Universidad de las Ciencias Informáticas

Facultad 5

Título: "Desarrollo de un módulo de reportes para el software de replicación REKO."

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

Autor: Julio Cesar Valdesuso Cepeda.

Tutor: Ing. Yusmary Companioni Sardiña.

Co-tutor: Ing. Albin Amat Reyes.

"Cuando en los hombres se encarna un grave pensamiento, un firme intento, una aspiración noble y legítima, los contornos del hombre se desvanecen en los espacios sin confines de la idea"

José Martí

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis 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 _____.

Julio Cesar Valdesuso Cepeda

_____

Firma del Autor

Ing. Yusmary Companioni Sardiña

_____

Firma del Tutor

Ing. Albin Amat Reyes.

_____

Firma del Co-Tutor

DATOS DE CONTACTO.

Ing. Yusmary Companioni Sardiña.

e-mail: ycompanioni@uci.cu

Graduado de Ingeniería en Ciencias Informáticas en el 2010 con 1 año de experiencia en la especialidad.

Ing. Albin Amat Reyes.

e-mail: aamat@uci.cu

Graduado de Ingeniería en Ciencias Informáticas en el 2009 con 2 años de experiencia en la especialidad.

AGRADECIMIENTOS.

Agradezco:

- *A mi mamá Mariela y mi papá Gerardo por ser la fuente de mi perseverancia, por ser mi mejor ejemplo a seguir, por hacer de mí el hombre que soy, sin ustedes no habría llegado lejos, los amo.*
- *A mi tía Alba, mi segunda mamá, por enseñarme que su cariño no tiene límite.*
- *A mi abuelo Gerardo por tener tanta paciencia conmigo, por construirme la base que ha permitido que llegue hasta aquí.*
- *A mi abuela Emelidad por enseñarme que "el que guarda siempre tiene", que aunque no esté físicamente, de mi corazón nunca saldrá.*
- *A mi hermana por recordarme que tengo a quien dar el ejemplo y quererme tanto.*
- *A mi familia por estar siempre tan unidad, en las buenas y en las malas.*
- *A mis amigos y compañeros de estudios por aguantarme con mis defectos y virtudes.*
- *A todos aquellos que de una forma u otra han hecho este sueño realidad*

A todos Gracias.

DEDICATORIA.

A mi familia por convertir el problema de uno en el de todos, por no parpadear a la hora de sacrificarnos por los que amamos, por sentir cada logro como un logro de todos.

Gracias por estar siempre cuando los necesito,

por hacerme sentir tan orgullos

de formar parte de ustedes.

RESUMEN.

En el presente trabajo se pretende agregar una nueva funcionalidad para mejorar la versión del Replicador REKO creado por la Comunidad Java (JEE) que pertenece al Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE) ubicado dentro de la Universidad de las Ciencias Informáticas (UCI). El software de réplica de datos implementado en la UCI surge como respuesta a la necesidad de mantener actualizadas un conjunto de bases de datos. Se fundamenta en la copia de información, de una localización a otra. Pretende cubrir las principales necesidades relacionadas con la distribución de datos entre los gestores más populares. El módulo de reportes para REKO se encarga de mantener una constante actualización con los procesos de transferencia de datos, sincronización, protección, centralización y recuperación de la información, así como errores y cualquier otra información sobre el proceso de réplica de datos. Para su implementación se utilizará: Visual Paradigm para el análisis y diseño del módulo. Como metodología de desarrollo OpenUP dadas las características de la aplicación a construir y de las ventajas que esta metodología proporciona; y UML como lenguaje de modelado. La herramienta de diseño para las plantillas será iReport apoyándose en la librería JasperReports. Java será el lenguaje de programación apoyados en Spring MVC y Dojo. Como IDE de desarrollo se utilizará Eclipse por ser de código abierto, extensible e incorporar funcionalidades tan interesantes como la refactorización de código y acepta además la inclusión de *plugins*[1] de desarrollo. La arquitectura que se usará será en capas.

PALABRAS CLAVE.

Bases de datos, Módulo de Reportes, REKO, Réplica de datos.

[1] Complemento que se adiciona a una aplicación y aportarle una función nueva y generalmente muy específica.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS.

ÍNDICE DE TABLAS.

ÍNDICE DE DIAGRAMAS.

Introducción:

En la actualidad la informática se ha convertido en una ciencia de suma importancia para el desarrollo de la sociedad, pues se puede aplicar en cualquier esfera el proceso de informatización y está en constante evolución, brindando nuevos servicios y funcionalidades en aras del desarrollo y el avance tecnológico.

La necesidad de extender las funcionalidades de los sistemas informáticos a distintas ubicaciones, unido a la acelerada evolución de la informática dio lugar al surgimiento de los sistemas informáticos distribuidos. Estos sistemas están diseñados para que muchos usuarios trabajen en forma conjunta y plantean en su arquitectura el uso de varios servidores de bases de datos separados geográficamente; de aquí surge la necesidad de mantener una actualización y sincronización adecuada de estos sistemas para su correcto funcionamiento.

En la actualidad, con la tendencia al desarrollo de sistemas distribuidos, ha aumentado la necesidad de la utilización de réplicas. La replicación es el proceso de copiado y mantenimiento de objetos de la base de datos, tales como las tablas, en bases de datos múltiples y simulan un sistema de bases de datos distribuido. Los cambios hechos en un sitio son capturados y aplicados en cada una de las locaciones remotas.

Cuba se ha introducido en un proceso de desarrollo informático, con el objetivo de lograr un progreso de la economía y el bienestar social de su pueblo, para ello ha desarrollado una verdadera revolución tecnológica. En este sentido la Universidad de las Ciencias Informáticas (UCI) nacida como universidad de nuevo tipo, juega un papel crucial. La visión de la UCI es la de una Ciudad Digital Avanzada, formando el capital humano especializado, investigando, produciendo software y servicios informáticos para la sociedad cubana.

El Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE), como parte de la infraestructura productiva de la UCI, desarrolló el software replicador de datos REKO. Esta herramienta multiplataforma pretende cubrir las principales necesidades relacionadas con la distribución de datos entre los gestores de datos más populares, dando solución a las principales necesidades de réplica tales como transferencia de datos, sincronización, protección, centralización y recuperación de la información. La replicación se utiliza para diseminar datos en un ambiente distribuido,

con el objetivo de obtener mejor rendimiento y confiablidad, mediante la reducción de dependencia de un sistema de base de datos centralizado.

Actualmente de la información generada dentro de los distintos procesos de réplica que ejecuta el software REKO no se persiste en su totalidad; impidiendo al usuario estar en constante actualización con los procesos de transferencia de datos, sincronización, protección, centralización y recuperación de la información así como errores; dificultando tomar decisiones e impidiendo a su vez conocer cuán cerca se está de alcanzar las metas trazadas en relación con el negocio.

Partiendo de lo planteado anteriormente surge el siguiente **problema a resolver:** ¿Cómo hacer uso de la información generada a partir del proceso de réplica de datos, para favorecer la toma de decisiones y el análisis de comportamientos de réplicas por parte de los usuarios del REKO?

Basado en el problema a resolver se define como **objeto de estudio:** Proceso de generación de reportes. El cual estará delimitado por el **campo de acción:** Proceso de generación de reportes en el software REKO.

Para dar solución al problema planteado se define como **objetivo general:** Desarrollar un módulo de reportes para el software de réplica REKO; que permita a los usuarios hacer uso de la información generada durante el proceso de réplica, para facilitar la toma de decisiones y análisis de comportamientos de réplicas.

Teniendo en cuenta el objetivo general se plantea como **Idea a defender:** La implementación del módulo Reportes para el software de réplica REKO facilitará y garantizará el rápido acceso a la información registrada en el sistema informático así como el trabajo con la información para la toma decisiones relacionadas con el negocio.

Para el cumplimiento del objetivo general se decide desglosar el mismo en los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación mediante el estudio y selección de las tecnologías necesarias para la construcción del módulo de reportes para REKO
- Desarrollar el diseño e implementación del módulo de reportes para REKO, poniendo en práctica la metodología y herramientas seleccionadas para la obtención de un producto acorde a las necesidades del cliente.

- Validar la solución propuesta mediante la aplicación de pruebas de software para el aseguramiento de la calidad del producto obtenido.

Estos objetivos específicos se les darán cumplimiento mediante las siguientes **tareas de investigación:**

- Estudio del estado de los sistemas para le generación de reportes.
- Análisis de las principales metodologías para el desarrollo de aplicaciones similares.
- Selección de herramientas para la solución del problema.
- Definición de los requisitos funcionales de la aplicación para el negocio.
- Generación de los artefactos para las fases de Diseño.
- Implementación del sistema.
- Validación de la solución.

A continuación se muestran una breve descripción de los métodos de investigación y empíricos utilizados para el desarrollo de la aplicación.

Métodos de Investigación:

Histórico - Lógico: Permite conocer el desarrollo evolutivo de los sistemas de gestión de reportes, de las distintas herramientas y tecnologías utilizadas para implementar los reportes.

Analítico - Sintético: Permite estudiar por separado las distintas herramientas que se tienen que emplear y luego integrarlas todas en la solución final.

Métodos Empíricos:

Entrevista: Permite recoger la información necesaria relacionada con el módulo de reportes para llevar a cabo la constitución de un producto con verdadera calidad y que cumpla las expectativas del cliente.

Para el desarrollo del presente trabajo se definieron tres capítulos, organizados de la siguiente forma:

Capítulo 1. Fundamentación Teórica: Se destacan las tendencias y tecnologías actuales sobre las cuales se apoya la propuesta del sistema informático, así como las diferentes metodologías de desarrollo de software, herramientas a utilizar y demás elementos asociados al estado del arte.

Capítulo 2. Características y diseño del Sistema: Incluye la descripción, diseño y análisis de la solución que se propone para darle respuesta a la problemática planteada.

Capítulo 3. Implementación y Pruebas: Descripción del flujo de implementación y estilos a utilizar, además de las pruebas que le serán aplicadas al sistema para demostrar la robustez del mismo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se hace alusión a algunos conceptos básicos relacionados con el dominio del problema a resolver; analizándose algunos sistemas informáticos existentes en el mundo que están relacionados con el problema actual, para diseñar una solución eficiente. También se realiza un estudio de las tecnologías, herramientas y metodologías que se utilizaran en el proceso de desarrollo, justificando el uso de cada una de ellas.

1.1 Marco Conceptual.

En el marco conceptual se pretende definir una serie de conceptos, los cuales están asociados al módulo de reportes, haciendo más amena la comprensión de estos términos por parte del lector.

Módulo: "Es un componente auto controlado de un sistema, el cual posee una interfaz bien definida hacia otros componentes; algo es modular si es construido de manera tal que se facilite su ensamblaje, acomodamiento flexible y reparación de sus componentes". {1}

Reporte: Información que se agrupa de acuerdo a un interés específico, basado en datos seleccionados en los filtros que han sido definidos en el sistema, los cuales pueden ser presentados de manera textual o a través de gráficos.
Los datos dentro de un reporte no pueden ser manipulados o modificados directamente, sino que tienen que ser afectados en alguna otra parte del sistema para que se reflejen los cambios una vez que el reporte sea generado nuevamente. Un reporte es generado dinámicamente, cada vez que se llama o se invoca desde el sistema, el reporte actualiza la información a los datos más recientes disponibles. (1)

Generación de reportes: Es la obtención de reportes en un formato personalizado de tal manera que le sea útil al personal al cual va dirigido, en este proceso interviene la acción de los generadores de reportes.

Generadores de reportes: Son programas para crear informes en una amplia variedad de formatos que no son rutinariamente producidos por un sistema de

información. Extraen datos de los archivos o de las bases de datos y crean reportes de acuerdo al formato definido, proporcionan más control, pueden manejar datos de cálculos y lógica compleja antes de darles la salida. (2)

Los generadores de reportes de manera estándar se caracterizan por dos elementos básicos, un diseñador o editor de informes y un motor que los genere como se muestra en la ***Fig. 1***.

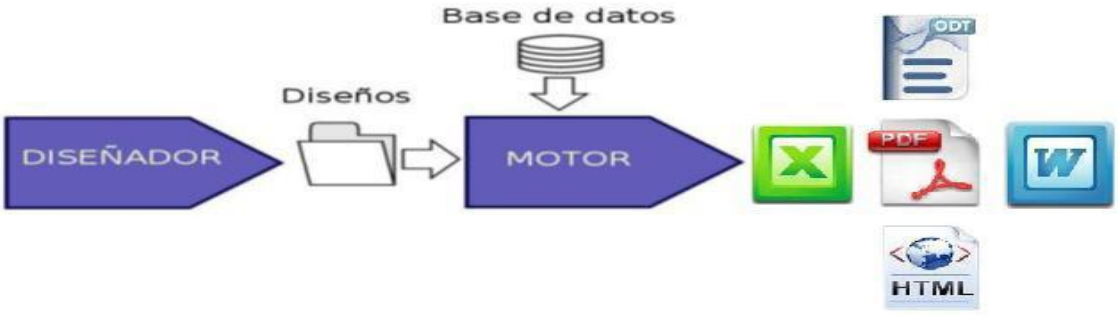


Fig. 1 Estructura general de un generador de reportes

De acuerdo al esquema de la ***Fig. 1*** las principales actividades o tareas que debe realizar un generador de reportes son las siguientes:

- Conformar el diseño de los reportes.
- Luego de obtener los datos se genera en el formato requerido el informe final de acuerdo a lo diseñado previamente.

1.2 Estado del Arte

A continuación se muestra un resumen de los sistemas automatizados compuestos por módulos de reportes para la gestión de información, los cuales permiten que la configuración deseada se exporte en plantillas con un formato determinado, de manera tal que el generador posteriormente las utilice para volcar la data proveniente de las bases de datos y lograr generar el reporte deseado con la apariencia definida en la edición y obtenida de la plantilla.

1.2.1 Sistemas automatizados que poseen módulos de reportes en el Mundo.

CENACAD: Sistema Censo Académico en Línea para la Automatización de la Evaluación a los Docentes, realizado por parte del personal de la Escuela Superior Politécnica del Litoral (ESPOL), Guayaquil, Ecuador. El sistema CENACAD 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 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.

El sistema tiene la capacidad de:

- Mostrar reportes en línea: que son de fácil entendimiento, con información mostrada de una manera amigable y que permite la descarga 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:

- Promedio General de ESPOL (por término).
- Promedio por Unidad (por término).
- Promedio por Profesor-Término (por término).
- Listado de mejores docentes en base a los mejores promedios obtenidos en la ESPOL por el período de un año.
- Promedio de la(s) facultad(es) o Instituto(s).

- 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, los cuales son presentados de forma gráfica. (3)

SAFYR: Sistema de Análisis Financiero y Rentable. Es una aplicación que tiene como objetivo principal la administración de portafolios de inversiones de títulos valores. Es una aplicación muy amigable, ágil, segura y lo más importante es su facilidad de manejo y operación. La interfaz gráfica y la automatización que se diseñó para la captura de datos hacen posible todas estas ventajas. Está desarrollado en Microsoft Visual Studio .Net ASP.NET (Páginas Activas de Servidor .NET), con motor de bases de datos en Microsoft SQL Server (Aplicación Cliente Servidor). Cumple con todos los requerimientos de trabajo en un entorno multiusuario, esto significa que varios usuarios desde diferentes máquinas o equipos pueden crear, modificar, ver y administrar datos simultáneamente sin generar ningún tipo de conflicto.

SAFYR dentro de los módulos por los que está compuesto, posee el módulo de consultas para comisionistas y el módulo de reportes.

El módulo de reportes se divide en las siguientes categorías:

- Informes Comerciales.
- Informes Financieros.
- Informes de Transacciones.
- Informes de Portafolios de Inversiones.
- Informes Por Centros de Costos.
- Informes Contables.

Cada una de estas categorías se compone por una variedad de informes o reportes. Destacar que el SAFYR incluye algunos tipos de reportes dentro de otros módulos, por lo que se considera que los reportes presentes en este sistema, no se encuentran centralizados en un mismo módulo. (4)

CELTA: Sistema para entidades de economía solidaria. Es el primer sistema integral de tipo Planeación de Recursos Empresariales (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, 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.(5)

El sistema CELTA se compone por 23 módulos, dentro de los que se destaca el módulo de reportes por su gran utilidad al 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, submódulos y funcionalidades para desplegar los informes asociados. Al realizar una selección, se carga la información en un reporte, que en gran parte de los casos permite ingresar diversos parámetros que amplían su funcionalidad. (6)

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 de ETECSA (Empresa de Telecomunicaciones de Cuba S.A). Mediante una interface 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.

Dentro de los objetivos a la hora de pensar el sistema están:

- Realizar una aplicación que sea capaz de administrar los diferentes flujos de trabajo que se generan en cada servicio solicitado.
- Lograr un sistema integrado de gestión vía Web en el que se conjugarán tres interfaces fundamentales, de usuario, traductor y administrador.
- Permitir que la aplicación realice búsquedas dentro de los datos almacenados, de acuerdo a diferentes criterios de selección.
- Generar reportes para saber el estado actual de las solicitudes que se realizan.
- Incluir la opción de que el usuario pueda ver y modificar sus datos y los documentos siempre que no hayan sido asignados a un traductor. (7)

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. (8)Los 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 maneja actualmente la información docente en la UCI. Akademos 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. (9)

KAINOS Sistema de gestión de la información de la FEU. Hace sólo aproximadamente tres años en la UCI, la FEU decidió desarrollar e implementar un sistema capaz de brindarle un mejor servicio a la comunidad universitaria para mantenerlos informados de lo acontecido en esta importante organización estudiantil de manera rápida y organizada. Actualmente el sistema de gestión de Kainos de la FEU Nacional es el encargado de gestionar todo lo referente a los estudiantes universitarios en el país. Kainos contó desde sus inicios con tres módulos esenciales: módulo de seguridad, el módulo estadístico y el módulo de funcionamiento. Independientemente de estos módulos, el sistema gestiona un conjunto de reportes estáticos, aunque estos no se encuentran de forma centralizada en el sistema.

En el desarrollo del sistema se utilizaron las herramientas y tecnologías siguientes:

- Java como lenguaje de programación.
- Postgre SQL como gestor de base de datos.
- Eclipse como entorno de desarrollo.
- JasperReports como herramienta para generar los reportes.
- Framework Hibernate para el acceso a los datos.
- Framework Spring para el negocio.

Actualmente en el sistema automatizado de gestión de información de la FEU Nacional se están realizando nuevos diseños de módulos como es el caso del módulo de reportes que será capaz de crear reportes dinámicos, lo que beneficiará considerablemente a los usuarios del sistema. (10)

1.2.4 Conclusiones del estudio de los antecedentes

Características	CENACAD	SAFYR	CELTA	Babel	GestCult	AKADEMOS	KAINOS
Exportar Reportes	X						
Reportes Gráficos	X				X		X
Reportes de fácil entendimiento	X					X	X
Parametrización de Reportes	X		X	X		X	X
Módulo de reportes por categorías		X	X				

Tabla 1 Características de sistemas que contiene módulos de reportes.

A partir del estudio realizado, se hizo una selección de sus características, tomando las más ventajosas para nuestra aplicación (**Tabla 1**) y adicionando otras solicitadas por el cliente como la generación de reportes en diferentes formatos. El estudio de estos sistemas sirvió como base para saber el estado actual de los módulos de reportes; permitiendo reducir el grupo de tecnologías, herramientas y librerías más usadas para la creación de los mismos, estos aspectos nos facilitaran el trabajo ya que serán tomados en cuenta a la hora del diseño de la aplicación.

1.3 Herramientas para la generación de reportes

Con el propósito de conocer las principales características, ventajas y desventajas de algunos de las herramientas para la generación de reportes, se realizó un estudio de las mismas, con el objetivo de seleccionar la de mayor afinidad para el desarrollo de la aplicación.

1.3.1 Data Visión:

Data Visión es una herramienta de reporte de código abierto. Los reportes pueden ser diseñados mediante una interfaz gráfica agradable al usuario. Pueden ejecutar los reportes, ver e imprimir desde la aplicación o exportar como HTML, XML, PDF, Excel o archivos de texto delimitados por tabuladores o comas. Data Visión está escrito en Java y es multi-plataforma. Puede generar informes de bases de datos o archivos de datos de texto. Es funcional con cualquier gestor de base de datos con un driver JDBC disponibles: Oracle, PostgreSQL, MySQL, Informix, HSQLDB, Microsoft Access, y muchos más. (11)

1.3.2 Agata Report:

Agata Report es un generador de reportes multi-plataforma, una herramienta de consulta y generación de gráficos que se conecta a varios gestores de bases de datos, como PostgreSQL, MySQL, Oracle, DB2, MS-SQL, Informix, InterBase, Sybase, o Frontbase y permite exportar los reportes en formatos como PostScript, HTML, XML, PDF o CSV (StarCalc, Excel). Permite definir niveles de datos, subtotales y totales para el relatorio. Permite crear documentos, como cartas y conjugar dinámicamente con los datos provenientes del reporte, así como crear etiquetas de direccionamiento y hasta generar un diagrama entidad relación completo a partir de su banco de datos. (12)

1.3.3 iReport:

La herramienta iReport es un constructor/diseñador de informes visual muy poderoso para JasperReports librería Java para la generación de reportes. Este instrumento permite que los usuarios corrijan visualmente informes complejos con cartas, imágenes, subinformes, etc. Está integrado con JFreeChart, una de las bibliotecas gráficas de código abierto más difundidas para Java. Tiene asistentes para las plantillas. Los datos pueden ser recuperados por varios caminos incluso múltiples uniones JDBC, TableModels, JavaBeans, XML, etc.

Características de iReport

- 100% escrito en JAVA y además OPENSOURCE y gratuito.

- Maneja el 98% de las etiquetas de JasperReports.
- Permite diseñar con sus propias herramientas: rectángulos, líneas, elipses, campos de los textfields, cartas, subreportes.
- Soporta JDBC.
- Soporta JavaBeans como orígenes de datos (éstos deben implementar la interface RDataSource).
- Incluye Wizards (asistentes) para crear de informes automáticamente, generación de subreportes y plantillas.
- Facilidad de instalación. (13)

1.3.3.1 JasperReports:

JasperReports es una librería Java para la generación de reportes, pudiéndolos exportar a diferentes formatos (usualmente PDF). Es una de las herramientas de reportes open source más usadas, y es específicamente para y realizada en Java.
Posee una muy buena documentación, así como muchísimos ejemplos (en cada uno de estos ejemplos se puede apreciar las variantes que permite este software). Los reportes además de poder ser impresos, pueden ser exportados a:

- PDF
- HTML
- XLS
- CVS
- XML

JasperReports permite organizar la información obtenida desde una base de datos relacional, a través de contenedores JDBC, en diseños de reportes predefinidos en un formato XML. (14)
El diseño de un reporte se realiza editando un archivo XML, que representa la estructura de dicho reporte, este archivo XML puede ser editado a través de una herramienta integrada al JasperReports llamado iReport, herramienta visual que permite el diseño y construcción de los reportes. (15)

1.3.4 Conclusiones de las herramientas para la generación de reportes:

Después de haber realizado una investigación de las principales herramientas de generación de reportes y teniendo en cuenta las necesidades del cliente en cuanto a

las características que debe presentar el sistema se decide hacer uso de la herramienta iReport, por tener una excelente integración con la librería JasperReports ya que maneja un 98% de las etiquetas de esta librería, esta escrito100% en java, es *opensource*[2] y gratuito. Permite la visualización y exportación de los reportes en diferentes formatos.

1.4 Metodología de desarrollo

Las metodologías de desarrollo de software definen el cómo trabajar eficientemente evitando las problemáticas que conllevan al fracaso de muchos proyectos de software. El objetivo fundamental de una metodología es aumentar la calidad del software, haciendo énfasis en la calidad y menor tiempo de construcción del software o lo que es lo mismo "producir lo esperado en el tiempo esperado y con el coste esperado".

1.4.1 Proceso Unificado de Desarrollo (RUP).

"RUP es una plataforma de procesos completos que contiene una base de conocimientos y herramientas para la edición, la configuración y el despliegue de procesos configurados". {2}

Los proyectos de desarrollo de software no están definidos en sus inicios, y requieren ciclos severos para arribar a la solución final. RUP provee un marco genérico que puede especializarse para una gran cantidad de sistemas de software. Se caracteriza por ser:

- Iterativo e incremental: Cada fase se desarrolla en iteraciones. Se divide el trabajo en partes pequeñas o mini-proyectos, donde cada uno es una iteración que resulta en un incremento.
- Centrado en la arquitectura: La arquitectura describe los elementos del modelo que son importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

[2] Código abierto

- Guiado por los 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. {3}

RUP describe cuatro fases: Inicio, Elaboración, Construcción y Transición, fases que un proyecto va a atravesar (Fig. 2).

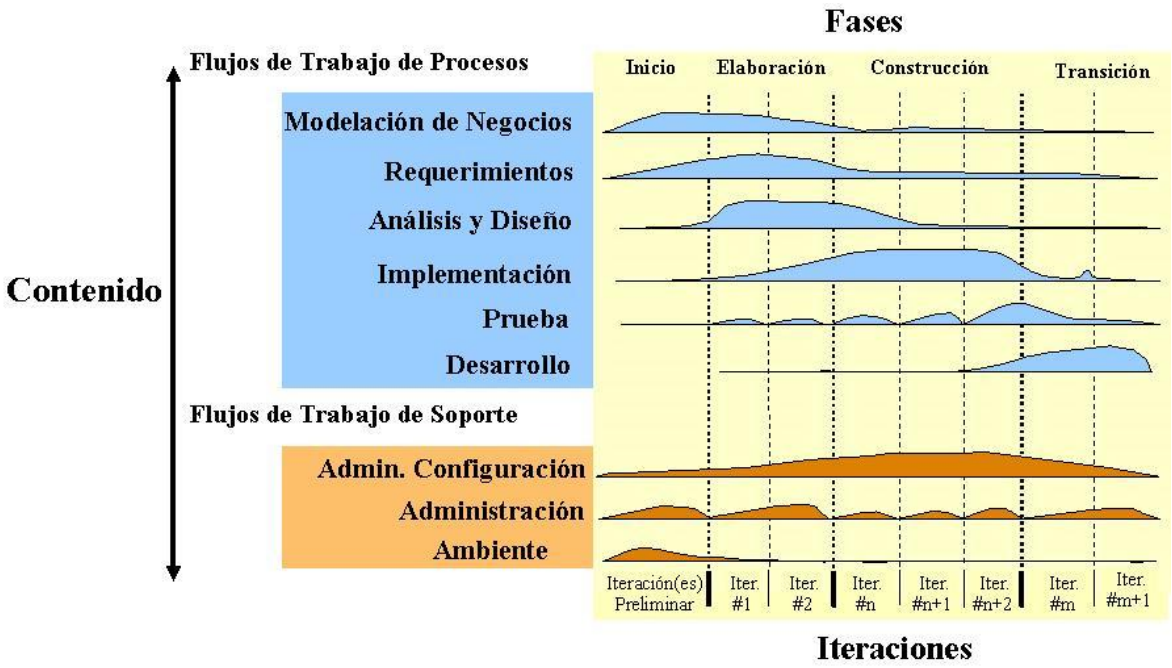


Fig. 2 Fases e Iteraciones de la Metodología RUP

Los objetivos de cada una de las fases son los siguientes:

- Inicio: Describir el negocio y delimitar el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.
- Elaboración: Definir la arquitectura del sistema y obtener una aplicación ejecutable que responde a los casos de uso que la comprometen.

- Construcción: Obtener un producto listo para su utilización que está documentado y tiene un manual de usuario. Obtener uno o varios release[3] del producto que han pasado las pruebas. Poner estos release a consideración de un subconjunto de usuarios.
- Transición: Instalar el release en las condiciones reales. (16)

En RUP se han agrupado las actividades en grupos lógicos definiéndose una serie de flujos de trabajo. Algunos de estos flujos son:

- Modelamiento de negocio: describe los procesos del negocio, identificando quiénes participan y las actividades que requieren automatizar.
- Requerimientos: define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las instrucciones que se imponen.
- Análisis y diseño: describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- Implementación: define cómo se organizan la clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación de ellos en los componentes y la estructura de capas de la aplicación.
- Pruebas: busca los defectos a lo largo del ciclo de vida del producto de software.
- Despliegue: produce liberaciones del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales. (17)

1.4.2 Extreme Programing (XP)

Programación extrema del inglés Extreme Programing es una de las conocidas metodologías ágiles de desarrollo del software. Esta metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requerimientos para llegar al éxito del proyecto.

[3] *Entrega del producto terminado.*

Características de XP:

- Pruebas unitarias: se basan en las pruebas realizadas a los principales procesos, de tal manera que adelante en algo hacia el futuro, se pueda hacer pruebas de las fallas que puedan ocurrir. Es como si se adelantara a obtener los posibles errores.
- Re-fabricación: se basa en la reutilización de código, para la cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- Programación en pares: 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.

XP plantea que el análisis y diseño no se debe ausentar en un ciclo de desarrollo de software. El análisis se efectúa de forma muy ligera, mientras que el diseño, si se lleva a profundidad como la mayoría de las metodologías. El análisis se transforma en la exposición por parte del cliente de las historias de usuarios para lograr un entendimiento de lo que se quiere. Luego continúa el diseño global del sistema, en el que se profundiza hasta el nivel necesario para que los desarrolladores sepan exactamente que van a hacer. (18)

Para lograr el éxito esperado al aplicar esta metodología, el cliente debe estar presente en el ambiente de desarrollo. Requiere además de recursos suficientes para el desarrollo de software en pares. Estos requisitos constituyen inconvenientes para el equipo de desarrollo.

1.4.3 OpenUP

Es un Proceso Unificado iterativo e incremental, que se centra en el desarrollo de software colaborativo. Está basado en casos de uso, la gestión del riesgo, y una arquitectura centrada a impulsar el desarrollo. {4}

Principios de OpenUp

Los cuatro principios básicos en los que se fundamenta OpenUp son los siguientes:

- Promover la colaboración para alinear los intereses comunes y difundir el conocimiento: Este principio promueve la colaboración, desarrollo y

compartición de un conocimiento sobre el proyecto en toda la organización, así como un entorno de trabajo saludable.

- Balanceo de las prioridades competitivas para maximizar el valor de los participantes del proyecto: Se intentan promover prácticas que permitan a los participantes del proyecto el desarrollo de una solución que maximice sus beneficios aportando el máximo valor de negocio.
- Desarrollo de una arquitectura al principio con el fin de minimizar riesgos y planificar el desarrollo: Ejercer prácticas que permitan al equipo focalizarse en la arquitectura del sistema con el fin de reducir riesgos y que permitan planificar la estrategia de desarrollo.
- Evolucionar para obtener un *feedback*[4] continuo en aras de una mejora continua: Con este principio se intentan ejecutar prácticas que permitan obtener un feedback desde el inicio del proyecto y de forma continua, así como mostrar el valor de negocio a los participantes del proyecto.

OpenUP describe cuatro fases: Inicio, Elaboración, Construcción y Transición. Cada una de sus fases tiene como objetivo lo siguiente ***Fig. 3***

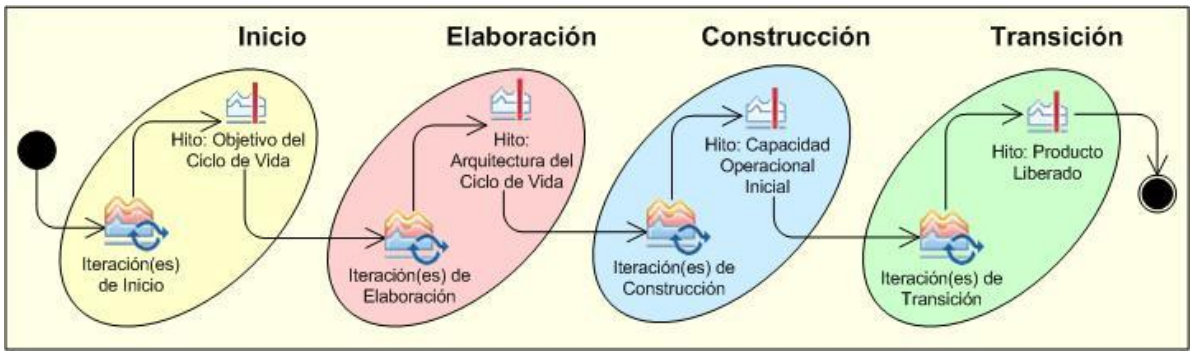


Fig. 3 Ciclo de Vida

- **Fase de inicio:** Las necesidades de cada participante del proyecto son tenidas en cuenta y son plasmadas en objetivos del proyecto. Se deben definir el ámbito del proyecto, los límites del mismo y el criterio de aceptación del proyecto. Los casos de uso críticos, aquellos que dirigen la funcionalidad del

[4] Retroalimentación.

sistema, son definidos en esta fase, así como una estimación inicial del coste del proyecto y un boceto de la planificación.

- **Fase de elaboración:** En esta fase se realizan tareas de análisis del dominio y definición de la arquitectura del sistema. Si se decide continuar con el proyecto se debe elaborar un plan de proyecto en esta fase, para lo cual se deben establecer unos requisitos y arquitectura estables. Por otro lado el proceso de desarrollo, las herramientas, la infraestructura a utilizar y el entorno de desarrollo también se especifican en detalle en esta fase. Al final de la fase se debe tener una definición clara y precisa de los casos de uso, los actores, la arquitectura del sistema y un prototipo ejecutable de la misma.
- **Fase de construcción:** Todos los componentes y funcionalidades del sistema que falten por implementar son realizados, testeados e integrados en esta fase. Los resultados obtenidos en forma de incrementos ejecutables deben ser desarrollados de la forma más rápida posible sin dejar de lado la calidad de lo desarrollado.
- **Fase de transición:** Cuando el producto está lo suficientemente maduro (algo que es establecido por ejemplo en función del número de peticiones de cambio por parte de los clientes) como para ser introducido en la comunidad de usuarios, el proyecto se encuentra en esta fase. Las fases de la transición constan de sub-fases de testeo de versiones beta, pilotaje y capacitación de los usuarios finales y de los encargados del mantenimiento del sistema. En función de la respuesta obtenida por los usuarios puede que haya que realizar cambios en las entregas finales o implementar alguna funcionalidad más.

El proceso es mínimo pues solamente el contenido fundamental es incluido; completo porque puede ser manifestado como todo el proceso para construir un sistema y es extensible pues puede ser utilizado como fundamento sobre el cual el contenido de procesos se pueda agregar o adaptar según lo necesitado.

Los roles de OpenUp que recogen el conjunto esencial de habilidades necesarias para realizar los desarrollos son los siguientes:

- **Cliente:** Representan el conjunto de individuos o entidades cuyas necesidades se verán cubiertas por el proyecto. Cualquier individuo que se vea afectado por el resultado del proyecto puede adoptar este rol.

- **Analista:** Dentro del equipo es el encargado de representar al cliente y sus necesidades. Es el encargado de recoger las impresiones de los clientes con el fin de entender el problema y establecer las prioridades de los requisitos identificados.
- **Arquitecto:** Es el encargado de realizar la arquitectura general del sistema y tomar las decisiones técnicas claves que influenciaran en gran medida el futuro diseño e implementación del proyecto.
- **Desarrollador:** El diseño, implementación, testeo unitario e integración de una parte del sistema es una tarea llevada a cabo por este rol.
- **Testeador:** Como su nombre indica, es el rol encargado de identificar, diseñar e implementar y ejecutar los tests necesarios, además de dejar registro de los resultados obtenidos en dichos tests.
- **Gestor del proyecto:** Es el encargado de dirigir la planificación del proyecto en colaboración con el equipo y los clientes. Asimismo debe conseguir que el equipo se mantenga focalizado en cumplir los objetivos planificados. (19)

1.4.4 Conclusiones de la Metodología de desarrollo:

Se decide optar por OpenUp, metodología que permite abordar ágilmente el desarrollo de proyectos pequeños y tiene un enfoque centrado en el cliente con iteraciones cortas. Este proceso de desarrollo unificado está basado en las mejores prácticas de RUP, lo que permite que el tiempo de capacitación (curva de aprendizaje) del personal sea mínimo ya que el mismo posee experiencia con el uso de RUP. Permite el desarrollo de artefactos ligeros y apropiados, utilizando el lenguaje de modelado unificado (UML), aplica un enfoque iterativo e incremental dentro de su estructura del ciclo de vida, tiene la peculiaridad de tener solamente cuatro fases. Debido a estas características el nivel de compatibilidad con REKO es muy alto, pues el software fue desarrollado basado casos de usos (CU) y pose un equipo de trabajo pequeño.

1.5 Herramientas CASE para el diseño

Las herramientas CASE facilitan el modelamiento del sistema a los ingenieros de software y analistas el desarrollo del software de principio a fin o en alguna de sus

fases. Fueron creadas para automatizar las distintas etapas del producto y facilitar la organización de las tareas a cumplir durante su ciclo de vida. Aceleran el desarrollo de los sistemas y aumentan la calidad del software. A continuación se exponen las principales características de tres de las herramientas CASE utilizadas en la UCI:

1.5.1 Rational Rose Enterprise Edition

Rational Rose es una herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que 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.

Dentro de las funcionalidades que soporta el Rational Rose están:

- Soporte para compilación y decompilación de las construcciones más habituales de Java 1.5.
- La generación de código ADA, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo-código configurables.
- Publicación web y generación de informes para optimizar la comunicación dentro del equipo.

Se integra con herramientas Ambiente de Desarrollo Integrado (IDE, por sus siglas en inglés) como:

- Borland Builder versiones 7.0 a 10.0.
- Sun Forte for Java Community y Enterprise Editions.
- Microsoft Visual Studio 6.
- Microsoft Visual Studio 2005.
- Wind River Tornado.
- Green Hills MULTI. (20)

1.5.2 Enterprise Architect

Es una herramienta CASE orientada a objetos que provee su alcance para el desarrollo de sistemas, administración de proyectos y análisis de negocios. Maneja totalmente el ciclo de vida de desarrollo de software, utilizando el UML como leguaje de modelado. Facilita y soporta el levantamiento de requerimientos, el análisis y diseño, las pruebas y mantenimiento del software en desarrollo. Soporta un

impresionante rango de lenguajes de desarrollo, incluyendo ActionScript, C, C++, C# y VB.NET, Java, Visual Basic 6, Python, PHP, XSD, WSDL y otros más. Gestiona la ingeniería de código, normal e inversa, además de una efectiva documentación compatible con Microsoft Word.

Dentro de las funcionalidades que soporta el Enterprise Architect se encuentran:

- Creación de diagramas UML.
- Creación de diagramas de casos de uso, y modelos lógicos, dinámicos y físicos.
- Extensiones personalizadas para modelado de procesos.
- Documentación de alta calidad compatible con Microsoft Word.
- Modelado de Datos.
- Ingeniería directa de Base de Datos a DDL e ingeniería inversa de Base de Datos desde ODBC.
- Soporte de pruebas.
- Aplicación Multi-usuario, con sistema de seguridad. (21)

1.5.3 Visual Paradigm for UML

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

Dentro de las funcionalidades que soporta el Visual Paradigm se encuentran:

- Administración de requerimientos.
- Modelado de procesos del negocio.
- Modelado de base de datos.
- Generación de código.
- Ingeniería inversa.
- Diseño de prototipos de interfaz de usuario.

Se integra con herramientas Java como:

- Eclipse.
- JBuilder.
- NetBeans/sun ONE.
- Weblogic Workshop.
- JDeveloper. (22)

1.5.4 Conclusiones de la Herramienta CASE para el diseño:

Se decide emplear como herramienta CASE Visual Paradigm, debido a que cuenta con la ventaja de ser multiplataforma a diferencia de las otras dos herramientas explicadas; soporta el ciclo de vida completo del desarrollo de software. Permite la integración con Eclipse IDE. La herramienta también proporciona abundantes tutoriales y demostraciones. Aunque Visual Paradigm es una herramienta privativa la UCI posee las licencias necesarias para su uso.

1.6 Conclusiones del CAPÍTULO 1:

Basado en los aspectos abordados durante desarrollo de este capítulo se puede concluir que:

- ✓ El análisis de las características en las diferentes aplicaciones que contienen módulos de reportes utilizadas a nivel mundial, en Cuba y en la UCI, permitieron saber el estado actual del proceso de generación de reportes.

- ✓ Se expusieron las diferentes herramientas y metodología que se usarán en el desarrollo de la aplicación, para lograr un resultado satisfactorio. Se caracterizó cada una de ellas para su mejor entendimiento y en breves palabras se explica el porqué de su adopción.

CAPÍTULO 2: CARACTERÍSTICAS Y DISEÑO DEL SISTEMA

2.1 Introducción:

En este capítulo se describe una propuesta de solución del módulo de reportes para REKO, siguiendo el problema científico planteado. Además se recogen los requisitos funcionales y no funcionales del sistema, también se definen los actores y casos de uso del sistema.

2.2 Requisitos del software

Un requerimiento (requisito) es una "condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen. Se clasifican en funcionales y no funcionales." {5}

2.2.1 Requisitos funcionales del sistema.

RF_1 Crear Reporte de Acciones de Réplica.

RF_1.1 Filtrar reporte por tipo de acción.
RF_1.2 Filtrar reporte por dirección.
RF_1.3 Filtrar reporte por ID del nodo.
RF_1.4 Filtrar reporte por rango de fecha.
RF_1.5 Seleccionar formato del reporte: Word, PDF o HTML.
RF_1.6 Generar reporte.
RF_1.7 Visualizar reporte en el formato seleccionado.
RF_1.8 Exportar reporte en el formato seleccionado.
RF_1.9 Imprimir reporte.

RF_2 Crear Reporte Gráfico de Acciones de Réplica.

RF_2.1 Filtrar reporte gráfico por: Tipo de acción o Tipo de dirección.
RF_2.2 Seleccionar formato del reporte: Word o PDF.
RF_2.3 Generar reporte.

RF_2.4 Visualizar reporte en el formato seleccionado.

RF_2.5 Exportar reporte a formato seleccionado.

RF_2.6 Imprimir reporte.

RF_3 Crear Reporte de Conflictos.

RF_3.1 Filtrar reporte por tipo de acción.

RF_3.2 Seleccionar Formato del reporte: Word, PDF o HTML.

RF_3.3 Generar reporte.

RF_3.4 Visualizar reporte en el formato seleccionado.

RF_3.5 Exportar reporte a formato seleccionado.

RF_3.6 Imprimir reporte.

RF_4 Crear Reporte Gráfico de Conflictos.

RF_4.1 Filtrar reporte gráfico por: Tipo de acción o Tipo de conflicto.

RF_4.2 Seleccionar formato del reporte: Word o PDF.

RF_4.3 Generar reporte.

RF_4.4 Visualizar reporte en el formato seleccionado.

RF_4.5 Exportar reporte a formato seleccionado.

RF_4.6 Imprimir reporte.

RF_5 Crear Reporte de Sincronización.

RF_5.1 Seleccionar Base de dato que desea usar: BD_ConflicSync o BD_Sync.

RF_5.2 Filtrar reporte por tipo de acción.

RF_5.3 Seleccionar Formato del reporte: Word, PDF o HTML.

RF_5.4 Generar reporte.

RF_5.5 Visualizar reporte en el formato seleccionado.

RF_5.6 Exportar reporte a formato seleccionado.

RF_5.7 Imprimir reporte.

RF_6 Crear Reporte de Exportación.

RF_6.1 Filtrar reporte por ID del nodo.

RF_6.2 Filtrar reporte por rango de fecha.

RF_6.3 Seleccionar formato del reporte: Word, PDF o HTML.

RF_6.4 Generar reporte.

RF_6.5 Visualizar reporte en el formato seleccionado.

RF_6.6 Exportar reporte en el formato seleccionado.

RF_6.7 Imprimir reporte.

2.2.2 Requisitos no funcionales del sistema

Los requerimientos no funcionales son “propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable”. {6}

- Seguridad: La información manejada por el módulo contará de protección ante intrusos y accesos no autorizados, será vista únicamente por aquellos usuarios que tengan autorización. El módulo controlará los diferentes niveles de acceso y funcionalidad de usuarios al sitio; o sea, prioriza la identificación del usuario antes de que sea capaz de realizar cualquier acción sobre el sistema.

- Portabilidad: El módulo será multiplataforma, es decir, debe poder ejecutarse sobre los Sistemas Operativos Windows y Linux.

- Usabilidad: La aplicación Web podrá ser navegada por cualquier usuario con conocimientos básicos de computación y sobre el ambiente Web. Será de fácil aprendizaje, logrando que los usuarios tengan una plena satisfacción con su uso.

- Rendimiento: La aplicación para cada solicitud del usuario tendrá una respuesta en pocos segundos para lograr que la gestión de la información sea efectiva. Las páginas solicitadas no contendrán grandes volúmenes de imágenes para evitar retrasos innecesarios. Además, debe permitir conexiones simultáneas.

- Apariencia o interfaz externa: El diseño de la interfaz será sencillo, amigable, de fácil navegación para el usuario y con reconocimiento visual a través de elementos visibles que identifiquen cada una de sus acciones. El producto

debe ser legible con colores agradables a la vista del usuario y mantener un formato estándar en todas las páginas.

- Confiabilidad: Dentro de este requerimiento se encuentran la integridad y la disponibilidad.
  - Integridad: La información será protegida contra corrupción y estados inconsistentes.
  - Disponibilidad: Estará disponible las 24 horas del día durante todo el año. Los usuarios autorizados tendrán acceso a la información siempre que lo deseen.

- Software: Hay que tener en cuenta los programas que deben tener instalado la PC cliente y la PC servidora.
  - Navegador: Mozilla Firefox 3.0.
  - Sistema operativo: Linux o Windows 98 o superior.
  - Servidor de ftp: ProFTPd
  - Servidor de mensajería: Active MQ
  - Servidor de Base de Datos: Oracle o Postgres.

- Hardware: Hay que tener en cuenta los requerimientos mínimos para el cliente y para el servidor.
  - Microprocesador: Pentium III o superior.
  - RAM: 128MB
  - Tarjeta de red.

2.3 Modelo de casos de uso del sistema

El modelo de casos de uso del sistema contiene los actores, casos de uso y sus relaciones. Cada caso de uso del modelo se describe detalladamente, mostrando paso a paso el modo en que el sistema interactúa con los actores y lo que el sistema hace en el caso de uso. Identificar los actores que interactuarán con el sistema es una de las tareas más importantes, permite definir las fronteras y esquematizar la funcionalidad del sistema.

2.3.1 Actores del Sistema

Son terceros fuera del sistema que interactúan con él.

Actores	Descripción
Usuario	Es el encargado de realizar las operaciones sobre los reportes, favoreciéndose de las ventajas que ofrece la aplicación para obtener información.

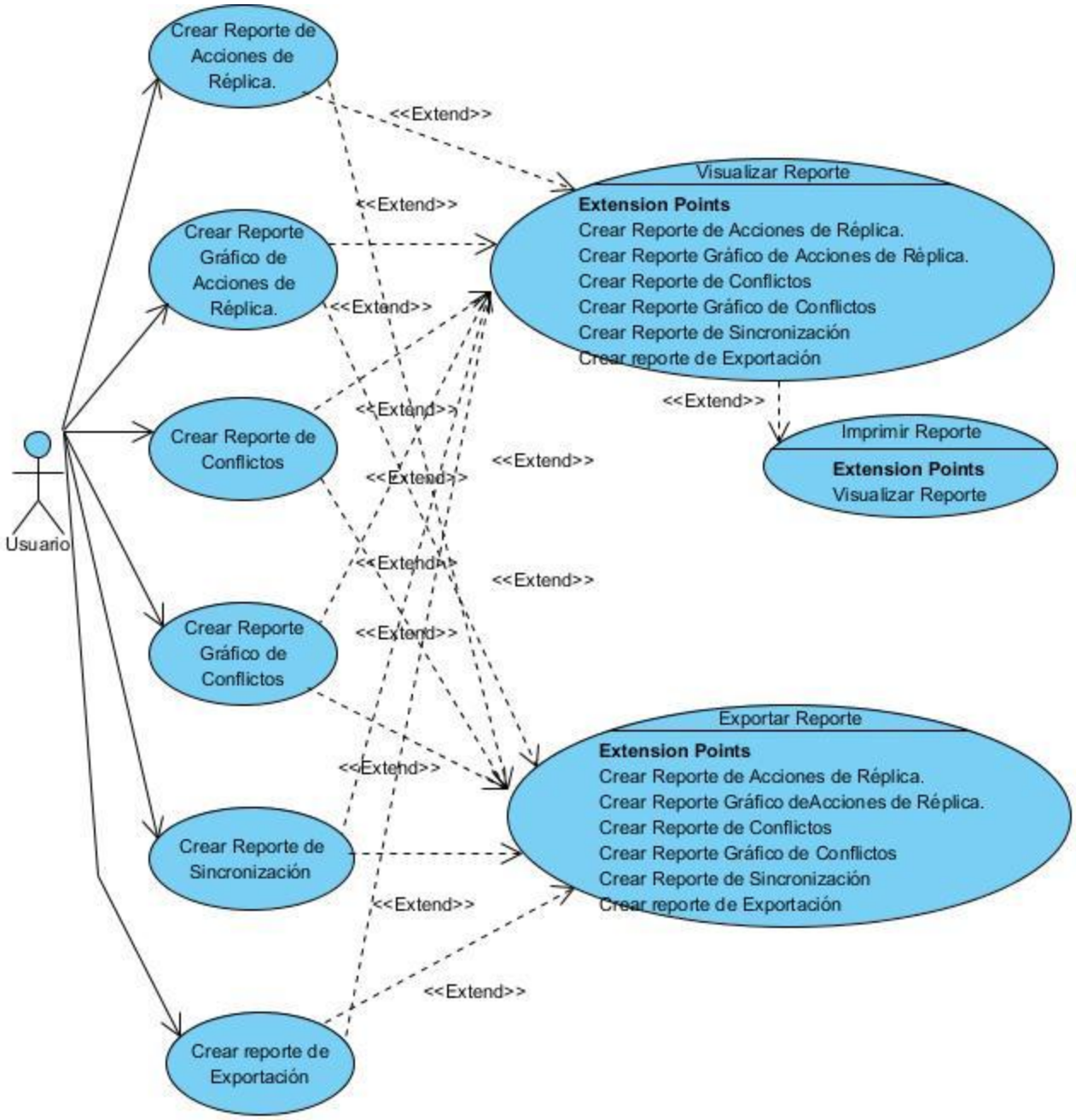
Tabla 2 Actores del Sistema

2.3.2 Casos de Uso del Sistema

1. Crear Reporte de Acciones de Réplica.
2. Crear Reporte Gráfico de Acciones de Réplica.
3. Crear Reporte de Conflictos.
4. Crear Reporte Gráfico de Conflictos.
5. Crear Reporte de Sincronización.
6. Crear Reporte de Exportación.
7. Visualizar Reporte
8. Exportar Reporte
9. Imprimir Reporte

2.3.3 Diagrama de Casos de Uso del Sistema

Los diagramas de casos de uso del sistema representan a los actores, casos de uso y sus relaciones.



Diag. 1 Diagrama de Casos de Uso

2.3.4 Descripción textual de los CU del Sistema

Caso de uso del negocio:	Crear Reporte de Acciones de Réplica
Casos de uso asociados:	1. Visualizar Reporte

	2. Exportar Reporte 3. Imprimir Reporte
Resumen: El caso de uso inicia cuando el Usuario se ha autenticado en el sistema y necesita crear un reporte de acciones de réplica.	
Referencias:	RF_1
Prioridad:	Crítico
Actores del negocio:	Usuario
Precondición:	El usuario debe haberse autenticado en el sistema.
Curso normal de eventos **Acción del actor**	**Respuesta del negocio**
Escenario: Crear Reporte de Acciones de Réplica.	
1. El usuario selecciona Crear Reporte de Acciones de Réplica.	2. Muestra un formulario con los parámetros disponibles para este tipo de reporte.
3. El usuario selecciona los parámetros.	
4. Oprime el botón "Generar Reporte".	5. Muestra una ventana con la opción de mostrar o exportar el reporte.
6. El usuario selecciona mostrar reporte.	7. Muestra el Reporte según los parámetros especificados
Flujos Alternos 1	
8. El usuario selecciona exportar reporte.	9. Muestra una ventana con la dirección donde se guardará el

	reporte.
10. El usuario guarda el reporte.	
Flujos Alternos 2	
10. El usuario selecciona imprimir Reporte	11. Se imprime el Reporte

Tabla 3 Descripción textual del CU Crear Reporte de Acciones de Réplica.

Caso de uso del negocio:	Crear Reporte Gráfico de Acciones de Réplica
Casos de uso asociados:	1. Visualizar Reporte 2. Exportar Reporte 3. Imprimir Reporte
Resumen: El caso de uso inicia cuando el Usuario se ha autenticado en el sistema y necesita crear un reporte gráfico de acciones de réplica.	
Referencias:	RF_2
Prioridad:	Crítico
Actores del negocio:	Usuario
Precondición:	El usuario debe haberse autenticado en el sistema.
Curso normal de eventos **Acción del actor**	**Respuesta del negocio**
Escenario: Crear Reporte Gráfico de Acciones de Réplica.	

1. El usuario selecciona Crear Reporte Gráfico de Acciones de Réplica.	2. Muestra un formulario con los parámetros disponibles para este tipo de reporte.
3. El usuario selecciona los parámetros.	
4. Oprime el botón "Generar Reporte".	5. Muestra una ventana con la opción de mostrar o exportar el reporte.
6. El usuario selecciona mostrar reporte.	7. Muestra el Reporte según los parámetros especificados
Flujos Alternos 1	
6. El usuario selecciona exportar reporte.	7. Muestra una ventana con la dirección donde se guardará el reporte.
8. El usuario guarda el reporte.	
Flujos Alternos 2	
8. El usuario selecciona imprimir Reporte	9. Se imprime el Reporte

Tabla 4 Descripción textual del CU Crear Reporte Gráfico de Acciones de Réplica.

Caso de uso del negocio:	Crear Reporte de Conflictos
Casos de uso asociados:	1. Visualizar Reporte 2. Exportar Reporte

	3. Imprimir Reporte
Resumen: El caso de uso inicia cuando el Usuario se ha autenticado en el sistema y necesita crear un reporte de conflictos.	
Referencias:	RF_3
Prioridad:	Crítico
Actores del negocio:	Usuario
Precondición:	El usuario debe haberse autenticado en el sistema.
Curso normal de eventos **Acción del actor**	**Respuesta del negocio**
Escenario: Crear Reporte de Conflictos.	
1. El usuario selecciona Crear Reporte de Conflictos.	2. Muestra un formulario con los parámetros disponibles para este tipo de reporte.
3. El usuario selecciona los parámetros.	
4. Oprime el botón "Generar Reporte".	5. Muestra una ventana con la opción de mostrar o exportar el reporte.
6. El usuario selecciona mostrar reporte.	7. Muestra el Reporte según los parámetros especificados
Flujos Alternos 1	
6. El usuario selecciona exportar reporte.	7. Muestra una ventana con la dirección donde se guardará el reporte.

8. El usuario guarda el reporte.	
Flujos Alternos 2	
8. El usuario selecciona imprimir Reporte	9. Se imprime el Reporte

Tabla 5 Descripción textual del CU Crear Reporte de Conflictos.

Caso de uso del negocio:	Crear Reporte Gráfico de Conflictos.
Casos de uso asociados:	1. Visualizar Reporte 2. Exportar Reporte 3. Imprimir Reporte
Resumen: El caso de uso inicia cuando el Usuario se ha autenticado en el sistema y necesita crear un reporte gráfico de conflictos.	
Referencias:	RF_4
Prioridad:	Crítico
Actores del negocio:	Usuario
Precondición:	El usuario debe haberse autenticado en el sistema.
Curso normal de eventos **Acción del actor**	**Respuesta del negocio**
Escenario: Crear Reporte Gráfico de Conflictos.	
1. El usuario selecciona Crear Reporte Gráfico de Conflictos.	2. Muestra un formulario con los parámetros disponibles para este tipo de reporte.
3. El usuario selecciona los	

parámetros.	
4. Oprime el botón "Generar Reporte".	5. Muestra una ventana con la opción de mostrar o exportar el reporte.
6. El usuario selecciona mostrar reporte.	7. Muestra el Reporte según los parámetros especificados
Flujos Alternos 1	
6. El usuario selecciona exportar reporte.	7. Muestra una ventana con la dirección donde se guardará el reporte.
8. El usuario guarda el reporte.	
Flujos Alternos 2	
8. El usuario selecciona imprimir Reporte	9. Se imprime el Reporte

Tabla 6 Descripción textual del CU Crear Reporte Gráfico de Conflictos.

Caso de uso del negocio:	Crear Reporte de Sincronización.
Casos de uso asociados:	1. Visualizar Reporte 2. Exportar Reporte 3. Imprimir Reporte
Resumen: El caso de uso inicia cuando el Usuario se ha autenticado en el sistema y necesita crear un reporte de sincronización.	
Referencias:	RF_5
Prioridad:	Crítico

Actores del negocio:	Usuario
Precondición:	El usuario debe haberse autenticado en el sistema.
Curso normal de eventos **Acción del actor**	**Respuesta del negocio**
Escenario: Crear Reporte de Sincronización.	
1. El usuario selecciona Crear Reporte de Sincronización.	2. Muestra un formulario con los parámetros disponibles para este tipo de reporte.
3. El usuario selecciona los parámetros.	
4. Oprime el botón "Generar Reporte".	5. Muestra una ventana con la opción de mostrar o exportar el reporte.
6. El usuario selecciona mostrar reporte.	7. Muestra el Reporte según los parámetros especificados
Flujos Alternos 1	
6. El usuario selecciona exportar reporte.	7. Muestra una ventana con la dirección donde se guardará el reporte.
8. El usuario guarda el reporte.	
Flujos Alternos 2	
8. El usuario selecciona imprimir Reporte	9. Se imprime el Reporte

Tabla 7 Descripción textual del CU Crear Reporte de Sincronización.

Caso de uso del negocio:	Crear Reporte de Exportación.
Casos de uso asociados:	4. Visualizar Reporte 5. Exportar Reporte 6. Imprimir Reporte
Resumen: El caso de uso inicia cuando el Usuario se ha autenticado en el sistema y necesita crear un reporte de Exportación.	
Referencias:	RF_6
Prioridad:	Crítico
Actores del negocio:	Usuario
Precondición:	El usuario debe haberse autenticado en el sistema.
Curso normal de eventos **Acción del actor**	**Respuesta del negocio**
Escenario: Crear Reporte de Exportación.	
6. El usuario selecciona Crear Reporte de Exportación.	7. Muestra un formulario con los parámetros disponibles para este tipo de reporte.
8. El usuario selecciona los parámetros.	
9. Oprime el botón "Generar Reporte".	10. Muestra una ventana con la opción de mostrar o exportar el reporte.
8. El usuario selecciona mostrar reporte.	9. Muestra el Reporte según los parámetros especificados

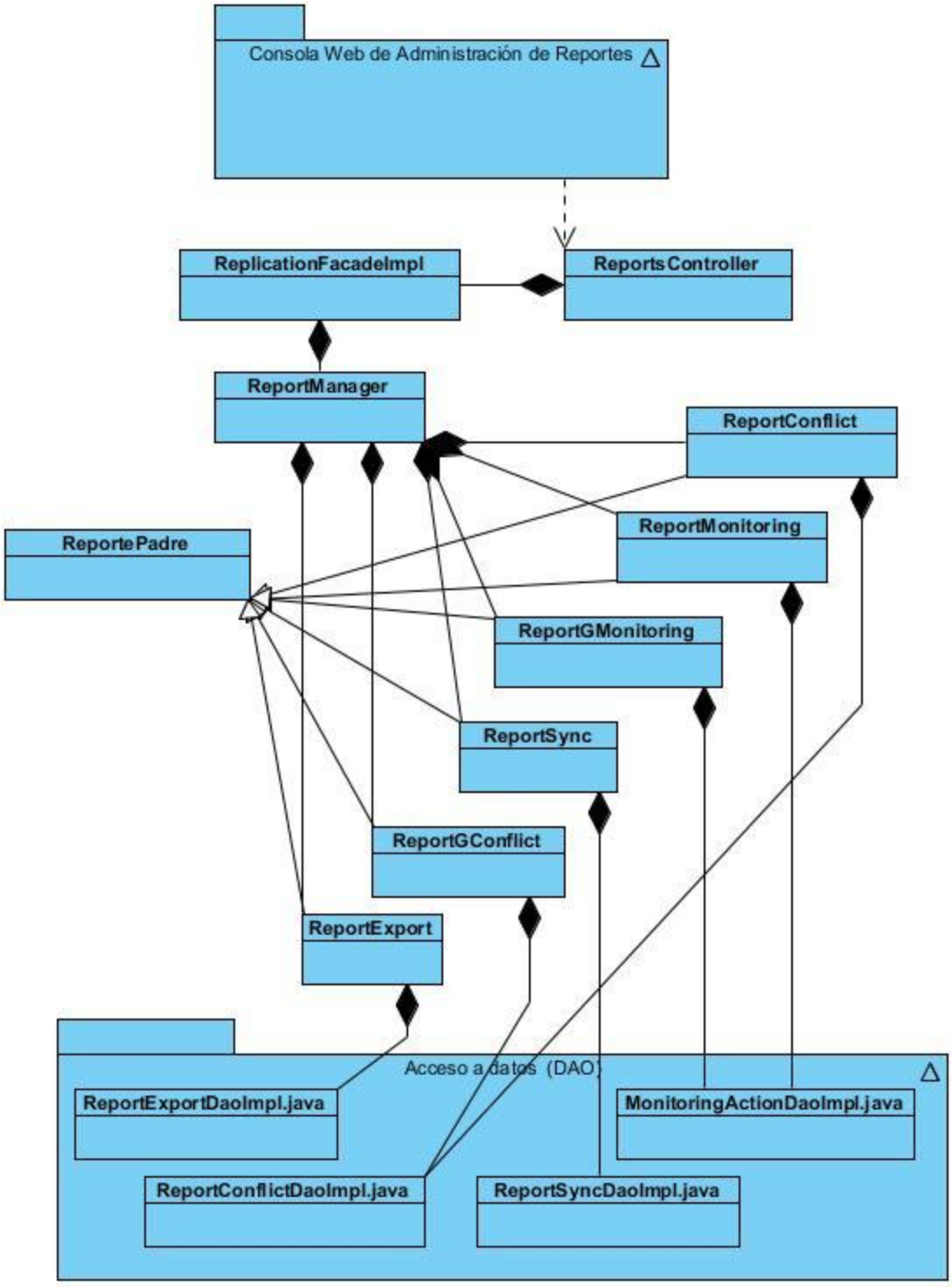
Flujos Alternos 1	
9. El usuario selecciona exportar reporte.	10. Muestra una ventana con la dirección donde se guardará el reporte.
11. El usuario guarda el reporte.	
Flujos Alternos 2	
10. El usuario selecciona imprimir Reporte	11. Se imprime el Reporte

Tabla 8 Descripción textual del CU Crear Reporte de Exportación.

2.3.5 Descripción de las Clases del Modelo de Sistema.

- DAO: contiene la información acerca del proceso de réplica, que se utiliza para la generación de los reportes.
- Reporte Padre: contiene un conjunto de métodos y atributos para generar los reportes.
- ReportMonitoring: contienen accesos a los datos específicos para los reportes relacionados con el módulo de monitoreo.
- ReportGMonitoring: contienen accesos a los datos específicos para los reportes gráficos relacionados con el módulo de monitoreo.
- ReportConflict: contienen accesos a los datos específicos para los reportes gráficos relacionados con el módulo de conflicto.
- ReportGConflict: contienen accesos a los datos específicos para los reportes relacionados con el módulo de conflictos.
- ReportSync: contienen accesos a los datos específicos para los reportes relacionados con el módulo de sincronización.
- ReportExport: contienen accesos a los datos específicos para los reportes relacionados con el módulo de exportación.
- ReportManager: contiene las instancias de las sub-clases Report para centralizar el acceso a las mismas.

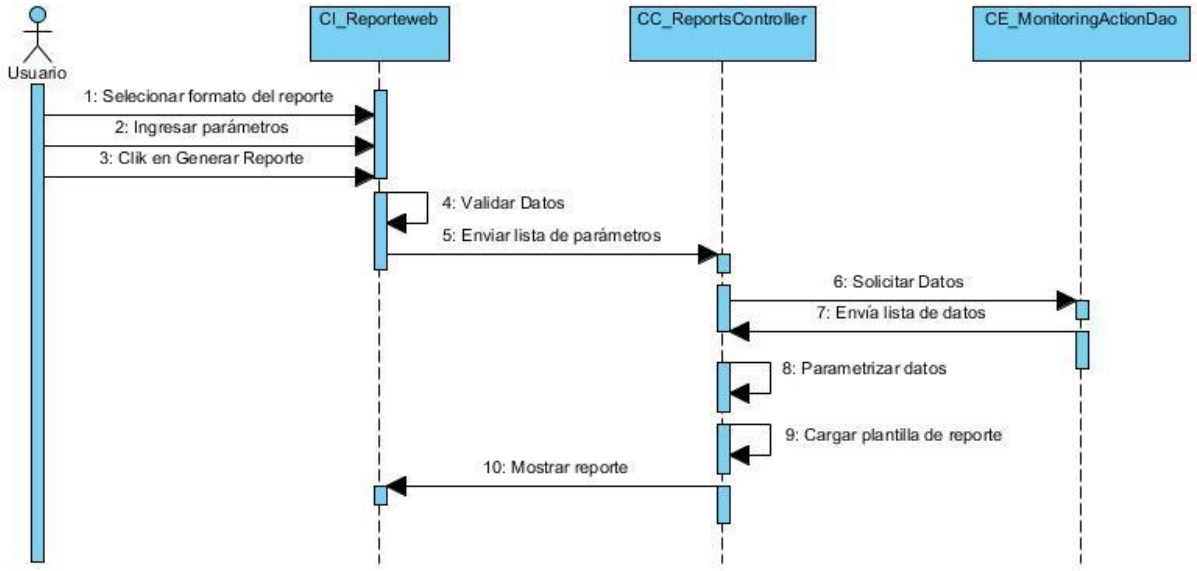
- ReplicationFacadeImpl: contiene las instancias de todos los módulos de REKO para tener un acceso centralizado a las mismas.
- ReportsController: es la clase controladora encargada de la integración entre las interfaces web y las clases entidades.



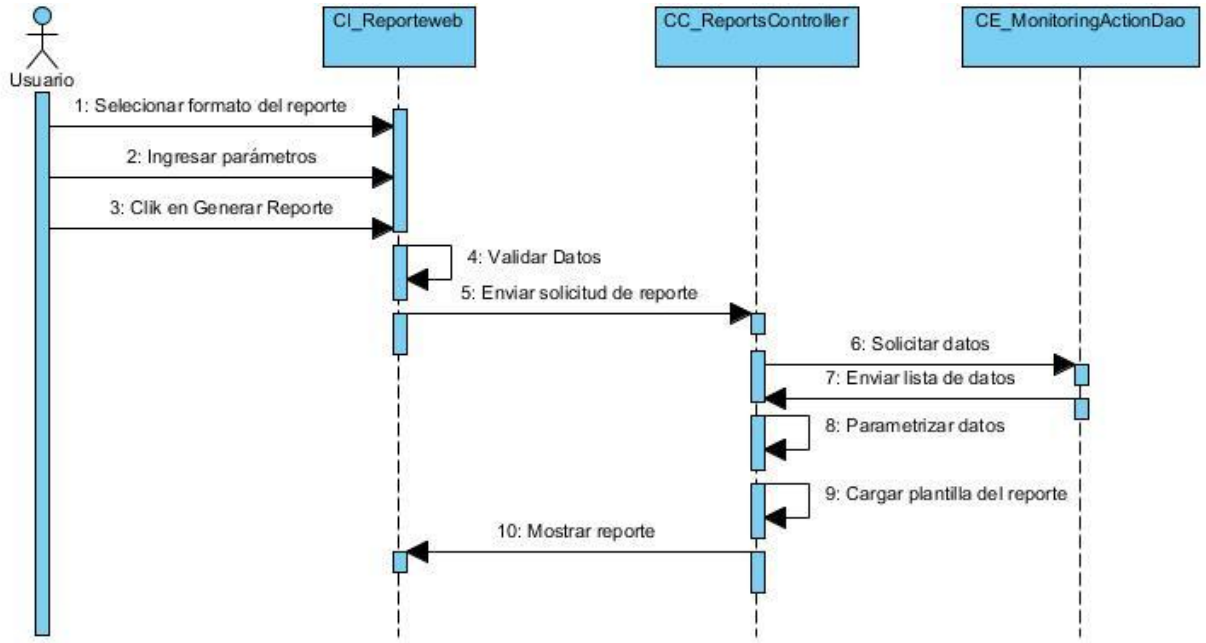
Diag. 2 Diagrama de Clases

2.4 Diagramas de Secuencia.

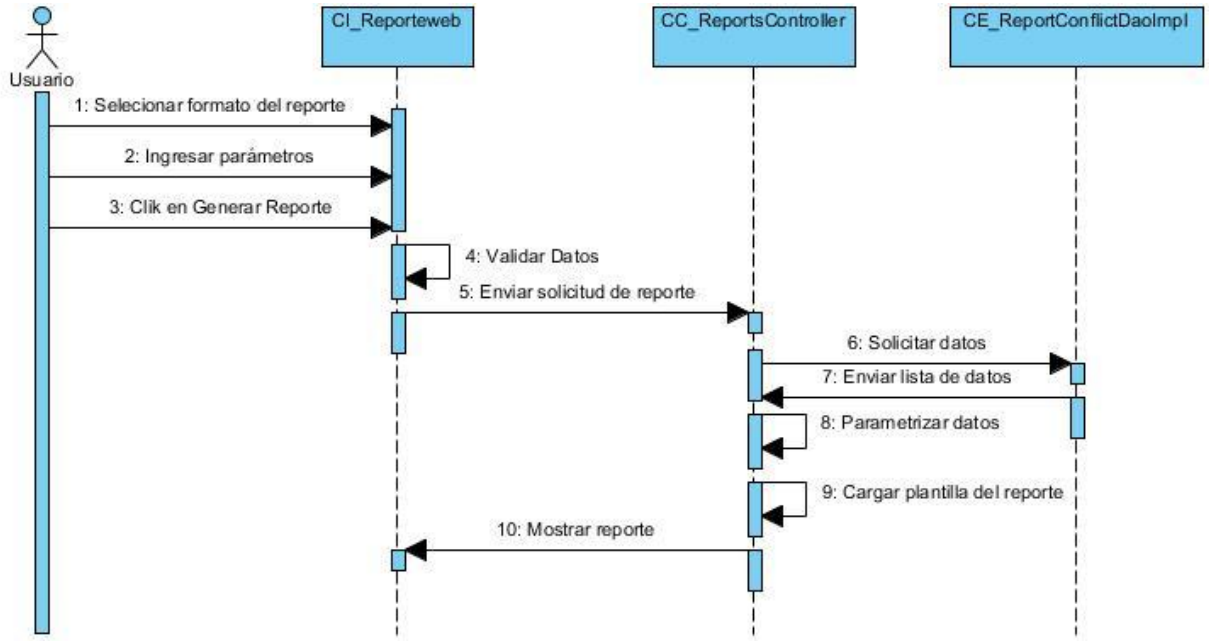
Los diagramas de secuencia muestran las interacciones entre los objetos organizadas en una secuencia temporal. En particular muestra los objetos participantes en la interacción y la secuencia de mensajes intercambiados. {7}



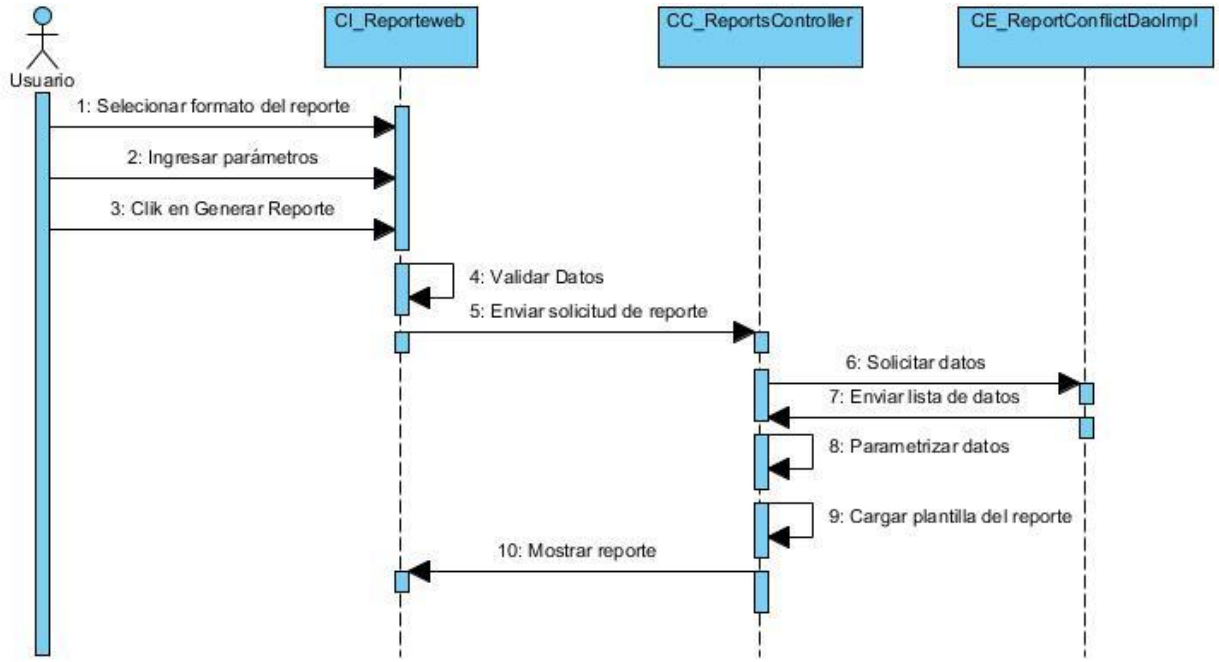
Diag. 3 Diagrama de Secuencia para el CU Crear Reporte de Acciones de Réplica.



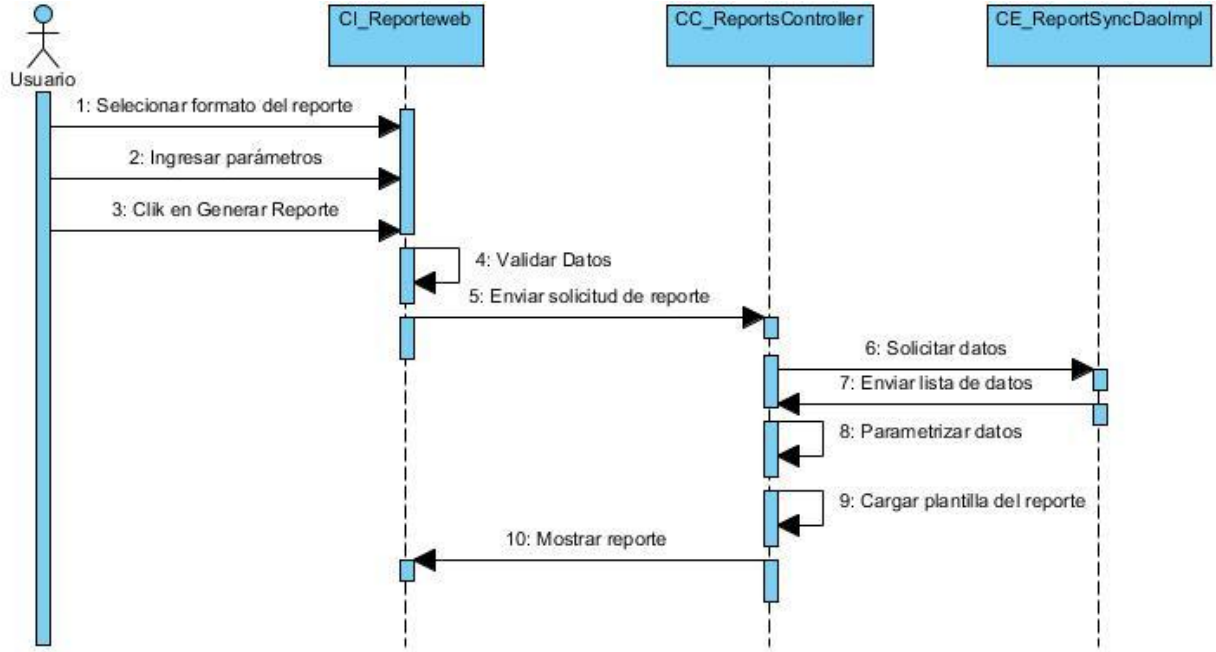
Diag. 4 Diagrama de Secuencia para el CU Crear Reporte Gráfico de Acciones de Réplica.



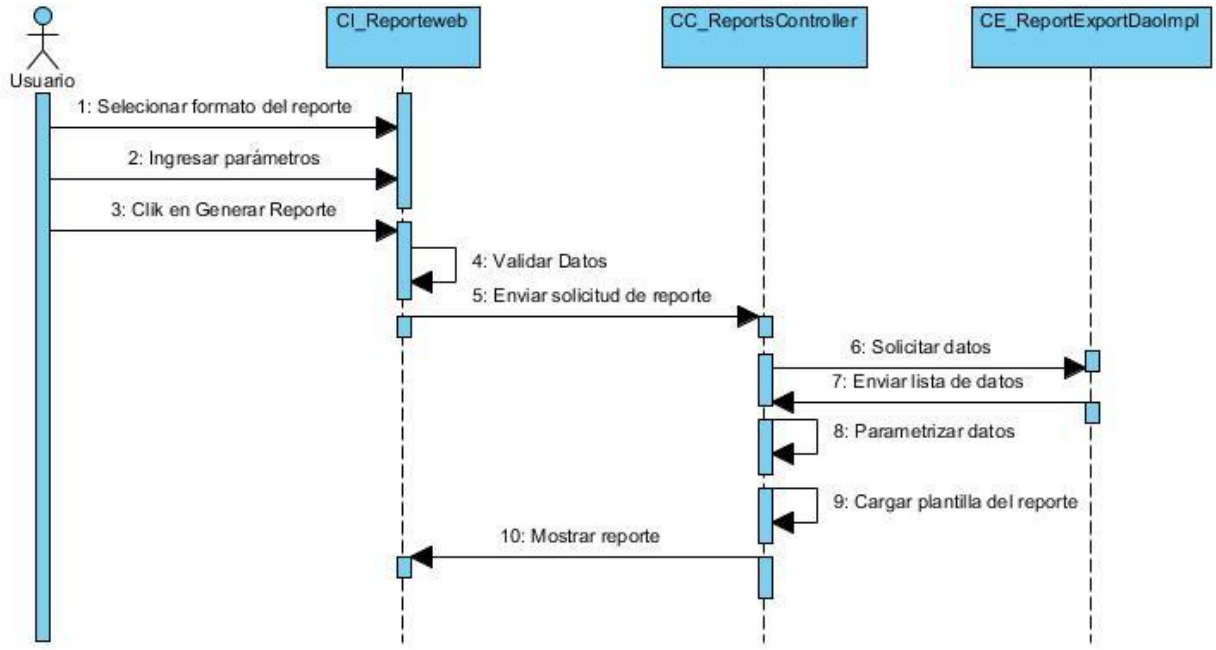
Diag. 5 Diagrama de Secuencia para el CU Crear Reporte de Conflictos.



Diag. 6 Diagrama de Secuencia para el CU Crear Reporte Gráfico de Conflictos.



Diag. 7 Diagrama de Secuencia para el CU Crear Reporte de Sincronización



Diag. 8 Diagrama de Secuencia para el CU Crear Reporte de Exportación

2.5 Patrones arquitectónicos y de diseño

"Un patrón de diseño es una descripción de clases y objetos que se comunican entre sí, adaptada para resolver un problema general de diseño en un contexto particular". Estos identifican las Clases, Instancias y la distribución de responsabilidades, son solución estándar para un problema común de programación. Los patrones arquitectónicos son patrones de software los cuales ofrecen soluciones arquitectónicas al problema. Describen los elementos y le tipo de relación entre ellos. Los patrones arquitectónicos expresan esquemas estructurales sobre la organización del software. Además incorporan diversas ventajas de calidad y eficiencia. (23)

2.5.1 Arquitectura en Capas

Los sistemas o arquitecturas en capas constituyen uno de los estilos que aparecen con mayor frecuencia mencionados en internet. En [GS94] Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inferior.

Características:

- El estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
- Admite muy naturalmente optimizaciones y refinamientos.
- Proporciona amplia reutilización.
- Ayuda a controlar y encapsular aplicaciones complejas.

Capas:

Capa de presentación: es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. (24)

2.5.2 Patrón Modelo Vista Controlador

El patrón conocido como Modelo-Vista-Controlador (MVC) separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes las cuales se aprecian en la:

- Modelo. El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
- Vista. Maneja la visualización de la información.
- Controlador. Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación de los sistemas jerárquicos en capas permite construir y probar el modelo independientemente de la representación visual. Entre las ventajas del estilo señaladas en la documentación de Patterns & Practices de Microsoft están las siguientes:

- Soporte de vistas múltiples. Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente.
- Adaptación al cambio. Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. (24)

2.5.3 Patrón Cliente – Servidor

La tecnología Cliente/Servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales. Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información de forma transparente aún en entornos multiplataforma.

Un sistema Cliente/Servidor es un sistema de información distribuido basado en las siguientes características:

- Servicio: unidad básica de diseño. El servidor los proporciona y el cliente los utiliza.
- Recursos compartidos: muchos clientes utilizan los mismos servidores y, a través de ellos, comparten tanto recursos lógicos como físicos.
- Protocolos asimétricos: los clientes inician “conversaciones”. Los servidores esperan su establecimiento pasivamente.
- Transparencia de localización física de los servidores y clientes: el cliente no tiene por qué saber dónde se encuentra situado el recurso que desea utilizar.
- Sistemas débilmente acoplados: interacción basada en envío de mensajes.
- Encapsulamiento de servicios: los detalles de la implementación de un servicio son transparentes al cliente.
- Escalabilidad horizontal (añadir clientes) y vertical (ampliar potencia de los servidores). se puede aumentar la capacidad de clientes y servidores por separado.
- Integridad: datos y programas centralizados en servidores facilitan su integridad y mantenimiento.
- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. (25)

2.5.4 Patrón acceso a datos (DAO)

El problema que viene a resolver este patrón es el de contar con diversas fuentes de datos (base de datos, archivos, servicios externos, etc.). De tal forma que se encapsula la forma de acceder a la fuente de datos. Este patrón surge históricamente de la necesidad de gestionar una diversidad de fuentes de datos, aunque su uso se extiende al problema de encapsular no sólo la fuente de datos, sino además ocultar la forma de acceder a los datos. Se trata de que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los datos o de cuál es la fuente de almacenamiento. Un DAO define la relación entre la lógica de presentación y empresa por una parte y por otra los datos. El DAO tiene un interfaz común, sea cual sea el modo y fuente de acceso a datos. No es imprescindible, pero en proyectos de

cierta complejidad resulta útil que el DAO implemente una interfaz. De esta forma los objetos cliente tienen una forma unificada de acceder a los DAO.

Algunas características:

- El DAO accede a la fuente de datos y la encapsula para los objetos clientes. Entendiendo que oculta tanto la fuente como el modo (JDBC) de acceder a ella.
- El Transfer Object encapsula una unidad de información de la fuente de datos. El ejemplo sencillo es entenderlo como un "bean de tabla", es decir, como una representación de una tabla de la base de datos, por lo que se representan las columnas de la tabla como atributos del Transfer Object. El DAO crea un Transfer Object (o una colección de ellos) como consecuencia de una transacción contra la fuente de datos. Por ejemplo, una consulta sobre ventas debe crear tantos objetos (Transfer Object) de la clase Venta como registros de la consulta; el DAO devolverá la colección de Transfer Object de la clase Venta al objeto Cliente. También puede ocurrir que el objeto Cliente mande un Transfer Object para parametrizar una consulta o actualización de datos por parte del DAO. (26)

2.5.5 Patrones GRASP:

Creador:

Consiste en asignarle a una clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

• B agrega los objetos A.

• B contiene los objetos A.

• B registra las instancias de los objetos A.

• B utiliza específicamente los objetos A.

• B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A). B es un creador de los objetos A.

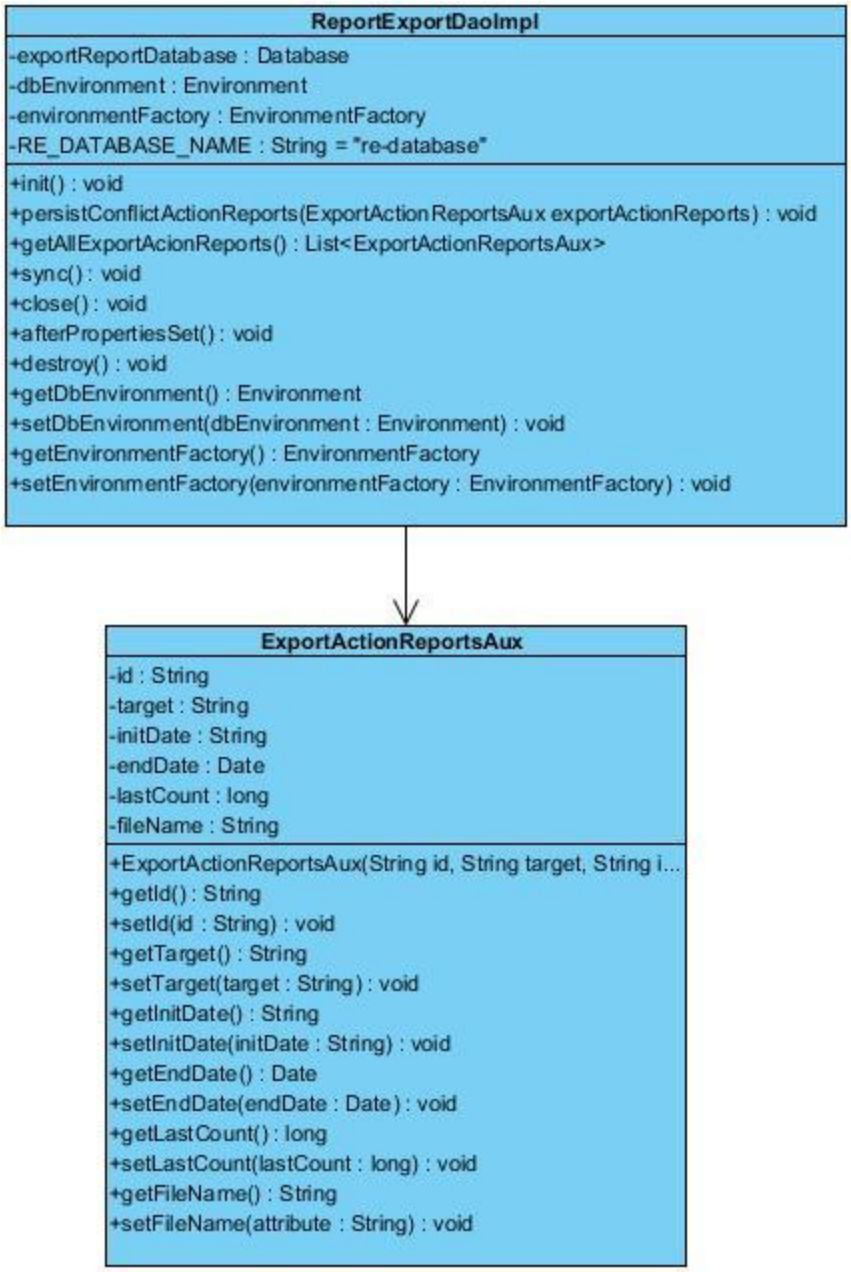


Fig. 4 Patrón Creador

Controlador:

Se encarga de asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas, facilitando la centralización de actividades. Con la utilización del módulo Spring MVC se pone de manifiesto este patrón.

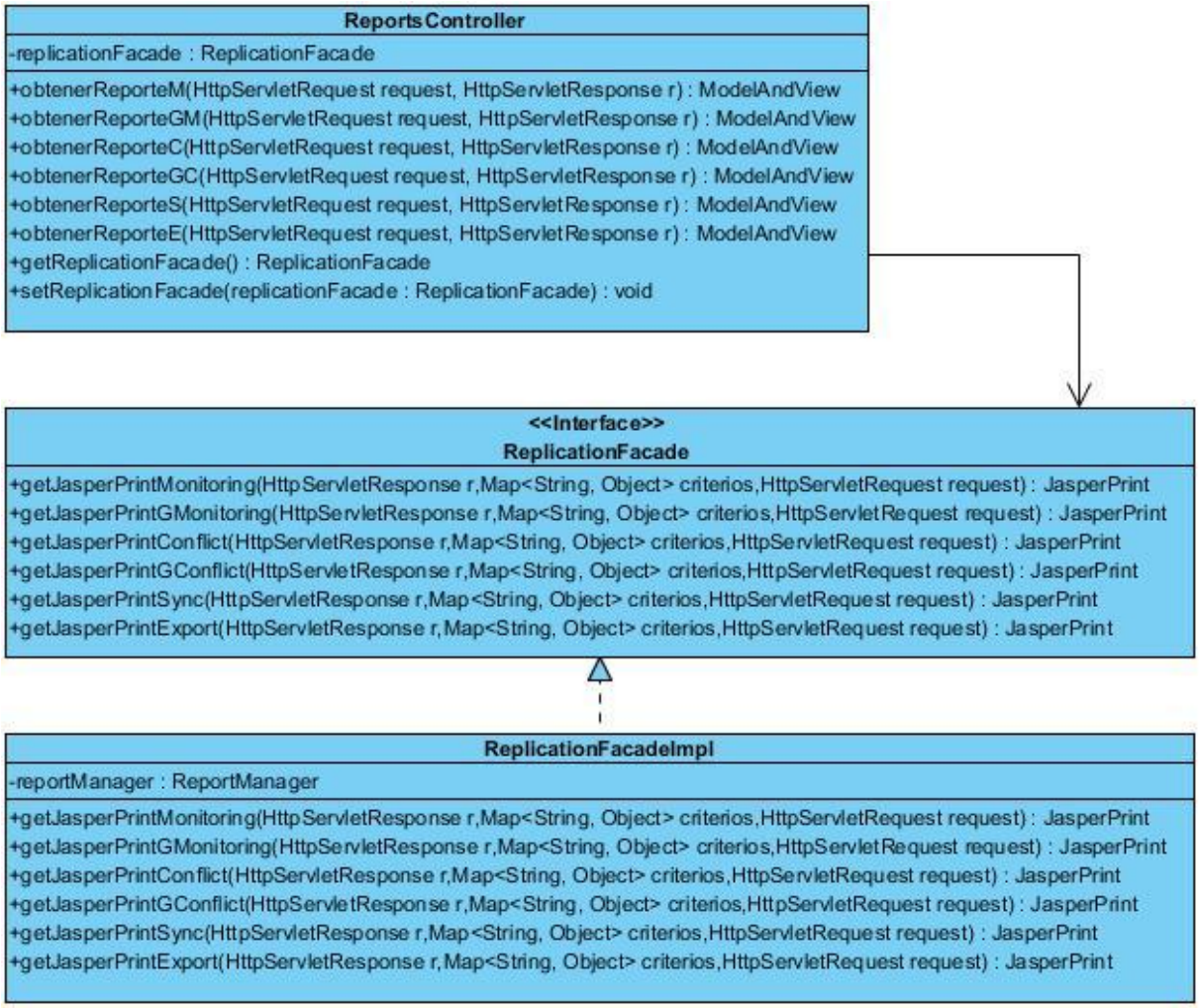


Fig. 5 Patrón Controlador

2.5.7 Patrones GOF:

Estructurales: Describen cómo las clases y los objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades.

Fachada (*Facade*): Provee de una interfaz unificada simple, para acceder a una interfaz o grupo de interfaces de un subsistema. Es utilizado para reducir la dependencia entre clases, ofreciendo un punto de acceso, de manera que si estas cambian o se sustituyen por otras, solo hay que actualizar la clase Fachada sin que el cambio afecte a las aplicaciones cliente. Fachada no oculta las clases, sino que ofrece

una forma más sencilla de acceder a las mismas y en los casos que se requiera, permite acceder directamente a ellas.

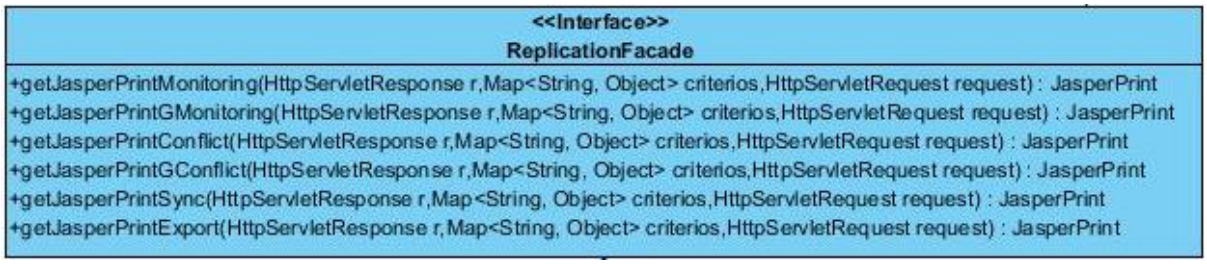


Fig. 6 Patrones Estructurales

2.6 Conclusiones del CAPÍTULO 2:

En este capítulo se mostraron las características y el diseño de la solución propuesta para el módulo de Reportes de REKO, arribándose a las siguientes conclusiones:

- ✓ Luego de analizados los requisitos obtenidos por el equipo de desarrollo así como la descripción de cada caso de uso del sistema, se realizó el diseño de las clases necesarias para la implementación del módulo.
- ✓ El diseño de los diagramas de clases utilizando la herramienta Visual Paradigm, facilitó la implementación de las clases. Este elemento contribuyó a disminuir el tiempo de desarrollo, convirtiéndose en la principal entrada para la implementación.
- ✓ Se logró uniformidad en la elaboración de los diagramas de clases, obteniéndose un lenguaje uniforme entre los diseñadores y mayor entendimiento por parte de los desarrolladores a la hora de leer la documentación.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

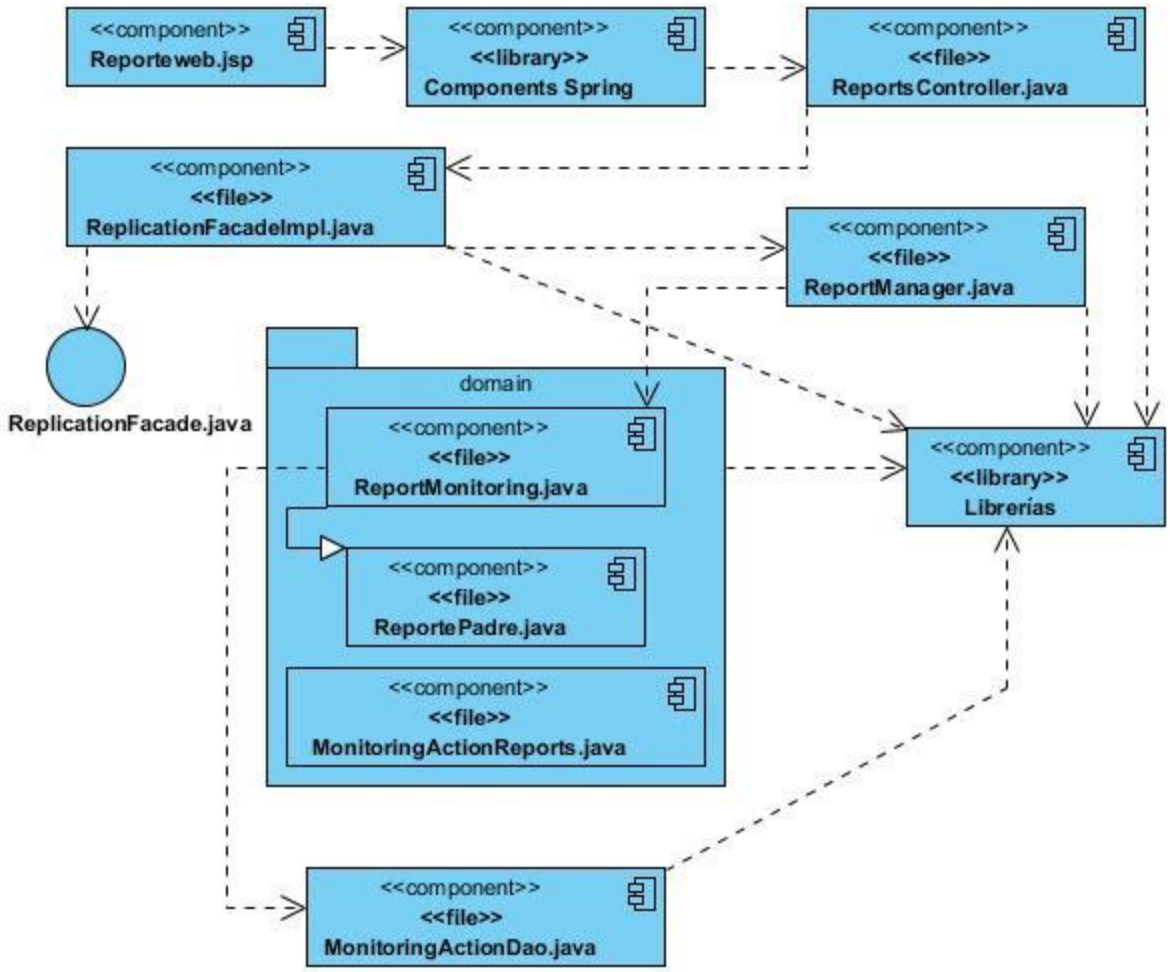
3.1 Introducción:

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, etc. El desarrollo del software ha de ir acompañado de alguna actividad que garantice su calidad; la prueba es un elemento crítico para la garantía de calidad del software.

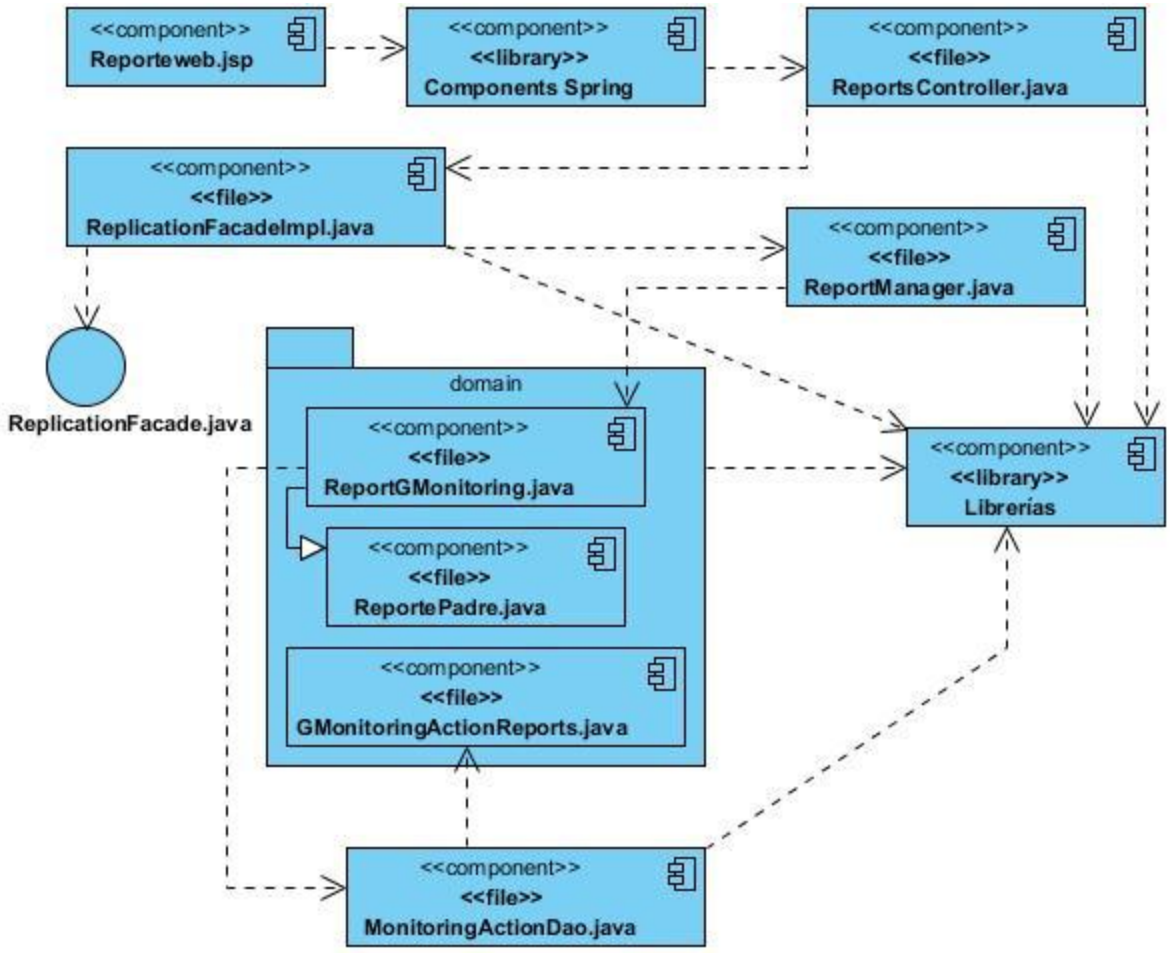
En el presente capítulo se muestran los diagramas de componentes para las capas: Presentación, Negocio y Acceso a Datos. Se exponen los resultados obtenidos al aplicar las pruebas de caja negra.

3.2 Diagrama de Componentes.

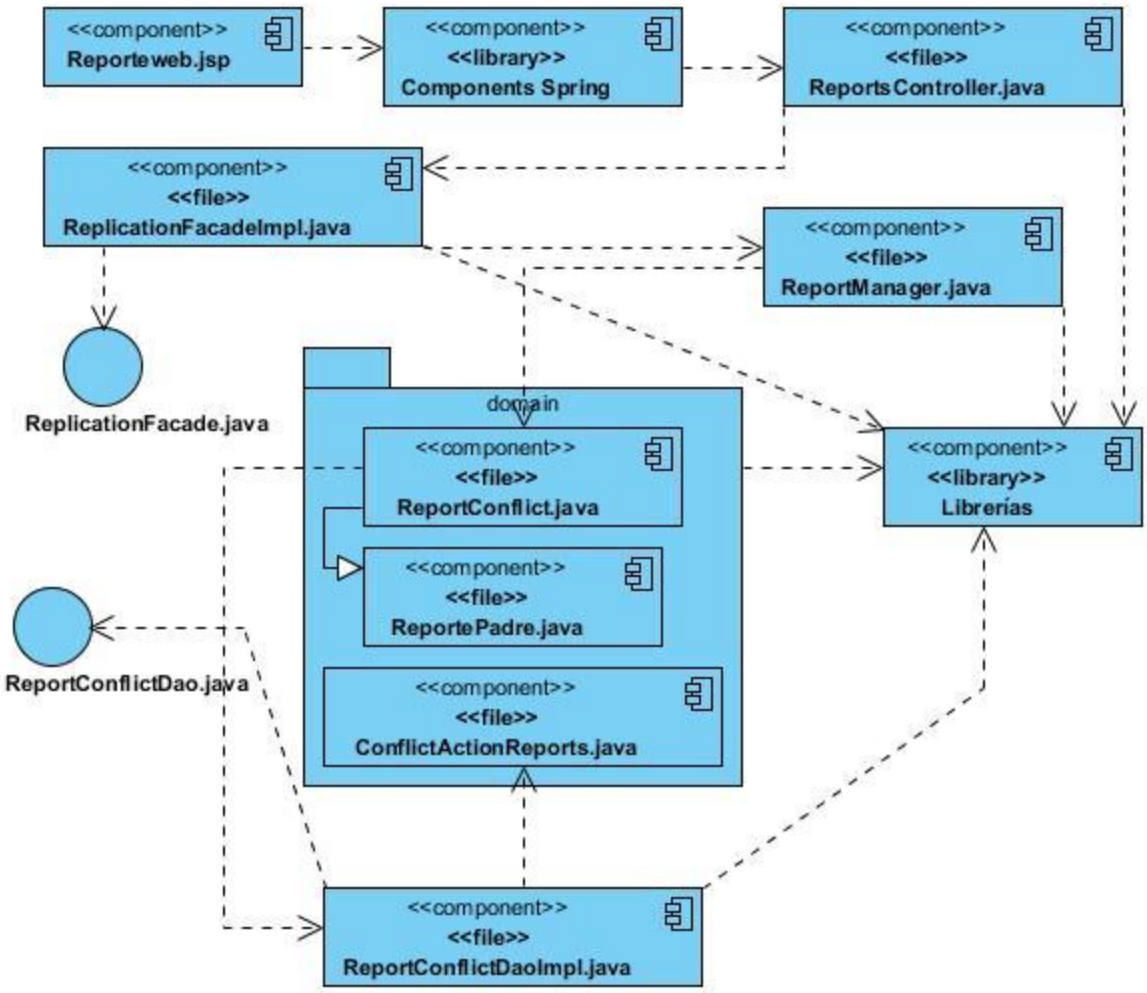
Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el Modelo de Diseño. *"Los diagramas de componentes describen los elementos físicos y sus realizaciones en el entorno"* de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros. (27)



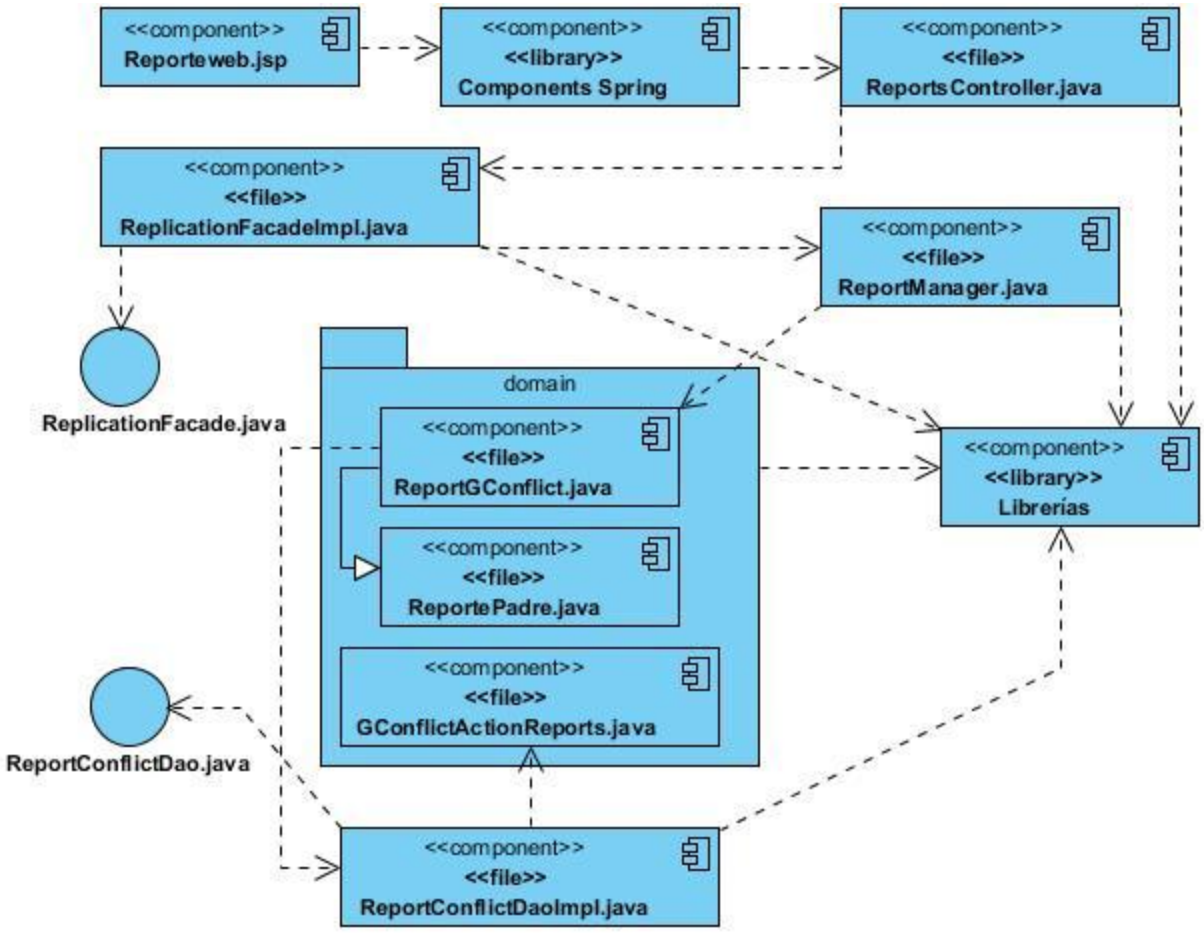
Diag. 9 Diagrama de Componentes para el CU Crear Reporte de Acciones de Réplica.



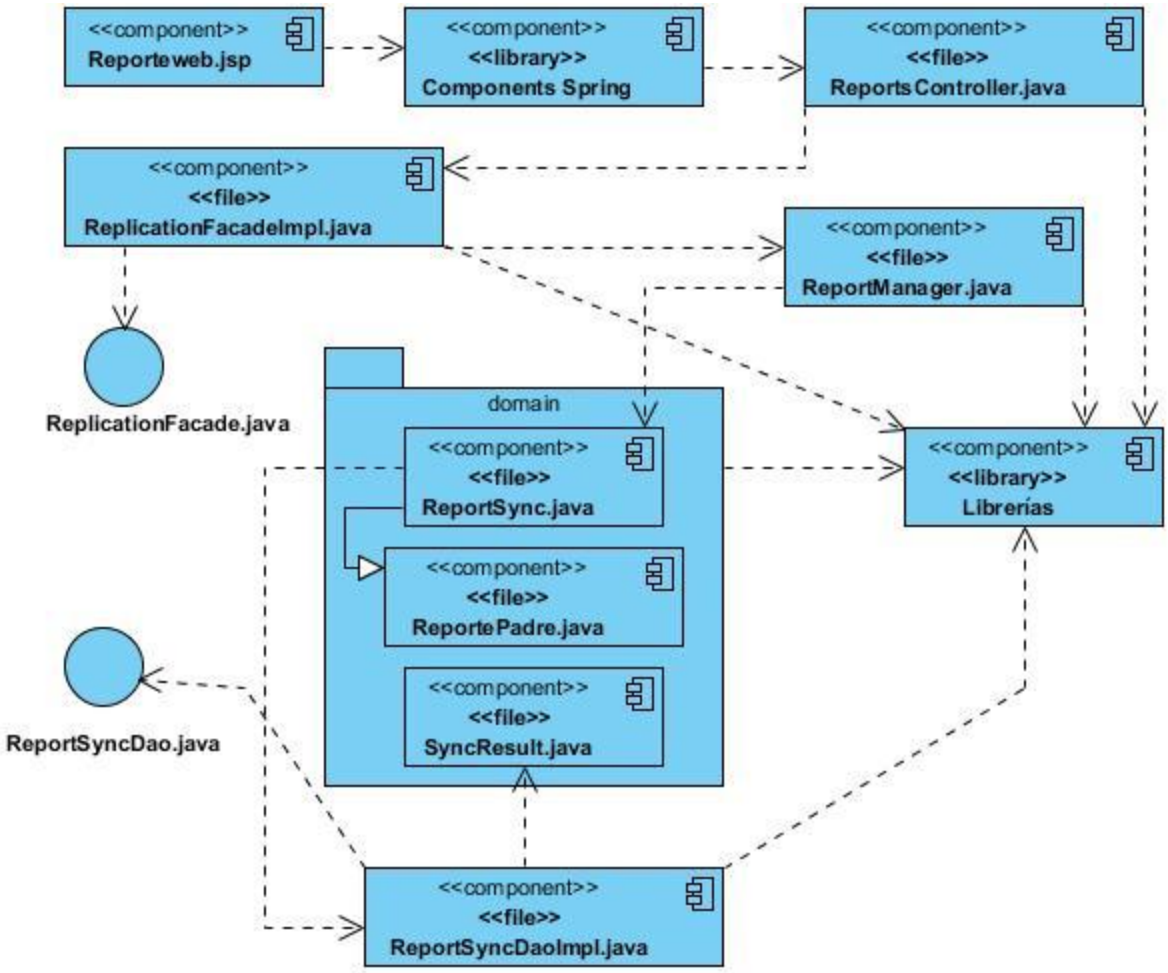
Diag. 10 Diagrama de Componentes para el CU Crear Reporte Gráfico de Acciones de Réplica.



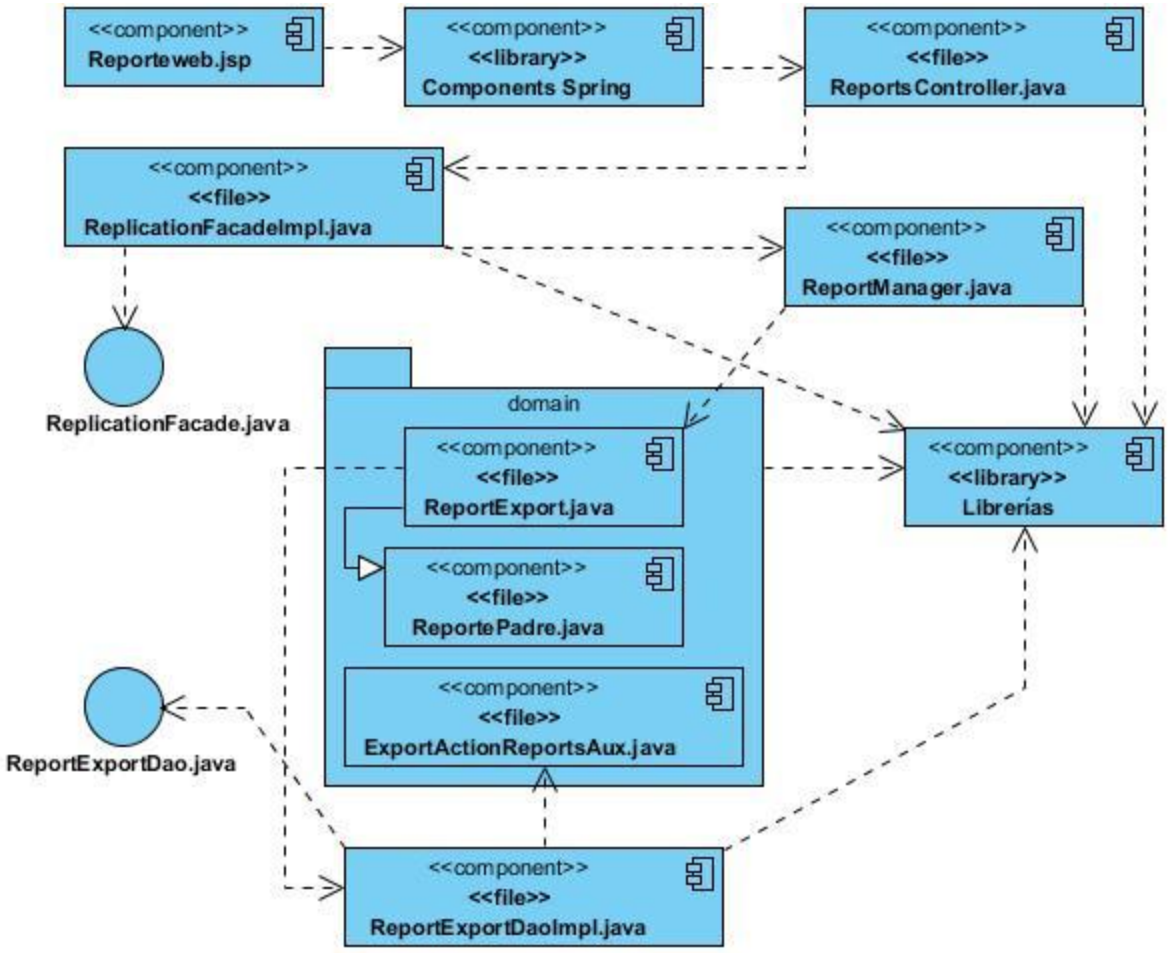
Diag. 11 Diagrama de Componentes para el CU Crear Reporte de Conflictos.



Diag. 12 Diagrama de Componentes para el CU Crear Reporte Gráfico de Conflictos.



Diag. 13 Diagrama de Componentes para el CU Crear Reporte de Sincronización.



Diag. 14 Diagrama de Componentes para el CU Crear Reporte de Exportación.

3.3 Pruebas del Sistema.

Una vez generado el código fuente, es necesario probar el software para descubrir y corregir la mayor cantidad de errores posible antes de entregarlo al cliente. Su objetivo es diseñar una serie de casos de prueba que tengan una alta probabilidad de encontrar errores y garantizar así una alta calidad del software desarrollado. Aquí es donde entran las técnicas de prueba del software. Estas técnicas proporcionan directrices sistemáticas para pruebas de diseño que comprueben la lógica interna y las interfaces de todo componente del software y comprueben los dominios de entrada y salida del programa para descubrir errores en su función, comportamiento y desempeño. (28)

3.3.1 Configuración del entorno de prueba.

La configuración del entorno donde se vayan a ejecutar las diferentes pruebas que se realizan a un software es un aspecto muy importante dentro del proceso de pruebas, pues si no se analizan bien los recursos de software y hardware que necesita el producto que se está construyendo, a la hora de probarlo se prescindirá de los elementos necesarios para la ejecución de un proceso de pruebas exitoso. Por ello se tuvo en cuenta algunos requerimientos de hardware y de software que hicieron posible un mejor desarrollo de las pruebas, en aras de que se lograran minimizar los errores de la aplicación en desarrollo. Los requerimientos que se consideraron necesarios para las pruebas se referencian a continuación:

- **Requerimientos de software:**

  · PC con Windows XP o superior.

  · PC con Linux

  · Máquina Virtual de Java 1.5 o superior.

  · Servidor JMS.

  · Servidor de base de datos PostgreSQL.

- **Requerimientos de hardware:**

  · PC con Microprocesador Pentium IV o superior.

  · Se requiere para Windows XP Professional (SP1) una memoria de 128MB y espacio en el disco de 2GB

  · Linux una memoria de 128MB y espacio en el disco de 2GB

3.3.2 Pruebas de Caja Negra

Para desarrollar las pruebas al sistema se decidió utilizar el método de caja negra, que se refiere a las pruebas llevadas a cabo sobre la interfaz del software. Es decir, a través de ellas se pretende demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. En general, este método se centra principalmente en los requisitos funcionales del software. (29)

La prueba de la caja negra intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructura de datos o en acceso a base de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Para confeccionar los casos de prueba de Caja Negra existen distintas técnicas. Entre las que se encuentran:

- *Particiones de Equivalencia:* Es un método de prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Esta se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada.
- *Análisis de Valores Límite:* El Análisis de Valores Límites (AVL) es una técnica de diseño de casos de prueba que complementa la partición equivalente. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida.
- *Métodos Basados en Grafos:* Empieza creando un grafo de objetos importantes y sus relaciones, y después diseñando una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir los errores.
- *Guiada por Casos de Prueba:* Verifican las especificaciones funcionales y no consideran la estructura interna del programa. Se realiza sin el conocimiento interno del producto. No validan funciones ocultas por tanto los errores asociados a ellas no serán encontrados.

Con el objetivo de comprobar el correcto funcionamiento de las funcionalidades correspondientes al Módulo de Reportes, el software fue sometido a una serie de pruebas, a través de las cuales se pudo corroborar que el sistema responde

positivamente a toda actividad que el usuario desee realizar sobre ella. A continuación se muestran los resultados obtenidos al aplicar las pruebas de caja negra a la aplicación utilizando la técnica de Particiones de Equivalencia.

Para el caso de uso Crear Reporte de Acciones de Réplica

Descripción General

El caso de uso inicia cuando el Usuario necesita crear un reporte de acciones de réplica para favorecerse de la información que este brinda.

Condiciones de Ejecución.

- El sistema debe estar instalado y ejecutándose correctamente.
- El actor debe estar autenticado con los permisos necesarios.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Generar Reporte	EC 1.1: Generar Reporte exitosamente.	Se generar el reporte introduciendo los datos correctamente.
	EC 1.2: Generar Reporte con datos incorrectos.	Al insertar datos para generar el reporte por los criterios determinados se pueden insertar incorrectamente.

Tabla 9 Secciones a probar en el CU Crear Reporte de Acciones de Réplica.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Formato del reporte	Botón de radio	Puede	Falso o verdadero
[2]	Tipo de acción	Filtrado de selección	Puede	Letras
[3]	Dirección	Filtrado de selección	Puede	Letras
[4]	Nodo	Campo de Texto	Puede	Letras y números
[5]	Fecha inicial	Calendario	No	Números con el siguiente formato: día-mes-año.
[6]	Fecha final	Calendario	No	Números con el siguiente formato: día-mes-año.

Tabla 10 Descripción de variable.

Escenario	Formato del reporte	Tipo de acción	Dirección	Nodo	Fecha inicial	Fecha final	Respuesta del Sistema	Resultado de la Prueba
EC 1.1: Generar Reporte exitosamente.	V PDF	V Inserción	V Salida	V Nodo 01	V 08/05/ 2012	V 11/05/ 2012	El sistema genera el reporte correspondiente a los criterios de búsqueda definidos.	Satisfactorio
	V WORD	V Campo vacío n	V Campo vacío	V Campo vacío	V 08/05/ 2012	V 11/05/ 2012	El sistema genera el reporte correspondiente a los criterios de búsqueda definidos.	Satisfactorio
EC 1.2: Generar Reporte con datos incorrectos.	V WORD	V Actualización	V Entrada	V Campo vacío	I 1-5-12	I 2012/05/12	El sistema mostrará un mensaje avisándole al usuario que no puede introducir campos incorrectos.	Satisfactorio
	V HTML	V Eliminación	V Entrada	V Nodo 02	I Campo vacío	V 11/05/ 2012	El sistema mostrará un mensaje avisándole al usuario que no puede dejar	Satisfactorio

							campos vacíos.	
	V PDF	V Eliminación	V Salida	V Nodo 03	V 08/07/ 2012	I Campo vacío	El sistema mostrará un mensaje avisándole al usuario que no puede dejar campos vacíos.	Satisfactorio
	V PDF	V Eliminación	V Salida	I Nodo 011	V 08/07/ 2012	V 11/05/ 2012	El sistema mostrará un mensaje avisándole al usuario que no puede introducir campos incorrectos.	Satisfactorio
	V PDF	V Eliminación	I Pedro	V Nodo 03	V 08/07/ 2012	V 11/05/ 2012	El sistema mostrará un mensaje avisándole al usuario que no puede introducir campos incorrectos.	Satisfactorio
	V PDF	I Juan	V Salida	V Nodo 03	V 08/07/ 2012	V 11/05/ 2012	El sistema mostrará un mensaje avisándole al usuario que no puede introducir	Satisfactorio

							campos incorrectos.	

Tabla 11 Matriz de Datos.

3.4 Resumen de Pruebas:

En el proceso de pruebas efectuado se diseñaron 6 Casos de Prueba con el objetivo de verificar el correcto funcionamiento de la aplicación y su cumplimiento con las principales funcionalidades. Se obtuvo un total de 4 no conformidades, las cuales fueron solucionadas en su totalidad. Como resultado final se logró una aplicación que cumple con los requisitos funcionales.

3.5Conclusiones del CAPÍTULO 3:

Una vez terminada la implementación, aplicadas las pruebas y el diseño del sistema, se concluyó lo siguiente:

- ✓ Se logró implementar la funcionalidad con éxito, presentándose el Modelo de Implementación, conformado por los Diagramas de Componentes correspondientes a los Casos de Uso.
- ✓ Las pruebas de caja negra aplicadas mostraron una respuesta efectiva en cuanto a la obtención de la información requerida por el usuario, validando de esta forma el cumplimiento de los requisitos funcionales.

CONCLUSIONES GENERALES

Durante el desarrollo del presente trabajo se llevaron a cabo una serie de fases que permitieron dar cumplimiento al objetivo planteado y que abarcaron todas las tareas investigativas propuestas.

Por lo tanto se concluye que:

- ✓ El estudio de las metodologías, herramientas y tecnologías permitieron el desarrollo de una solución que cumple con las especificaciones de los casos de uso y los requisitos asociados.
- ✓ Nuestra herramienta ha tenido un efecto grato por parte del cliente al atenuar las deficiencias que existían en la obtención de reportes con información filtrada.
- ✓ Ciertamente, la herramienta obtenida ha creado una condición laboral mejorada en el aspecto de obtener información específica de los procesos de REKO, tan indispensable para la toma de decisiones.

RECOMENDACIONES

- ✓ Profundizar en el estudio de los generadores de reportes dinámicos con el objetivo de encontrar nuevas funcionalidades que el sistema pudiera brindar al generar los reportes en versiones futuras.
- ✓ Continuar el desarrollo de la herramienta con el objetivo de incorporar otras funcionalidades que permitan el diseño de reportes de mayor complejidad.
- ✓ Agregar nuevas funcionalidades como puede ser la de una vista previa del reporte, y evaluar la posibilidad de mejorar el diseño de la interfaz.

CITAS BIBLIOGRAFÍCAS

1. Isaac Newton. Universidad de Palermo**.** [En línea] 2011 [Citado el: 12 de 6 de 2012] http://fido.palermo.edu/servicios_dyc/publicacionesdc/vista/detalle_articulo.php?id_lior o=13&id_articulo=5255

2. Jacobson, Ivar. *El proceso unificado de desarrollo de software.* La Habana: Félix Varela, 2004.

3. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso unificado de desarrollo de software.* S.l.: Addison Wesley, 2000.

4. Balduino, Reicardo. Introduction to OpenUp (Open Unified Process). Eclipse. www.eclipse.org/epf/general/OpenUp.pdf.

5. *Oracle® Database Advanced Replication 11g Release 2 (11.2) E10706-02.* [En línea] 12 de 2009. [Citado el: 8 de 2 de 2012.] http://download.oracle.com/docs/cd/E11882_01/server.112/e10706.pdf.

6. Requerimientos. [Citado el 10 de 3 de 2012] http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos

7. UML. [En línea] [Citado el: 31 de 3 de 2012.] http://www.neuronsrl.com.ar/training/uml/uml_secuencia.html.

REFERENCIAS BIBLIOGRAFÍCAS

1. reportes. [En línea] [Citado el: 4 de 11 de 2011.] http://sipec.sep.gob.mx/WebHelp/reportes/reporte.htm.

2. Software De Computo. [En línea] [Citado el: 4 de 11 de 2011.] http://www.mitecnologico.com/Main/SoftwareDeComputo

3. CENACAD. [En línea] 10 de 2006. [Citado el: 5 de 11 de 2011.] http://www.cenacad.espol.edu.ec/index.php

4. SAFYR. [En línea] 2010. [Citado el: 5 de 11 de 2011.] http://www.safyr.co/Portal/

5. Goldentech Ltda. Goldentech. [En línea] [Citado el: 6 de 11 de 2011.] http://www.goldentech-e.com/Celta.html

6. Goldentech Ltda. Goldentech. [En línea] [Citado el: 6 de 11 de 2011.] http://www.goldentech-e.com/Celta_Reportes.html

7. Álvarez, Elda Jeny Báez. [En línea] 2006. [Citado el: 8 de 11 de 2011.] http://www.bibliociencias.cu/gsdl/collect/eventos/index/assoc/HASH25c5.dir/doc.pdf

8. Vázquez, Yusliel García. [En línea] 2006. [Citado el: 6 de 12 de 2011.] http://www.congresoinfo.cu/UserFiles/File/Info/Info2006/Ponencias/37.pdf.

9. marquina88. [En línea] 25 de 9 de 2010. [Citado el: 12 de 6 de 2012.] http://marquina88.wordpress.com/2012/06/06/mineria-de-datos/

10. Lastre, Reyvis Ernesto. *Análisis y Diseño del Módulo de Reportes para el Portal de la UJC en la UCI,* 2008.

11. datavision. [En línea] [Citado el: 9 de 11 de 2011.] http://datavision.sourceforge.net/.

12. gnuempresa. [En línea] [Citado el: 10 de 11 de 2011.] http://gnuempresa.wordpress.com/2007/10/28/agata-report/.

13. jasperforge. [En línea] 12 de 1 de 2010. [Citado el: 4 de 12 de 2011.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=ireport.

14. cepeu. [En línea] [Citado el: 3 de 12 de 2011.] http://www.cepeu.edu.py/LIBROS_ELECTRONICOS_3/lpcu089%20-%2001.pdf.

15. dosideas. [En línea] [Citado el: 22 de 10 de 2011.] http://www.dosideas.com/wiki/JasperReports.

16. Entorno Visual de Aprendizaje. [En línea] 10 de 2009. [Citado el: 10 de 11 de 2011.] http://eva.uci.cu/mod/resource/view.php?id=21621&subdir=/El_proceso_unificado_de_desarrollo_de_software.

17. *Rational Software Corporation. Rational Unified Process.* 2003.

18. Cortizo Pérez, José Carlos, Expósito Gil, Diego y Ruiz Leyva, Miguel. *Extreme Programming.* España: s.n., 2004.

19. [En línea] 2006. [Citado el: 12 de 11 de 2011.] http://www.fing.edu.uy/inco/cursos/gestsoft/Presentaciones/ProcesosAgiles-4/ProcesosAgiles.ppt.

20. Rational. [En línea] 2011. [Citado el: 12 de 11 de 2011.] www.rational.com.

21. SparxSystems. [En línea] 2 de 2011. [Citado el: 12 de 11 de 2011.] http://sparxsystems.com.ar/products/ea.html.

22. Visual-Paradigm. [En línea] 2 de 2011. [Citado el: 12 de 11 de 2011.] www.visual-paradigm.com.

23. Prieto, Félix. *Patrones de diseño.* Valladolid : s.n., 2008.

24. Reynoso, Carlos y Kicillof, Nicolas. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.*

25. temariotic. [En línea] 8 de 06 de 2011. [Citado el: 15 de 11 de 2011.] http://temariotic.wikidot.com/la-arquitectura-cliente-servidor.

26. proactiva-calidad. [En línea] [Citado el: 15 de 11 de 2011.] http://www.proactiva-calidad.com/java/patrones/DAO.html.

27. Visconti, Marcello y Hernán, Astudillo. Departamento de Informática. [En línea] [Citado el: 29 de Abril de 2012.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/15-Implementacion.pdf

28. Pressman, Roger S. *Ingenieria del software, un enfoque practico.* España : s.n.

29. Asignatura: Ingeniería de Software II Curso: 2011-2012 Conferencia 4. Título: Flujo de trabajo de Prueba. [Citado el 4 de noviembre de 2011].