Universidad de las Ciencias informáticas

Facultad 15

Título: Diseño e implementación de los reportes de SINAPSIS.

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

Autores: Franklin René Rivero García.
Enrique Trujillo Larrauri.

Tutor: Ing. Dalgis Rogelio López Góngora.
Co-tutor: Martín Marín Villalón Cruzata.

Ciudad de la Habana
Marzo 2010

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Enrique Trujillo Larrauri

_____

Firma del Autor

Franklin René Rivero García

_____

Firma del Autor

Dalgis Rogelio López Góngora

_____

Firma del Tutor

Dedicatoria

***A** mi mamá especialmente por todo el amor y el apoyo que me ha dado siempre.*

***A** mi hermano Roger que aunque no estés con nosotros siempre te querré por haber sido más que un hermano.*

***A** mi abuela y mis hermanos.*

Enrike.

***A** toda mi familia, que han compartido el sueño de ser ingeniero conmigo.*

Franklin.

Agradecimientos

A mi mamá por siempre estar a mi lado, por todos los sacrificios y las cosas imposibles que ha hecho por mí. A ti te debo todo lo que soy, esto es por ti.

A mi hermano ito por salvarme siempre en los momentos malos.

A mi abuela Graciellita por ser siempre mi compañía.

A mis tios Chaly y Aramís por la ayuda y los consejos que siempre me han dado que me han servido de mucho. Ahh y las clases de guitarra!!!.

A todos mis hermanos.

A todos mis amigos especialmente a los que han seguido hasta hoy y los que ya no están.

Al viejo Juan, al Luisma y Ariel: el piquete del tetrafósforo.

A todos los profesores y a aquellos que me han ayudado profesionalmente en mi carrera.

Y a todo el que cuando lea esto sienta que falta, es que son muchos y se olvida. Je!!!!

Enrike.

A mi mamá y mi papá que durante los 17 años de estudio me han guiado.

A mis tíos y primos de la Habana: Haydee, Nito, Elsa, Yanu, Nana, Lauren, Lando.

A mi tía Mayra que siempre me ha ayudado.

A toda mi familia en general que siempre me han apoyado.

A mis vecinos y amigos del barrio, a las amistades que he conocido durante mis años de estudiante de la primaria, secundaria, pre-uniersitario y Universidad.

A los profesores que han contribuido a mi formación profesional.

A Google (http://www.google.com.cu/).

Franklin.

Resumen

La República Bolivariana de Venezuela y El Ministerio del Poder Popular para la Planificación y Desarrollo han reconocido la necesidad de realizar un sistema informático para el seguimiento de las inversiones realizadas en los proyectos realizados en dicha república. Para lo cual se crea el proyecto SINAPSIS (Sistema Nacional Público para el Seguimiento de las Inversiones y Sectores) el cual consta de 7 módulos entre los cuales se encuentra el módulo Reportes. Éste módulo consta de un conjunto de reportes necesarios para llevar a cabo un mayor nivel de gestión, control y seguimiento de los proyectos que se ejecutan.

En el presente trabajo se realiza el diseño e implementación de los reportes del proyecto SINAPSIS con el objetivo de obtener una buena implementación del mismo y con la calidad que éste requiere. Por lo que se realizó un estudio del arte de las metodologías de desarrollo, el entorno de desarrollo y las métricas para la implementación. Se utilizaron estándares de codificación y se generaron los siguientes artefactos correspondientes al diseño e implementación: diagrama de clases del diseño, diagrama de despliegue, diagrama de componentes y código fuente. Finalmente se validan los artefactos obtenidos para que la implementación termine con la calidad que se requiere.

Índice de Figuras

Índice

Introducción

En la República Bolivariana de Venezuela se lleva a cabo el presupuesto por proyectos. A través del presupuesto el gobierno proporciona bienes y servicios para atender las necesidades de la comunidad; cancela servicios que permiten su funcionamiento, paga a proveedores y contratistas, resuelve problemas de pasivos laborales y contractuales con sus trabajadores.

Para llevar a cabo la gestión del presupuesto por proyectos, el Ministerio del Poder Popular para la Planificación y Desarrollo, conjuntamente con el Ministerio de Economía y Finanzas y la Vicepresidencia de la República, decidieron implantar un sistema informático para el reporte de planes, acciones y avances de los proyectos, así como el registro, seguimiento y control de todos los proyectos que formarían parte del presupuesto anual de la nación, este sistema fue denominado Nueva Etapa.

El sistema Nueva Etapa no fue pensado para que los usuarios ingresen los datos referentes a los proyectos en todo momento, sino que cuentan con períodos de tiempos que se establecen de forma manual sobre el sistema; estos períodos de tiempo son muy cortos por lo que la información registrada en el mismo no está actualizada en todo momento. La información referente a la planificación de los proyectos puede ser modificada libremente sin tener en cuenta fechas o períodos de modificación alguno, lo que trae como consecuencia que la planificación histórica de un proyecto pueda ser modificada, lo que brinda reportes falsos del seguimiento que se le ha dado a los proyectos, afectando grandemente a la planificación de presupuestos.

Dada la necesidad que representa un sistema eficiente que cumpla con todos los requerimientos para llevar a cabo la gestión del presupuesto por proyectos, los principales organismos rectores de Venezuela deciden cambiar el actual software, surgiendo así el Sistema Nacional Público para el Seguimiento de Inversiones y Sectores (SINAPSIS).

Dentro de las funcionalidades que debe presentar dicho sistema está la de obtener información integra de la ejecución de los proyectos a través de reportes. Actualmente este proceso se realiza manualmente o de forma semiautomática. Se hacen mediante vías no formales (Llamadas telefónicas, documentos, email, etc.).

De todo lo expuesto anteriormente se determinó el siguiente **Problema Científico**:
¿Cómo generar y presentar la información gestionada en el proyecto SINAPSIS?

Objeto de Estudio

El proceso de desarrollo de software.

Campo de acción

Diseño e Implementación del módulo Reportes del proyecto SINAPSIS.

Objetivo general

Desarrollar el módulo de Reportes de SINAPSIS para obtener información íntegra de la ejecución de los proyectos correspondientes al presupuesto anual de la nación.

Tareas de la investigación:

- Revisión bibliográfica de las herramientas de diseño de reportes.
- Integración de la herramienta de generación de reporte seleccionada con los frameworks Hibernate y Spring.
- Realización de los artefactos del diseño.
- Implementación las clases, interfaces y plantillas necesarias para desarrollar el módulo Reportes.
- Validación de la implementación mediante las pruebas de caja negra.

Hipótesis

Si se desarrolla adecuadamente el módulo Reportes para el proyecto SINAPSIS entonces se facilitará obtener información integra de la ejecución de los proyectos a través de reportes apoyando así en la toma de decisiones.

Métodos científicos

Histórico-Lógico: Para conocer las tendencias actuales de los sistemas de gestión de reportes, así como conocer los antecedentes de las distintas herramientas y tecnologías utilizadas para implementar los reportes.

Analítico-Sintético: Se utilizó para el procesamiento de la información permitiendo analizar los documentos y la extracción de los elementos más importantes acerca del proceso de generación de reportes, para precisar características del diseño propuesto, así como para el arribo de las conclusiones de la investigación.

Hipotético - Deductivo: Se utilizó para a partir del problema concreto y objetivo plantear la hipótesis que será puesta a contrastación en todo el proceso investigativo.

Métodos empíricos

Observación: Se utilizó para realizar una valoración a partir de la percepción en la etapa exploratoria de la realidad estudiada para determinar cómo ocurre realmente el proceso de gestión de reportes.

Experimentación: Para realizar pruebas y validar la solución propuesta corrigiendo los errores que puedan encontrarse y de esta forma comprobar la validez de la hipótesis

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se realizará una revisión bibliográfica acerca de los sistemas automatizados con módulos de reportes. Se justificarán las herramientas, tecnologías y metodología, así como los frameworks, el lenguaje, la plataforma y el servidor de aplicación que se utilizará en el desarrollo de éste trabajo. Se realizará además un estudio acerca de las diferentes herramientas para la generación de reportes, así como también se abordará sobre cómo se deben realizar los sistemas de reportes para el desarrollo de los productos de software.

1.2 Revisión bibliográfica

A continuación se muestra un estudio de una serie de sistemas automatizados compuestos por módulos de reportes para la gestión de información:

1.2.1 Sistemas automatizados compuestos por módulos de reportes en el Mundo

CENACAD: Sistema Censo Académico en Línea para la Automatización de la Evaluación a los Docentes.

CENACAD fue realizado por parte del personal de la Escuela Superior Politécnica del Litoral (ESPOL), Guayaquil, Ecuador.

El sistema CENACAD (Censo Académico en Línea) es la innovación de las evaluaciones docentes mediante el uso de nueva tecnología en el manejo de sistemas en ambiente Web, creando un repositorio de datos, diferentes accesos, diferentes reportes, siendo así una herramienta para encontrar una buena aproximación de la enseñanza en la Escuela Superior Politécnica del Litoral (ESPOL). El sistema nace de la necesidad de brindar una solución informática sobre las evaluaciones docentes en la ESPOL, permitiendo que el estudiante pueda evaluar de una forma mucho más tranquila y sin necesidad de interrumpir sus horas de estudio, mediante el uso de Internet, garantizando de este modo la confiabilidad de los datos. El sistema fue diseñado no sólo para comodidad de los estudiantes, profesores y directivos, sino, además para reducir costos y tiempo, permitiendo la ampliación de los servicios del Centro de Investigaciones y Servicios Educativos (CISE), dando la posibilidad de implementar este sistema a otras instituciones educativas. (J. Vásquez, 2009)

El sistema tiene la capacidad de:

- Mostrar reportes en línea, que son de fácil entendimiento, con información mostrada de una manera amigable y que pueden ser descargadas desde Internet.
- Generar documentos de Excel con información específica.

El módulo de Reportes de CENACAD permite la obtención de diferentes tipos de reportes a los usuarios, los que son mostrados a continuación:

Reporte de dominio público:

a) Promedio General de ESPOL (por término).
b) Promedio por Unidad (por término).
c) Promedio por Profesor-Término (por término).
d) Listado de mejores docentes en base a los mejores promedios obtenidos en la ESPOL por el período de un año.
e) Promedio de la(s) facultad(es) o Instituto(s).
f) Promedios obtenidos en cada área que conforman cada una de las encuestas realizadas a cada materia. Estos promedios deberán ser: general, de toda la facultad o unidad. Además son presentadas de manera gráfica.

CELTA: Sistema para entidades de economía solidaria.

CELTA ® es el primer sistema integral de tipo ERP desarrollado en Colombia y orientado en su totalidad a entidades de economía solidaria, dentro de las que se encuentran Fondos de Empleados, Cooperativas, Pre cooperativas, Asociaciones Mutuales, Organismos de Representación, Administraciones Públicas Cooperativas, Instituciones Auxiliares Especializadas y Organismos de Segundo Grado de Carácter Económico. En virtud de ser un Sistema tipo ERP (Planeación de Recursos Empresariales), el sistema CELTA está conformado por 23 módulos que enmarcan las áreas gerencial, comercial, administrativa, operativa, financiera y fiscal, permitiendo automatizar los procesos propios de la gestión de cada área, brindando la funcionalidad requerida por cada una de las áreas y dando manejo centralizado a la información, de forma tal que se encuentre disponible para la consulta de los usuarios, permitiendo explotar la información y reduciendo drásticamente las cargas operativas. (Goldentech, 2010)

El sistema Celta se compone por 23 módulos, dentro de los que se destaca el módulo de reportes por su gran utilidad a la hora de gestionar información.

Módulo de Reportes de CELTA**:** La generalidad de los reportes del Sistema CELTA® se encuentran centralizados en este módulo. El sistema permite seleccionar módulos, y funcionalidades para desplegar los informes asociados. Al realizar una selección, se carga la información en un reporte en Cristal Reports, que en gran parte de los casos permite ingresar diversos parámetros que amplían su funcionalidad. (Goldentech)

1.2.2 Sistemas automatizados compuestos por módulos de reportes en Cuba

Babel: Sistema Automatizado de Gestión de Información para los servicios de traducción e interpretación.

Babel es un sistema que integra las tecnologías de la información a la gestión de solicitudes de los servicios de traducción e interpretación del Centro de Información. Mediante una interfaz de comunicación amigable los usuarios pueden realizar un intercambio de datos entre todas las funciones implicadas en este proceso y así aprovechar adecuadamente las sinergias que se producen entre todas y cada una de las funciones. Este sistema da a sus usuarios la información precisa sobre el estado de su solicitud y además, las competencias del traductor, al aumentar el valor añadido de cada recurso que interviene en el proceso. Esta herramienta de trabajo permite la organización, clasificación de la información, la recuperación de documentos con oportunas normas de seguridad. Babel v1.0 es una aplicación programada en PHP (Preprocesador de Hipertexto), lenguaje de programación de aplicaciones Web. (Álvarez, 2006)

Dentro de los objetivos a la hora de pensar el sistema podemos encontrar el de generar reportes para saber el estado actual de las solicitudes que se realizan.

GestCult: Sistema de gestión de información de actividades culturales.

GestCult fue diseñado e implementado a petición del Ministerio Nacional de Cultura para solventar una serie de dificultades a la hora de organizar los diferentes tipos de actividades que se realizan constantemente como parte de la expansión territorial de la cultura y recreación en nuestro país.

Ejemplo de estas actividades son las diferentes giras artísticas de actores, elencos de telenovelas, agrupaciones musicales, humoristas por señalar sólo algunos, que conforman ese gran potencial cultural que se extiende por cada rincón de nuestro archipiélago regalándole a los cubanos su trabajo y dedicación a su vocación. (Vázquez, 2006)

Módulo de reportes estadísticos de GestCult: Todo sistema de gestión de información debe brindar la posibilidad de acceder a la información generada y esto es precisamente lo que se puede hacer mediante el módulo de reportes estadísticos. Aquí se generan un grupo de reportes solicitados por la dirección de programas culturales y que forman parte de toda la información que hasta el momento se consolida de forma manual en la dirección.

1.2.3 Sistemas automatizados compuestos por módulos de reportes en la UCI

AKADEMOS Sistema Automatizado de Gestión Académica

Akademos es el Sistema Automatizado para la Gestión Académica por el cual se maneja actualmente la información docente en la Universidad de Ciencias Informáticas. El sistema de gestión de información académica cuenta en la actualidad con una serie de módulos que engloban diferentes tipos de áreas. A través de la red interna de la universidad, estudiantes, secretarias docentes y profesores, hacen uso de las ventajas ofrecidas por el sistema para la gestión de información.

Como tecnologías básicas de desarrollo para el sistema Akademos se destacaron:

- SQL Server 2000 como gestor de bases de datos.
- Plataforma de desarrollo .Net
- Servicios WEB XML (lenguaje de marcas extensible) para el intercambio con otras aplicaciones.
- IIS (Servidor de Información del Internet) como servidor web.

Dentro de los módulos con que cuenta el sistema, se encuentra el módulo de reportes, elemento que permite a los usuarios del sistema la obtención de reportes de forma rápida y confiable.

SICCV: Sistema Integral de Cooperación Cuba-Venezuela

La República de Cuba y la República Bolivariana de Venezuela, en aras de intensificar sus relaciones diplomáticas en los marcos del ALBA, mantienen en conjunto un acuerdo de colaboración, los cuales comprenden la concepción y ejecución de varios proyectos. El control de los proyectos se efectúa de forma semiautomática, mediante reuniones de coordinación, vía telefónica y por correo electrónico, para darle soporte a la gestión de estos, se crea el proyecto “Convenio Integral de Cooperación Cuba-Venezuela” (CICCV), el cual consta de 7 módulos entre los cuales se encuentra el módulo Reportes el cual consta

de un conjunto de reportes necesarios para llevar a cabo un mayor nivel de gestión, control y seguimiento de los proyectos que se ejecutan.

1.2.4 Conclusiones del estudio de los antecedentes del tema investigativo

A partir del estudio realizado sobre sistemas compuestos por módulos de reportes en el Mundo, Cuba y la Universidad de Ciencias Informáticas, a pesar de reconocerse que algunos de los sistemas poseen características ventajosas a la hora de crear reportes, se determinó que ninguno de estos sistemas está compuesto por un módulo de reporte acorde a lo que se quiere diseñar según las necesidades expuestas por parte de los clientes.

A continuación se realizará un análisis de las metodologías, lenguajes y herramientas de diseño y generación de reportes con el objetivo de seleccionar la que satisfaga los requisitos del cliente.

1.3 Metodologías de desarrollo de software

En la concepción de cada proyecto es de gran importancia realizar un buen proceso de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software (**Ver Figura 1**). El proceso de desarrollo de software requiere por un lado un conjunto de conceptos, una metodología y un lenguaje propio.

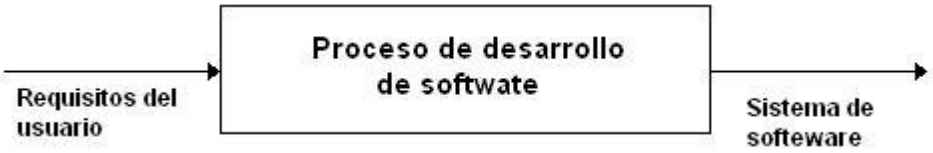

Figura 1. **Proceso de desarrollo de software**

Una metodología de desarrollo de software es un conjunto de pasos y procedimientos que deben seguirse para desarrollar software. Normalmente consiste en un conjunto de fases descompuestas en subfases (módulos, etapas, pasos, etc.). Esta descomposición del proceso de desarrollo guía a los desarrolladores en la elección de las técnicas que debe elegir para cada estado del proyecto, y facilita la planificación, gestión, control y evaluación de los proyectos. Dentro de las metodologías más utilizadas se encuentra Rational Unified Process (RUP).

1.3.1 RUP

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del trabajo de muchas metodologías utilizadas por los clientes. Fue creado por Jacobson, Rumbaugh y Booch. Se inserta dentro de las metodologías orientadas a objetos, es una opción a tener en cuenta para desarrollar grandes y complejos proyectos.

RUP es un proceso de desarrollo de software. Sin embargo, el Proceso Unificado es más que un simple proceso; es un marco genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto (Jacobson, et al., 2000)

RUP define **quién** (trabajadores) hace **qué** (artefactos), **cómo** los hace (Actividades) y **cuándo** los hace (flujo de actividades). Además está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes interconectados a través de interfaces bien definidas. RUP utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema software. De hecho UML es una parte esencial de él. (Jacobson, 2000) Sin embargo, los rasgos que caracterizan el ciclo de vida de RUP son:

Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. Sin embargo, los casos de uso no son sólo una herramienta para especificar los requisitos de un sistema. También guían su diseño, implementación, y prueba; esto es, guían el proceso de desarrollo. (Jacobson, 2000)

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción. La arquitectura le da la forma necesaria al sistema para que los casos de uso aporten las funcionalidades requeridas por el usuario, con la necesaria adaptabilidad a cambios futuros. Además, aparte del estudio de la plataforma, de los protocolos para comunicación en red y del sistema de gestión de base de datos requeridos para que el sistema funcione, la arquitectura se desarrolla a partir de los caso de uso significativo, por lo que existe una fuerte relación entre ellos

Iterativo e Incremental: como sugiere Dijkstra, la técnica de dominar la complejidad se conoce desde tiempos remotos: divide et impera (divide y vencerás) (Dijkstra, 1979). De

esta forma RUP divide un proyecto de software en partes más pequeñas o mini proyectos. Cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada es por eso que se dice que son mini proyectos (Jacobson, 2000). En cada iteración se realiza un proceso de gestión de riesgos, garantizando así que los costes por algún problema afecten sólo a la iteración y no al producto completo. RUP define nueve flujos de trabajo, seis ingenieriles y tres de apoyo, a la vez estos flujos se encuentran relacionados de forma bidimensional con cuatro fases. El ciclo de vida de un producto de software en RUP está determinado por las iteraciones y los hitos de cada fase (**Ver Figura 2**).

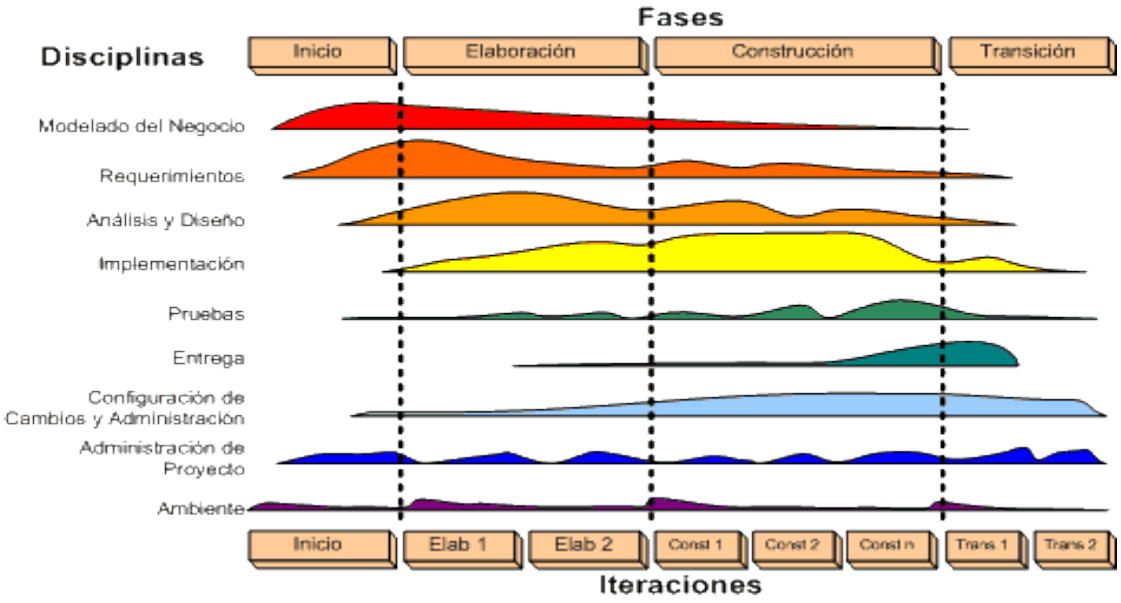

Figura 2. Gráfica bidimensional entre flujos de trabajo y fases de RUP

En RUP, cada ciclo produce una nueva versión del sistema, y cada versión es un producto preparado para su entrega. Consta de su cuerpo de código fuente incluido en componentes que puede compilarse y ejecutarse, además de manuales y otros productos asociados. Sin embargo, el producto terminado no sólo debe ajustarse a las necesidades de los usuarios, sino también a las de todos los interesados, es decir. Toda la gente que trabajara con el producto. El producto software debería ser algo más que el código máquina que se ejecuta. (Jacobson, 2000). Así, a través de la abundante documentación y los artefactos que RUP ofrece, se hace más fácil afrontar futuros cambios en los requisitos, en el sistema operativo, base de datos u otros. De esta forma los mismos desarrolladores o incluso otros desarrolladores podrán partir de las funcionalidades los casos de usos ya descritos, fusionar los cambios y darle forma al sistema. Resulta un proceso menos costoso que si se tiene que rehacer el sistema desde el comienzo

Por otra parte el Proceso Unificado de Desarrollo realiza una mitigación temprana de posibles altos riesgos, sostiene un progreso visible en las primeras etapas, posee una temprana retroalimentación que se ajusta a las necesidades reales, manteniendo la gestión de la complejidad de los proyectos y hace posible que el conocimiento adquirido en una iteración puede aplicarse de iteración a iteración.

El flujo de trabajo de implementación abarca la definición de la organización del código en términos de subsistemas de implementación organizados en capas, la implementación de los elementos de diseño en términos de elementos de implementación (códigos fuentes, ejecutables, entre otros), la prueba de los componentes desarrollados como unidades y la integración de los resultados de los programadores o equipos de programadores, en un sistema ejecutable. Es precisamente, el rol de Programador, el encargado de implementar y probar componentes de acuerdo con los estándares definidos en el proyecto, para la integración en subsistemas más amplios (Rational Software Corporation, 2003). Este flujo de trabajo cuenta con varios roles, entre ellos el rol de implementador. El implementador debe tener conocimiento del paradigma y el lenguaje de programación escogidos, así como dominar el estándar de codificación que más se ajusta a este lenguaje. Además debe conocer con profundidad las potencialidades del IDE de desarrollo y saber aplicarlas a la hora de implementar. Debe dominar la programación utilizando una estructura de soporte definida con bibliotecas y librerías (frameworks) que le puedan ayudar a implementar los componentes que le corresponden. El programador debe dominar los pasos para implementar los elementos de diseño necesarios para obtener componentes ejecutables que proporcionen funcionalidad al sistema. Además necesita saber cómo llevar los patrones modelados en el diseño a la implementación si desea obtener código reusable y legible. Paralelamente debe probar la implementación utilizando herramientas que permitan realizar pruebas de unidad a su código. También el programador debe llevar a cabo prácticas como el Proceso de Software Personal (PSP) que le permitan planificar su trabajo, detectar y corregir errores, y de esta forma lograr una disminución del tiempo en la implementación.

1.3.2 XP Programación Extrema

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con

requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (Informatizate.net.).

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración. El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega *(release)*, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

La metodología se basa en:

• Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.

• Re fabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

• Programación en pares**:** una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

Luego de haber realizado un estudio de ambas metodologías se decide usar RUP, puesto que es necesario obtener los artefactos entregables de RUP como parte del contrato con el cliente. El cliente exige que el proyecto se entregue por iteraciones lo cual hace a esta metodología idónea.

1.4 Herramientas Case

1.4.1 Rational Rose Enterprise Edition

Rational Rose Enterprise Edition es una herramienta CASE que soporta el modelado visual con UML, ofreciendo distintas perspectivas del sistema. Da soporte al Proceso Unificado de Rational (RUP). Algunas de sus características son:

- Diseño dirigido por modelos que redundan en una mayor productividad de los desarrolladores.
- Diseño centrado en casos de uso y enfocado al negocio, que generan un software de mayor calidad.
- Cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables.
- Solo permite trabajar sobre el sistema operativo Windows.
- Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.
- Desarrollo Iterativo.
- Generador de Código.
- Ingeniería Inversa.

1.4.2 Visual Paradigm

Visual Paradigm: Esta herramienta es multiplataforma y está diseñada para desarrollar software con programación orientada a objetos. Visual Paradigm es una herramienta CASE orientada a UML, soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Visual Paradigm genera toda la documentación de lo que se hace cumpliendo con estándares establecidos. Brinda la posibilidad de generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así como obtener diagramas a partir de código. Esta es precisamente una gran ventaja puesto que el sistema será desarrollado en Java. El análisis textual es una técnica útil y práctica para la captura de los requisitos del sistema y la identificación de las clases candidatas, Visual Paradigm es una

de las pocas herramientas CASE que soporta el análisis textual. Tiene disponibilidad en múltiples plataformas y en múltiples versiones.

Esta característica es muy importante pues por ejemplo el Rational Rose, que es una herramienta muy recomendada y además profesional, tiene como desventaja que obliga al usuario a desarrollar sobre el sistema operativo Windows, mientras que el Visual Paradigm está disponible para varios sistemas operativos como Windows, Linux, Unix. Se decidió seleccionar esta herramienta CASE fundamentalmente por esta característica pues el proyecto se va a desarrollar completamente sobre el sistema operativo GNU/Linux siendo una política del país migrar todo el desarrollo hacia plataformas libres. Aunque el Visual Paradigm no es una herramienta libre la Universidad cuenta con las licencias necesarias para desarrollar con la misma.

1.5 Lenguajes de programación

1.5.1 PHP

Es un lenguaje *script* interpretado en el lado del servidor utilizado para la generación de páginas Web dinámicas. Principalmente es usado en la interpretación del lado del servidor (*server-side scripting*), pero puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK. PHP es un lenguaje sencillo, de sintaxis cómoda y similar a la de otros lenguajes como C o C++, es rápido a pesar de ser interpretado, multiplataforma y dispone de una gran cantidad de librerías que facilitan el desarrollo de las aplicaciones. Posee otras características, como son:

- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados extensiones).
- 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.

Desventajas:

- Las herramientas de generación de reporte existentes en este lenguaje son menos potentes que las de otros lenguajes como Java que se explica a continuación.

1.5.2 Java

Java es un lenguaje de desarrollo de propósito general, y como tal es válido para realizar todo tipo de aplicaciones profesionales. A continuación se verán algunas características del lenguaje

Es orientado a objetos:

Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Las plantillas de objetos son llamadas, como en C++, clases y sus copias, instancias.

Java incorpora funcionalidades como la resolución dinámica de métodos. En C++ se suele trabajar con librerías dinámicas (DLL) que obligan a recompilar la aplicación cuando se retocan las funciones que se encuentran en su interior. Este inconveniente es resuelto por Java mediante una interfaz específica llamada RTTI (RunTime Type Identification) que define la interacción entre objetos excluyendo variables de instancias o implementación de métodos. Las clases en Java tienen una representación en el tiempo de ejecución que permite a los programadores interrogar por el tipo de clase y enlazar dinámicamente la clase con el resultado de la búsqueda.

Para ejecutarlo, es necesario un “intérprete”, la JVM (Java Virtual Machine) máquina virtual Java. De esta forma, es posible compilar el programa en una estación UNIX y ejecutarlo en otra con Windows utilizando el intérprete lo que lo hace portable y multiplataforma, puesto que la ejecución no depende ni del Sistema Operativo ni la arquitectura de la máquina pudiendo incluso tener un cliente hasta en un teléfono celular.

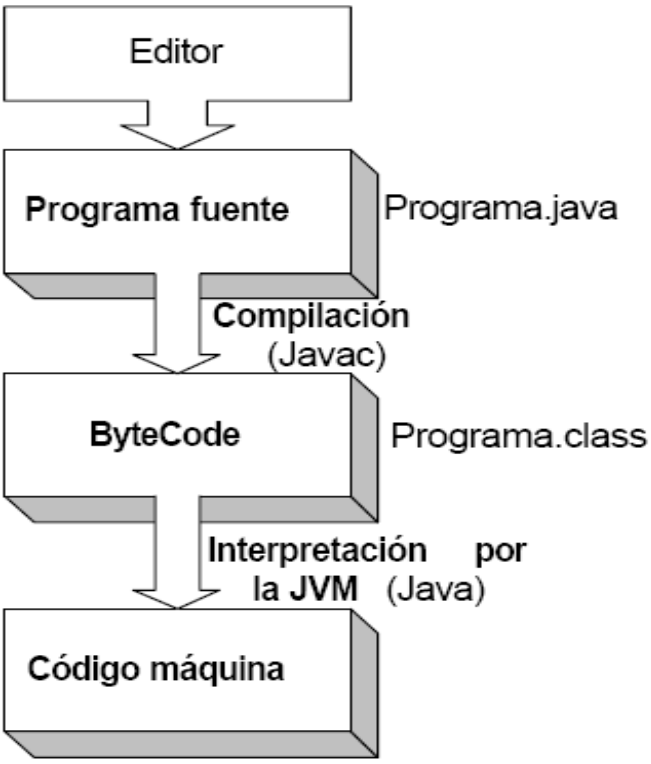

Figura 3. Proceso compilación e interpretación del código Java.

Algunas de las características principales de Java son:

Es Seguro:

Las aplicaciones de Java resultan extremadamente seguras, ya que no acceden a zonas delicadas de memoria o de sistema, con lo cual evitan la interacción de ciertos virus. Java no posee una semántica específica para modificar la pila de programa, la memoria libre o utilizar objetos y métodos de un programa sin los privilegios del núcleo del sistema operativo. Además, para evitar modificaciones por parte de los crackers de la red, implementa un método seguro de autentificación por clave pública. El Cargador de Clases puede verificar una firma digital antes de realizar una instancia de un objeto. Por tanto, ningún objeto se crea y almacena en memoria, sin que se validen los privilegios de acceso. Es decir, la seguridad se integra en el momento de compilación, con el nivel de detalle y de privilegio que sea necesario.

Es robusto:

Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo. Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las

preocupaciones por parte del programador de la liberación o corrupción de memoria. También implementa los arreglos auténticos, en vez de listas enlazadas de punteros, con comprobación de límites, para evitar la posibilidad de sobrescribir o corromper memoria resultado de punteros que señalan a zonas equivocadas. Estas características reducen drásticamente el tiempo de desarrollo de aplicaciones en Java.

Desventajas:

- Los entornos de desarrollo potentes de este lenguaje consumen gran cantidad de memoria RAM.

1.5.3 Conclusiones

Para resumir una comparación entre estos lenguajes se presenta la siguiente gráfica.

En este caso concreto las variables más importantes son la seguridad, la mantenibilidad y la integración.

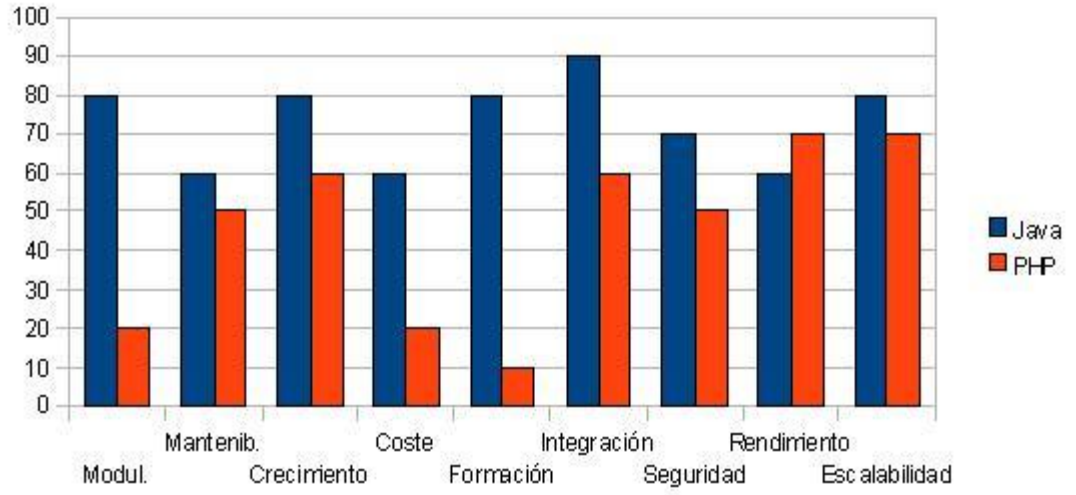

Figura 4. Comparación Java-PHP (López, 2009)

Como se puede apreciar en la gráfica Java posee una gran cantidad de características que superan a PHP: es un lenguaje mucho más seguro, robusto y mantenible. Por esto se decidió adoptar como lenguaje de programación Java.

1.6 Entorno de Desarrollo Integrado (IDE).

1.6.1 NetBeans

El entorno de desarrollo integrado (IDE) NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Existe además un número importante de módulos para extender este IDE NetBeans.

Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en control de versiones y refactorización.

NetBeans Enterprise Pack soporta el desarrollo de aplicaciones empresariales con JEE 5, incluyendo herramientas de desarrollo visuales de SOA, orientación a servicios web, herramientas de esquemas XML y modelado UML.

Sun Studio, Sun Java Studio Enterprise y Sun Java Studio Creator de Sun Microsystems han sido todos basados en el IDE NetBeans. Desde Julio de 2006 éste IDE es licenciado bajo la Common Development and Distribution License (CDDL), una licencia basada en la Mozilla Public License (MPL).

1.6.2 Eclipse

El SDK (Sun Development Kit) de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un entorno de desarrollo integrado (IDE) con un compilador de Java interno y un modelo completo de los archivos fuente, lo que permite técnicas avanzadas de refactorización y análisis de código(Storkel, 2002). La versión actual de Eclipse dispone de las siguientes características:

- Editor de texto.
- Resaltado de sintaxis.
- Compilación en tiempo real.
- Pruebas unitarias con JUnit.
- Control de versiones con CVS.
- Integración con Ant.
- Asistentes (wizards): para creación de proyectos, clases y pruebas.
- Refactorización.

- A través de "plugins" libremente disponibles es posible añadir: Control de versiones con Subversion, además de completamiento de código para ExtJS mediante SPKet y completamiento de XML para Hibernate.

- Integración con Hibernate.

Se decidió utilizar el IDE eclipse, pues el desarrollo del proyecto será sobre la plataforma J2EE, la cual se integra perfectamente con éste. La arquitectura de plugins permite incorporarle múltiples funcionalidades como control de versiones mediante el plugin Subeclipse. Facilita la integración con distintos frameworks de JavaScript como ExtJS, JQuery, Prototype, entre otros.

Para la implementación del software se utilizan herramientas basadas en software libre, lo cual trae como ventaja fundamental que no esté presente un gasto por cuestiones de licenciamientos sin descuidar su calidad.

1.7 El servidor de aplicaciones Apache.

Un servidor de aplicaciones es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general a través de Internet y utilizando el protocolo http. Los servidores de aplicación se distinguen de los servidores Web por el uso extensivo del contenido dinámico y por su frecuente integración con bases de datos. Un servidor de aplicación maneja la mayoría de las transacciones relacionadas con la lógica y el acceso a los datos de una aplicación. La ventaja principal de un servidor de aplicaciones es la facilidad para desarrollarlas, puesto que éstas no necesitan ser programadas y en cambio, se arman a partir de módulos previstos por el servidor de aplicaciones.

Apache -TOMCAT

Apache Tomcat es sin lugar a dudas el proyecto de software libre más famoso escrito en Java. Creado a partir de código donado por SUN Microsystems a la Apache Software Foundation, ha crecido hasta convertirse en la implementación oficial de referencia de Servlets y JSP sustituyendo a la implementación de SUN original. Esto lo convierte, en el contenedor de Servlets sobre el que todos los demás prueban su adhesión a las especificaciones. Aunque siempre se le ha acusado de un rendimiento pobre, el caso es que a partir de las nuevas versiones ha mejorado considerablemente su rendimiento. Puede integrarse sin demasiados problemas con el servidor web Apache y para aplicaciones empresariales grandes con servidores de aplicaciones como JBoss o JOnAS (Foundation, The Eclipse). Se puede integrar con muchos otros productos con licencias propietarias. En estos momentos es el contenedor web que más se está utilizando en servidores de

aplicaciones comerciales, donde gigantes como IBM con su todopoderoso WebSphere han elegido a Tomcat como contenedor web. Es mantenido y desarrollado por miembros de la Apache Software Foundation y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Apache Software License.

1.8 Frameworks de desarrollo.

Los frameworks son arquitecturas de software bien definidas creados tanto para organizar, como para integrar diferentes componentes de un proyecto con la idea de facilitar su desarrollo. El desarrollo de frameworks en la actualidad posee gran aceptación entre los desarrolladores, pues éste brinda la posibilidad de reutilizar el código del diseño y el código fuente.

1.8.1 Spring

El framework Spring facilita a los desarrolladores la implementación del software, promoviendo buenas prácticas de programación. Maneja objetos del negocio y además se integra fácilmente con otros frameworks. El framework Acegi está íntimamente ligado a Spring, el cual se encarga de manejar la seguridad de la aplicación. Éste framework se hizo open source en febrero de 2003. (Walls, 2005)

Tiene el objetivo de facilitar la construcción de aplicaciones en Java, se puede utilizar en cualquier tipo de aplicaciones, no solamente es utilizado en aplicaciones web y es ligero debido al poco impacto que tiene en las aplicaciones. Lo que trae varios beneficios arquitectónicos como son:

- Puede organizar eficazmente la capa intermedia de cualquier aplicación, tanto de escritorio como web.

- Las aplicaciones con Spring son fáciles de testear.

- Facilita el desarrollo de buenas prácticas, reduce el costo de programación en una aplicación.

Spring está basado en el principio de inyección de dependencias, esta técnica hace externo el manejo y la creación de las dependencias de los componentes, logrando así obtener una mayor claridad y limpieza en el código(Harrop, 2005). Como ventajas fundamentales que trae la utilización de Spring están:

❖ Reduce potencialmente el código de enlace entre los diferentes componentes de la aplicación.

❖ Permite la reconfiguración de las dependencias sin necesidad de compilar todo el código.

❖ Disminuye el margen de errores.

❖ Fomenta un buen diseño de la aplicación.

La principal característica que posee Spring es el soporte a la programación orientada a aspectos (AOP), la cual es una tecnología del momento en el mundo de Java, esto posibilita el manejo de trazas, de la seguridad y de las transacciones. Sin lugar a duda es una buena opción para el desarrollo de aplicaciones, pues como se puede observar facilita un bajo acoplamiento y permite obtener un código limpio y claro. (Johnson, 2005)

1.8.2 EJB

Los Enterprise JavaBeans (también conocidos por sus siglas EJB) son una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE (ahora JEE 5.0) de Oracle Corporation (inicialmente desarrollado por Sun Microsystems). Su especificación detalla cómo los servidores de aplicaciones proveen objetos desde el lado del servidor que son, precisamente, los EJB.

A continuación se muestra un estudio comparativo entre los frameworks de desarrollo Spring y EJB.

Característica	Spring	EJB
Manejo de transacciones	Solo se puede usar JTA Transaction Manager	Se puede usar tanto JTA Transaction Manager como el manejador de transacciones del ORM (Hibernate, JDO, JDBC, OJB)
Persistencia de entidades	Define su propio manejo de persistencia, posibilidad de emplear anotaciones en ORM, EJB QL y sentencias SQL nativas, además de integración	Usa implementaciones ORM de terceros como Hibernate, IBATIS, JDO, OJB.

	con Hibernate	
Seguridad	Soporta seguridad declarativa por medio de anotaciones de metadatos y declaraciones en el descriptor de despliegue	Provee integración con la solución Open Source Acegi Security Framework, el mismo que soporta seguridad declarativa basada en el uso de IoC y AOP.
Precio	Primariamente los productos son pagos, la implementación de EJB de JBOSS es gratuita	Productos Open-Source son gratuitos.
Flexibilidad de servicios	Depende de la implementación de EJB. Si el servidor provee una estructura modular entonces sólo se requieren los servicios que este puede usar.	Cualquier servicio puede ser ensamblado usando un archivo XML de configuración

1.8.3 Hibernate

Hibernate es una herramienta que realiza el mapeo entre el mundo orientado a objetos de las aplicaciones y el mundo entidad-relación de las bases de datos en entornos Java. Se encarga de gestionar la capa de Acceso a Datos de un software y realiza la transición de los datos de un modelo relacional a un modelo orientado a objetos. Está basado en la implementación del patrón DAO, pues crea una capa separada que se ocupa del acceso a datos con total independencia del gestor y la base de datos, lo que brinda la posibilidad de trabajar con varios gestores y bases de datos dentro de la misma aplicación (Bauer, 2005). Soporta todos los sistemas gestores de bases de datos SQL y se integra sin restricciones con los contenedores web más conocidos. Constituye un motor de persistencias que implementa múltiples funcionalidades y fue liberado bajo la licencia GPL.

Tiene como principales características las siguientes:

- Soporta el paradigma de orientación a objetos de una manera natural, como es caso de utilizar herencia, composición, polimorfismo y las librerías de colecciones de Java.
- Posee lenguaje de consultas, lo que proporciona una independencia del SQL de cada base de datos ya sea para la recuperación como para el almacenamiento de objetos.
- Permite una gran variedad de mapeos para colecciones y objetos dependientes.
- Posee una cache de dos niveles, un alto rendimiento y puede ser utilizado en un clúster.

Presenta un mecanismo de persistencia transparente lo que trae consigo que los objetos desconocen su capacidad de persistencia, no necesita un contenedor de aplicaciones, son fáciles de manejar y pueden ser usados para transportar los datos a cualquier capa de la aplicación. Estas son algunas de las razones por las que ha ganado en preferencia y popularidad entre los grandes desarrolladores de aplicaciones (Heudecker, 2003). Posee además una amplia documentación.

1.8.4 JDO

El API Java Data Objects (JDO) proporciona una forma estándar y sencilla de conseguir la persistencia de objetos en una Base de Datos Relacional con la tecnología Java. En líneas generales, JDO utiliza una combinación práctica de metadatos XML y bytecodes mejorados para hacer más sencilla la complejidad y al sobrecarga, comparado con otras tecnologías de unión de objetos.

JDO resume en unas interfaces de programación los servicios necesarios para alcanzar la persistencia de objetos Java sobre distintos sistemas de gestión de datos, de forma uniforme, sin necesidad de conocer los mecanismos de persistencia realmente utilizados.

La transparencia de datos de JDO, no necesita alterar los fuentes Java, pero requiere de la definición de la correspondencia datos-objetos y la inclusión del código necesarios para las clases cuyos objetos serán persistentes.

JDO propone para ello el uso de la meta información necesaria en archivos XML, separados del código, y la posibilidad del uso de un procesador de clases compiladas, para alcanzar un buen nivel de persistencia de los datos. (Atiwiki).

JDO no permite que objetos remotos se hagan persistentes algo que es necesario en el proyecto a desarrollar lo que hace que no sea un candidato idóneo para seleccionar.

1.8.5 ExtJS

ExtJS es un framework de JavaScript para el desarrollo de grandes aplicaciones web.

Empezó siendo un conjunto de librerías y extensiones para YUI (Yahoo! User Interface) que agregaban una API (Application Programming Interface) sencilla y fácil de usar, así como una serie de componentes personalizables ideales para la construcción de aplicaciones web con un ambiente renovado dentro de las aplicaciones de la web 2.0.

Entre las principales características podemos citar:

- Ext provee una interfaz de usuario fácil de usar, con un gran parecido a las interfaces de las aplicaciones de escritorio, lo cual permite a los desarrolladores concentrarse en la funcionalidad de la aplicación web.
- Provee un mecanismo para la compatibilidad entre navegadores que posibilita abstraerse de los distintos navegadores haciendo menos engorroso el desarrollo de aplicaciones web.
- Permite la interacción de usuario con el navegador a través de manejadores de eventos que responden a los mismos.
- Posibilita una fácil comunicación con el servidor sin tener que refrescar la página a través de AJAX (Frederick S., 2008).
- Posee una amplia documentación.

Todas estas características han hecho de este framework el líder en el desarrollo de aplicaciones en JavaScript.

1.8.6 Dojo

Es más que un conjunto de herramientas, es un Framework que contiene APIs (Applications Program Interface) y widgets (controles) para soportar el desarrollo de aplicaciones Web. Contiene un sistema de empaquetado inteligente, widget APIs, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX. Dojo resuelve asuntos de usabilidad comunes como la navegación y detección del navegador, soportar cambios de URLs para luego regresar a ellas (bookmarking), y la habilidad de degradar cuando AJAX/JavaScript no es completamente soportado en el cliente. Proporciona una gama más amplia de opciones en una sola biblioteca y hace un trabajo muy bueno que apoya los nuevos y viejos navegadores.

Por lo planteado anteriormente se propuso que se apliquen los siguientes frameworks, los cuales brindan ciertas facilidades y servicios para gestionar las funcionalidades de cada una de las capas en las que está organizada la aplicación.

Para la implementación de la capa de Negocio se decidió utilizar Spring por todas las ventajas mencionadas anteriormente que posee, las cuales lo hacen el candidato idóneo para el desarrollo del negocio de la aplicación.

Como actualmente en las aplicaciones informáticas, el acceso a datos representa un serio problema, Hibernate fue el seleccionado para implementar la capa de Acceso a Datos pues proporciona una solución factible a dicho inconveniente.

Para el desarrollo de la interfaz se usará ExtJS pues permite crear aplicaciones Web con un entorno parecido al de escritorio permitiendo al programador concentrarse en las funcionalidades de la aplicación. Posee una amplia documentación y presenta una interfaz mucho más atractiva que la de otros frameworks de JavaScript.

1.9 Herramientas de generación de reportes

1.9.1 PATDSI

PATDSI es un paquete de herramientas para la ayuda a la toma de decisiones. El mismo es concebido como una plataforma Web única de inteligencia de negocio, que integra en sí las funcionalidades más recurrentes y específicas necesarias para la toma de decisiones en diferentes contextos. Este paquete incluye un Generador de Reportes Dinámicos.

Dicho Generador aún se encuentra en fase de desarrollo en nuestra Universidad y no presenta algunas funcionalidades necesarias como dar el formato adecuado a la agrupación de datos y no permite la generación de subreportes aún. Los reportes se elaboran en tiempo de diseño por lo que también está orientado al programador y no al usuario pues no permite la modificación dinámica del diseño del reporte.

El generador de Reportes Dinámicos está instalado en diferentes proyectos de la Universidad a de las Ciencias Informática (UCI) como son:

- El ERP Cubano.
- Informatización
- Proyecto Alfa Omega (Facultad 8)
- Proyecto Aduana (Facultad 4)
- Proyecto Aduana (Facultad 15)

1.9.2 JasperReport

Es una herramienta para la generación de informes. Está implementada en Java, es de código abierto y fue desarrollada por Teodor Danciu para facilitar el agregar capacidades de reporte a las aplicaciones Java. No es una herramienta por sí sola ya que no se puede instalar. Para utilizarla es necesario añadirla a las aplicaciones Java por medio de la inclusión de su librería al classpath (Variable de entorno que permite a la JVM conocer dónde localizar sus clases) de la aplicación, la cual no tiene ningún tipo de dependencia con las librerías de Java, lo que posibilita utilizar JasperReport también para aplicaciones de escritorio. Permite incluir datos de texto, imágenes, gráficos, subreportes, tablas, entre otras características básicas en los reportes, para que estos tengan un aspecto profesional. Las principales características que provee son:

1- **Permite una diagramación flexible de los reportes:** los reportes se pueden dividir en secciones opcionales que son: título, encabezado de página, una sección para los detalles del reporte, pie de página y una sección de resumen que aparece al final del reporte.

2- **Permite que los desarrolladores le surtan datos en varias formas:** los desarrolladores pueden pasar datos a los reportes por medio de parámetros. Estos parámetros pueden ser instancia de cualquier clase de Java.

3- **Pueden generar sub-reportes:** permite la creación de subreportes dentro de los reportes lo que facilita el diseño.

4- **Los reportes son capaces de presentar los datos de manera textual o a través de gráficos:** son capaces de mostrar los datos que le son pasados y pueden generar o calcular con esos datos otros datos de forma dinámica y mostrarlos.

5- **Pueden generar marcas de agua:** permite generar textos o imágenes de fondo para utilizarlo como marcas de agua con el propósito de identificar el reporte o simplemente por motivos de seguridad.

6- **Se pueden exportar los reportes a una multitud de formatos:** los reportes generados pueden ser exportados a una multitud de formatos como PDF, XLS, RTF, HTML, XML, CVS y texto plano. (Heffelfinger, 2009)

Se integra perfectamente con JFreeChart que es una librería libre para la generación de todo tipo de gráficas.

Se pueden crear estilos predefinidos para que sean reutilizados en varios reportes aumentando la productividad y dando mayor facilidad al mantenimiento de los reportes, quedando separado el diseño del reporte con el formato de presentación.

Esta herramienta cuenta con una amplia y detallada documentación además de múltiples ejemplos donde se muestran claramente lo que se puede hacer con la misma y el cómo hacerlo,lo que es muy importante para conocer a fondo sus capacidades.

Requerimientos de JasperReport:

- Se requiere tener instalado en el equipo el JDK 1.3 o posterior. No basta con tener instalado el J2RE (Run Time Environment).

1.9.3 BIRT

BIRT es un sistema para la generación de reportes para aplicaciones Web basado en eclipse. Posee dos componentes principales: un diseñador de reportes basado en eclipse y un motor de generación de reportes para agregar a la aplicación. Ofrece además un motor de gráficos que permite agregar gráficas a los reportes.

Con BIRT se pueden crear una gran variedad de reportes al igual que con JasperReport pero no presenta una documentación ni una guía de ejemplos como la de JasperReport lo que deja a imaginación del usuario algunas funcionalidades que son necesarias para cumplir los requisitos del cliente.

Analizando las posibilidades de cada una de las herramientas se propone el JasperReports puesto que satisface las necesidades de los reportes deseados, posee una amplia documentación y una gran variedad de ejemplos que sirven para ilustrar las potencialidades del mismo, además de ayudar en la productividad al usarlo.

1.10 Herramientas para diseñar los reportes.

A continuación se muestra un análisis de dos potente herramientas para diseñar reportes a generar por JasperReport

Herramienta	Licencia	Características	Desventajas
DynamicJasper	GPL	-Mantiene las principales características del JasperReport ya que trabaja directamente con él. -Añade cierto grado de dinamismo	-Deficiente robustez y eficacia para reportes complejos. - Baja productividad

		a los reportes en tiempo de ejecución. -Reduce la complejidad del JasperReport para reportes simples y de mediana complejidad.	
Ireport	GPL	- Herramienta visual de diseños de informes para JasperReport potente, intuitiva y fácil de usar. -Requiere el JDK 1.4 o superior. -Permite editar informes complejos con gráficas, imágenes y subinformes. -Permite obtener una vista previa del reporte con datos reales en todos los formatos soportados por JasperReport. -Permite el diseño de reportes mediante el arrastre de componentes hacia el área de trabajo. -Tiene asistentes para las plantillas. -Soporta internacionalización. -Soporta JDBC. -Permite crear estilos para el formato de los reportes.	-No es integrado a los IDE Java.

Mediante el estudio de estas herramientas y analizando sus principales características y desventajas se decidió utilizar para diseñar las plantillas de los reportes la herramienta Ireport. Es una herramienta bajo licenciamiento libre, es muy usada actualmente en el desarrollo de los reportes en grandes industrias de software. Posee una interfaz muy sencilla abstrayendo al usuario del lenguaje XML que representa el diseño final del reporte.

1.11 Conclusiones

En este capítulo se hizo una revisión bibliográfica de los sistemas automatizados compuestos por módulos de reportes, se abordaron los lenguajes, entornos de desarrollo integrado y frameworks a utilizar. Se hizo una caracterización de las herramientas de generación y diseño de reporte, comparando y seleccionando en cada caso la más adecuada y de esta forma obtener un producto con los costes de tiempo y calidad requeridos, además de que responda a las necesidades del cliente.

Se trabajó utilizando las referencias bibliográficas necesarias para garantizar la autenticidad de la información expuesta en el mismo, así el lector podrá auxiliarse de las diferentes fuentes de información consultadas en caso de que requiera mayor información acerca del tema expuesto.

Capítulo 2: Propuesta de la solución

2.1 Introducción

En este capítulo se abordará en la solución desarrollada para la generación de los reportes, se tomará como punto de partida el diagrama de Casos de Uso (CU) y los prototipos de interfaz elaborados durante los flujos previos. Se dará una explicación de la elaboración de algunas de las plantillas y de los elementos que la conforman.

2.2 Descripción de la situación actual

Actualmente en la República Bolivariana de Venezuela se utiliza un sistema llamado Nueva Etapa para la gestión del presupuesto por proyectos, dicho sistema crea reportes falsos de seguimiento de los proyectos debido a que no es flexible a cambios que pueden ocurrir durante la ejecución de los proyectos. Muchos reportes se hacen por vías informales como el teléfono o el correo, por lo que surge la necesidad de implementar un sistema que brinde toda la información necesaria en los reportes que en él se generen.

2.3 Estándar de Codificación

Los estándares, estilos o convenciones de codificación son importantes para los programadores debido a que ofrecen:

Extensibilidad: La facilidad con que se adapta el software a cambios de especificación. Un buen estilo de código fomenta programas que no sólo resuelven el problema, sino que también reflejan claramente la relación problema/solución. Esto tiene como efecto que muchos cambios simples en el problema reflejen de forma obvio los cambios a hacer en el programa.

Verificabilidad: La facilidad con que pueden comprobarse propiedades de un sistema. Si el estilo de código hace obvia la estructura del programa, eso ayuda a verificar que el comportamiento sea el esperado.

Reparabilidad: La posibilidad de corregir errores sin demasiado esfuerzo.

Capacidad de evolución: La capacidad de adaptarse a nuevas necesidades.

Comprensibilidad: La facilidad con que el programa puede ser comprendido.

El estándar utilizado en el proyecto, fue la convención para el lenguaje de programación Java brindada por **Sun Microsystems**. A continuación se muestra una tabla con las convenciones de nombres pertenecientes a este estilo de codificación.

Tipos de identificadores	**Reglas para nombres**	**Ejemplos**
Paquetes	El prefijo del nombre de un paquete se escribe siempre con letras ASCII en minúsculas, y debe ser uno de los nombres de dominio de alto nivel, actualmente *com*, *edu*, *gov*, *mil*, *net*, *org*, o uno de los códigos ingleses de dos letras que identifican cada país como se especifica en el ISO Standard 3166, 1981. Los subsecuentes componentes del nombre del paquete variarán de acuerdo a las convenciones de nombres internas de cada organización. Dichas convenciones pueden especificar que algunos nombres de los directorios correspondan a divisiones, departamentos, proyectos o máquinas.	ve.sinapsis.dao
Clases	Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Intentar mantener los nombres de las clases simples y descriptivas. Usar palabras completas, evitar acrónimos y abreviaturas (a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML).	Class FichaProyecto;
Interfaces	Los nombres de las interfaces siguen la misma regla que las clases.	Interface FichaDao; Interface ContratoDao;
Métodos	Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula.	crear(); cargarPorId();

Variables	Excepto las constantes, todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman (si son compuestas) empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres subguión	Integer i; String s; Double d; DfichaProyecto fichaProyecto;

2.4 Elementos previos al diseño

Para llevar a cabo el diseño e implementación es necesario basarse en algunos elementos de fases anteriores. A continuación se muestra el diagrama de paquetes y los diagramas de Casos de Uso correspondientes del módulo en cuestión.

2.4.1 Diagrama de casos de uso

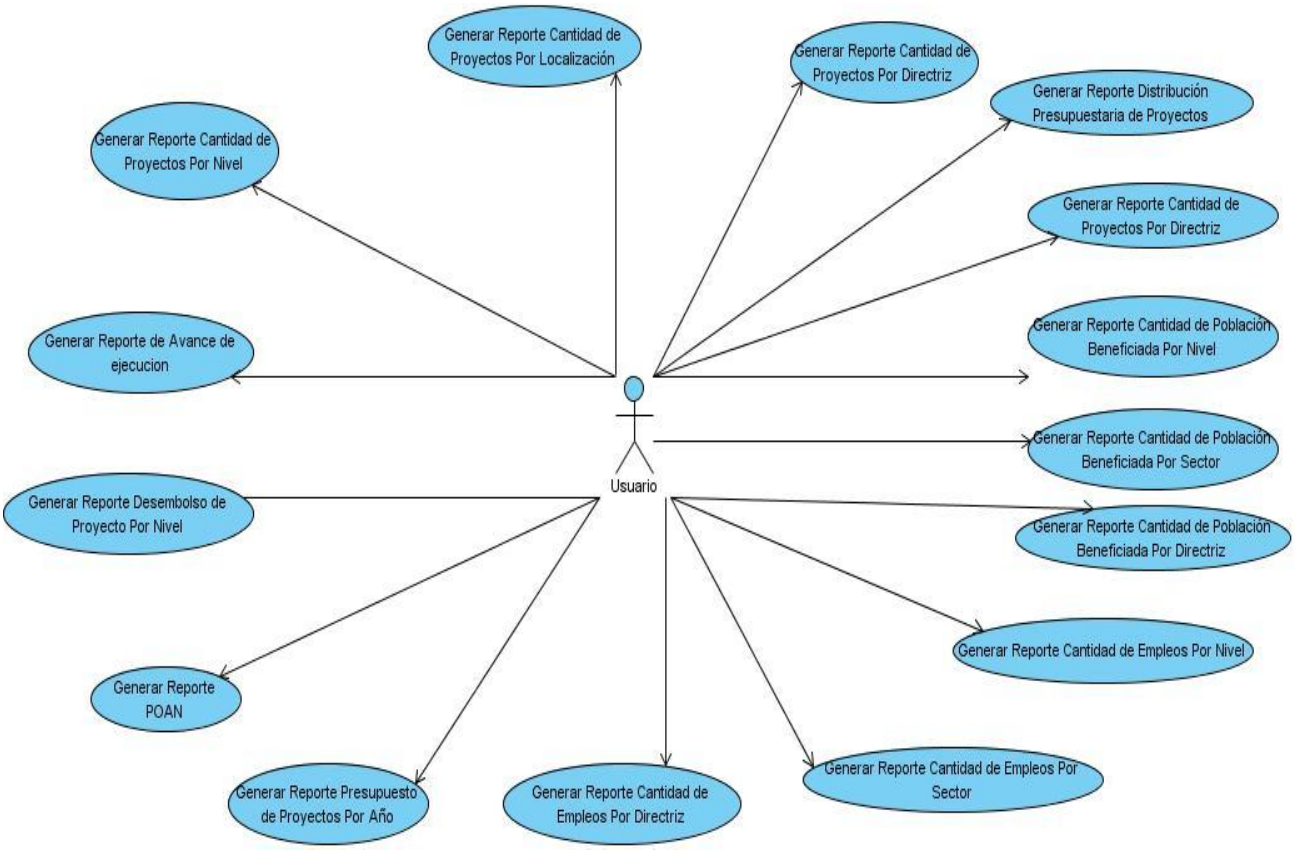

Figura 5. Diagrama de Casos de Uso

2.5 Convenciones de Archivos y Paquetes

El proyecto Web que se creará en el Eclipse para el desarrollo del sistema será "web_sinapsis", al igual que la URL para acceder al mismo. Todas las clases, ficheros XML, páginas JSP y otros ficheros del sistema estarán agrupados en una estructura de carpetas y paquetes. Las clases y ficheros de recursos se encontrarán dentro de la carpeta **src**, haciendo uso de las convenciones de nombrado de paquetes y adaptándolas al sistema. Quedaría como paquete principal: ***ve.sinapsis.*** A partir de este nivel se agregaría la separación por subsistemas o módulos: ***ve.sinapsis.reportes.***

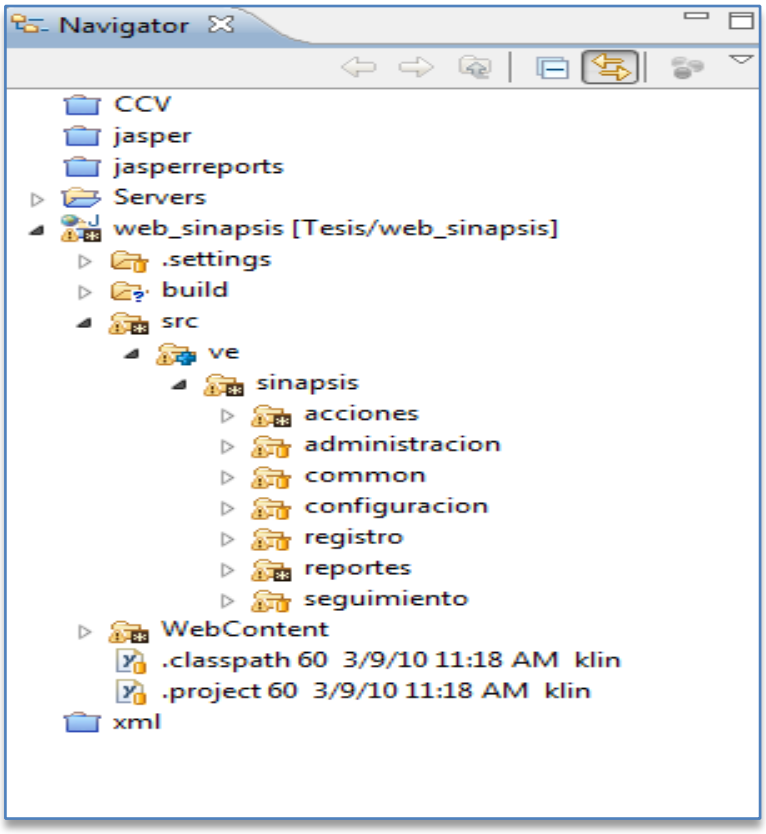

Figura 6. Organización de paquetes y subsistemas

Dentro de cada subsistema se agregaría la separación de las clases por capas de cada uno de ellos, en este caso se muestra como se vería en el subsistema reportes. ***ve.sinapsis.reportes.web*** (Tiene los controladores) ***ve.sinapsis.reportes.bussines (****Contiene las clases que ejecutan alguna lógica de negocio como es el caso de la clase exportadora****) ve.sinapsis.reportes.configuration*** (ficheros xml de configuración de Spring)

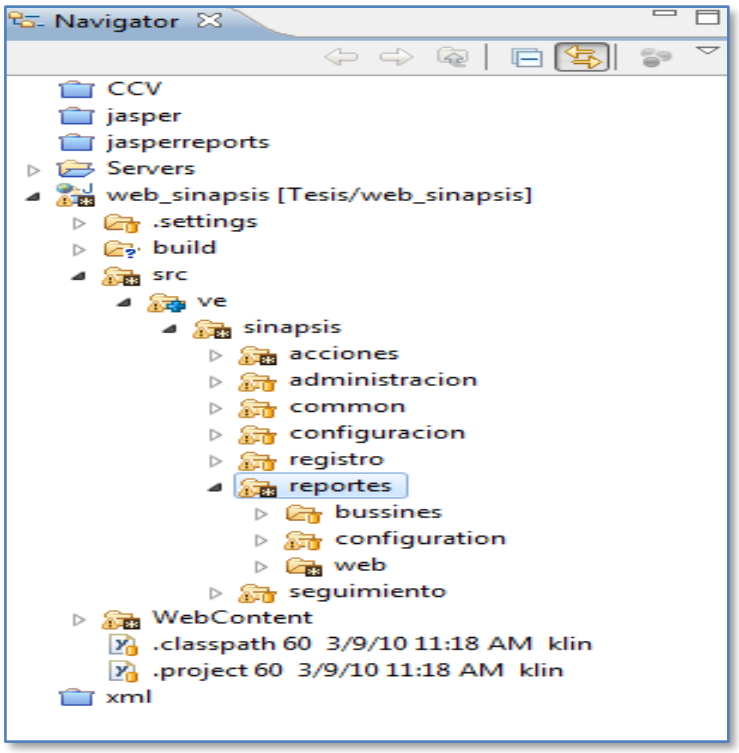

Figura 7. Separación de las clases del subsistema de reportes

2.6 Diagrama de paquetes

A continuación se muestra el diagrama de paquetes de la solución. Un diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema.

Los Paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes.

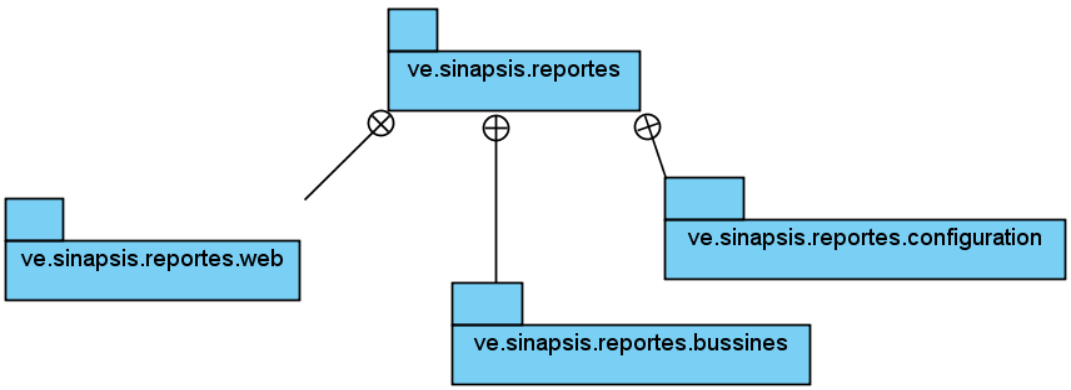

Figura 8. Diagrama de Paquetes

Para comprender esta estructura es necesario conocer el ciclo de vida de las peticiones web en el framework Spring MVC, por lo que será presentada a continuación:

Ciclo de Vida de una petición Web

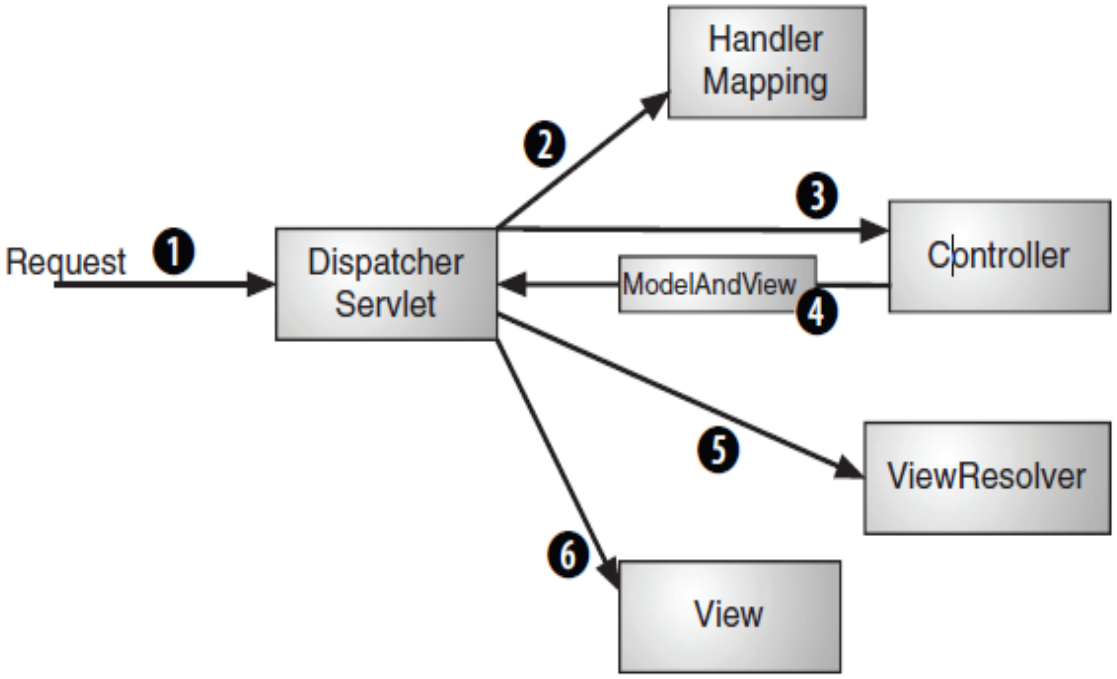

Figura 9. Ciclo de vida de una petición Web en Spring MVC

Donde primero se detiene una petición Web en Spring MVC es en el **DispatcherServlet** de Spring, controlador frontal por el cual son analizadas todas las peticiones. Un controlador frontal es un patrón común en aplicaciones Web donde se delega la responsabilidad a otros componentes para que realicen el procesamiento real. En Spring MVC, la clase DispatcherServlet es dicho controlador.

DispatcherServlet envía el trabajo de la petición a un controlador el cual escoge consultando uno o más **HandlerMapping.** Los **HandlerMapping** son clases que provee el framework Spring para mapear las url solicitadas con los controladores. Estos ficheros de configuración están ubicados en el paquete **ve.sinapsis.reportes.configuration** y los controladores dentro del paquete **ve.sinapsis.reportes.web**. A continuación se muestra el fichero sinapsis-reportes-web-context.XML donde se configura el objeto **HandlerMapping** del módulo el cual es de tipo **SimpleUrlHandlerMapping** que basa el mapeo en la url solicitada.

```
<bean id="mappingReportes"
    class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <props>
        <!-- Report Controllers -->
            <prop key="/distribucionPresupuestaria.htm">DistribucionPresupuestariaController</prop>
            <prop key="/empleosPorDirectriz.htm">EmpleosPorDirectrizController</prop>
            <prop key="/empleosPorNivel.htm">EmpleosPorNivelController</prop>
            <prop key="/empleosPorSector.htm">EmpleosPorSectorController</prop>
            <prop key="/poan.htm">POANController</prop>
            <prop key="/poblacionPorDirectriz.htm">PoblacionPorDirectrizController</prop>
            <prop key="/poblacionPorNivel.htm">PoblacionPorNivelController</prop>
            <prop key="/poblacionPorSector.htm">PoblacionPorSectorController</prop>
            <prop key="/presupuestoPorAgno.htm">PresupuestoPorAgnoController</prop>
            <prop key="/proyectosPorDirectriz.htm">ProyectosPorDirectrizController</prop>
            <prop key="/proyectosPorLocalizacion.htm">ProyectosPorLocalizacionController</prop>
            <prop key="/proyectosPorNivel.htm">ProyectosPorNivelController</prop>
            <prop key="/proyectosPorSectores.htm">ProyectosPorSectoresController</prop>
            <prop key="/reporteDesembolso.htm">ReporteDesembolsoController</prop>
        </props>
    </property>
    <property name="order" value="25" />
</bean>
```

En el controlador se realiza el procesamiento o se delega a otros objetos del negocio, los cuales están en **ve.sinapsis.reportes.bussines** y finalmente encapsula el nombre de la vista junto con los datos que se deseen mostrar al usuario encapsulados ambos en un objeto de tipo **ModelAndView.** Con el nombre de la vista mediante un **ViewResolver** se encuentra la **Vista** real a mostrar.

2.7 Modelo de Datos

El modelo de datos describe el comportamiento lógico y físico de los elementos persistentes de utilidad para dar soporte de información al sistema.

El modelo de datos representa toda la información persistente que se maneja en el proyecto. Para llevar a cabo los reportes es necesario el dominio total del mismo pues los datos se encuentran distribuidos en los diferentes módulos y no solo en el que se registra el proyecto, por ejemplo el estado del proyecto y los por cientos de ejecución física y financiera se encuentran en el módulo de Seguimiento y Control, los datos correspondientes a las trazas se encuentran en el módulo de Administración, etc.

A continuación se muestra el modelo de datos del sistema:

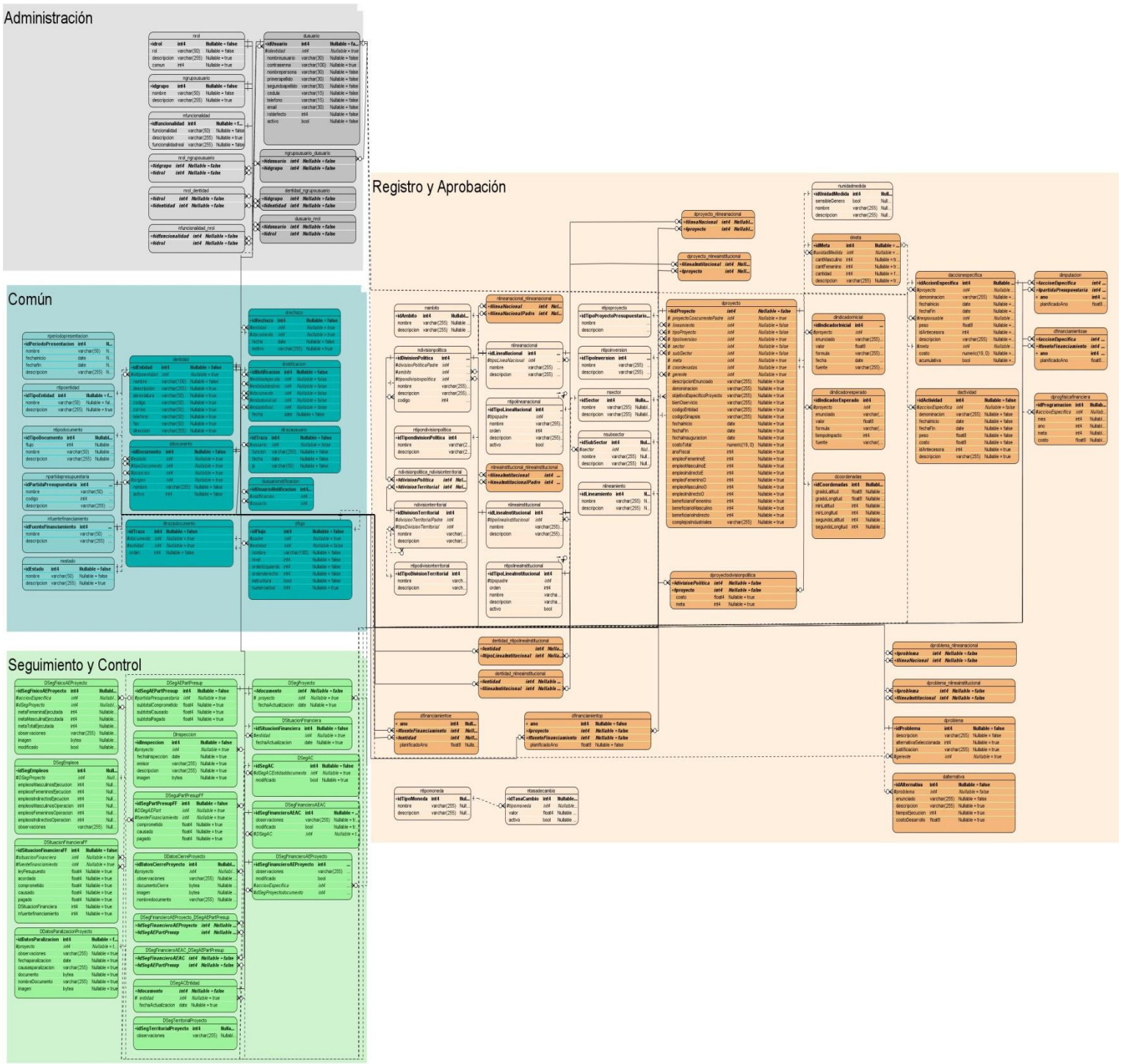

Figura 10. Modelo de datos

2.8 Diagramas de clases

Para realizar el diseño de las clases correspondientes a los Casos de Uso es necesario contar con las descripciones de los Casos de Uso además de los prototipos de interfaz y el modelo de negocio realizados en flujos anteriores.

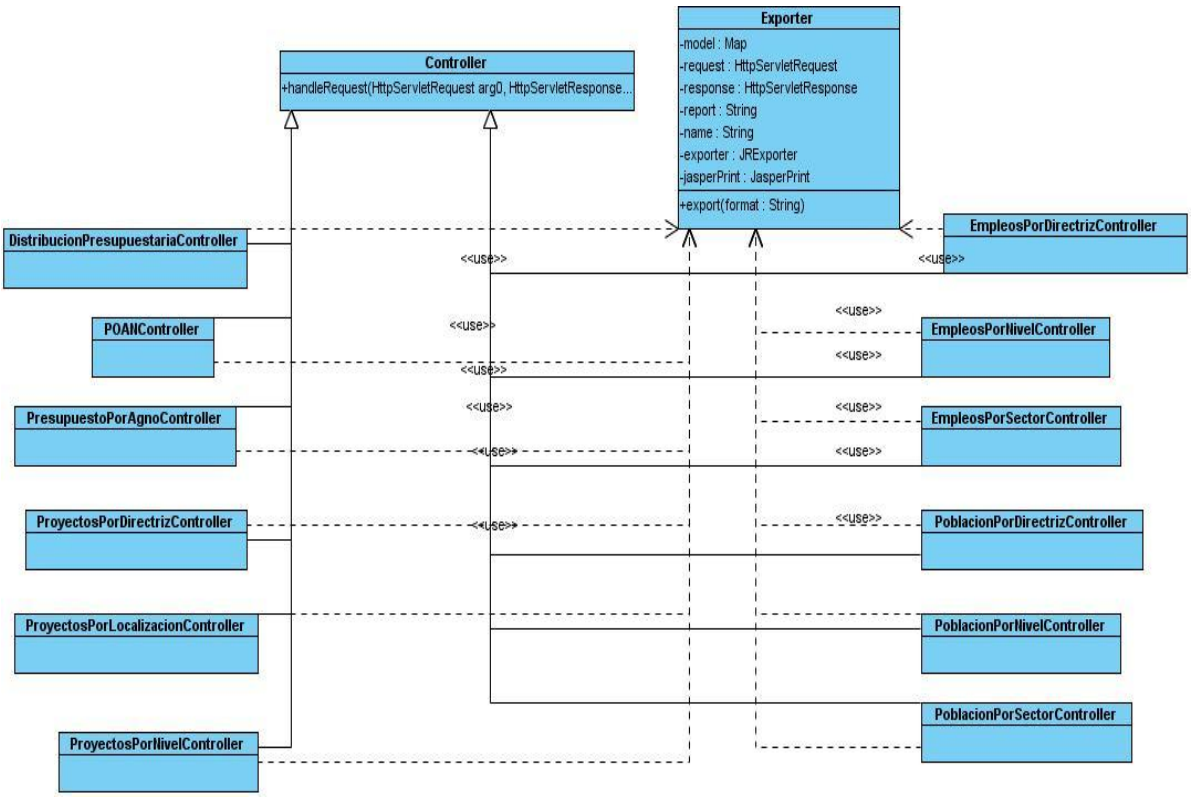

Figura 11. Diagrama de clases

Una vez conocido el ciclo de las peticiones Web en Spring MVC y las características del JasperReport para la generación de reportes se crea solamente un controlador para cada reporte. Dicho controlador se apoya en la clase **Exporter** para presentar la información en los formatos deseados. Se usa una clase auxiliar para la exportación y no el mecanismo de integración que provee Spring para la exportación puesto que hasta la versión actual del framework no permite pasarle parámetros al exportador.

De las diferentes vías permitidas para el llenado de los informes se usan las consultas en lugar de las clases especiales **DataSource**. Las consultas se escribirán en el lenguaje “hql” proporcionado por hibernate lo cual está soportado por JasperReport.

2.8.1 Diagrama de clases de la vista

Como se expresó en el capítulo anterior la vista será implementada usando el framework de JavaScript "ExtJS". A continuación se muestra el diseño de clases de la interfaz que proveerá el sistema. El mismo está implementado solamente en JavaScript posibilitando una fácil migración hacia cualquier framework de JavaScript.

JavaScript es un lenguaje interpretado que no soporta clases, ni posee mecanismos de Programación Orientada a Objetos. Es un lenguaje basado en objetos, funciones y variables, pero el comportamiento de dichas funciones, así como el de los objetos **prototype**, permite simular los mecanismos de la POO como la herencia, el polimorfismo, etc.

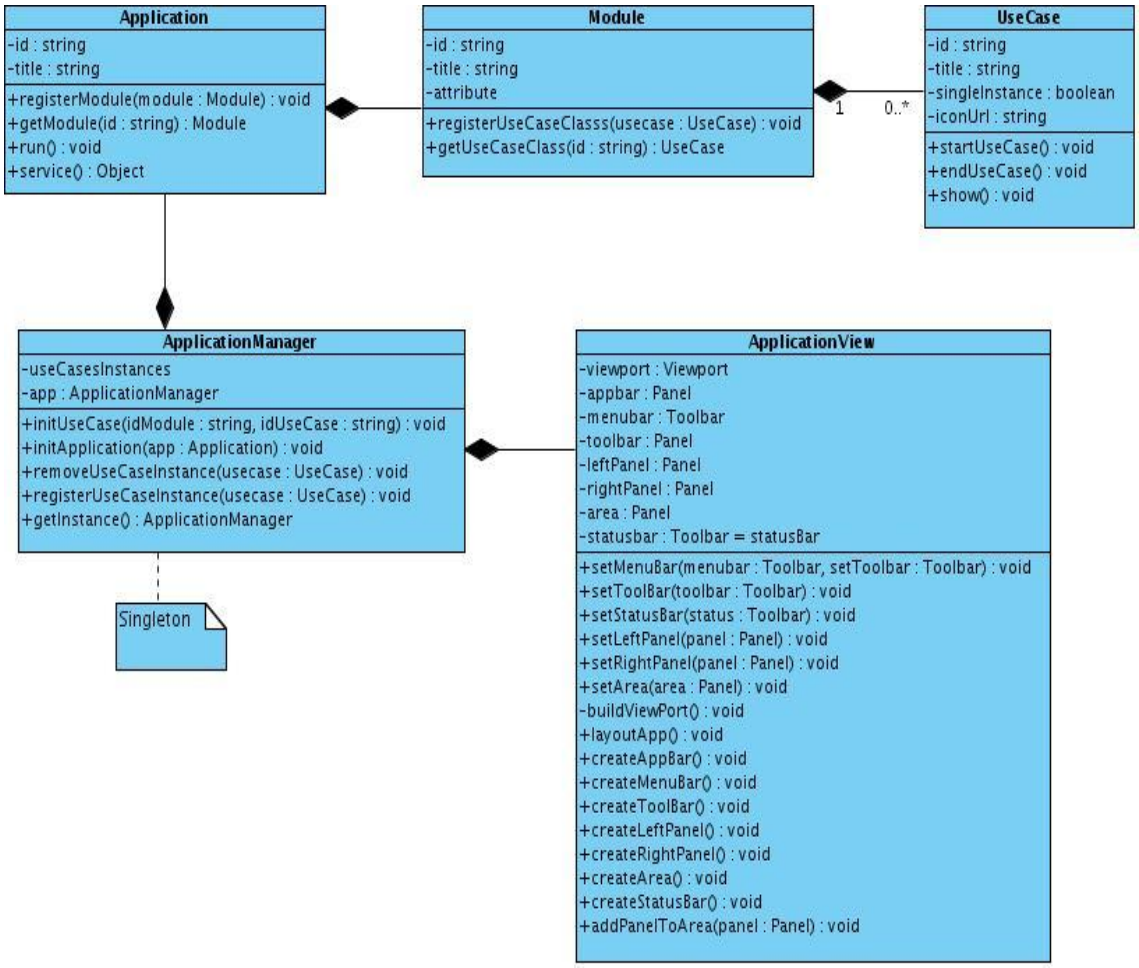

Figura 12.Diagrama de clases de la interfaz

Como se aprecia el diseño se basa en 5 clases fundamentales:

UseCase: Define y encapsula la representación de un Caso de Uso. Esta clase es una representación abstracta de un caso de uso. Cada caso de uso debe devolver su representación visual una vez haya sido inicializado. Se debe heredar de dicha clase si se necesita implementar un caso de uso concreto.

Module: Define la representación de un módulo como un conjunto de casos de usos. Esta es la clase contenedora de los distintos casos de uso que dicho módulo posea. De los casos de usos solo se registra la clase y sólo se instancian los mismos cuando sean requeridos, lo que permite un mejor aprovechamiento de la memoria.

Application: Define la representación de la Aplicación como un conjunto de subsistemas (módulos). Esta clase contiene todos los módulos de una aplicación.

ApplicationView: Define la representación de la interfaz visual de la aplicación. Es una clase abstracta de la cual se debe heredar creando una vista concreta. (ExtJS, Prototype, JQuery, etc).

ApplicationManager: Gestiona las propiedades en tiempo de ejecución de la aplicación. Esta clase es la que gestiona la comunicación entre la aplicación y la vista. La misma es la encargada de crear la vista, inicializar y finalizar los casos de usos según sea requerido.

Este conjunto de clases gestiona todo el núcleo de la vista, su implementación en JavaScript lo hace independiente del framework que se quiera usar.

A través de este conjunto de clases podemos crear nuestra aplicación, los módulos que contiene la misma, así como los casos de usos relacionados con cada módulo. Los módulos sólo registran las clases de los casos de uso, los cuales sólo se serán instanciados cuando sean requeridos. La clase **ApplicationManager** es la encargada de inicializar los casos de usos y finalizar los mismos. Cada caso de uso a su vez es encargado de gestionar su propia representación visual la cual será adicionada a la vista de la aplicación una vez inicializado el caso de uso y se eliminará de la vista una vez que concluya.

2.9 Diseño de las plantillas

Las plantillas definen la ubicación y el formato de la información a mostrar en el documento que el proceso de llenado del reporte produce. Para la creación de las plantillas se usará la herramienta iReport.

El proceso mediante el cual se produce un reporte en el JasperReport es el siguiente:

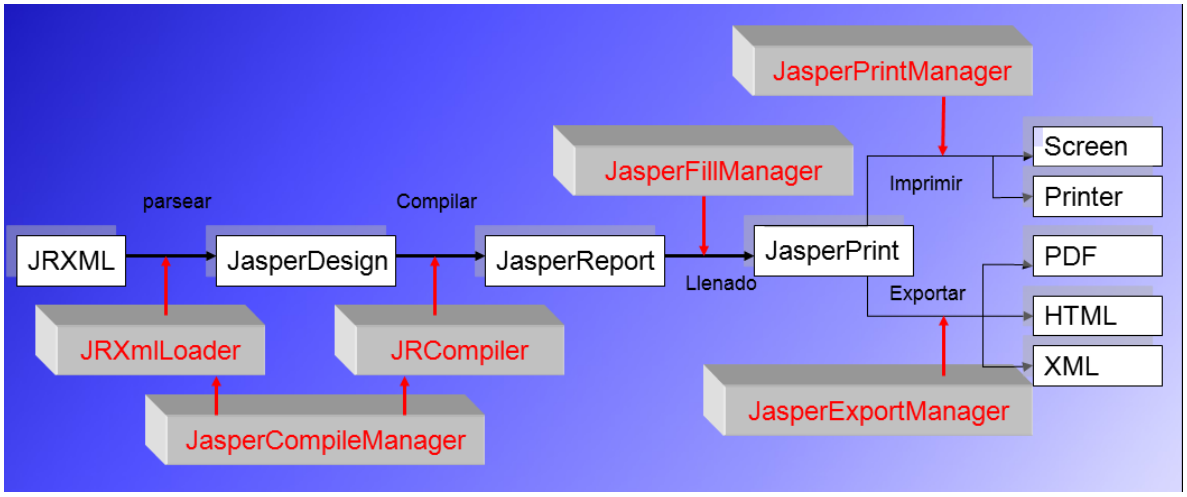

Figura 13. Ciclo de vida de un reporte

Inicialmente se crea la plantilla del reporte que no es más que un XML con la definición del mismo. Este XML puede ser cargado por un objeto JasperDesign mediante el API de JasperReport (JRXmlLoader). Tanto el archivo XML como el objeto JasperDesign pueden ser compilados a través de la clase JasperCompileManager, la cual produce un archivo de salida con extensión .jasper. El proceso de compilado no necesariamente debe almacenar un archivo para su salida, puede trabajar además con objetos de tipo JasperReport, aunque la integración de JasperReport con Spring permite desplegar las plantillas en XML, el mismo se hará de manera pre compilada para ganar en rendimiento. Luego de tener un compilado se procede a llenar el reporte lo que puede hacerse mediante diferentes vías, ya sea a través de clases especiales llamadas "**datasources**" o a través de una conexión la cual se le pasa como parámetro a la clase encargada del llenado de los objetos JasperReport, dicha clase es JasperFillmanager la cual produce un archivo .jrprint o un objeto JasperPrint los cuales pueden ser exportados o impresos según se desee.

Las plantillas contienen 9 secciones fundamentales, también conocidas como bandas, estas son:

- **Title**: Es el título del reporte y aparece una sola vez en el informe.
- **PageHeader:** Es la cabecera de la página, se repite cada vez que se muestre una página nueva.
- **ColumnHeader:** Es la cabecera de las columnas y su comportamiento es análogo a **PageHeader**.
- **Detail:** Es la encargada de mostrar los elementos que tienen alguna repetición, estos elementos se mostrarán en los subinformes.

- **ColumnFooter:** Pie de la columna. Su comportamiento es análogo a **ColumnHeader**.
- **PageFooter:** Pie de página, se repite una vez por página. Su comportamiento es análogo a **PageHeader**.
- **Summary:** Sólo se repite una vez por informe en la última página del mismo. Su comportamiento es análogo a **Title**.
- **Background:** Es una sección especial la cual se renderiza en todas las páginas y es solapada por el resto de las bandas, o sea, todas las bandas se renderizan una tras otra y esta no interfiere y puede ser usada para imprimir marcas de agua en todas la páginas del reporte.
- **NoData:** En esta sección se define que mostrar en caso de que no hayan datos en el reporte.

Nota: Ninguna de estas bandas es obligatoria que esté presente en el diseño del

2.9.1 Plantillas de los reportes

Dado los prototipos de interfaz tomados del documento: ”E. Modelo de Sistema v1.0” (Daynier Ruiz, 2009) se generaron las plantillas de los reportes, aquí se verán algunas plantillas de los CU más importantes. El resto de las plantillas se pueden encontrar en la aplicación y en el documento Modelo de Diseño (Larrauri, 2010).

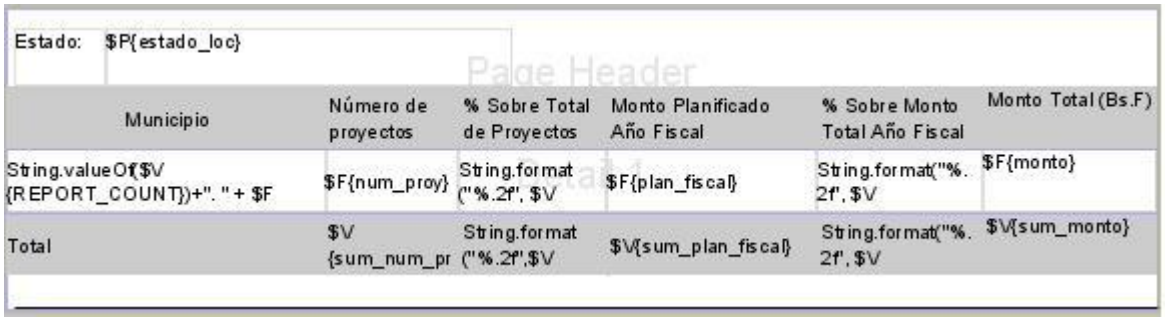

Figura 14. Cantidad de Proyectos por estado

En la figura se muestra el subreporte Cantidad de Proyectos por estado, el cual recibe como parámetro un estado y genera el reporte para ese estado.

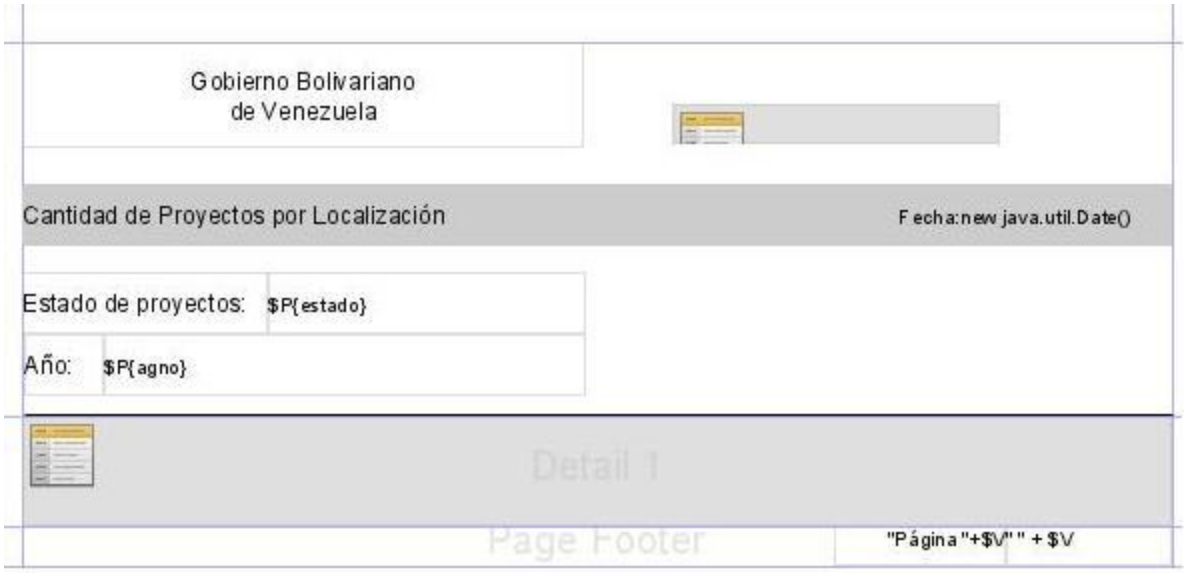

Figura 15. Cantidad de Proyectos por Localización

La principal función de este reporte es generar dentro de su contenido un "reporte de Cantidad de proyectos por estado" para todos los estados existentes para lo cual el mismo ejecuta una consulta para obtener los estados y genera un subreporte por cada estado.

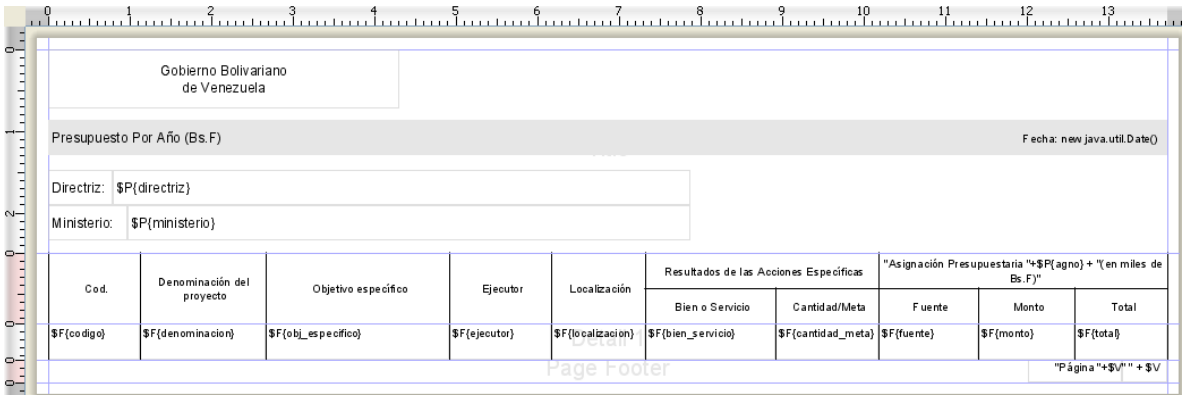

Figura 16. Plantilla del CU Generar Reporte POAN

Figura 17. Plantilla del CU Generar Reporte Cantidad de Empleos Por Nivel

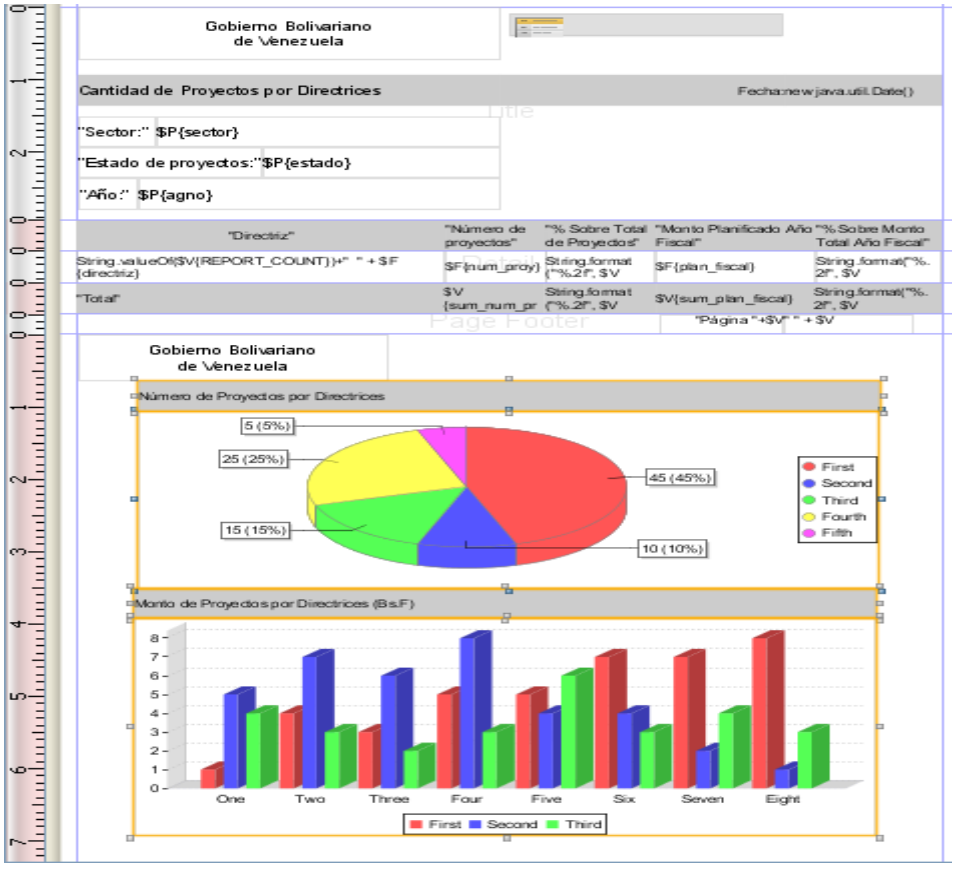

Figura 18. Plantilla del CU Generar Reporte Cantidad de Proyectos Por Directriz

En la figura 18 se hace uso de una de las gráficas que facilita la herramienta, en éste caso se ve una gráfica de pastel en 3D, también existen otros tipos de gráficas como las de líneas (2D y 3D), entre otras.

Para la realización de esta plantilla se usan la mayoría de los elementos y técnicas usadas en el proyecto de manera general, por lo que la misma será analizada en detalle.

2.9.2 Detalles Plantilla Cantidad de Proyectos Por Directriz

Las plantillas no son más que un xml al cual por convenio se le pone de extensión ".jrxml" haciendo alusión a las iniciales del JasperReport, a continuación se muestran las partes correspondientes al xml que representan los elementos explicados para la plantilla del Reporte Cantidad de Proyectos Por Directriz:

Parámetros:

```
<parameter name="sector" class="java.lang.String" isForPrompting = "false">
            <defaultValueExpression><![CDATA[]]></defaultValueExpression>
</parameter>
```

Mediante los parámetros este reporte recibe varios datos como la dirección donde se encuentra ubicado el estilo que se le aplica al reporte en el parámetro style_dir, el sector del cual se desea obtener los datos, el año, el estado de los proyectos que se quiere revisar y otros filtros opcionales como el nivel, el objetivo, etc.

Variables:

```
<variable name="chart_proy_pc" class="java.lang.Float" resetType="Group"
resetGroup="counter">         <variableExpression><![CDATA[((float
$F{num_proy}*100) /
      $V{total_  proy} ]]> </variableExpression>
</variable>
```

Se crearon variables con el objetivo de realizar algunos cálculos de manera automatizada como es el caso de los totales de cada columna de la tabla del reporte y los datos referentes a las columnas de por ciento cuyas fórmulas se pueden apreciar en el xml.

Grupos:

```
<group name="counter">
      <groupExpression><![CDATA[new Boolean(true)]]> /groupExpression>
</group>
```

Para que se creara un reporte en forma de tabla y con los totales al final de toda la información se hizo necesario crear un grupo que se encargara de brindar la estructura necesaria para la representación de la misma.

Subreporte:

```
<subreport>
      <reportElement x="288" y="6" width="181" height="23"/>
      <connectionExpression><![CDATA[$P{REPORT_CONNECTION}]]></connectionEx
pression>
      <returnValue subreportVariable="total_proy" toVariable="total_proy"/>
      <returnValue subreportVariable="monto_total"
toVariable="monto_total"/>
      <subreportExpression
class="java.lang.String"><![CDATA[$P{sub_report_dir}]]></subreportExpressio
n>
</subreport>
```

Para la obtención de los por cientos necesarios en este reporte se necesita no solo los datos correspondientes a los proyectos de los sectores, niveles, año o cualquier otro filtro sino que el dato correspondiente al total de proyectos existentes es necesario también. Para lograr este objetivo se creó un subreporte para que en el mismo se ejecutara la consulta correspondiente a la obtención de totales. Gracias a la capacidad que brinda JasperReport de retornar valores desde subreportes, estos totales se pasan a variables del reporte original y se pueden usar para calcular los por cientos necesarios.

Gráficas:

```
<pie3DChart>
      <chart isShowLegend="true" renderType="draw" theme="default">
            <reportElement style="borde" x="41" y="98" width="492"
height="163"/>
            <chartTitle position="Top"/>
            <chartSubtitle/>
            <chartLegend position="Right"/>
      </chart>
      <pieDataset>
            <keyExpression><![CDATA[$F{directriz}]]></keyExpression>
            <valueExpression><![CDATA[$F{num_proy}]]></valueExpression>
            <labelExpression><![CDATA[$F{num_proy} +
"("+String.format("%.2f", $V{chart_proy_pc})+ "%)"]]></labelExpression>
      </pieDataset>
      <pie3DPlot depthFactor="0.15" isCircular="false" labelFormat="{1}
({2})">
            <plot/>
            <itemLabel color="#000000" backgroundColor="#FFFFFF"/>
      </pie3DPlot>
</pie3DChart>
```

En la actualidad es muy extendido el uso de gráficas, como requisito de este reporte el cliente pidió que la información, además de mostrarse en forma tabular, se represente en forma de gráfica circular.

TextField (Campos de Texto)

```
<textField isStretchWithOverflow="true">
      <reportElement positionType="Float"  mode="Transparent" x="1" y="139"
width="123" height="31" />
```

```
        <textElement verticalAlignment="Middle">
                <font size="12" isBold="false"/>
        </textElement>
        <textFieldExpression class="java.lang.String"><![CDATA["Estado de
proyectos:"]]></textFieldExpression>
</textField>
```

Mediante estos elementos se muestra la información obtenida de la consulta, al igual que con las variables.

StaticText (Texto estático)

```
<staticText>
        <reportElement x="282" y="67" width="188" height="31" />
        <textElement textAlignment="Right" verticalAlignment="Middle">
                <font isBold="false"/>
        </textElement>
        <text><![CDATA[Fecha:]]></text>
</staticText>
```

El título, las etiquetas de los datos, los encabezados de la tabla que muestra los datos del reporte y otros elementos cuyo valor no cambia nunca en el reporte se muestran mediante este componente.

2.10 Diagrama de despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. A continuación se muestra como quedó conformado el diagrama de despliegue del sistema.

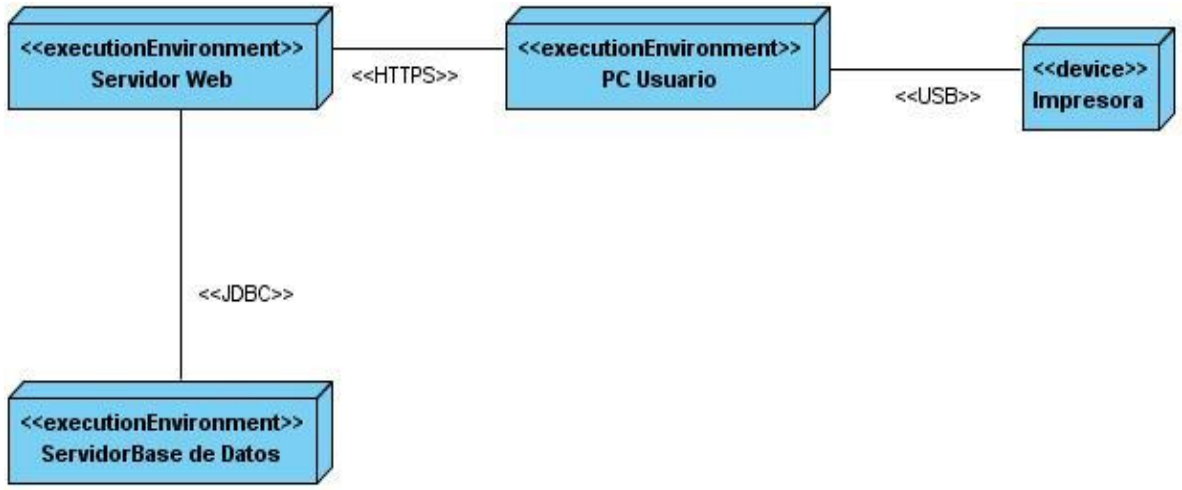

Figura 19. Diagrama de Despliegue

2.11 Diagrama de Componentes

Un subsistema de implementación es una colección de componentes y otros subsistemas de implementación usados para estructurar el modelo de implementación y dividirlos en pequeñas partes que pueden ser integradas y probadas de forma independiente. Los subsistemas de implementación incluyen dependencias y otras informaciones. Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación, especificando:

Los subsistemas de implementación, sus dependencias a la hora de importar código y organizar los subsistemas de implementación en capas.

Estos se utilizan para mostrar las dependencias de compilación de los ficheros de código, relaciones de derivación entre ficheros de código fuente y ficheros.

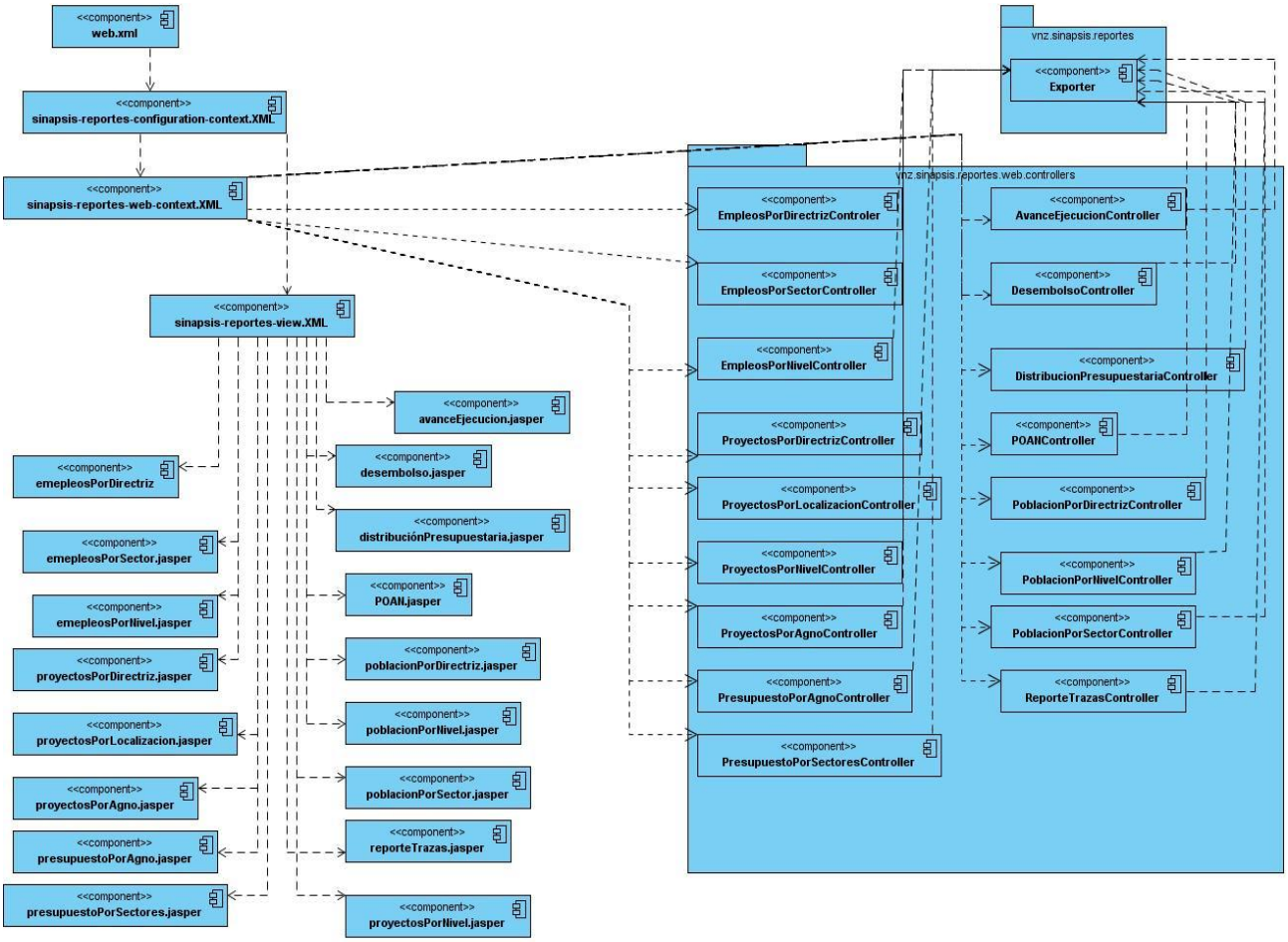

Figura 20. Diagrama de Componentes

2.12 Flujo para la obtención de reportes

Para obtener un reporte el cliente selecciona el módulo de reportes en la paleta izquierda y se muestran los filtros:

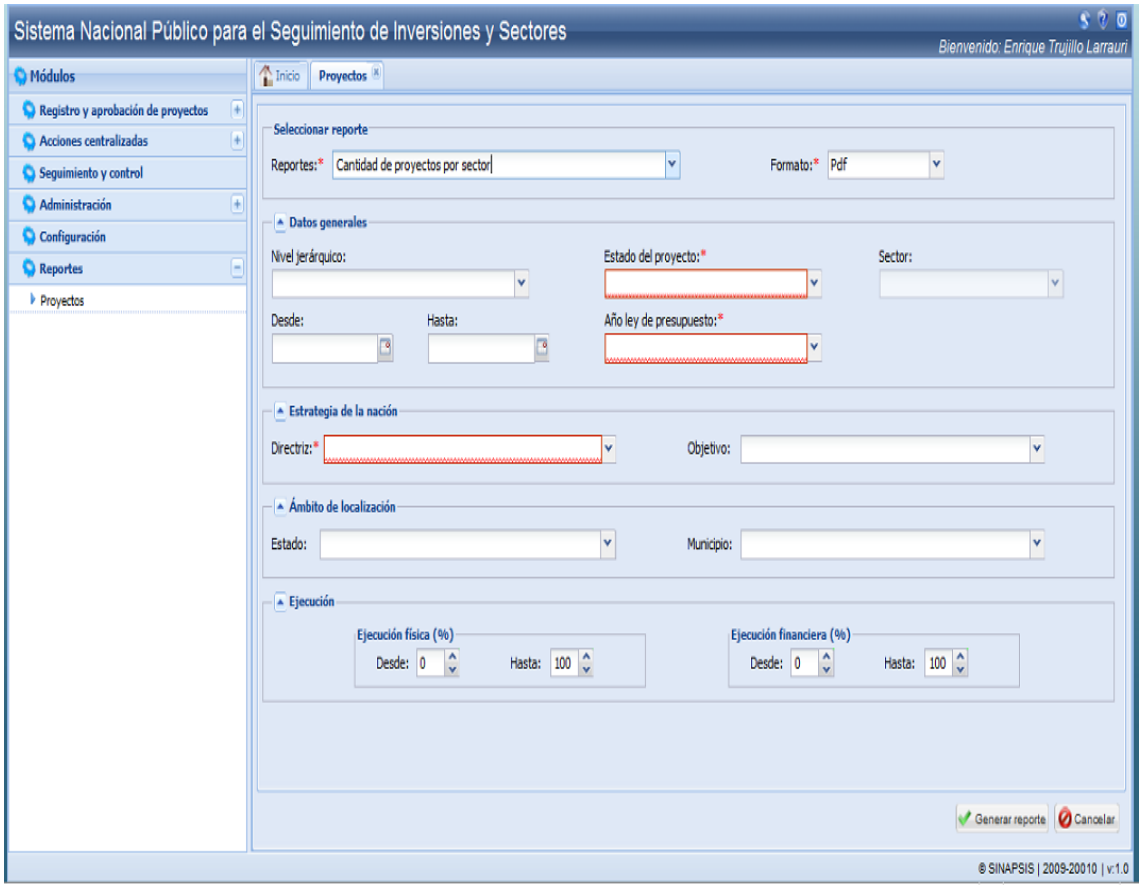

Figura 21. Módulo de reportes

Al seleccionar el reporte que el usuario desea, los campos de filtro se configuran automáticamente para indicar cuáles son obligatorios o cuáles no pueden ser usados en el filtro, ejemplo de esto es que si el usuario desea obtener la cantidad de proyectos por cada directriz de la nación, el filtro directriz no es aplicable.

Se brinda además la posibilidad de seleccionar el formato en que se desea el informe final admitiéndose pdf, html, excel, word y odt.

En el 0 se muestra el reporte de cantidad de proyectos por Localización, donde se muestra los montos invertidos en cada municipio de cada estado existente y los por ciento que representan.

2.13 Conclusiones

En este capítulo se obtuvo una representación del flujo de implementación a través de diferentes diagramas como el de despliegue del sistema que muestra la situación física de la aplicación, el diagrama de componentes que representan cada parte modular del sistema y las relaciones entre ellas, así como los diagramas de clases necesarios para lograr una buena implementación.

Se pusieron en práctica los estándares de codificación definidos para la implementación de la solución propuesta, lo que ayudó a tener un código más uniforme, claro y fácil de mantener.

El diseño gráfico establecido para la aplicación cumple con las expectativas del cliente y con los prototipos firmados en el flujo anterior.

Capítulo 3: Validación de la solución propuesta.

3.1 Introducción

En el desarrollo del software, las posibilidades de error son innumerables. Pueden darse por una mala especificación de los requisitos funcionales, uso indebido de las estructuras de datos, errores al enlazar módulos, etc. El desarrollo del software ha de ir acompañado de alguna actividad que garantice la calidad; la prueba es un elemento crítico para la garantía de calidad del software.

La prueba y validación de los resultados no es un proceso que se realiza una vez desarrollado el software, sino que debe efectuarse en cada una de las etapas de desarrollo. Es fundamental medir la cobertura de las pruebas, es decir, la determinación de cuando se han realizado las suficientes pruebas. Si se siguen encontrando errores cada vez que se procesa el programa, las pruebas deben continuar.

Durante el mantenimiento debe de existir documentación de pruebas que incluyan casos de prueba y resultados esperados. Si se producen modificaciones en el programa, habrá que probar de nuevo todas las partes del programa afectadas por las modificaciones.

El desarrollo de este capítulo tiene como objetivo especificar las pruebas que serán hechas al software y describir todo lo encontrado en las mismas.

3.2 Pruebas de software

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa. Involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados, es por eso que la realización de las mismas a los software es un factor de vital importancia.

La fase de pruebas es la que añade al producto final el valor para afirmar que ya se ha alcanzado la calidad requerida. Gran porciento de los programas que se desarrollan tienen errores, y es en la fase de pruebas donde se descubren, ese es el valor que añade esta etapa. Es por ello que probar es una de las fases más importantes para que un producto salga con la calidad máxima y sin errores.

Probar es un proceso de ejecución de un programa con la intención de descubrir un error. Una prueba tiene éxito si se descubre un error no detectado hasta entonces (Myers, 1979)

3.2.1 Objetivos

"La prueba del software es un elemento crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. Además esta etapa implica:

- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Diseñar pruebas que sistemáticamente saquen a la luz diferentes tipos de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo." (Pruebas)

La prueba no es una actividad sencilla, no es una etapa del proyecto en la cual se asegura la calidad, sino que la prueba debe ocurrir durante todo el ciclo de vida: probar la funcionalidad de los primeros prototipos, probar la estabilidad, cobertura y rendimiento de la arquitectura, probar el producto final, entre otras. Lo que conduce al principal beneficio de la prueba: proporcionar retroalimentación mientras haya todavía tiempo y recursos para hacer algo.

3.2.2 Alcance

Se lleva a cabo la prueba y se evalúan los resultados obtenidos frente a los resultados esperados. Si se descubren datos erróneos implica que hay un error y hay que corregirlo y empieza el proceso de depuración de errores. Se basa en las estructuras de control del diseño procedimental para generar los casos de prueba que:

- Garanticen que se recorren por lo menos una vez todos los caminos independientes de cada módulo.
- Se ejecutan todas las decisiones lógicas en su parte verdadera y en su parte falsa. Se recorren todos los bucles.
- Se utilizan las estructuras de datos internas para garantizar su validez.
- Se invierte tiempo y esfuerzo en los detalles de control debido a que: Los errores suelen estar en situaciones fuera de las normales.
- A menudo caminos que se piensa que tienen pocas posibilidades de recorrerse, son recorridos regularmente.

- Los errores tipográficos son aleatorios. Puede que no sean detectados por los procesadores de la sintaxis del lenguaje particular y estar presentes en cualquier camino lógico.

3.3 Descripción de las pruebas de Unidad

Las pruebas de unidad son la manera de comprobar el funcionamiento correcto de determinado módulo de código y ayuda a independizar el mismo. Esto significa que se pueden probar los módulos independientemente uno de otros. Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del módulo. Si los datos no fluyen correctamente, todas las demás pruebas no tienen sentido.

Realizar uno de estos test es necesario probar el flujo de datos desde la interfaz del componente. Si los datos no se comportan correctamente al ser introducidos en la aplicación, todas las demás pruebas no cumplen función hacerlas.

3.3.1 Prueba de Caja Blanca o Estructurales

A este tipo de técnicas se le conoce también como Técnicas de Caja Transparente o de Cristal. Este método se centra en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa. Se examina así la lógica interna del programa sin considerar los aspectos de rendimiento.

El objetivo de la técnica es diseñar casos de prueba para que se ejecuten, al menos una vez, todas las sentencias del programa, y todas las condiciones tanto en su vertiente verdadera como falsa.

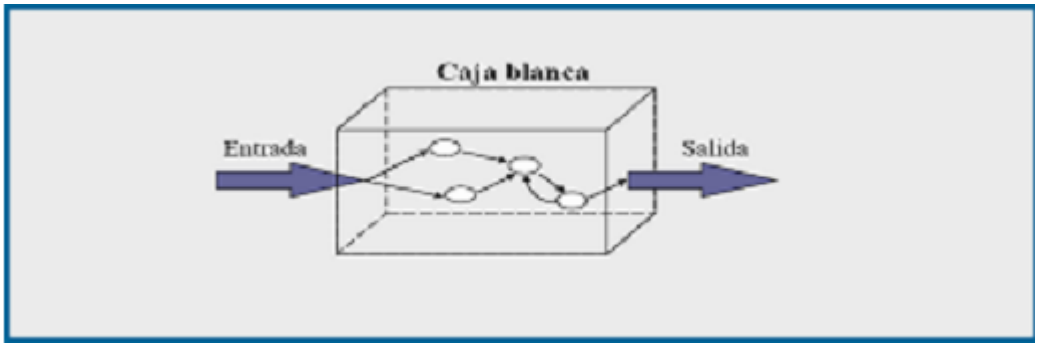

Figura 22. Representación de pruebas de Caja Blanca.

En resumen, mediante la prueba de caja blanca se puede obtener casos de prueba que:

1. Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
2. Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Algunas técnicas de prueba de Caja Blanca son:

Prueba de Condición: Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.

Prueba de Flujo de Datos: Se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

Prueba de Bucles: Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

Prueba del Camino Básico: Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto básico.

3.3.2 Pruebas de Caja Negra o Funcionales

También conocidas como Pruebas de Comportamiento o Pruebas Inducidas por los Datos, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve determinado estudiando sus entradas y las salidas obtenidas a partir de ellas. No obstante, como el estudio de todas las posibles entradas y salidas de un programa sería impracticable, se selecciona un conjunto de ellas sobre las que se realizan las pruebas. Para seleccionar el conjunto de entradas y salidas sobre las que trabajar, hay que tener en cuenta que en todo programa existe un conjunto de entradas que causan un comportamiento erróneo en el sistema, y como consecuencia producen una serie de salidas que revelan la presencia de defectos. Entonces, dado que la prueba exhaustiva es imposible, el objetivo final es encontrar una serie de datos de entrada cuya probabilidad de pertenecer al conjunto de entradas que causan dicho comportamiento erróneo sea lo más alto posible.

El objetivo de realizar este tipo de prueba al sistema es para detectar el incorrecto o incompleto funcionamiento de este, así como los errores de interfaces, rendimiento y errores de inicialización y terminación.

El proceso de pruebas de caja negra se va a centrar principalmente en los requisitos funcionales del software para verificar el comportamiento de la unidad observable externamente y la calidad funcional.

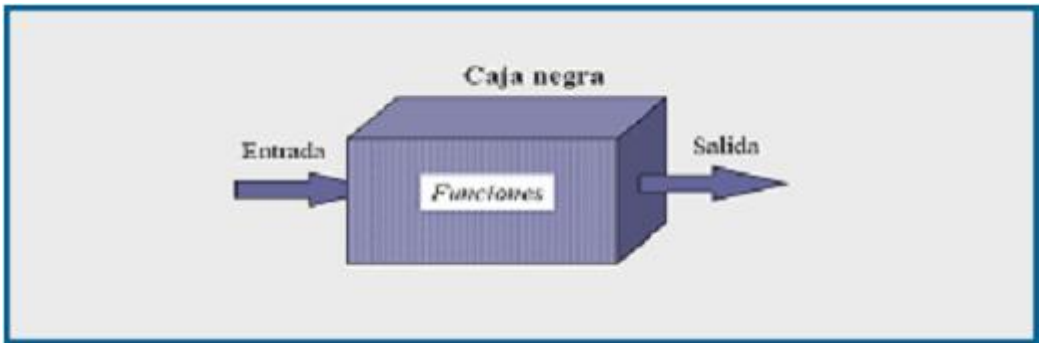

Figura 23. Representación de pruebas de Caja Negra.

Algunas técnicas utilizadas en la prueba de caja negra son:

Partición de Equivalencia: Técnica que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Análisis de Valores Límite: La experiencia muestra que los casos de prueba que exploran las condiciones límite producen mejor resultado que aquellos que no lo hacen. Las condiciones límite son aquellas que se hallan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el análisis de valores límite como técnica de prueba. Esta técnica nos lleva a elegir los casos de prueba que ejerciten los valores límite.

Las pautas para desarrollar casos de prueba con esta técnica son:

- Si una condición de entrada especifica un rango de valores, se diseñarán casos de prueba para los dos límites del rango, y otros dos casos para situaciones justo por debajo y por encima de los extremos.
- Si una condición de entrada especifica un número de valores, se diseñan dos casos de prueba para los valores mínimo y máximo, además de otros dos casos de prueba para valores justo por encima del máximo y justo por debajo del mínimo.
- Aplicar las reglas anteriores a los datos de salida.

- Si la entrada o salida de un programa es un conjunto ordenado, habrá que prestar atención a los elementos primero y último del conjunto. Grafos de Causa-Efecto: Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Se aplicarán las pruebas de caja negra pues los algoritmos internos son lineales y de poca complejidad lógica puesto los cálculos y fórmulas se manejan dentro de los reportes.

3.4 Aplicación de pruebas de caja negra

A continuación se mostrará el diseño de pruebas para el caso de uso Generar Reporte Cantidad de Proyectos por Directriz pues es uno de los que posee un mayor número de entradas obligatorias, así como de restricciones, lo cual aumenta la probabilidad de un comportamiento erróneo en el sistema.

3.4.1 Validaciones de los campos de entrada.

Antes de que se realice el diseño de pruebas es sumamente importante contar con las validaciones de los campos de entrada de cada caso de uso.

A continuación observarán las validaciones de los casos de uso Generar Reporte Cantidad de Proyectos por Directriz.

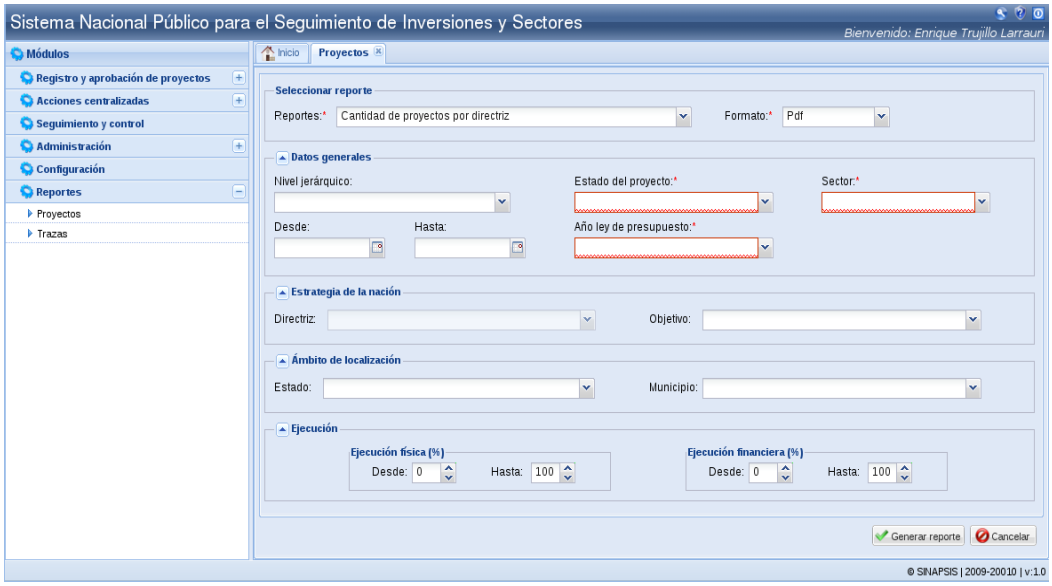

Figura 24. Vista del filtro del caso de uso Generar Reporte Cantidad de Proyectos por Directrices.

Los filtros de búsqueda son los siguientes:

Formato: Se debe seleccionar el formato al cual se desea exportar el reportar. Es obligatoria la selección

Nivel jerárquico: Se puede seleccionar el nivel jerárquico del proyecto.

Estado del proyecto: Se debe seleccionar el estado del proyecto. Es obligatoria la selección.

Sector: Se debe seleccionar el sector al que pertenece el proyecto. Es obligatoria la selección.

Desde: Se puede seleccionar la fecha de inicio del proyecto.

Hasta: Se puede seleccionar la fecha de culminación del proyecto.

Año ley presupuesto: Se debe seleccionar el año de ley de presupuesto. Es obligatoria la selección.

Directriz: Se deshabilita este filtro pues se deben mostrar los proyectos por directrices.

Objetivo: Se puede seleccionar el objetivo del proyecto.

Estado: Se puede seleccionar el estado al que pertenece el proyecto.

Municipio: Se puede seleccionar el municipio al que pertenece el proyecto. Depende de la selección del estado.

Ejecución física (desde): Se puede seleccionar un número entre 0 y 100.

Ejecución física (hasta): Se puede seleccionar un número entre 0 y 100. Debe ser mayor que el campo **Ejecución física (desde)**

Ejecución financiera (desde): Se puede seleccionar un número entre 0 y 100.

Ejecución financiera (hasta): Se puede seleccionar un número entre 0 y 100. Debe ser mayor que el campo **Ejecución financiera (hasta).**

Las validaciones de los campos de entrada de cada caso de uso de los reportes se pueden encontrar en el documento: (Trujillo Larrauri, 2010).

3.4.2 Diseño de casos de prueba.

Partiendo de las validaciones de los campos de entrada de cada CU se puede pasar al diseño de casos de prueba, ahora seguidamente se observará el diseño de casos de prueba que se le realizó al caso de uso Generar Reporte de Cantidad de Proyectos por Directrices.

Caso de Uso: Generar Reporte Cantidad de Proyectos por Directrices.

Descripción General

El caso de uso consiste en hacer un reporte con datos de los proyectos agrupados según las directrices de la nación.

Condiciones de Ejecución:

El sistema debe estar instalado y ejecutado correctamente.

Secciones a probar en el Caso de Uso:

Nombre de la sección	**Escenarios de prueba de la sección**	**Descripción de la funcionalidad**	**Flujo central**
SC1: Reporte Cantidad de Proyectos por Directriz.	EP1: Reporte Cantidad de Proyectos por Directriz	El sistema muestra el menú del módulo según los permisos del usuario.	
	EP2: Al usuario se le muestra el menú : **Reportes->Proyecto.**	Se muestra la interfaz de reportes con los siguiente filtros: • Reportes • Formato • Nivel jerárquico • Estado del proyecto • Sector • Lineamiento • Año ley de presupuesto • Desde (Fecha inicio) • Hasta (Fecha fin) • Directriz • Objetivo • Estado • Municipio • % Ejecución Física(desde y hasta) • % Ejecución Financiera (desde y hasta) Se muestra los	

		botones “Generar reporte” y “Cancelar”	
	EP3: El usuario selecciona el Reporte: “Cantidad de proyectos por Directriz”	Se resaltan los filtros que son requeridos de manera obligatoria: • Formato • Estado del proyecto • Año ley de presupuesto • Sector Se desactiva: • Directriz	
	EP4: El usuario presiona el botón “Cancelar”	Se elimina de la interfaz principal la interfaz de reportes.	
	EP5: Se presiona el botón “Generar reporte” dejando algunos filtros con valores no requeridos	Se muestra un cuadro de diálogo de error mostrando el siguiente mensaje: “Existen campos con valores no válidos”.	
	EP6: Se presiona el botón “Generar reporte” llenando todos los filtros no requeridos	Se muestra el reporte Cantidad de Proyectos por Directriz en el formato indicado.	

SC1: Reporte Cantidad de Proyectos por Directriz.

Id Escenario de prueba	Escenario	Respuesta del sistema	Resultado de la prueba
EP1	Reporte Cantidad de Proyectos por Directriz	El sistema muestra el menú del módulo según los permisos del usuario.	El sistema muestra el menú del módulo.
EP2	Al usuario se le muestra el menú :	Se muestra la interfaz de reportes con los siguiente filtros:	Se muestra la interfaz de reportes con los siguiente filtros:

	Reportes->Proyecto.	• Reportes • Formato • Nivel jerárquico • Estado del proyecto • Sector • Lineamiento • Año ley de presupuesto • Desde (Fecha inicio) • Hasta (Fecha fin) • Directriz • Objetivo • Estado • Municipio • % Ejecución Física(desde y hasta) • % Ejecución Financiera (desde y hasta) Se muestra los botones "Generar reporte" y "Cancelar"	• Reportes • Formato • Nivel jerárquico • Estado del proyecto • Sector • Lineamiento • Año ley de presupuesto • Desde (Fecha inicio) • Hasta (Fecha fin) • Directriz • Objetivo • Estado • Municipio • % Ejecución Física(desde y hasta) • % Ejecución Financiera (desde y hasta) Se muestra los botones "Generar reporte" y "Cancelar"
EP3	El usuario selecciona el Reporte: "Cantidad de proyectos por Directriz"	Se resaltan los filtros que son requeridos de manera obligatoria: • Formato • Estado del proyecto • Año ley de presupuesto • Sector Se desactiva: Directriz	Se resaltan los filtros que son requeridos de manera obligatoria: • Formato • Estado del proyecto • Año ley de presupuesto • Sector Se desactiva: Directriz
EP4	El usuario presiona el	Se elimina de la interfaz principal la interfaz de	Se elimina de la interfaz principal la interfaz de

	botón “Cancelar	reportes.	reportes.
EP5	Se presiona el botón “Generar reporte” dejando algunos filtros con valores no requeridos	Se muestra un cuadro de diálogo de error mostrando el siguiente mensaje: “Existen campos con valores no válidos”.	Se muestra un cuadro de diálogo de error mostrando el siguiente mensaje: “Existen campos con valores no válidos”.
EP6	Se presiona el botón “Generar reporte” llenando todos los filtros no requeridos	Se muestra el reporte Cantidad de Proyectos por Directriz en el formato indicado.	Se muestra el reporte Cantidad de Proyectos por Directriz en el formato indicado.

3.4.3 No Conformidades Detectadas.

Una vez que se obtiene el diseño de pruebas y se realizan las pruebas correspondientes a los casos de uso, se conforma un listado de No Conformidades Detectadas. A continuación se puede apreciar algunas no conformidades del módulo de reportes, las demás se encuentran los Modelos de pruebas especificados por casos de usos.

Tabla de No Conformidades Detectadas.

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Importancia
aplicación	1	Error al escribirse el mensaje: “12/160/2010 no es una fecha válida”	Proyectos	1ra etapa	
aplicación	2	No existe concordancia entre la fecha de inicio y la de fin	Proyectos	1ra etapa	
aplicación	3	No existe concordancia entre los % de ejecución física.	Proyectos	1ra etapa	
aplicación	4	No existe concordancia	Proyectos	1ra etapa	

		entre los % de ejecución financiera.			
aplicación	5	No existe concordancia entre el estado y el municipio seleccionado.	Proyectos	1ra etapa	

Algunas de estas no conformidades se muestran a continuación en la aplicación para una mejor comprensión y visualización.

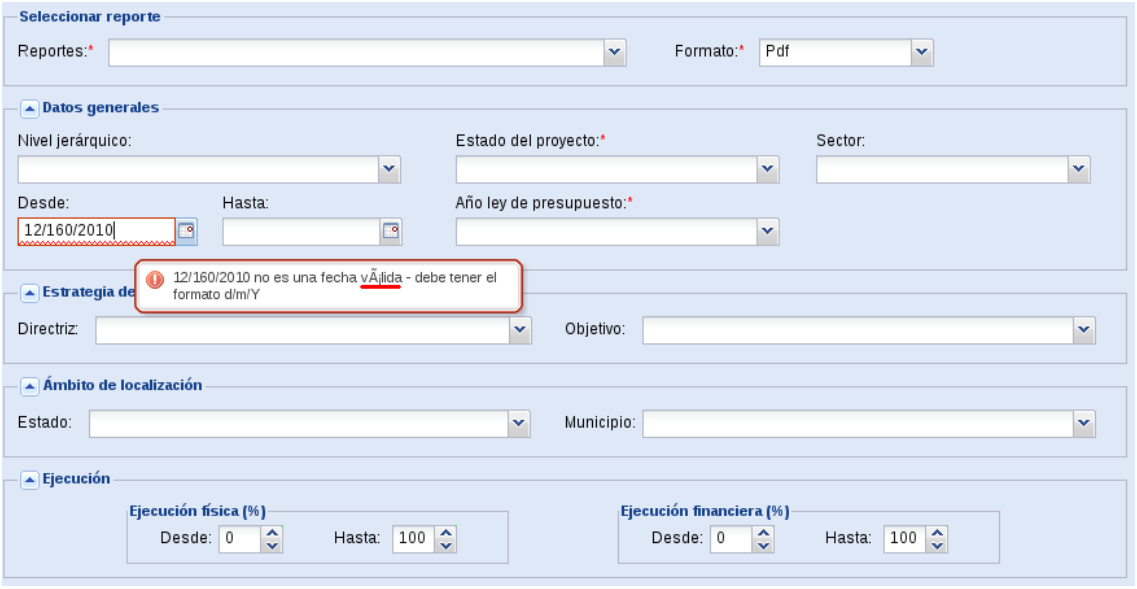

Figura 25. Error al escribir el mensaje "12/160/2010 no es una fecha válida…"

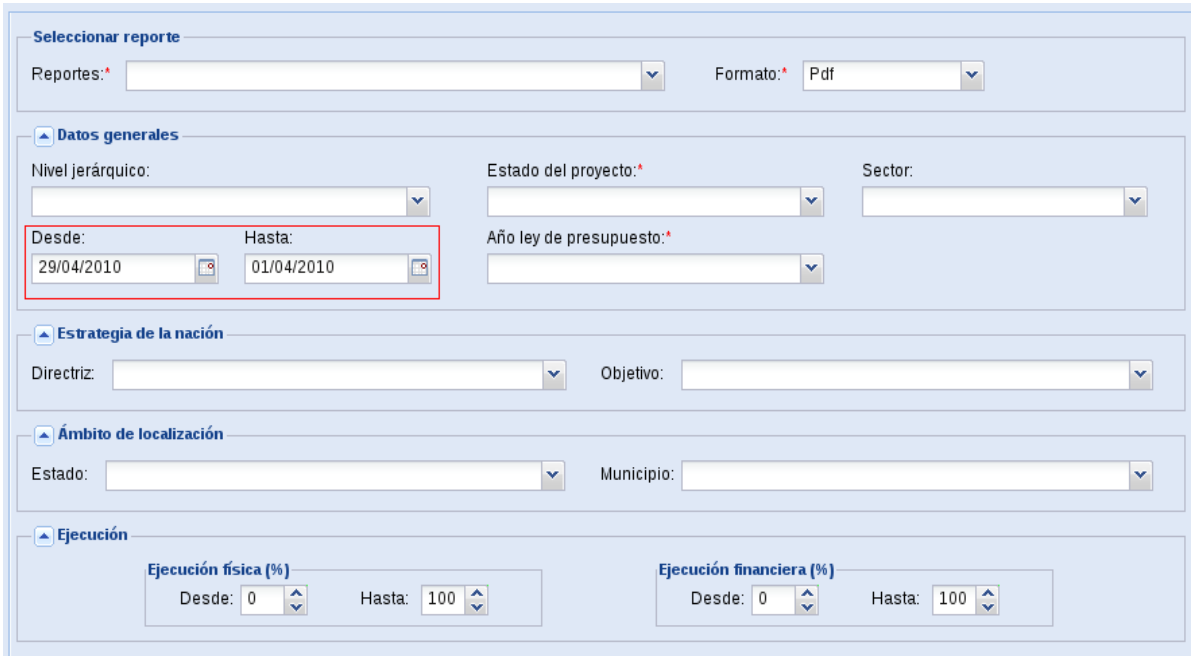

Figura 26. No existe concordancia entre las fecha de inicio y fin.

Estas no conformidades detectadas cuando se realizaron las pruebas al módulo fueron vistas, analizadas y corregidas. Una vez que fueron corregidas se les realizó nuevamente el plan de prueba donde no se encontraron errores quedando así liberado el software por el grupo de calidad del proyecto.

3.5 Conclusiones

En el desarrollo de este capítulo se hizo un análisis de diferentes aspectos como el significado de las pruebas de software, sus objetivos y alcance.

Se realizó una descripción de los test de unidad, dentro de los cuales se analizaron los tipos de prueba: Caja Blanca y Caja Negra.

A través de la ejecución de cada prueba salieron a relucir errores no detectados hasta el momento, errores que se corrigieron dándole mayor calidad al producto.

Conclusiones generales

Con el desarrollo de este trabajo se ha demostrado la necesidad de disponer de un sistema para la gestión de toda la información correspondiente a la planificación del presupuesto por proyectos anual de la República Bolivariana de Venezuela. A causa del problema planteado se realizó un estudio de todas las herramientas y las tecnologías actuales, para finalmente determinar cuáles serían las herramientas necesarias para implementar el sistema. Se utilizó RUP como metodología de desarrollo, Visual Paradigm como herramienta de modelado, java como lenguaje de programación, Eclipse como IDE de desarrollo, JasperReport como herramienta de generación de reportes e Ireport para el diseño de las plantillas. Acompañado de las herramientas se decide utilizar algunos frameworks como ExtJS para la interfaz del lado del cliente, Spring para la arquitectura MVC del lado del servidor e hibernate para el acceso a datos.

Se realizaron los artefactos correspondientes a los flujos de trabajo de diseño e implementación, los cuales fueron liberados por el equipo de calidad de la facultad, demostrando así su validez.

La aplicación fue probada mediante las pruebas de caja negra cuyo documento de casos de prueba fue validado y liberado también por el equipo de calidad.

Recomendaciones

Una vez realizado el trabajo se propone como recomendación darle mantenimiento a los reportes para que estos se mantengan actualizados según la necesidad de los clientes.

Referencias bibliográficas

(s.f.). Recuperado el 5 de 1 de 2010, de Atiwiki: http://150.185.75.30/atiwiki/index.php/JDO#Objetivos_fundamentales_de_JDO

Pruebas. (s.f.). Recuperado el 5 de Marzo de 2009, de http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/pruebas_d.php

Rational Software Corporation. (2003). *Rational Unified Process (Rational Help).*

Álvarez, E. J. (2006). Obtenido de http://www.congresoinfo.cu/UserFiles/File/Info/Info2006/Ponencias/37.pdf

Bauer, C. (2005). *Hibernate in Action.* M.P. Co.

DIGIKOL TM .Crystal Solutions.Crystal Reports XI. (s.f.). Recuperado el 21 de 1 de 2010, de http://www.crystalsolutions.com.ar/productos/crystalreports.html

Dijkstra, E. W. (1979). *Programming considered as a human activity.* New York : YourdonPress, 1979.

Enhydra, C. 2. (2007). *Open Source Java solutions.* Recuperado el 11 de 2 de 2010, de http://www.enhydra.org/index.php

Flanagan, D. (2006). *JavaScript: The Definitive Guide, 5th Edition.*

Foundation, The Eclipse. (s.f.). Recuperado el 1 de febrero de 2009, de Eclipse: http://www.eclipse.org/

Frederick S., C. R. (2008). *Learning Extjs.*

Goldentech, G. L. (s.f.). Recuperado el 15 de 01 de 2010, de http://www.goldenteche.com/Celta_Reportes.html

Goldentech, G. L. (2010). Recuperado el 20 de 12 de 2010, de http://www.goldenteche.com/Celta.html

Harrop, R. (2005). *Pro Spring.* Apress.

Heffelfinger, D. R. (2009). *JasperReportsfor Java Developers.*

Heudecker, N. (2003). Recuperado el 15 de 2 de 2010, de TheServerSide.COM: http://www.theserverside.com/

Informatizate.net. (s.f.). *Metodologías De Desarrollo De Software XP.* Recuperado el 12 de 02 de 2010, de http://www.informatizate.net

J. Vásquez, F. E. (20 de 12 de 2009). *Revista Tecnológica ESPOL.* Obtenido de Sistema Censo Académico en Línea (CENACAD) para la Automatización.: http://www.rte.espol.edu.ec/archivos/Revista_2007/19-248Final.pdf

Jacobson, I. B. (2000). *El Proceso Unificado de Desarrollo de Software.*

Janis Graudins, L. Z. *Comparative Analysis of EJB3 and Spring Framework.*

Johnson, R. 2. (2005). Recuperado el 12 de 2 de 2010, de TheServerSide.COM: http://www.theserverside.com

López, R. G. (2009). *PHP vs Java.*

Myers, G. (1979). *The Art of Software Testing.*

Ramon, J. (2009). *Ext JS 3.0 Cookbook.* PACKT.

Snook, J. (2007). *Apress Accelerated DOM Scripting with Ajax APIs and Libraries.* Apress.

Stefanov, S. (2008). *Object Oriented JavaScript.* PACKT.

Storkel, S. (2002). *An Introduction to the Eclipse IDE. .*

Trujillo Larrauri, E. (2009). *Diseño casos de prueba presentación v1.0.*

Vázquez, Y. G. (2006). Recuperado el 20 de 01 de 2010, de http://www.congresoinfo.cu/UserFiles/File/Info/Info2006/Ponencias/37.pdf

Walls, C. (2005). *Spring in Action.* M.P. Co.

Anexos

Anexo.1 : Acta de aceptación del módulo Reporte del sistema SINAPSIS.

PDVSA

Caracas, 07 de Diciembre de 2009.

Señor **Juan Carlos Montané Izaguirre**
Jefe del Proyecto "**SISTEMA NACIONAL PÚBLICO PARA EL SEGUIMIENTO DE INVERSIONES Y SECTORES**".

Estimado: **Juan Carlos Montané Izaguirre**

Después de analizado los documentos:

- Especificación de requisitos de software. Módulo de Reporte.
- Modelo de Sistema. Módulo de Reporte.

Visión del Proyecto concretado en el **Anexo 3** al **Convenio PDVSA-ALBET**, manifestamos nuestra conformidad.

Atentamente,

Saludos,

07/12/09

Saúl Chirinos Gutiérrez
Gerente del Centro de Servicios Comunes-Occidente.
Jefe de Proyecto Sistema Nacional Público para el Seguimiento de Inversiones y Sectores.

PETROLEOS DE VENEZUELA
Avenida Libertador, edificio Petróleos de Venezuela, Torre Este, La Campiña. Apartado Portal 61373. Caracas 1060-A. Venezuela
Teléfono: (02) 708.1111 Fax: (02) 708.3257 www.pdvsa.com

Anexo.2 : Acta de liberación de calidad del módulo Reporte del sistema SINAPSIS.

UCi

Informaticas

Acta de Liberación de Artefactos, Grupo de Calidad Centro CEGEL de la Facultad 15 de la Universidad de las Ciencias Informáticas.

Martes, 8 de junio de 2010.

Luego de haber efectuado 3 iteraciones de revisiones a los artefactos: Modelo de Sistema, Modelo de diseño y Modelo de Despliegue del módulo Reportes del proyecto SINAPSIS del Centro CEGEL de la Facultad 15 y haberse detectado un promedio de 20 No Conformidades, se puede afirmar que se han corregido los defectos encontrados, por lo que quedan liberados los artefactos.

Firma del Asesor y Jefe del Grupo de Calidad Centro CEGEL

Ing. Raúl Velázquez Alvarez

UCi Universidad de las Ciencias Informáticas

Decanato Facultad

Anexo.3: Reporte Cantidad de Proyectos Por Localización

Gobierno Bolivariano
de Venezuela

Cantidad de Proyectos por Localización Fecha: 22/03/2010

Estado de proyectos: Aprobados

Año: 2009

Estado: Bolivar

Municipio	Número de proyectos	% Sobre Total de Proyectos	Monto Planificado Año Fiscal	% Sobre Monto Total Año Fiscal	Monto Total (Bs.F)
1. Ciudad Bolivar	84	16,15	223578	17,20	207334
2. Puerto Ordaz	85	16,35	193951	14,92	225279
3. San Felix	58	11,15	161043	12,39	122199
Total	227	43,65	578572	44,52	554812

Estado: Monagas

Municipio	Número de proyectos	% Sobre Total de Proyectos	Monto Planificado Año Fiscal	% Sobre Monto Total Año Fiscal	Monto Total (Bs.F)
1. Libertador	80	15,38	225962	17,39	191220
2. Maturin	93	17,88	217680	16,75	258173
3. Uracoa	72	13,85	170239	13,10	182637
Total	245	47,12	613881	47,23	632030