de Software Dirigido por Modelos, conceptos teóricos y su aplicación práctica.* **Pons, Dra. Claudia y Giandini, Dra. Roxana .** 2007.
- ✓ *Desarrollo de Software Dirigido por Modelos.* **manrubia Díez, Jorge y Cueva Lovelle, Juan Manuel .** Diciembre 2006, Vol. Memoria Segundo Semestre. FICYT.
- ✓ **Desarrollo de Software Dirigido por Modelos-manrubia Díez, Jorge y Cueva Lovelle, Juan Manuel.** *Desarrollo de Software Dirigido por Modelos.* Diciembre 2006.
- ✓ Developers code with Microsoft. [En línea] Microsoft, 2011. [Citado el: 4 de Junio de 2001.] http://msdn.microsoft.com/es-es/.
- ✓ **dsadada.** http://www.cetic.guerrero.gob.mx/pics/art/articles/113/file.TiposPruebasSoftware.pdf. *http://www.cetic.guerrero.gob.mx/pics/art/articles/113/file.TiposPruebasSoftware.pdf.* [En línea]
- ✓ EcuRed. [En línea] http://www.ecured.cu/index.php/Herramienta_CASE.
- ✓ EcuRed. [En línea] http://www.ecured.cu/index.php/Prueba_de_RUP.
- ✓ Entorno Virtual de Aprendizaje. [En línea] 2011. http://eva.uci.cu.
- ✓ **Eric Freeman, Elisabeth Freeman.** *Head First Design Patterns.*
- ✓ **Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides.** *Design Patterns: ElementS ofReusable Object-Oriented Software.* s.l. : Addison-Wesley, 1995. ISBN 0-201-63361-2.
- ✓ **Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.** *Design Pattens-Elements of Reusable Object Oriented Software.*
- ✓ *Estructura de datos III TP 1:Algoritmos de ordenamiento.* **Kennedy, Universidad Argentina John F.**
- ✓ **FLORES, MIRIAN MILAGROS DÍAZ.** XP Extreme Programming. [En línea] [Citado el: 1 de Febrero de 2011.] http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf.
- ✓ **Fowler, Martin.** *Pattern of Enterprises Architecture.*
- ✓ Free Download Manager. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.freedownloadmanager.org/es/.
- ✓ **Garcia, Joaquin.** IngenieroSoftware. [En línea] 27 de Mayo de 2005. [Citado el: 1 de Febrero de 2011.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.
- ✓ **Hernández, Pedro Veloso.** *Uso de patrones de arquitectura.*
- ✓ **Hernández, Yanoski Agneri Martínez.** *Sistema para la generación de reportes en la plataforma alasGRATO: Desarrollo del módulo "Reportador".* Ciudad de la Habana : s.n., 2010.

- ✓ **Hernández, Yasmany Hernández y Infante, Jenny Frometa.** *Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos.* Ciudad Habana : s.n., 2009.
- ✓ Hooping.net. [En línea] 2008. [Citado el: 1 de Febrero de 2011.] http://www.hooping.net/glossary/paquete-92.aspx.
- ✓ http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf. [En línea]
- ✓ http://www.exa.unicen.edu.ar/catedras/modysim/teoria/casos_de_uso_a.pdf. [En línea]
- ✓ http://www.fi.unju.edu.ar/materias/materia/IS1/document/Clases_Teoricas_2010/Introduccion-2010.pdf. [En línea]
- ✓ http://www.mitecnologico.com/Main/ModelosDeProcesoDeSoftware. [En línea]
- ✓ http://www.sparxsystems.com.ar/resources/tutorial/use_case_model.html. [En línea]
- ✓ *http://www.upedu.org/process/artifact/ar_desmd.htm.*
- ✓ **IBM Corporation & Sun Microsystems, Inc.** Eclipse Platform Technical Overview. *Sitio web oficial de la comunidad de desarrollo Eclipse.* [En línea] http://www.eclipse.org/whitepapers/eclipse-overview.pdf.
- ✓ Inavex. [En línea] 21 de Abril de 2008. [Citado el: 1 de Febrero de 2011.] http://ivanex.wikidot.com/patron-arquitectura.
- ✓ **Ingeniería, Facultad de Informática-Universidad Politécnica de Madrid-Unidad Doc de.** *Patrones del "Grang of Four ".* España-Madrid : s.n.
- ✓ **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de Software.* s.l. : Editorial Felix Varela, 1999.
- ✓ **Jaque, Miguel.** http://miguelJaque.com/index.php/tecnicas/tecnicasmodnegocio/37-modelado_negocio/46-modelo-de-dominio. [En línea] Diciembre del 2007.
- ✓ **JORGE GALVES, JUAN PABLO GOMEZ GALLEGO.** Scribd. [En línea] 16 de septiembre de 2007. [Citado el: 1 de Febrero de 2011.] http://www.scribd.com/doc/297224/RUP.
- ✓ Json. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.json.org/json-es.html.
- ✓ JSON. *Sitio oficial.* [En línea] http://www.json.org/json-es.html.
- ✓ **l.Jacobson, G.Booch, J.Rumbauggh.** *El Proceso Unificado de Desarrollo.* s.l. : Addison Wesley, 2000.
- ✓ **L.Nahuel, L.Ocaranza, M.Pinasco.** *Herramientas de soporte al proceso de desarrollo dirigido por modelos .* La Plata, Buenos Aires, Argentina : s.n.
- ✓ **La-Diversidad-de-Los-Objetos.** http://es.scribd.com/doc/25171397/1-La-Diversidad-de-Los-Objetos-1-1-Caracteristicas. [En línea]
- ✓ **La-Diversidad-de-Los-Objetos-1-1-Caracteristicas.** http://es.scribd.com/doc/25171397/1-La-Diversidad-de-Los-Objetos-1-1-Caracteristicas. [En línea]
- ✓ **Lago, Ramiro.** Introducción a JSF. [En línea] 2007. http://www.proactiva-calidad.com/java/jsf/introduccion.html.

- ✓ **Larman, Craig.** *Applying UML and Patterns.* s.l. : Prentice Hall - Estados Unidos. ISBN: 0131489062.
- ✓ **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* s.l. : México: Prentice Hall, 1999.
- ✓ **Lobo, Armando Robert.** *Arquitectura de Software para el Sistema Integrado de Gestión Estadística 2.0 Nuragas.* Ciudad de la Habana : s.n., 2009.
- ✓ Lobo_Tuerto. [En línea] [Citado el: 2 de Febrero de 2011.] http://lobotuerto.com/blog/2009/07/13/ext-js-3-0-ya-esta-aqui/.
- ✓ mailXmail.com. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.mailxmail.com/curso-desarrollo-aplicaciones-dispositivos-inalambricos-j2me/lenguaje-programacion-java.
- ✓ **Mann, Kito D.** *JavaServer Faces in Action.* s.l. : Greenwich: Manning Publications Co., 2005.
- ✓ **Marciniak, J.J.** *Process Models in Software Engineering.* s.l. : 2nd Edition, John Wiley and Sons, New York, December 2001.
- ✓ **Marleysis López Duque, Raidel Oceguera Ravelo.** *Tesis: Herramienta en Matlab para la obtención de información de la base de datos y ficheros electroencefalograma del proyecto Mapeo Cerebral Humano Cubano.* La Habana, Cuba : s.n., Junio, 2010.
- ✓ *MDA y el papel de los modelos en el proceso de desarrolo de software.* **Quintero, Juan Bernardo y Anaya, Raquel.** Colombia, Medelli : s.n : ISSN 1794-1237., Diciembre 2007. Número 8.
- ✓ **MDA.** http://es.scribd.com/doc/47014355/MDA. [En línea]
- ✓ **Modelos, Arquitectura de Software Dirigida por.** *Arquitectura de Software Dirigida por Modelos.pdf.*
- ✓ **ModelosDeProcesoDeSoftware.** http://www.mitecnologico.com/Main/ModelosDeProcesoDeSoftware. [En línea]
- ✓ **Molpeceres, Alberto.** javaHispano. [En línea] 15 de Octubre de 2002. [Citado el: 1 de Febrero de 2011.] http://www.javahispano.org/contenidos/es/patrones_de_software_parte_1/.
- ✓ NetBeans. [En línea] [Citado el: 20 de Noviembre de 2010.] http://netbeans.org/community/releases/70/index.html.
- ✓ **netbeans.org.** http://netbeans.org/index_es.html. [En línea]
- ✓ **netbeans.org.** netbeans.org. [En línea] www.netbeans.org.
- ✓ **Object Management Group.** UML. [En línea] http://www.uml.org/.
- ✓ **Olivares, José Rolando Lafaurie.** *Sistema para la generación de reportes en la plataforma alasGRATO: Desarrollo del módulo Reportador.* Habana : s.n., 2008.
- ✓ OpenUP. *OpenUP.* [En línea] [Citado el: 11 de Junio de 2011.] http://epf.eclipse.org/wikis/openup/index.htm.

- ✓ O'Really XML.com. [En línea] 2010. [Citado el: 2 de Febrero de 2011.] http://www.xml.com/.
- ✓ **Orellanos, Norah Velazco.** *Estadísticas de Inversión Extranjera Directa en los Países de la Comunidad Andina.*
- ✓ **Ortiz, Kadir Hector.** http://www.eumed.net/libros/2009c/583/Representacion%20del%20Modelo%20de%20Objetos%20de%20Dominio.htm. [En línea]
- ✓ **Padill, David Fuller.** Scribd. [En línea] 2003. [Citado el: 1 de Febrero de 2011.] http://www.scribd.com/doc/6371079/Roles-Desarrollo-Software.
- ✓ PentahoCorporation, 2005. [Citado el: 15 de 02 de 2011.] http://Pentah.com.
- ✓ **Pereira Ojeda, Maikel y Gago Martínez, Yudanis.** *Análisis, Diseño e Implementación del módulo Experticias Criminalísticas del Sistema de Investigación e Información Policial.* Ciudad Habana : s.n., 2008.
- ✓ **Pérez, Roberto Sarmiento.** *Diseño de la Base de Datos para un Sistema Integral de Gestión de Portafolios de Proyectos.* La Habana : s.n., 2010.
- ✓ *Perfiles UML y Desarrollo Dirigido por Modelos: Desafíos y Soluciones para Utilizar UML como Lenguaje de Modelado.* **Giachetti, Giovanni, Marín, Beatríz y Pastor, Oscar.** Valencia, España : SISTEDES, 2008. ISSN 1988–3455.
- ✓ pgAdmin. [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.pgadmin.org/index.php.
- ✓ **Prueba., Disciplina de.** http://eva.uci.cu, Conferencia #7.Disciplina de Prueba. Ingeniería de Software II. *http://eva.uci.cu, Conferencia #7.Disciplina de Prueba. Ingeniería de Software II.* [En línea]
- ✓ **PruebaCasoDePrueba.** http://www.mitecnologico.com/Main/PruebaCasoDePruebaDefectoFallaErrorVerificacionValidacion. *http://www.mitecnologico.com/Main/PruebaCasoDePruebaDefectoFallaErrorVerificacionValidacion.* [En línea]
- ✓ **pruebas, tipos de.** http://www.cetic.guerrero.gob.mx/pics/art/articles/113/file.TiposPruebasSoftware.pdf. *http://www.cetic.guerrero.gob.mx/pics/art/articles/113/file.TiposPruebasSoftware.pdf.* [En línea]
- ✓ **Pruebas_Funcionales.pdf.** http://carolina.terna.net/ingsw3/datos/Pruebas_Funcionales.pdf. *http://carolina.terna.net/ingsw3/datos/Pruebas_Funcionales.pdf.* [En línea]
- ✓ Qdiario. [En línea] [Citado el: 20 de Noviembre de 2010.] http://www.aplicacionesempresariales.com/postgresql-84.html.

- ✓ **Quintana, Abelardo López.** *Diseño e implementación del módulo de Gestión de indicadores del proyecto del Ministerio del Poder Popular de la Comunicación y la Información.* Ciudad de la Habana : s.n., 2010.
- ✓ **requirements.** http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf. [En línea]
- ✓ **Rivero Guevara, Humberto.** *Análisis, Diseño e Implementación del Módulo Aprehensión del SIIPOL.* Ciudad Habana : s.n., 2008.
- ✓ **Rodríguez Brito, Dayami.** *Plan de Aseguramiento de la Calidad.* Ciudad de la Habana : s.n., 2011.
- ✓ **Roque, Diana Monné.** *Aplicación de las pruebas de liberación al Sistema Informático de Genética Médica.* Habana : s.n., 2009.
- ✓ **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Madrid: Pearson Educación, 2000.
- ✓ **Sanchez, María A. Mendoza.** Informatizate. [En línea] 7 de Junio de 2004. [Citado el: 1 de Febrero de 2011.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.
- ✓ Secretaria de Economia . [En línea] http://www.economia.gob.mx/swb/work/models/economia/Resource/516/1/images/EstadInverMexicoUE.pdf.
- ✓ sesame. *sesame.* [En línea] [Citado el: 28 de 2 de 2011.] http://www.openrdf.org/doc/sesame/users/.
- ✓ SOA agenda. [En línea] http://soaagenda.com/journal/articulos/que-son-los-frameworks/.
- ✓ symfony. [En línea] [Citado el: 20 de Noviembre de 2010.] http://www.symfony-project.org/.
- ✓ **Systems, Por Popkin Software and.** *Modelado de Sistemas com UML.*
- ✓ **TiposPruebasSoftware.** http://www.cetic.guerrero.gob.mx/pics/art/articles/113/file.TiposPruebasSoftware.pdf. [En línea]
- ✓ **tutorial2.** http://www.adrformacion.com/cursos/metod5s/leccion1/tutorial2.html. [En línea]
- ✓ **UCI.** Entorno Virtual de Aprendizaje curso de Historia de la Informatica. [En línea] http://eva.uci.cu/mod/resource/view.php?id=9287&subdir=/Temas_Generales.
- ✓ **Universidad de las Ciencias Informáticas.** *Especificación de Requisitos de Software V2.0 del Generador Dinámico de Reportes.* s.l. : Sistema de Información de Gobierno.
- ✓ **Vázquez Pérez, Amed y Galvez Alonso, Arlan.** *Análisis, Diseño e Implementación del submódulo Evidencia del módulo Registro y Control del SIIPOL.* Ciudad Habana : s.n., 2009.

- ✓ Versión Cero. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.
- ✓ **Visual Paradigm.** *Sitio oficial Visual Paradigm.* [En línea] http://www.visual-paradigm.com.
- ✓ Visual Parading. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.visual-paradigm.com/.
- ✓ **visual-paradigm.** http://www.visual-paradigm.com/product/vpsuite/. [En línea]
- ✓ webmaster Club Planeta. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.cad.com.mx/historia_del_lenguaje_java.htm.
- ✓ **Wilson, Leslie B.** *Comparative Programming Languages, Second Edition.*
- ✓ **www.mastermagazine.info/termino/7006.php.** http://www.mastermagazine.info/termino/7006.php. [En línea]
- ✓ **www.slideshare.net/guest83f0d26/mda-2596889.** http://www.slideshare.net/guest83f0d26/mda-2596889. [En línea]
- ✓ **Yasmany Hernández Hernández, Ariel Sobrino Fernández.** *MODELO DE SISTEMA DEL SISTEMA ESTADÍSTICO PARA LA AYUDA A LA TOMA DE DECISIONES.* Habana : s.n., 2010. PR4-IM-CC-057.
- ✓ **Yasmany Hernández Hernández, Ernesto Sarduy Alonso, Fidel Alejandro Alberto Calafet.** *Especificación de requisitos de software del Sistema Estadístico para la Ayuda a la Toma de Decisiones.* Habana : s.n., 2010. IS-SW-DR-001.

Referencias Bibliograficas

1. NetBeans. [En línea] [Citado el: 20 de Noviembre de 2010.] http://netbeans.org/community/releases/70/index.html.
2. symfony. [En línea] [Citado el: 20 de Noviembre de 2010.] http://www.symfony-project.org/.
3. **Hernández, Yasmany Hernández y Infante, Jenny Frometa.** *Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos.* Ciudad Habana : s.n., 2009.
4. Qdiario. [En línea] [Citado el: 20 de Noviembre de 2010.] http://www.aplicacionesempresariales.com/postgresql-84.html.
5. pgAdmin. [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.pgadmin.org/index.php.
6. **Padill, David Fuller.** Scribd. [En línea] 2003. [Citado el: 1 de Febrero de 2011.] http://www.scribd.com/doc/6371079/Roles-Desarrollo-Software.
7. **Oceguera, Raidel Ravelo y López, Marleysi Duque.** *"Herramienta en Matlab para la obtención de información de la base de datos y ficheros electroencefalograma del proyecto Mapeo Cerebral Humano Cubano".* Habana : s.n., 2010.
8. OpenUP. *OpenUP.* [En línea] [Citado el: 11 de Junio de 2011.] http://epf.eclipse.org/wikis/openup/index.htm.
9. **Corp, IBM.** *Rational Software Architect.* 2006.
10. **Universidad de las Ciencias Informáticas.** *Especificación de Requisitos de Software V2.0 del Generador Dinámico de Reportes.* s.l. : Sistema de Información de Gobierno.
11. **Larman, Craig.** *Applying UML and Patterns.* s.l. : Prentice Hall - Estados Unidos. ISBN: 0131489062.
12. **Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides.** *Design Patterns: ElementS ofReusable Object-Oriented Software.* s.l. : Addison-Wesley, 1995. ISBN 0-201-63361-2.
13. **Garcia, Joaquin.** IngenieroSoftware. [En línea] 27 de Mayo de 2005. [Citado el: 1 de Febrero de 2011.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.
14. Developers code with Microsoft. [En línea] Microsoft, 2011. [Citado el: 4 de Junio de 2001.] http://msdn.microsoft.com/es-es/.
15. Visual Parading. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.visual-paradigm.com/.
16. Json. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.json.org/json-es.html.
17. O'Really XML.com. [En línea] 2010. [Citado el: 2 de Febrero de 2011.] http://www.xml.com/.
18. **Yasmany Hernández Hernández, Ernesto Sarduy Alonso, Fidel Alejandro Alberto Calafet.** *Especificación de requisitos de software del Sistema Estadístico para la Ayuda a la Toma de Decisiones.* Habana : s.n., 2010. IS-SW-DR-001.
19. **Yasmany Hernández Hernández, Ariel Sobrino Fernández.** *MODELO DE SISTEMA DEL SISTEMA ESTADÍSTICO PARA LA AYUDA A LA TOMA DE DECISIONES.* Habana : s.n., 2010. PR4-IM-CC-057.
20. **Roque, Diana Monné.** *Aplicación de las pruebas de liberación al Sistema Informático de Genética Médica.* Habana : s.n., 2009.

21. **Lobo, Armando Robert.** *Arquitectura de Software para el Sistema Integrado de Gestión Estadística 2.0 Nuragas.* Ciudad Habana : s.n., 2009.
22. Free Download Manager. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.freedownloadmanager.org/es/.
23. Hooping.net. [En línea] 2008. [Citado el: 1 de Febrero de 2011.] http://www.hooping.net/glossary/paquete-92.aspx.
24. Inavex. [En línea] 21 de Abril de 2008. [Citado el: 1 de Febrero de 2011.] http://ivanex.wikidot.com/patron-arquitectura.
25. Lobo_Tuerto. [En línea] [Citado el: 2 de Febrero de 2011.] http://lobotuerto.com/blog/2009/07/13/ext-js-3-0-ya-esta-aqui/.
26. mailXmail.com. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.mailxmail.com/curso-desarrollo-aplicaciones-dispositivos-inalambricos-j2me/lenguaje-programacion-java.
27. Versión Cero. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.
28. webmaster Club Planeta. [En línea] [Citado el: 2 de Febrero de 2011.] http://www.cad.com.mx/historia_del_lenguaje_java.htm.
29. **Blog, Miguel Matas.** [En línea] 5 de Junio de 2008. [Citado el: 1 de Febrero de 2011.] http://www.miguelmatas.es/blog/2008/05/06/como-definir-un-estandar-de-codificacion-yo-trabajo/.
30. **Conran, Aaron.** Sencha. [En línea] 3 de Mayo de 2009. [Citado el: 2 de Febrero de 2011.] http://www.sencha.com/blog/ext-js-30-rc11-released.
31. **FLORES, MIRIAN MILAGROS DÍAZ.** XP Extreme Programming. [En línea] [Citado el: 1 de Febrero de 2011.] http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf.
32. **JORGE GALVES, JUAN PABLO GOMEZ GALLEGO.** Scribd. [En línea] 16 de septiembre de 2007. [Citado el: 1 de Febrero de 2011.] http://www.scribd.com/doc/297224/RUP.
33. **Molpeceres, Alberto.** javaHispano. [En línea] 15 de Octubre de 2002. [Citado el: 1 de Febrero de 2011.] http://www.javahispano.org/contenidos/es/patrones_de_software_parte_1/.
34. **Sanchez, María A. Mendoza.** Informatizate. [En línea] 7 de Junio de 2004. [Citado el: 1 de Febrero de 2011.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.
35. **Olivares, José Rolando Lafaurie.** *Sistema para la generación de reportes en la plataforma alasGRATO: Desarrollo del módulo Reportador.* Habana : s.n., 2008.
36. **Corp, IBM.** *Rational Software Architect.* 2006.
37. **Lobo, Armando Robert.** *Arquitectura de Software para el Sistema Integrado de Gestión Estadística 2.0 Nuragas.* Ciudad de la Habana : s.n., 2009.
38. **Hernández, Pedro Veloso.** *Uso de patrones de arquitectura.*
39. **netbeans.org.** netbeans.org. [En línea] www.netbeans.org.
40. **Quintana, Abelardo López.** *Diseño e implementación del módulo de Gestión de indicadores del proyecto del Ministerio del Poder Popular de la Comunicación y la Información.* Ciudad de la Habana : s.n., 2010.
41. **Universidad de las Ciencias Informáticas.** *Especificación de Requisitos de Software V2.0 del Generador Dinámico de Reportes.* s.l. : Sistema de Información de Gobierno.

42. **Hernández, Yanoski Agneri Martínez.** *Sistema para la generación de reportes en la plataforma alasGRATO: Desarrollo del módulo "Reportador".* Ciudad de la Habana : s.n., 2010.
43. *Estructura de datos III TP 1:Algoritmos de ordenamiento.* **Kennedy, Universidad Argentina John F.**
44. *Búsqueda binaria y Ordenación.* **Programación, Departamento Docente Central de Técnicas de Programación Asignatura: Introducción a la.** Ciudad de la Habana : s.n., 2010.
45. **Pérez, Roberto Sarmiento.** *Diseño de la Base de Datos para un Sistema Integral de Gestión de Portafolios de Proyectos.* La Habana : s.n., 2010.
46. **Universidad de las Ciencias Informáticas.** Tesis.uci. [En línea] 2010. http://tesis.uci.cu.
47. —. Entorno Virtual de Aprendizaje. [En línea] 2011. http://eva.uci.cu.
48. [En línea] PentahoCorporation, 2005. [Citado el: 15 de 02 de 2011.] http://Pentah.com.
49. [En línea] 2009. [Citado el: 19 de Junio de 2011.] http://msdn.microsoft.com/es-es/library/ms145514.aspx.
50. [En línea] [Citado el: 25 de 11 de 2010.] http://antares.inegi.org.mx/metadatos/metadat1.htm.
51. [En línea] [Citado el: 25 de 11 de 2010.] http://www.sinnexus.com/business_intelligence/datawarehouse.aspx.
52. [En línea] [Citado el: 10 de 12 de 2010.] econlink.com.a.
53. [En línea] [Citado el: 24 de 5 de 2011.] file:///C:/Documents%20and%20Settings/Ary/Escritorio/Documentacion%20Cap%204/Cap%204/Calidad%20de%20Software.htm.
54. **Álvarez Marañón, Gonzalo.** ¿Qué es Java? *Sitio oficial de la IEC.* [En línea] 1999. http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html.
55. **Breidenbach, Craig Walls y Ryan.** *Spring in Action. Second Edition.* s.l. : Greenwick: Manning Publications Co., 2008.
56. **Browne, Paul.** O'Relly on Java.com. *The independent Source for Enterprise Java.* [En línea] 2007. http://www.oreillynet.com/onjava/blog/2007/09/red_hat_developer_studio_good.html.
57. **Cáceres Cruz, Yander y Alea Boffill, Adonys.** *Análisis, Diseño e Implementación del Módulo de Investigaciones Internas del SIIPOL.* Ciudad Habana : s.n., 2009.
58. EcuRed. [En línea] http://www.ecured.cu/index.php/Herramienta_CASE.
59. EcuRed. [En línea] http://www.ecured.cu/index.php/Prueba_de_RUP.
60. **IBM Corporation & Sun Microsystems, Inc.** Eclipse Platform Technical Overview. *Sitio web oficial de la comunidad de desarrollo Eclipse.* [En línea] http://www.eclipse.org/whitepapers/eclipse-overview.pdf.
61. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de Software.* s.l. : Editorial Felix Varela, 1999.
62. JSON. *Sitio oficial.* [En línea] http://www.json.org/json-es.html.
63. **Lago, Ramiro.** Introducción a JSF. [En línea] 2007. http://www.proactiva-calidad.com/java/jsf/introduccion.html.
64. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* s.l. : México: Prentice Hall, 1999.
65. **Mann, Kito D.** *JavaServer Faces in Action.* s.l. : Greenwich: Manning Publications Co., 2005.

66. **Object Management Group.** UML. [En línea] http://www.uml.org/.
67. **Pereira Ojeda, Maikel y Gago Martínez, Yudanis.** *Análisis, Diseño e Implementación del módulo Experticias Criminalísticas del Sistema de Investigación e Información Policial.* Ciudad Habana : s.n., 2008.
68. **Rivero Guevara, Humberto.** *Análisis, Diseño e Implementación del Módulo Aprehensión del SIIPOL.* Ciudad Habana : s.n., 2008.
69. **Rodríguez Brito, Dayami.** *Plan de Aseguramiento de la Calidad.* Ciudad de la Habana : s.n., 2011.
70. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Madrid: Pearson Educación, 2000.
71. SOA agenda. [En línea] http://soaagenda.com/journal/articulos/que-son-los-frameworks/.
72. **Vázquez Pérez, Amed y Galvez Alonso, Arlan.** *Análisis, Diseño e Implementación del submódulo Evidencia del módulo Registro y Control del SIIPOL.* Ciudad Habana : s.n., 2009.
73. **Visual Paradigm.** *Sitio oficial Visual Paradigm.* [En línea] http://www.visual-paradigm.com.
74. **Wilson, Leslie B.** *Comparative Programming Languages, Second Edition.*
75. Wikipedia, la enciclopedia libre. [En línea] http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado.
76. Wikipedia, the free encyclopedia. [En línea] 3 de Junio de 2011. http://en.wikipedia.org/wiki/Library_%28computing%29.
77. Lazarus wiki. [En línea] http://wiki.lazarus.freepascal.org/Lazarus_For_Delphi_Users#Delphi_-.3E_Lazarus.

Glosario

***camelCase*:** estilo de escritura que se aplica a frases o palabras compuestas.

CU: caso de uso.

***Extensible Markup Language* (XML):** *extensible markup language* (Lenguaje de Marcas Extensible).

***Frameworks*:** marco de trabajo.

GDR: generador dinámico de reportes.

IDE: entorno desarrollo integrado**.**

JavaScript Object Notation (JSON): notación de objetos de *JavaScript* es un formato ligero de intercambio de datos.

OpenUp: metodología para el proceso del desarrollo del software.

Php: h*ypertext pre-processor* lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas.

Reporte: documento caracterizado por contener información u otra materia reflejando el resultado de una investigación adaptado al contexto de una situación.

Report_Manager: es el módulo encargado de responder las peticiones realizadas al Administrador de Reportes del Generador Dinámico de Reportes (V 2.0).

UML: *unified modeling language* (Lenguaje Unificado de Modelado).