Universidad de las Ciencias Informáticas


Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

**Título: Componente de generación de reportes dinámicos basados en conceptos para el módulo Estadísticas del Sistema de Información Hospitalaria del CESIM.**

**Autores:** Alejandro Vázquez Menéndez

Jorge Gaspar Ramírez Segura

**Tutores:** Ing. Alain Ramos Medina

Ing. Osmany Castro Espinosa

La Habana, Junio 2014

"Año 56 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los _____ del mes de junio del año 2014:

| | |
|---|---|
| _____ | _____ |
| Alejandro Vázquez Menéndez | Jorge Gaspar Ramírez Segura |
| **Autor** | **Autor** |
| _____ | _____ |
| Ing. Alain Ramos Medina | Ing. Osmany Castro Espinosa |
| **Tutor** | **Tutor** |

DATOS DE CONTACTO

**Tutores:**

Ing. Alain Ramos Medina: Ingeniero en Ciencias Informáticas, graduado en la Universidad de las Ciencias Informáticas en el año 2007. Posee la categoría docente de profesor asistente. Ha impartido asignaturas de Proyectos de Investigación y Desarrollo III, Programación I, Introducción a la Programación y Aplicaciones Informáticas en el Sector de la Salud. Ha participado en varios proyectos de desarrollo vinculados al perfil de salud. Actualmente labora en el Departamento de Sistemas de Gestión Hospitalaria del Centro de Informática Médica (CESIM), desempeñándose como jefe de proyecto.
Correo Electrónico: aramos@uci.cu

Ing. Osmany Castro Espinosa: Ingeniero en Ciencias Informáticas, graduado en la Universidad de las Ciencias Informáticas en el año 2010. Ha participado en varios proyectos de desarrollo vinculados al perfil de salud. Actualmente labora en el Departamento de Sistemas de Gestión Hospitalaria del Centro de Informática Médica (CESIM), desempeñándose como desarrollador.
Correo Electrónico: ocespinosa@uci.cu

## AGRADECIMIENTO

De Alejandro

A mis amigos y amigas que me acompañaron durante estos 5 años y en general a todas las personas que de un modo u otro contribuyeron a mi formación.

A mis tutores por guiarme a través de todo el camino.

De Jorge Gaspar

A mis amigos del antiguo 89107 Osvaldo, Fidel y el Malva.

A todos mis amigos del 79106 en especial para Froilán, Vázquez y Alfredo.

A Rosa por su incondicional apoyo en los momentos más difíciles de mi vida universitaria.

A Marla y a Nely por ser madres, amigas y estar pendiente de cada paso que doy en mi vida.

A Arianna Gómez por brindarme su ayuda.

A mi familia por estar a mi lado en todo momento.

A mi abuelo César, a mis tíos Arévalo y Jorge Segura.

A todos los que de una forma u otra ayudaron a que lograra graduarme.

A mis tutores por guiarme a través de todo el camino.

Un agradecimiento especial a mi hermanita querida, una de las esquinas del triángulo de mujeres que componen mi vida.

## DEDICATORIA

De Alejandro

A mi familia por apoyarme y estar a mi lado en todo momento, incluso por muy lejos que pudieran haber estado.

A mi padre específicamente por ser siempre mi modelo a seguir.

De Jorge Gaspar

A mi abuela y a mamá.

## RESUMEN

Los generadores de reportes son herramientas complementarias de los sistemas de información, utilizan una especie de lenguaje transparente para el usuario, que se emplea para generar y obtener de forma organizada la información del reporte.

Las estructuras que se usan para construir los mismos se denominan conceptos, éstos son la unión de varias estructuras de la base de datos para la representación de información referida a la persona o a la población y el proceso de todas las actividades que identifican la atención médica, evaluados en un período de tiempo y aplicándole un grupo de funciones que respondan a las exigencias estadísticas.

El Centro de Informática Médica de la Universidad de las Ciencias Informáticas desarrolla un Sistema de Información Hospitalaria, que genera una serie de reportes de forma estática, lo que obliga al personal de estadísticas a elaborar la información necesaria y luego enviarla al equipo de desarrollo que se encarga de la construcción de la plantilla, complejizando el proceso de generación del reporte. Existen algunos trabajos de investigación relacionados con la generación de reportes dinámicos, un ejemplo es el Generador Dinámico de Reportes pero no constituye una solución a la problemática planteada.

Por esta razón el objetivo del trabajo de investigación es desarrollar un Componente de generación de reportes dinámicos basados en conceptos para el módulo Estadísticas del Sistema de Información Hospitalaria del Centro de Informática Médica**.**

El generador de reportes dinámicos basado en conceptos permite agilizar la construcción de los reportes de acuerdo a las necesidades del personal de salud, ayudando a la toma de decisiones y estudios estadísticos de comportamientos.



Palabras clave: Centro de Informática Médica, reportes dinámicos, conceptos.

# TABLA DE CONTENIDO

# ÍNDICE DE TABLAS

# ÍNDICE DE FIGURAS

# INTRODUCCIÓN

Las nuevas tendencias del uso de las Tecnologías de la Información y las Comunicaciones (TIC, por sus siglas) (1) dentro de la sociedad, han aportado las herramientas necesarias para gestionar la información generada en el sector de la salud. Con este fin se crean soluciones informáticas especializadas en los procesos que tienen lugar en los distintos departamentos que conforman los sistemas de salud.

Los sistemas de salud se dividen en tres formas de atención. La Atención Primaria, es el más básico de todos y el de menor complejidad. La Atención Secundaria, se encuentra orientada a un segmento menor de la población y el nivel de Atención Terciaria o Especializada de Salud, atiende casos con patologías excepcionales. Para la gestión de los procesos en las instituciones hospitalarias que pertenecen al segundo nivel de atención a la salud son usados los Sistemas de Información Hospitalaria (HIS, por sus siglas en inglés) (2), estos son los encargados de la recolección, almacenamiento, procesamiento, recuperación y comunicación de la información obtenida como parte de los procesos de atención a pacientes en dichas instituciones.

Los HIS tienen como principio fundamental elevar el nivel de los servicios brindados al paciente por parte del personal médico, adicionalmente separan la información obtenida de las áreas de investigación, la clínica, la docencia y la administración. Estos sistemas juegan un rol importante donde el paciente es el más beneficiado y los profesionales de la salud encuentran un recurso amigable y flexible que responde a las necesidades de información de la institución hospitalaria o de salud.

Idóneamente los HIS abarcan todas las áreas y especialidades que se encuentran a nivel secundario de atención médica, entre los que se pueden identificar Consulta Externa, Hospitalización, Bloque Quirúrgico, Estadísticas entre muchos otros.

El uso de las estadísticas en los Sistemas de Información Hospitalaria complementa las exigencias de control de los procesos administrativos y de salud que se llevan a cabo en la institución, siempre teniendo en cuenta y haciendo uso de indicadores médicos (3) correctamente elaborados y fundamentados por especialistas de esta área del conocimiento. De esta forma se hace fácil la obtención de datos críticos, por su importancia, en la toma de decisiones a este nivel.

En los últimos años el MINSAP ha llevado a cabo el proceso de informatización de la atención a la salud publica en Cuba, para ello ha puesto la tarea en manos de personal capacitado para asumir dicha responsabilidad.

En el Centro de Informática Médica (CESIM) de la Universidad de Ciencias Informáticas durante la última década se ha enfocado en el desarrollo de un HIS. El desarrollo del mismo ha priorizado la interacción de todas las áreas y especialidades de atención médica dentro de una institución hospitalaria, así como se mencionaba con anterioridad. Hasta la fecha cuenta con varios módulos tales como: Banco de sangre, Epidemiología, Anatomía Patológica, Bloque quirúrgico, Emergencia, Hospitalización, Consulta Externa y Estadísticas entre otros, dichos módulos como función principal capturan la información perteneciente al flujo del proceso de atención al paciente en la institución, que adicionalmente puede ser utilizada para la toma de decisiones.

En el módulo Estadísticas se procesan grandes cantidades de información con el objetivo de generar indicadores de cobertura, disponibilidad, eficacia, eficiencia, productividad, uso y utilización que permiten mantener un control y evaluación constante del funcionamiento de la institución de salud.

Este módulo genera una serie de reportes cuyos marcos o diseños están previamente definidos. Cuando el usuario presenta la necesidad de obtener información que difiere en formato o datos estadísticos a los actualmente generados, debe ponerse en contacto con el equipo de desarrollo del sistema, el cual tramita la necesidad del cliente. De aprobarse la personalización que el cliente requiere, se procede a su implementación por parte del equipo de desarrollo siendo obligatorio el uso de aplicaciones y herramientas externas al HIS. Posteriormente es necesaria la actualización del sistema con el resultado de la implementación, para lo cual el funcionamiento del HIS debe interrumpirse momentáneamente afectando el servicio en la institución de salud. Este proceso descrito anteriormente no necesariamente culmina con la aprobación de la solicitud de personalización, o bien puede demorarse un tiempo relativamente largo por lo que el cliente en cualquiera de las variantes termina insatisfecho y se afectan actividades como los procesos científicos, de investigación y toma de decisiones en las instituciones.

Los escenarios anteriormente descritos evidencian la necesidad de permitirles a los usuarios de los sistemas de gestión hospitalaria la generación de reportes acorde a sus necesidades, en tiempo real, sin

depender del conocimiento de diseños de reportes, la localización exacta de los datos en las bases de datos o el dominio de algún lenguaje de consulta y programación.

El presente trabajo surge para darle solución a la situación antes expuesta, por lo que el ***problema a resolver*** queda formulado de la siguiente forma: ¿Cómo permitir que los usuarios puedan elaborar nuevos diseños de reportes de forma dinámica?

Considerando como ***objeto de estudio*** la generación de reportes estadísticos. Enmarcado en el ***campo de acción*** el proceso de generación dinámica de reportes estadísticos basados en conceptos.

Para solucionar el problema identificado se define como ***objetivo general*** desarrollar un componente de generación de reportes dinámicos basados en conceptos para el módulo Estadísticas del Sistema de Información Hospitalaria del Centro de Informática Médica.

Para dar cumplimiento al objetivo planteado, se precisan las siguientes ***tareas de la investigación:***

- Analizar las tendencias actuales de las soluciones informáticas relacionadas a la generación de reportes.
- Caracterizar las tendencias actuales de los indicadores de gestión en instituciones de atención médica.
- Asimilar la arquitectura definida por el departamento Sistemas de Gestión Hospitalaria para el desarrollo de sus aplicaciones.
- Obtener mediante el Proceso Unificado de Desarrollo, los artefactos correspondientes a los flujos de trabajo “Gestión de Requerimientos”, “Modelo de Dominio”, “Diseño” e “Implementación”.
- Implementar el proceso de negocio relacionado con la generación dinámica de reportes a partir de conceptos.

Con el desarrollo del componente de generación dinámico de reportes basado en conceptos se esperan obtener los siguientes ***beneficios:***

- Con el componente generador de reportes dinámicos basados en conceptos se espera que los usuarios cuenten con una herramienta intuitiva y organizada, que permita construir los reportes al personal médico de acuerdo a sus necesidades, favoreciendo la toma de decisiones y la investigación científica.

El documento se encuentra dividido en cuatro capítulos, estructurados de la siguiente manera:

**Capítulo 1: Fundamentación Teórica del Componente de Generación de reportes dinámicos basado en conceptos:** se presentan las principales definiciones que constituyen las bases para el desarrollo del generador dinámico de reportes basado en conceptos. Se realiza un estudio preliminar de varios sistemas relacionados con la generación de reportes. Se fundamentan las tecnologías, metodologías y herramientas de desarrollo a utilizar.

**Capítulo 2: Características del Componente de Generación de reportes dinámicos basado en conceptos:** se describen las principales características del sistema. Se presenta el Modelo de Dominio de la aplicación, conjuntamente con la especificación de los requerimientos funcionales y no funcionales y el modelo de casos de usos del sistema.

**Capítulo 3: Análisis y diseño del Componente de Generación de reportes dinámicos basado en conceptos:** Se realiza una descripción y análisis de la estructura de la solución que se propone para dar respuesta a la problemática planteada. Se fundamenta la arquitectura empleada, así como las estrategias de integración a tener en cuenta.

**Capítulo 4: Implementación del Componente de Generación de reportes dinámicos basado en conceptos:** Se introduce el flujo de trabajo de implementación, partiendo de los resultados obtenidos en el diseño. Se detalla el Modelo de Datos, en el que se ve la estructura donde se almacena toda la información requerida en el sistema y se exponen aspectos referentes a la seguridad del mismo, las estrategias de codificación, así como la forma en que se tratarán los errores.

# Capítulo 1. Fundamentación Teórica del Componente de Generación de reportes dinámicos basado en conceptos

En el presente capítulo se da a conocer la base teórica y conceptual para el cumplimiento de la generación dinámica de reportes en el módulo Estadísticas del Sistema de Información Hospitalario del CESIM. Como aspectos fundamentales se abordan los principales conceptos relacionados con el Dominio del problema, el campo de acción y los antecedentes existentes de sistemas o subsistemas. Se analizan las tecnologías, metodologías y herramientas a considerar en el proceso de desarrollo de software.

## 1.1 Conceptos básicos asociados a la generación de reportes dinámicos basados en conceptos

**Indicador hospitalario** (4)**:** es la representación de los datos numérico referido a la persona, a la población, a sus problemas, a sus situaciones y el procedimiento o proceso de todas las actividades que identifican la atención médica.

**Concepto:** (para el informático) es la agrupación de un conjunto de tablas o recursos bajo una clasificación que pertenece al campo de acción de los profesionales de la salud.

**Reporte**: son documentos que pretenden representar una información contenida en la base de datos, puede ser de forma digital o impresos, por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios.

**Reporte dinámico**: es la forma dinámica y simplificada de realizar reportes, ayuda a la toma de decisiones y permite al cliente abstraerse de los temas relacionados con el gestor de base de datos. Posee flexibilidad, lo que posibilita que el usuario estructure el reporte de acuerdo a sus necesidades y a los factores que requiera tener en cuenta en el estudio.

## 1.2 Sistemas de salud pública

La salud (5) constituye un derecho social fundamental de las personas y como parte del derecho a la vida, en el marco de un estado democrático, es un estado de completo bienestar físico, mental y social, y no solamente la ausencia de afecciones o enfermedades.

Los sistemas de salud pública (6) constituyen la base de la salud a nivel poblacional, cuyo objetivo fundamental es el de mantenerla y mejorarla. Se alcanza mediante la vigilancia, evaluación y análisis de la situación de salud de la sociedad, así como la acción inmediata sobre situaciones de emergencia sanitaria.

Para ello existen tres niveles básicos de atención:

1. *La Atención Primaria*: es el de mayor cobertura pero menor complejidad y está representado por los centros de atención más básico de salud familiar. Es la asistencia sanitaria esencial, accesible a todos los individuos y familias de la comunidad a través de medios aceptables para ellos, con su plena participación y de forma gratuita para la comunidad y el país. Es el núcleo del sistema de salud del país y forma parte integral del desarrollo socioeconómico general de la comunidad.

2. *La Atención Secundaria*: recibe para diagnóstico y tratamiento a los pacientes cuyas afectaciones no pueden ser resueltas en el nivel primario. Generalmente está estrechamente relacionado con los servicios clínicos de los hospitales y da cobertura a cerca del 15 % de los problemas de salud.

3. *La Atención Terciaria*: se encuentra representada por los establecimientos con condiciones para realizar acciones bajo régimen de atención cerrada, de ahí que su nivel de cobertura es menor y se aboca a manejar solo los casos con patología a excepcionales. A este nivel pertenecen los institutos y hospitales especializados y abarca alrededor del 5 % de los problemas de salud de la población.

## 1.3 La estadística en los sistemas de salud pública

En salud pública se emplean muchos conceptos estadísticos al adoptar decisiones relativas a diagnósticos clínicos, o bien al predecir probables resultados de un programa de intervención en la población. Considerando que la estadística es una excelente base para comprender muchos fenómenos reales y orientar la resolución de problemas relativos, es importante poder definirla mediante el conocimiento de diferentes autores. La MSc. Ligia Moya, en su libro "Introducción a la Estadística de la Salud" (7), la define como:

"La rama del saber que trata del desarrollo y aplicación de métodos eficientes de recolección, elaboración, presentación, análisis e interpretación de datos numéricos".

El Dr.Mark M. Spiegel Director del Centro de Estudios de la Cuenca del Pacífico en el Banco de la Reserva Federal de San Francisco, lo define como:

"…estudia los métodos científicos para recoger, organizar, resumir y analizar datos, así como para sacar conclusiones válidas y tomar decisiones razonables basadas en tal análisis".

Los autores de la presente investigación, asumen como concepto de estadística a utilizar y referenciar en el resto de la investigación al dado por la MSc. Ligia Moya.

## 1.4 Sistemas de Información Hospitalaria (HIS)

Un Sistema de Información Hospitalaria (8) “Es un sistema de información orientado a satisfacer las necesidades de generación de información, para almacenar, procesar y reinterpretar datos médico-administrativos de cualquier institución hospitalaria permitiendo la optimización de los recursos humanos y materiales”. Todo Sistema de Información Hospitalaria genera reportes e informes dependiendo del área o servicio para el cual se requiera, dando lugar a la retroalimentación de la calidad de la atención de los servicios de salud.

## 1.5 Sistemas de Información Hospitalaria y la generación de reportes

Tanto a nivel mundial como nacional existen diversos Sistemas de Información Hospitalaria que se especializan en las distintas áreas de atención a la salud, teniendo como característica principal la generación de reportes o informes estadísticos. Sobre algunos de ellos se hará mención seguidamente.

### 1.5.1 Galen Hospital

Galen Hospital (9) es un sistema que está orientado hacia la informatización de la gestión de pacientes como elemento básico de control para mejorar la atención médica, optimizar el uso personal, aumentar la calidad de los servicios hospitalarios y disminuir sus costos. Adicionalmente brinda la información requerida para la actividad gerencial a todos los niveles y la elaboración de reportes estadísticos.

Los pacientes pueden obtener su informe de resultados de forma unificada según lo solicitado. Los resultados de las solicitudes pueden ser obtenidos por varios criterios incluyendo examen o consulta, fecha e instituciones. Se pueden obtener estadísticas por exámenes, titulares, técnicos o médicos que realizan las pruebas o exámenes.

### 1.5.2 Comlogik HIMS

El Sistema de Información y Administración Hospitalaria (HIMS, por sus siglas en inglés) (10), es desarrollado totalmente por el Sistema Empresarial Comlogik y está concebido exclusivamente para la gestión de la información en instituciones de atención hospitalaria. Está basado en la arquitectura cliente servidor y es totalmente escalable para todos los niveles de atención en instituciones de salud.

HIMS almacena toda la información de los pacientes en una sola base de datos, lo que permite que la información sea visible de forma simultánea por varios especialistas a la vez y asegurando la actualización de esta al instante de su modificación o agregación. La centralización de los datos y su estructura de control también permiten el uso de técnicas avanzadas para la salva de seguridad de la información.

En el centro del HIMS se encuentra la herramienta Grabado Médico Electrónico (EMR, por sus siglas en inglés), este, gracias al diseño singular de la base de datos, integra toda la documentación generada por los procesos de atención a los pacientes desde su ingreso en la institución y genera una sola historia clínica por cada paciente. Esto permite la generación electrónica de reportes e informes gráficos para la obtención de resultados estadísticos, los cuales son vitales para la toma de decisiones de la entidad.

### 1.5.3 3M HIS

La solución 3M HIS (11) es un Sistema Inteligente de Gestión de Información Hospitalaria que cuenta con un conjunto de herramientas que ayudan a la recopilación y uso de la información clínica de los pacientes para mejorar los procesos de atención.

Entre las principales características del sistema se encuentra la de ser expertos en CIE-10 para la estandarización de enfermedades. Adicionalmente tiene aplicaciones innovadoras y servicios de consulta para el mejoramiento de la documentación clínica.

3M HIS posee excelentes herramientas para registros médicos de resúmenes para recolectar, agrupar y reportar datos de pacientes de la entidad médica. Dichas herramientas permiten mejorar las entradas de datos de pacientes y reducir el tiempo de proceso, así como proporcionar capacidades avanzadas, realizar informes clínicos y agilizar el flujo de trabajo. Este sistema presenta además una combinación de reportes analíticos especializados para el estudio de datos clínicos, lo que apoya la toma de decisiones a nivel administrativo.

### 1.5.4 Care2x

Care2x (12) es un Sistema de Información Hospitalaria de código abierto y multiplataforma. Es una aplicación web totalmente configurable y flexible para cualquier estructura clínica y está compuesto por varios módulos que gestionan, de forma general, la entrada y salida de datos de pacientes.

Cuenta con una comunidad, que está creciendo en tamaño, razón por la que gana en desarrollo y publicidad entre las instituciones de atención hospitalarias del mundo. Desafortunadamente esta aplicación aún no cuenta con un módulo de reportes o alguna especie de generadores de informes clínicos que responda a las necesidades de la administración para la toma de decisiones.

Luego de analizar detenidamente los HIS antes descritos, se concluye que ninguno cumple con las exigencias requeridas para la generación de reportes tal como se desea. Solo Care2x tiene carácter libre, siendo una premisa para el presente trabajo, pero el mismo, por ser una aplicación aún muy joven, no cuenta con el módulo de reportes. El resto de las aplicaciones analizadas son de carácter privativo, por esta razón se descarta el uso de algún componente que permitiera la creación de reportes, adicionalmente, en algunos casos son aplicaciones de escritorio o hacen uso de tecnologías no compatibles y no son sistemas multiplataforma.

Luego de analizar detenidamente los HIS antes descritos, se concluye que ninguno cumple con las exigencias requeridas para la generación de reportes, pues de conjunto ninguno posee una gestión personalizada de los reportes o informes que generan, estos se encuentran previamente diseñados o definidos quedando como única opción para el especialista de salud el uso de estos.

## 1.6 Sistemas de generación de reportes dinámicos

Independientemente de los Sistemas de Información Hospitalarios, son variadas las aplicaciones que se crean con el propósito de la generación de reportes con un fin determinado. Estas surgen generalmente con una idea específica y un campo de acción limitado, que va creciendo gradualmente en cuanto se desarrolla y gana tanto en utilidad como en integralidad con otros sectores y campos de la sociedad. En esta sección se hará mención de algunas aplicaciones dirigidas a la generación dinámica de reportes, haciendo especial enfoque en sus capacidades de integración con sistemas externos.

### 1.6.1 Generador Dinámico de Reportes

El Generador Dinámico de Reportes (GDR) (13) es una aplicación web que tiene como principal objetivo generar reportes de forma rápida e interactiva. Está compuesto por una serie de módulos independientes que le brindan total flexibilidad para acoplarse a sistemas empresariales que necesiten de sus servicios.

Entre sus principales ventajas se encuentran las de permitirle a los usuarios, agilizar la toma de decisiones y generar reportes en varios formatos y con gran variedad de opciones en su diseño. Adicionalmente, el sistema íntegro está desarrollado con tecnologías libres como el lenguaje de programación PHP del lado del servidor, el framework Symfony 1.2.7 y para el diseño de su interfaz gráfica de usuario se utilizó Ext Js 2.2. Las tecnologías suponen riesgos de compatibilidad y dificultad al integrar con el HIS en desarrollo.

El principal inconveniente del Generador Dinámico de Reportes es que no se encuentra totalmente orientado al usuario con conocimientos nulos en Bases de Datos, puesto que una parte de él basa su funcionamiento en la suposición de que el personal informático posee conocimientos previos en esta área, trabajar con vistas, tablas y estructurar consultas. Por tanto se crea la dependencia de un intermediario entre el usuario y los resultados que muestre el reporte, complejizando el proceso de generación del mismo.

### 1.6.2 Pentaho

Pentaho BI (Business Intelligence, Inteligencia de Negocios) (14) es un conjunto de herramientas libres que tienen como objetivo primario generar inteligencia a nivel empresarial. Este proceso fluye a través de la extracción de datos desde una fuente hasta el procesamiento y análisis de los mismos de forma correcta para brindar información relevante a los administrativos.

Las funcionalidades de Pentaho incluyen el Procesamiento Analítico en Línea (OLAP), aprendizaje inteligente junto a la minería de datos, un motor de presentación dirigido a la generación de informes, así como también, permite visualizar mediante una plataforma integrada los datos en cuestión en forma de informes o gráficas. Estas entre otras más son, a grandes rasgos, las virtudes y ventajas del uso de este conjunto de herramientas de aplicación empresarial.

De los sistemas de generación de reportes dinámicos antes expuestos se descartaron la posibilidad de utilizar dichas soluciones, pues a pesar de las ventajas que poseen, tienen una serie de inconvenientes a considerar. Las tecnologías utilizadas para el desarrollado de la aplicación web GDR no son compatibles

con las asumidas por el HIS del Centro de Informática Médica para la elaboración del reportador dinámico basado en conceptos.

Las herramientas de la Suite de Pentaho, son robustas y desafiante para sistemas informáticos empresariales de gran tamaño, que abarquen bases de conocimiento variadas y de grandes dimensiones, es bajo estas condiciones, cuando realmente se explotan a fondo sus potencialidades, de lo contrario solo sería una estructura compleja a nivel de almacenamiento de datos y con ello se alcanzaría la posible sobrecarga del sistema. Si se tiene en cuenta que el HIS almacena todos sus datos en una sola base de datos, no se puede hacer uso de esta solución por las características descritas.

## 1.7 Tendencias y tecnologías a considerar

Teniendo en cuenta la ausencia de soluciones informáticas que se acoplen fácilmente y que brinden respuesta a las necesidades planteadas, se hace evidente la búsqueda de otros métodos para generar reportes con la calidad y rapidez requerida.

Seguidamente se hace referencia a diferentes tecnologías de carácter libre para dar continuación a la soberanía tecnológica.

### 1.7.1 JasperReports

JasperReports (15) es una de las herramientas de código abierto más populares a nivel mundial y puede ser usada en gran variedad de aplicaciones de Java, incluyendo las web, para generar contenido dinámico. Está escrita en su totalidad en el entorno de Java y permite la extracción de datos desde cualquier fuente de base de datos, también permite visualizar de forma previa los reportes, así como imprimirlos o exportarlos hacia una variedad de formatos tales como PDF, Excel, HTML, Open Office y Word.

JasperReports tiene algunas dependencias externas del sistema como:

- Java Development Kit (JDK) 1.6 o superior.
- Java Database Conectivity (JDBC) 2.1 o superior.

### 1.7.2 DynamicReports

DynamicReports (16) es una herramienta de reportes de código abierto y escrito en Java, basado en JasperReports, permite crear diseños de reportes de forma dinámica sin necesidad de una interfaz gráfica

que medie esta operación, haciendo uso solamente de código en java. Adicionalmente tiene la capacidad de aplicar la herencia entre los reportes, lo que permite diseñar un reporte con las características de otro previamente creado. Está basado en JasperReports, posee todas sus facilidades y virtudes.

La facilidad de uso de DynamicReports radica no sólo en ahorrar tiempo invertido en el desarrollo de los reportes, sino también en incrementarla productividad del mismo. A través de esta se pueden visualizar los reportes o exportarlos a muchos de los formatos más comunes y corrientes tales como documentos Word, PDF, Excel.

### 1.7.3 DynamicJasper

DynamicJasper (17) es una herramienta libre escrita en java que corre sobre JasperReports y simplifica enormemente la realización de reportes, detrás de su desarrollo se encuentra una comunidad que está en constante búsqueda de mejoras que a esta se le puedan realizar.

Brinda posibilidades como agregar gráficas, encadenar reportes, añadir códigos de barras a las columnas, entre otras facilidades, además de poder usarse sin mucha complejidad. Su funcionamiento básico se centra en hacer más intuitivo y amigable JasperReports, del cual se abstrae totalmente el programador, o sea que se interactúa con DynamicJasper aunque realmente se sigue usando JasperReports, pero sin seguir todos los engorrosos pasos que supone usarlo. Es fácilmente integrable en proyectos que empleen el lenguaje Java. De forma adicional no es necesaria ninguna herramienta extra o complemento al margen del presente entorno de desarrollo.

## 1.8 Algoritmos de búsqueda de camino

Para el desarrollo del componente es necesario la utilización de un método que encuentre los posibles caminos entre las tablas que pueden conformar un concepto, teniendo en cuenta se puede relacionar con un conjunto de sus homólogos. Por lo que se hizo necesario el estudio de los siguientes algoritmos de búsqueda:

### 1.8.1 Algoritmo de Dijkstra

Dijkstra (18), también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista. Es un algoritmo ávido o glotón que trabaja por etapas y toma en cada una la mejor solución sin considerar

consecuencias futuras, el óptimo encontrado en una etapa puede modificarse posteriormente si surge una solución mejor.

### 1.8.2 Algoritmo de Floyd

Una primera solución para encontrar el camino mínimo para todo par de vértices pertenecientes a un grafo, consiste en usar repetidamente el algoritmo de Dijkstra variando el vértice inicial. Para solucionar esto está el algoritmo de Floyd (19) cuyo objetivo es encontrar el camino más corto entre todos los pares de nodos o vértices de un grafo.

### 1.8.3 Comparación entre los algoritmos de Dijkstra y Floyd

Dijkstra con matriz de adyacencia puede encontrar las caminos más cortos desde un vértice en un tiempo O(n*n), como el algoritmo de Floyd, también puede encontrar todos los caminos más cortos en un tiempo O(n*n*n). El compilador, la máquina y los detalles de realización determinarán las constantes de proporcionalidad.

Luego de analizar las características de los algoritmos se determinó por parte de los autores descartar el posible uso de estos para el desarrollo del componente, pues no encuentran todos los posibles caminos entre las tablas del sistema, teniendo en cuenta que cada par de tablas se pueden vincular a través de uno o varios atributos.

## 1.9 Arquitectura y patrón de diseño

La arquitectura en capas y el patrón Modelo Vista Controlador (MVC) (20), pueden relacionarse lógicamente mediante cada uno de sus elementos, donde la capa de presentación podría corresponderse con la Vista, la capa de negocio con el Controlador y la capa de datos con el Modelo. El MVC separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes.

## 1.10 Tecnologías utilizadas en el proceso de desarrollo

Para el desarrollo de la presente investigación se asumen las tecnologías, metodologías y arquitectura previamente definidas por el Centro de Informática Médica. Se utilizará la biblioteca de clases (DynamicReports) para la generación de reporte en forma dinámica. Son numerosas las ventajas que presenta esta librería, la más importante que posee es la de lograr con pocas líneas de código la

generación de reportes muy complejos, dicha ventaja es bastante útil para el programador y todo aquel personal que necesite entender el código en el momento de brindarle mantenimiento a la aplicación.

A continuación se presentan las tecnologías asumidas y lenguaje de programación, según su ubicación dentro de la arquitectura en capas (presentación, negocio, datos). Se mostrará además una relación de las herramientas propuestas para el desarrollo del proyecto en cuestión.

### 1.10.1 Java

Java (21) es un lenguaje orientado a objetos, con el objetivo de ser utilizado en la programación web. Presenta varias semejanzas con lenguajes como C, C++ y C#. Es altamente seguro e indiferente a la arquitectura pues está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red. Logra cumplir el principal objetivo que persigue el HIS del Centro de Informática Médica que es estar libre del costo relacionado con patentes de software**,** asociadas al servidor de aplicaciones, al servidor de base de datos, al sistema operativo huésped u otras herramientas o tecnologías utilizadas para su desarrollo.

### 1.10.2 Vista

La Vista (22) es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa preferentemente con el Controlador, pero es posible que trate directamente con el Modelo a través de una referencia al propio Modelo.

#### 1.10.2.1 Java Server Faces (JSF)

JSF (23) es una tecnología y un ambiente de desarrollo o corrida (framework), basado en el patrón MVC. Ofrece una clara separación entre el comportamiento y la presentación, une los componentes de la interfaz de usuario (UI, por sus siglas en inglés) con los conceptos de la capa web sin limitarse a una tecnología de script o lenguaje de marcas particular.

JSF brinda un conjunto de componentes de interfaz de usuario del lado del servidor para aplicaciones web basadas en Java. Permite el manejo de estados, eventos y la asociación entre datos de la interfaz y los de la aplicación web.

#### 1.10.2.2 RichFaces

*RichFaces* (24) es un conjunto de componentes visuales para JSF, posee un framework avanzado para la integración de funcionalidades Ajax en dichos componentes visuales, mediante el soporte de Ajax4JSF. RichFaces se integra perfectamente en el ciclo de vida de JSF, provee varios componentes como Core, Ajax, UI y una administración avanzada de recursos como imágenes, código JavaScript y Hojas de Estilo en Cascada (CSS). Con su uso es posible crear interfaces de usuario de manera rápida y eficiente, basado en componentes que están listos para usar y son altamente configurables. RichFaces, además, es un proyecto desarrollado con una arquitectura abierta lo que facilita la compatibilidad con la mayor cantidad de entornos.

#### 1.10.2.3 Ajax4JSF

*Ajax4JSF* (25) es una herramienta de código abierto que se integra totalmente a la arquitectura de JSF y extiende la funcionalidad de sus etiquetas dotándolas con tecnología Ajax de forma limpia y sin añadir código JavaScript. Mediante este framework se puede variar el ciclo de vida de una petición JSF, recargar determinados componentes de la página sin necesidad de recargarla por completo, realizar peticiones automáticas al servidor y controlar cualquier evento del usuario. Esta herramienta permite dotar a la aplicación JSF de contenido mucho más profesional con muy poco esfuerzo.

#### 1.10.2.4 Facelets

*Facelets* (26) es un framework para plantillas centrado en la tecnología JSF, permite el uso de plantillas en aplicaciones. No depende de un contenedor web. Las principales ventajas de Facelets son la construcción de interfaces basadas en plantillas, una rápida y fácil creación de funciones y componentes por composición.

#### 1.10.2.5 SeamUI

*SeamUI* (27) son una serie de controles JSF que adicionan varias mejoras, desde validación hasta integración de la navegación en la interfaz de usuario basada en flujo de páginas o procesos del negocio.

#### 1.10.2.6 Lenguaje de Marcado de Hipertexto Extensible (XHTML)

Lenguaje de Marcado de Hipertexto Extensible (XHTML, por sus siglas en inglés) (28), es una versión del Lenguaje de Marcado de Hipertexto (HTML, por sus siglas en inglés), que nace precisamente con el

objetivo de remplazar a HTML ante su limitación de uso con las herramientas basadas en el Lenguaje de Marcado Extensible (XML). Su objetivo es avanzar en el proyecto del World Wide Web Consortium (W3C) de lograr una web semántica, donde la información y la forma de presentarla estén claramente separadas.

### 1.10.3 Controlador

El controlador (29) interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

#### 1.10.3.1 JBossSeam

JBossSeam (27) es una poderosa plataforma de desarrollo de código abierto para la construcción de aplicaciones empresariales en Java, integra tecnologías como Ajax, JSF, Java Persistence API (JPA), Enterprise Java Beans (EJB 3.0) y Business Process Management (BPM). Fue diseñado para posibilitar a los desarrolladores de aplicaciones empresariales, ensamblar complejas aplicaciones usando simples clases Java, un rico conjunto de componentes de interfaces de usuario y muy poca configuración mediante archivos XML. Brinda un soporte único para las conversaciones.

### 1.10.4 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).

#### 1.10.4.1 Java Persistence API (JPA)

JPA (30) es una interfaz de programación de aplicaciones (por sus siglas en inglés API) de persistencia de POJOs (Plain Old Java Object), es decir, objetos simples que no heredan, ni implementan otras clases (como los EJBs). En su definición, combina ideas y conceptos de los principales frameworks de persistencia, como Hibernate, Toplink y JDO (Java Data Object) y de las versiones anteriores de EJB. Es una especificación del gestor de bases de datos Oracle para la persistencia de objetos Java a cualquier base de datos relacional.

Proporciona un estándar para gestionar datos relacionales en aplicaciones Java Platform, Standard Edition (por sus siglas en inglés Java SE) o Java Platform**,** Enterprise Edition (por sus siglas en inglés

Java EE), de forma que además se simplifique el desarrollo de la persistencia de datos. Esta API fue desarrollada para la plataforma JEE.

#### 1.10.4.2 Enterprise JavaBeans (EJB3)

Una de las metas de la arquitectura EJB (31) es la de poder escribir, de manera fácil, aplicaciones de negocio orientadas a objetos y distribuidas, basadas en el lenguaje de programación Java. El propósito de EJB3 es el de proveer el soporte de la arquitectura de EJB y al mismo tiempo reducir la complejidad para el desarrollo de aplicaciones empresariales.

#### 1.10.4.3 Hibernate

Hibernate (31) es una herramienta de Mapeo Objeto Relacional (ORM, por sus siglas en inglés) y un generador de sentencias SQL (Structured Query Language). Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos.

## 1.11 Tecnologías y metodologías horizontales

### 1.11.1 Proceso Unificado de Desarrollo (RUP)

RUP es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (por sus siglas en inglés UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El ciclo de vida de RUP (32) se caracteriza por ser dirigido por caso de uso, centrado en la arquitectura, iterativo e incremental.

### 1.11.2 Lenguaje de Modelado Unificado (UML)

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) (33), es la sucesión de una serie de métodos de análisis y diseño orientado a objetos, siendo como su nombre lo indica, un lenguaje de modelado y no un método. UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientado a objetos. Los autores de UML apuntan también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios. Se utiliza UML en su versión 2.1.

### 1.11.3 PostrgeSQL

Es un Sistema Gestor de Base de Datos (SGBD) (34) relacional de código abierto muy poderoso, con una arquitectura probada. Es un sistema multiplataforma, que puede ser utilizado en la mayoría de los

sistemas operativos actuales, consta de varias versiones, entre ellas la 8.4 que es la que se utiliza y hasta sus más recientes como la 9.1. Presenta características que no tienen otros SGBD como son los tipos de datos definidos por el usuario, la herencia y el uso de normas. Además cuenta con el apoyo de una comunidad de usuarios que colabora activamente con su desarrollo.

Utiliza el lenguaje SQL para la construcción de las consultas. Emplea un modelo cliente/servidor y usa multiprocesos en vez de multihilos, donde cada conexión tiene su propio proceso desde el punto de vista del servidor, para garantizar la estabilidad del sistema, por lo que un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

### 1.11.4 JBossApplication Server

JBossApplication Server (27) es uno de los servidores de aplicaciones de código abierto más ampliamente desarrollado del mercado. Por ser una plataforma certificada J2EE, presta servicios adicionales como clustering, caching y persistencia. JBoss es ideal para aplicaciones Java y aplicaciones basadas en la web, soporta Enterprise Java Beans (EJB), lo que hace que el desarrollo de las aplicaciones sea mucho más fácil. Se utiliza JBossApplication Server en su versión 4.2.2 por las características y ventajas que brinda.

## 1.12 Herramientas

Luego de un análisis de los elementos anteriormente expuestos, se ha definido las siguientes herramientas a utilizar en el ambiente de desarrollo:

### 1.12.1 Visual Paradigm

Visual Paradigm (33) para UML, es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, además de generar código desde diagramas y generar documentación. Se utiliza Visual Paradigm en su versión 8.0 dadas las características y ventajas que brinda.

### 1.12.2 PgAdmin

PgAdmin es una herramienta de código abierto para la administración de bases de datos, está diseñada para responder a las necesidades de la mayoría de los usuarios, desde escribir simples consultas SQL

hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostrgeSQL y hace simple la administración. Está disponible en más de una docena de lenguajes y para varios sistemas operativos, incluyendo Microsoft Windows y Linux.

### 1.12.3 JBossDeveloper Studio

JBoss Developer Studio (JBDS) (35) es una plataforma gratuita e independiente, de código abierto, utilizada para crear aplicaciones de cualquier tipo. Proporciona un rendimiento superior para todo el ciclo de vida de desarrollo. JBDS incluye un amplio conjunto de capacidades de las herramientas y soporte para múltiples modelos y estructuras de programación, incluyendo Java Enterprise Edition 6, RichFaces, JavaServer Faces (JSF), Enterprise JavaBeans (EJB), Java Persistence API (JPA), Hibernate, HTML5, y muchas otras tecnologías populares.

Proporciona elección al desarrollador en el apoyo al uso de varias Máquinas Virtuales de Java (JVM, Java Virtual Machine). Está totalmente probado y certificado para asegurar que todos sus plug-ins, componentes de tiempo de ejecución y sus dependencias sean compatibles entre sí. Se utiliza la presente plataforma en su versión 5.0.0.

## Conclusiones

En este capítulo se realiza un estudio caracterizador de diversos Sistemas de Información Hospitalarias y de otras aplicaciones para la generación de reportes dinámicos. Se lleva a cabo un análisis enfocado en los reportes e informes que estos generan, reportes que varían desde su aplicación, hasta la forma en que son implementados, así como en su capacidad de integración al HIS del CESIM. Como resultado se decide no utilizar ninguno de los algoritmos búsqueda estudiados para el desarrollo del componente y valerse de la librería de clases DynamicReports para generar los reportes de forma dinámica.

# Capítulo 2. Características del Componente de Generación de reportes dinámicos basado en conceptos

En el presente capítulo se describen, de forma general, las características del sistema, se indaga en los procesos de generación de reportes que ocurren dentro del módulo Estadística y se realiza un análisis crítico de su funcionamiento y ejecución en la actualidad.

Se describe el modelo de dominio o modelo conceptual que representa los diferentes entes participantes en el proceso de generación de reportes y la relación existente entre ellos. Se enumeran los requerimientos y se explica la propuesta de solución.

## 2.1 Proceso de generación de reportes en el HIS del CESIM

Actualmente el HIS del CESIM está compuesto por un conjunto de módulos interrelacionados, los cuales son los encargados de gestionar la información de los pacientes obtenida de la atención médica en las diferentes áreas y especialidades de una entidad hospitalaria. Los datos anteriormente adquiridos se utilizan en cada una de las de generación de informes o reportes estadísticos que tributan a la toma de decisiones a nivel administrativo.

El módulo de Estadística específicamente es el encargado de generar los reportes antes mencionados. El proceso tiene lugar una vez que el usuario entra al módulo y escoge unos de los indicadores hospitalarios previamente definidos, llena los campos necesarios para su confección y el sistema se encarga de generarlo. La estructura de los reportes no puede ser modificada, son desarrollados de forma estática. Para la elaboración de un informe que no coincida con los presentes en el sistema, el personal de salud debe presentar los requerimientos al equipo de desarrollo para la implementación de dicho reporte. Paralelamente al proceso de desarrollo el especialista, en aras de suplir la necesidad de información, genera los reportes que contienen la información base y utiliza herramientas externas como Excel para conformar a partir de la información antes mencionada el informe que necesita.

## 2.2 Modelo de Dominio

El modelo de dominio es un artefacto construido con las reglas del Lenguaje Unificado de Modelado durante la fase de concepción, que presenta uno a más diagramas de clases que contienen, no conceptos propios de un sistema de software sino de la propia realidad física.

Según el canadiense, M.Sc.Craig Larman (36): “Un modelo de dominio es una representación de las clases conceptuales del mundo real, no de componentes de software. No se trata de un conjunto de diagramas que describen clases de software, u objetos de software con responsabilidades.”

### 2.2.1 Conceptos fundamentales del dominio

A continuación se hará una breve descripción de clases conceptuales significativas en el dominio del problema planteado:

**Institución de salud:** establecimiento sanitario donde se atiende a los enfermos para proporcionar el diagnóstico y tratamiento que necesitan.

**Reporte:** documento que pretende brindar una información de carácter estadístico sobre algún indicador hospitalario con el objetivo de facilitar su estudio.

**Usuario:** es el personal que opera con la aplicación y construye el diseño del reporte.

**Concepto:** (para el personal de salud) es una definición, objeto o término asociado a su campo de acción.

### 2.2.2 Diagrama del Modelo de Dominio

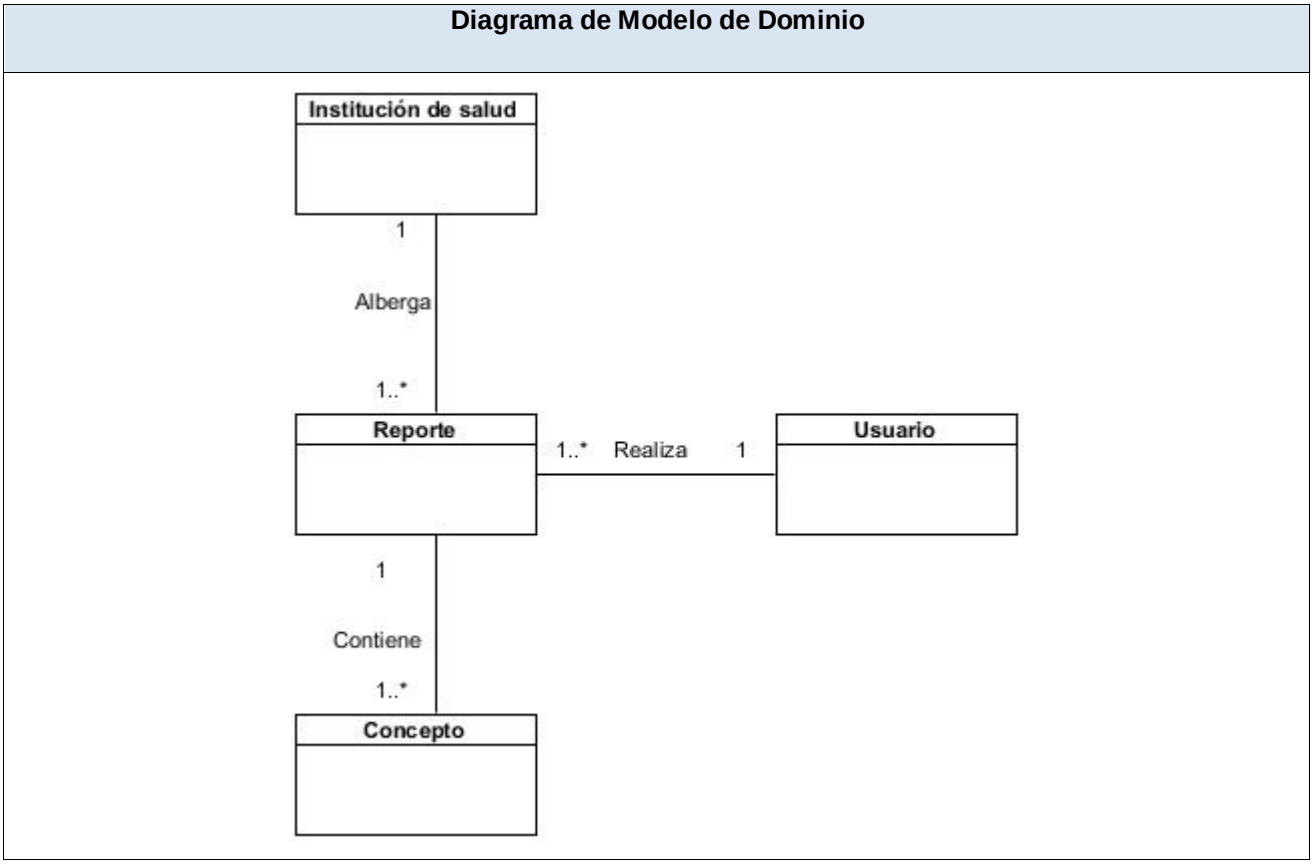



Figura 2.1 Diagrama de Modelo de Dominio

## 2.3 Especificación de requisitos de software

### 2.3.1 Requerimientos funcionales

Los requerimientos funcionales (37) especifican capacidades o condiciones que el sistema debe cumplir, sin tomar en consideración ningún tipo de restricción física, de manera que especifican el comportamiento de entrada y salida del sistema y surgen de la razón fundamental de la existencia del producto.

Se definen los siguientes requisitos funcionales:

| Nº | Nombre | Descripción |
|---|---|---|
| RF1 | Crear diseño de reporte | Permite al usuario seleccionar los datos con que se va a construir y persiste un diseño de reporte. |
| RF2 | Ver detalles de reporte | Permite ver los datos y la estructura con que se construyó el reporte. |
| RF3 | Listar diseño de reporte | Permite mostrar un listado de los diseños de reportes creados. |
| RF4 | Buscar diseño de reporte | Permite encontrar un diseño de reporte, si el mismo fue creado anteriormente y no ha sido eliminado. |
| RF5 | Eliminar diseño de reporte | Permite al usuario eliminar un reporte seleccionado. |
| RF6 | Generar reporte | Permite al usuario generar un reporte. |
| RF7 | Exportar reporte | Permite exportar un reporte en el formato deseado. |
| RF8 | Asignar privilegios de reportes | Permite a un usuario administrador añadir conceptos en la vista de otro usuario seleccionado para que pueda utilizarlos en la creación de los reportes. |

Tabla 2.1 Descripción de los requisitos funcionales.

### 2.3.2 Requerimientos no funcionales

Los requerimientos no funcionales (38) son restricciones que se imponen en el diseño o la implementación, propiedades o cualidades que el producto debe tener. Deben pensarse en estas propiedades como las características que hacen que el producto sea usable, rápido y confiable.

Los requerimientos no funcionales cumplen un papel fundamental en la aceptación del cliente y en el cumplimiento de las especificaciones del contrato. A continuación se especifican los requisitos no funcionales (RNF) que se deben tener en cuenta para lograr una solución de forma eficiente. Es necesario aclarar que se asumen aquellos definidos en el documento de Especificación de Requisitos de Software

Elementos Comunes perteneciente al departamento de Gestión Hospitalaria del CESIM. Los que se describen a continuación son aquellos que se adicionan como parte de la solución propuesta.

#### 2.3.2.1 Fiabilidad

El componente desarrollado inserta en la bitácora del sistema los eventos relacionados a la generación de los reportes por parte de los usuarios.

#### 2.3.2.2 Eficiencia

Las estructuras o diseños de reportes se almacenan compilados evitando el proceso de construcción de estos en el momento de la visualización del reporte.

## 2.4 Modelo de Casos de Uso del Sistema

### 2.4.1 Definición de los actores del sistema

Los actores de un sistema son agentes externos, roles que los usuarios o dispositivos juegan cuando interactúan con el componente de generación de reporte basados en conceptos.

| Actor | Descripción |
|---|---|
| Usuario | Usuario global que se autentica en la aplicación, el sistema lo valida y le asigna un rol con la posibilidad de generar un reporte basado en conceptos. |
| Súper Usuario | Se encarga de asignar el acceso de conceptos por usuarios y roles. |

Tabla 2.2 Definición de los actores del sistema

### 2.4.2 Vista global de actores del sistema

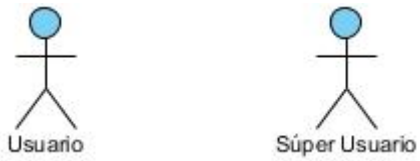
| Vista global de actores del sistema |
|---|
|  |

Figura 2.2 Vista global de actores del sistema

### 2.4.3 Diagrama de Caso de Uso del Sistema

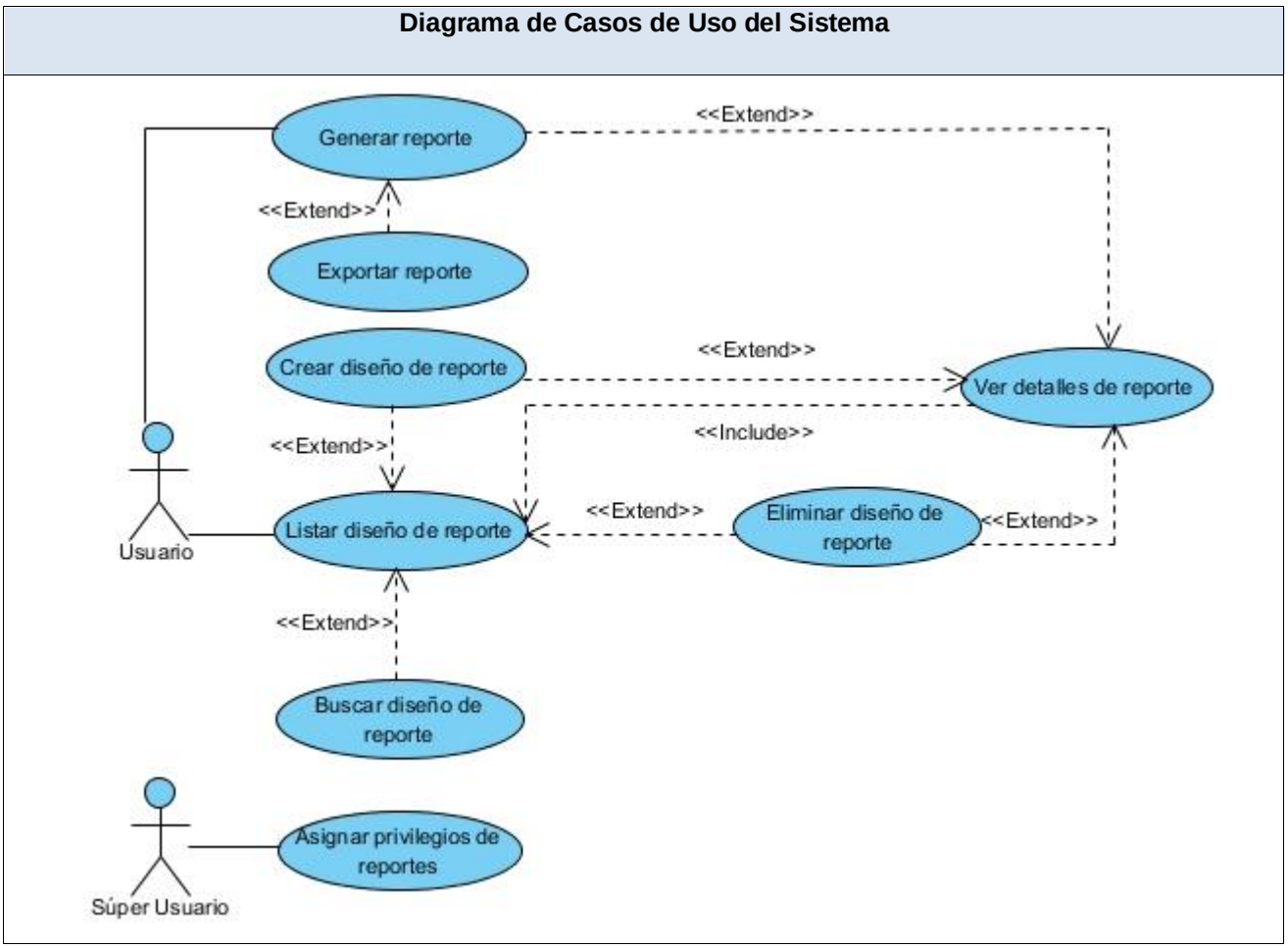



Figura 2.3 Diagrama de Casos de Uso del Sistema

El Modelo de casos de uso del sistema (39) es un artefacto de Ingeniería de Software que describe, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Esto permite el establecimiento de un acuerdo entre clientes y desarrolladores sobre las condiciones y requerimientos que debe cumplir el sistema.

Este modelo está formado por actores, casos de usos y las relaciones que se establecen entre ellos, es decir, representa gráficamente a los procesos y su interacción con los actores y constituye una entrada de gran valor para las siguientes fases de construcción de un software. A continuación se muestran una serie

de casos de usos, de los cuales, una breve selección de los más significativos para el sistema serán descritos, el resto de ellos se encuentran en artefacto 0114_Especificación de casos de uso.

### 2.4.4 Especificación de Casos de Uso

Descripción de los casos de usos principales, el resto se encuentra en el artefacto "Modelo de Casos de Uso del Sistema".

| **Caso de uso** | |
|---|---|
| **CU-1** | **Crear diseño de reporte** |
| **Propósito** | Construir un diseño de reporte. |
| **Actores** | Operador Informático. |
| **Resumen:** | El caso de uso inicia luego de que el actor selecciona la opción Crear reporte. El sistema muestra los conceptos habilitados para ese Operador Informático y una serie de formularios dependiendo de los conceptos seleccionados. |
| **Precondiciones** | No existe. |
| **Referencias** | RF2 Ver detalles de reporte. |

| **Flujo Normal de los Eventos** | |
|---|---|
| **Acción del actor** | **Respuesta del sistema** |
| 1. El caso de uso inicia cuando el actor accede a la opción Crear reporte. | |
| | 2. Muestra los datos predeterminados:<br>-Lista de conceptos a los que tiene acceso.<br>Y permite:<br>-Aceptar Crear diseño de reporte.<br>-Cancelar operación.<br>Ver **Alternativa 1**:"Cancelar operación". |

| | |
|---|---|
| 3. Selecciona los conceptos con que desea realizar el diseño del reporte. | |
| | 4. Muestra los conceptos seleccionados. |
| 5. Vincula los conceptos. | |
| | 6. Valida los datos. Si hay datos incorrectos, ver **Alternativa 2:** "Existen datos incorrectos.". |
| | 7. Muestra los caminos posibles para unir los conceptos seleccionados. |
| 8. Selecciona uno de los caminos. | |
| | 9. Muestra los campos de unión entre dos conceptos. |
| 10. Selecciona los campos de unión entre los conceptos. | |
| 11. Accede al botón Aceptar. | |
| | 12. Valida los datos. Si hay campos vacíos o sin seleccionar, ver **Alternativa 3:** "Campos vacíos.". |
| | 13. Muestra los valores contenidos en los conceptos. |
| 14. Selecciona los valores a incluir en el reporte. | |
| | 15. Muestra los valores seleccionados. |
| 16. Selecciona si el valor va estar ubicado en la fila o en la columna del reporte. | |
| 17. Accede al botón Aceptar. | |
| | 18. Valida los datos. Si hay campos vacíos o sin seleccionar, ver **Alternativa 3:** "Campos vacíos.". |
| | 19. Muestra la lista de valores que van estar ubicados en |

| | |
|---|---|
| | la fila del reporte. |
| 20. Selecciona el valor por el cuál agrupará los valores seleccionados anteriormente. | |
| 21. Accede al botón Agrupar. | |
| | 22. Valida los valores agrupados en las filas, ver **Alternativa 4:** “Agrupaciones duplicadas1.”. |
| | 23. Muestra la lista de valores que van estar ubicados en las columnas del reporte. |
| 24. Selecciona el valor por el cuál agrupará los valores seleccionados anteriormente. | |
| 25. Accede al botón Agrupar. | |
| | 26. Valida valores agrupados en las columnas, ver **Alternativa 5:** “Agrupaciones duplicadas2.”. |
| 27. Accede al botón Aceptar. | |
| | 28. Valida los datos. Si hay campos vacíos o sin seleccionar, ver **Alternativa 3:** “Campos vacíos.”. |
| | 29. Crea el diseño del reporte. |
| | 30. Muestra los datos de diseño del reporte. Ver caso de uso: **Ver detalles de reporte.** |
| | 31. Termina el caso de uso. |
| **Flujos alternos** | |
| Alternativa 1. **“Cancelar operación”** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. Selecciona la opción de Cancelar operación. | |

| | |
|---|---|
| | 2. Regresa a la vista anterior. |
| | 3. El caso de uso termina. |
| Alternativa 2. **"Existen datos incorrectos"** | |
| **Acción del actor** | **Respuesta del sistema** |
| | 1. Muestra un indicador sobre los campos incorrectos. |
| | 2. Regresa al paso 4 del **Flujo Normal de Eventos.** |
| Alternativa 3. **"Campos vacíos"** | |
| **Acción del actor** | **Respuesta del sistema** |
| | 1. Muestra un indicador sobre los campos vacíos. |
| | 2. Regresa al paso anterior del **Flujo Normal de Eventos.** |
| | |
| Alternativa 4. **"Agrupaciones duplicadas1."** | |
| **Acción del actor** | **Respuesta del sistema** |
| | 1. Valida que los valores ubicados en las filas no hayan sido agrupados anteriormente. |
| | 2. Regresa al paso 17 del **Flujo Normal de Eventos.** |
| Alternativa 5. **"Agrupaciones duplicadas2."** | |
| **Acción del actor** | **Respuesta del sistema** |
| | 1. Valida que los valores ubicados en las columnas no hayan sido agrupados anteriormente. |
| | 2. Regresa al paso 23 del **Flujo Normal de Eventos.** |
| **Postcondiciones** | Se crea un nuevo diseño de reporte. |

Tabla 2.3 Descripción textual del caso de uso: *Crear diseño de reporte*.

| **Caso de uso** | |
|---|---|
| **CU-6** | **Generar reporte** |
| **Propósito** | Generar el reporte a nivel de programación. |
| **Actores** | Operador Informático. |
| **Resumen:** | El caso de uso inicia luego de que el actor selecciona la opción Generar reporte. El sistema muestra el diseño del reporte. |
| **Precondiciones** | Tiene que existir el diseño de reporte que se quiere generar. |
| **Referencias** | RF7 Exportar reporte. |
| **Flujo Normal de los Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El caso de uso inicia cuando el actor accede a la opción Generar reporte. | |
| | 2. Permite:<br>-Generar un reporte.<br>-Exportar reporte.<br>Ver **Alternativa 1**:"Exportar reporte"<br>- Cancelar operación. Ver **Alternativa 2:** "Cancelar operación." |
| | 3. Genera el reporte. |
| | 4. Termina el caso de uso. |
| **Flujos alternos** | |
| Alternativa 1. **"Exportar reporte"** | |
| **Acción del actor** | **Respuesta del sistema** |

| | |
|---|---|
| 1. Selecciona la opción de Exportar. | |
| | 2. Permite exportar el reporte.<br><br>Ver caso de uso: **Exportar reporte.** |
| | 3. Termina el caso de uso. |
| Alternativa 2. **"Cancelar operación"** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. Selecciona la opción de Cancelar operación. | |
| | 2. Regresa a la vista anterior. |
| | 3. El caso de uso termina. |
| **Postcondiciones** | Se genera el reporte. |

Tabla 2.4 Descripción textual del caso de uso: *Generar reporte*.

| **Caso de uso** | |
|---|---|
| **CU-8** | **Asignar privilegios de reportes** |
| **Propósito** | Habilitar conceptos a usuarios o roles. |
| **Actores** | Súper Usuario |
| **Resumen:** | El caso de uso inicia luego de que el actor selecciona la opción Seguridad. El operador informático con súper privilegios es el encargado de adicionar o eliminar conceptos a un usuario o rol deseado. |
| **Precondiciones** | No existen. |
| **Referencias** | No existen. |
| **Flujo Normal de los Eventos** | |

| **Acción del actor** | **Respuesta del sistema** |
|---|---|
| 1. El caso de uso inicia cuando el actor accede a la opción Seguridad. | |
| | 2. Muestra los campos para la asignación de privilegios del reporte:<br>• Conceptos<br>• Usuarios<br>• Roles<br>Permite:<br>-Habilitar conceptos a un usuario o rol escogido. Ver **Alternativa 1**:"Habilitar conceptos".<br>-Deshabilitar conceptos a un usuario o rol deseado. Ver **Alternativa 2**:"Deshabilitar conceptos".<br>-Cancelar la operación. Ver **Alternativa 3**:"Cancelar operación". |
| | 3. Actualiza los cambios. |
| | 4. Termina el caso de uso. |
| **Flujos alternos** | |
| Alternativa 1. **"Habilitar conceptos"** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. Escoge un concepto y un usuario o rol. | |
| | 2. Muestra todos los usuarios y roles que están habilitados para dicho concepto médico. |
| 3. Escoge al usuario o rol que desea habilitar.<br>Accede a la opción Permitir. | |

| | |
|---|---|
| | 4. Verifica que el usuario o el rol al cuál se le quiere dar acceso no está permitido y le da permiso.<br>De lo contrario:<br>Muestra el mensaje de información "El usuario o rol al que se quiere permitir ya tiene acceso." |
| | Regresa al paso 2 del **Flujo Normal de Eventos**. |
| Alternativa 2. **"Deshabilitar conceptos"** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. Selecciona un concepto. | |
| | 2. Muestra todos los usuarios y roles que están habilitados para dicho concepto médico. |
| 3. Selecciona el usuario o rol que desea denegar. | |
| 4. Escoge la opción Eliminar. | |
| | Regresa al paso 2 del **Flujo Normal de Eventos**. |
| Alternativa 3. **"Cancelar operación"** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. Selecciona la opción de Cancelar operación. | |
| | 2. Regresa a la vista anterior. |
| | 3. El caso de uso termina. |
| **Postcondiciones** | Se adicionaron o eliminaron conceptos a un usuario o rol deseado. |

Tabla 2.5 Descripción textual del caso de uso: *Asignar privilegios de reporte*.

## 2.5 Propuesta de solución

Al analizar el proceso de generación de reportes en el HIS del CESIM y con el fin de eliminar las restricciones presentes en la versión existente, se hace necesario desarrollar un componente que permita generar los reportes de forma dinámica.

Por ello el componte propuesto contiene funcionalidades que permiten al administrador del sistema, gestionar los permisos sobre los conceptos, asignándole a cada usuario o rol solamente aquellos a los cuáles tiene acceso.

Luego de que el usuario posea permisos a uno o más conceptos, este podrá crear un reporte siguiendo el flujo que se describe a continuación. El usuario selecciona el o los conceptos que necesita para generar el reporte, de seleccionar más de uno, debe especificar de qué forma los desea vincular; a partir de la selección anterior define los campos o datos que desea aparezcan en el reporte. Se le brinda la posibilidad de generar reportes simples o complejos, para estos últimos debe especificar la ubicación de los datos en filas o columnas, la jerarquía de estos, y las operaciones de agregación a efectuar. El siguiente paso sería la visualización del reporte y brindar la opción de exportarlo a los formatos Pdf, Word o Excel.

El usuario luego de crear sus reportes puede volverlos a visualizar o eliminarlos en el momento en que desee.

## Conclusiones

En el desarrollo del presente capítulo se especificaron los requerimientos del sistema, tanto funcionales como no funcionales, los actores y los casos de uso con los que interactúan. Lo anterior posibilita la obtención de los artefactos correspondientes a dichos casos de uso, lo que permite iniciar el desarrollo de las actividades correspondientes al Análisis y Diseño del Sistema.

# Capítulo 3. Análisis y Diseño del Componente de Generación de reportes dinámicos basado en conceptos

En el desarrollo del capítulo se define la arquitectura utilizada en la construcción del sistema. Se presenta el diseño propuesto para la solución de la aplicación. Se analizan posibles implementaciones, componentes o módulos ya existentes que puedan ser rehusados, así como las estrategias de integración. Se describen los diagramas de diseño y de paquete donde se muestra la jerarquía lógica del sistema. La elaboración del diseño del sistema y su análisis ayudará a una mejor compresión para la implementación del sistema.

## 3.1 Descripción de la arquitectura

Cuando se habla de arquitectura de software, se hace alusión a la especificación de la estructura del sistema, entendida como la organización de componentes y relaciones entre ellos; los requerimientos que debe satisfacer el mismo y las restricciones a las que está sujeto. La arquitectura está compuesta por propiedades no funcionales del sistema y su impacto sobre la calidad del mismo; las reglas y decisiones de diseño que gobiernan esta estructura y los argumentos que justifican las decisiones tomadas. Es la columna vertebral del software, puede considerarse como el puente entre los requerimientos del sistema y la implementación, es por ello la importancia que se le atribuye para lograr que el producto cumpla con la calidad requerida.

El sistema que se desarrolla presenta una arquitectura basada en el patrón arquitectónico Modelo Vista Controlador (MVC), que permite separar los datos de la aplicación, la interfaz de usuario y la lógica de control, en tres componentes distintos. Este diseño independiente permite que se realicen cambios en una sola capa sin que afecte a todo el sistema. Es una arquitectura que divide los componentes del sistema en 3 capas principales: presentación, negocio y acceso a datos.

En el capítulo 1 se hizo mención de este patrón arquitectónico pero de forma breve, donde se especificó por cada capa cuáles eran las herramientas, lenguajes, conjunto de tecnologías, plataformas y framework que se utilizaban para lograr el objetivo de la investigación. Ahora se profundiza en cómo opera el patrón MVC, por la importancia que se le atribuye. Para un mejor entendimiento se muestra a continuación un gráfico donde se explica cómo funciona el mismo.

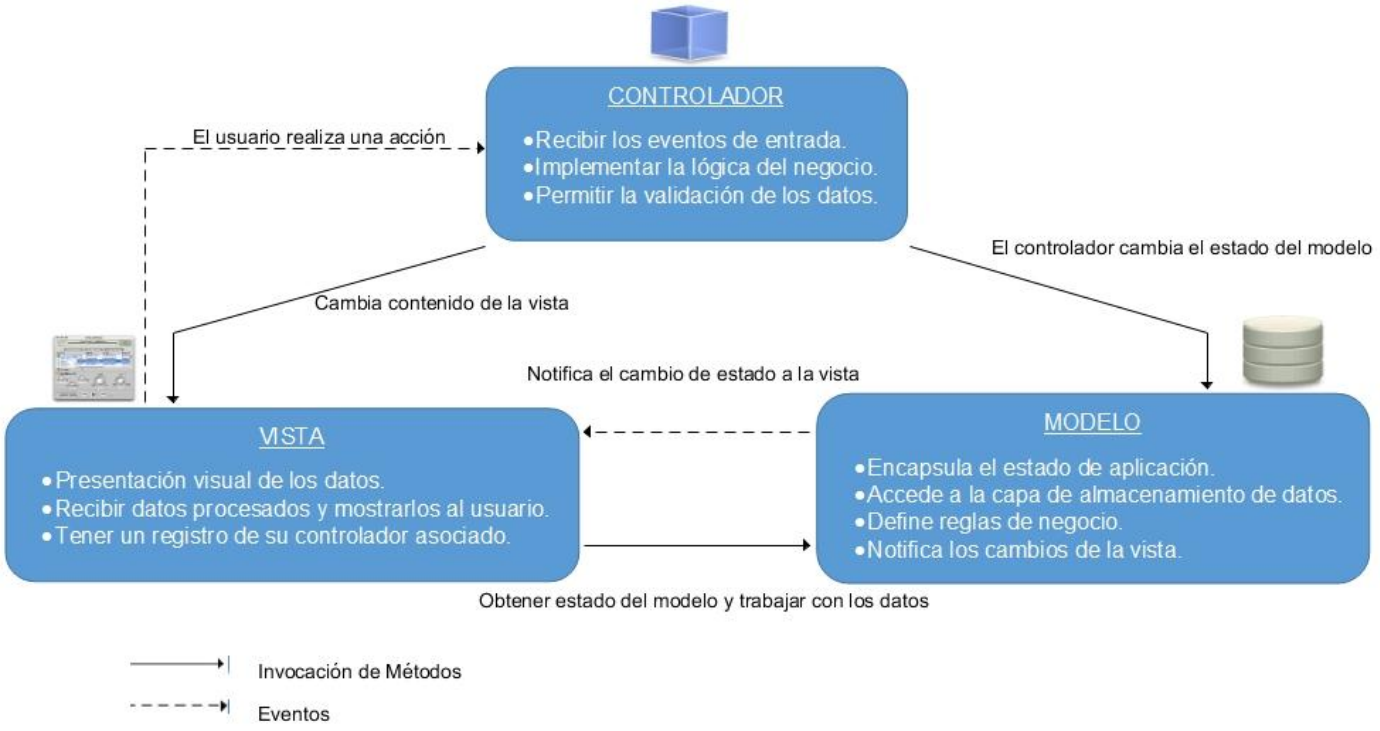



Figura 3.1 Patrón Arquitectónico Modelo Vista Controlador.

En el modelo o capa de datos se utiliza Hibernate como herramienta de mapeo objeto-relacional para la plataforma java, siendo la implementación para EJB en su versión 3 y la API para la persistencia de java o JPA. Utilizando las herramientas anteriormente mencionadas se generan las clases entidades las cuales contiene información que utiliza JPA para realizar las diferentes acciones sobre las bases de datos

La vista o capa de presentación está formada por las páginas XHTML, desarrolladas utilizando JavaServer Faces (framework basado en el patrón MVC), RichFaces y Facelets como motor de plantillas. Todas las tecnologías anteriormente mencionadas permiten el desarrollo de interfaces amigables y en corto período de tiempo.

En el controlador o capa de negocio se encuentra Seam como framework de integración entre los componentes visuales y los de acceso a datos, DynamicReports como biblioteca de clases para la generación de los reportes dinámicos y las clases controladoras autogeneradas y las encargadas de la lógica de generación de reportes.

## 3.2 Estrategias de integración

La reutilización de código es uno de los aspectos que puede ser más beneficioso en un proyecto de desarrollo de software, permite reducir tiempo y minimizar la redundancia del código. La forma más eficiente de la reutilización de código es la creación de componentes reutilizables para evitar la duplicidad del mismo. Para la implementación del componente se reutiliza el conjunto de clases desarrolladas para la generación de reportes en el sistema, específicamente las funcionalidades para exportar los reportes a Pdf, Word, Excel.

## 3.3 Modelo de diseño

Luego de establecer los requisitos del software, el diseño es la primera de dos actividades técnicas: diseño y codificación. Cada actividad transforma la información de forma que al final se obtiene un software validado. El diseño es técnicamente la parte central de la ingeniería del software, sin tenerlo en cuenta, se corre el riesgo que la construcción del sistema sea inestable, que falle cuando se realicen pequeños cambios, que sea difícil de probar, un sistema cuya calidad no pueda ser evaluada hasta más adelante, cuando quede poco tiempo y ya se haya gastado muchos recursos.

En el diseño se utilizaron los Patrones de Software para la Asignación General de Responsabilidad (GRASP, por sus siglas en inglés) (40). Los patrones de diseño son soluciones a problemas comunes en el diseño de aplicaciones, es de buenas prácticas de programación implementarlos, pues ahorran tiempo y mejoran el software haciéndolo más eficiente, dinámico y seguro. No es más que la solución efectiva a un problema en un momento dado y puede ser reusable aplicándose a diferentes problemas de diseño en distintas circunstancias.

En el desarrollo del componente a cada clase le fueron asignadas las tareas que podían realizar según la información que poseía principio básico que suele utilizarse en el diseño orientado a objetos, poniéndose de manifiesto el patrón Experto. Para crear las instancias de otras clases en correspondencia con la responsabilidad dada, se usó el patrón Creador. Con esto se logra conservar el encapsulamiento pues los objetos logran valerse de su propia información para realizar lo que se les pide. Se empleó Bajo Acoplamiento con el fin de tener las clases lo menos ligadas posibles, de tal forma que, en caso de producirse una modificación en alguna de ellas, tenga la mínima repercusión en el resto de las clases. Otro de los utilizados es el de Alta cohesión, que garantiza que la información contenida en las clases sea coherente. Por último se dispuso de la creación de las clases controladoras que facilitan realizar las

operaciones del sistema, debido a que estas operaciones reflejan los procesos del componente y no es factible manejarse en la capa de interfaz o presentación.

El Modelo de Diseño está compuesto por artefactos que engloban todas las clases del diseño, subsistemas, paquetes, colaboraciones, y las relaciones entre ellos. Para la elaboración del mismo se utilizó el mecanismo paquete, con el objetivo de describir los grupos de elementos o subsistemas. Un paquete es un conjunto de cualquier tipo de elementos de un modelo: clases, casos de uso y diagramas de colaboración.

Un paquete se muestra gráficamente como una carpeta con etiquetas. Los paquetes subordinados se incluyen en su interior. El nombre del paquete se encuentra dentro de la etiqueta, si el paquete describe sus elementos, en caso contrario, estará en el centro de la misma carpeta. A continuación se muestra cómo fue dividido el sistema a través del diagrama de paquetes y las dependencias que existen entre ellos:

**Diagrama de Paquetes**

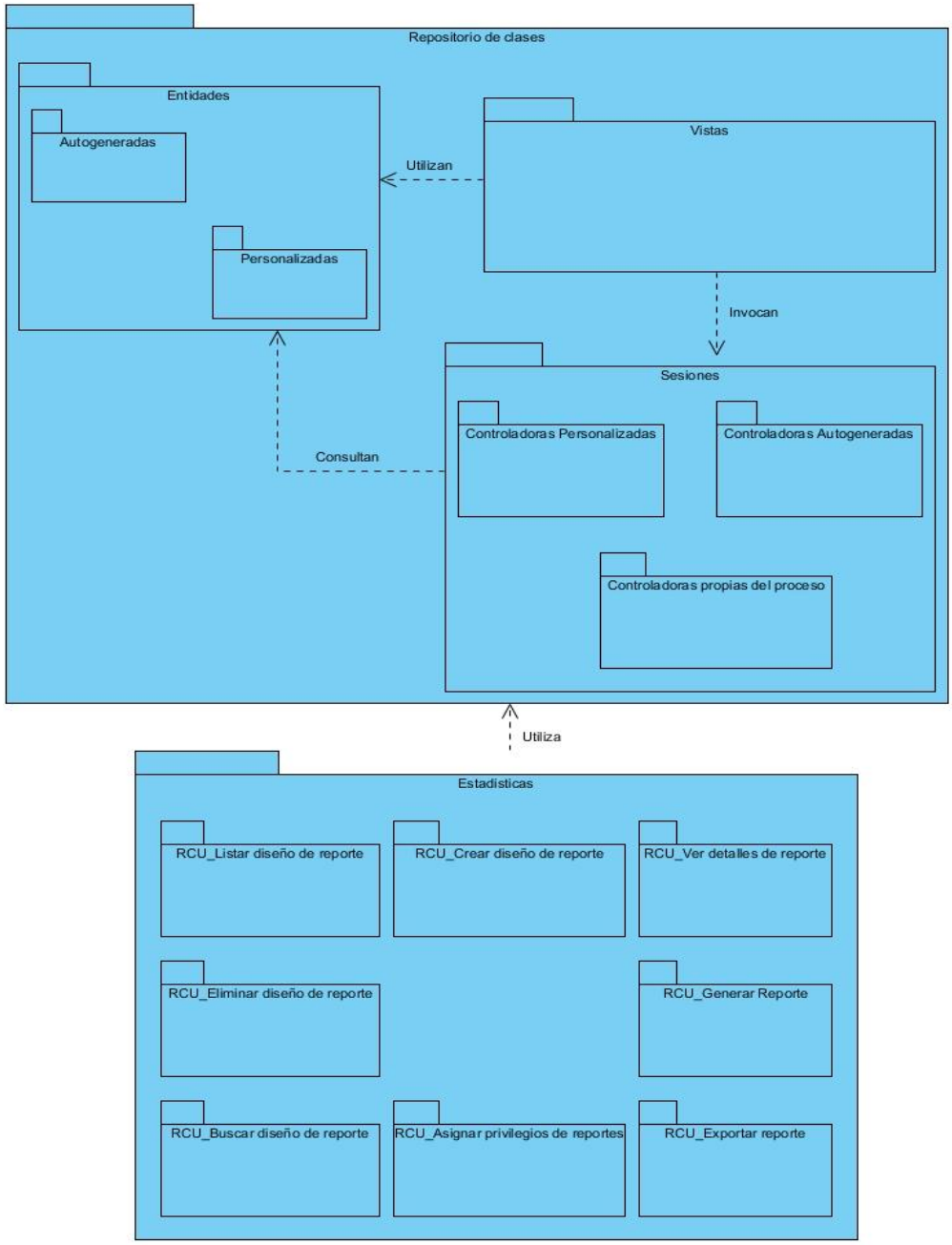



Figura 3.2 Diagrama de paquetes

### 3.3.1 Diagramas de Clases del Diseño

Los diagramas de clases de diseños se construyen mediante estereotipos web. La representación de las mismas está asociada al uso de UML para el modelado de aplicaciones web, por lo que son identificadas las siguientes clases: página servidora (server page), página cliente (client page), formulario (form).

Una página servidora (server page) construye una página cliente (client page), la cual contiene un formulario (form) que puede actualizar directamente a las entidades o enviar las peticiones a la página servidora, estas últimas invocan métodos y responsabilidades de las clases controladoras, las cuales pueden consultar o modificar las entidades.

La nomenclatura que se usó fue la siguiente: SP_<Nombre de la página (server_page)>, CP_<Nombre de la clase (client_page)>, FR_<Nombre de la clase (form)>, CC_<Nombre de la clase controladora>,

CE_<Nombre de la clase entidad>.

**DCD_Crear diseño de reporte**

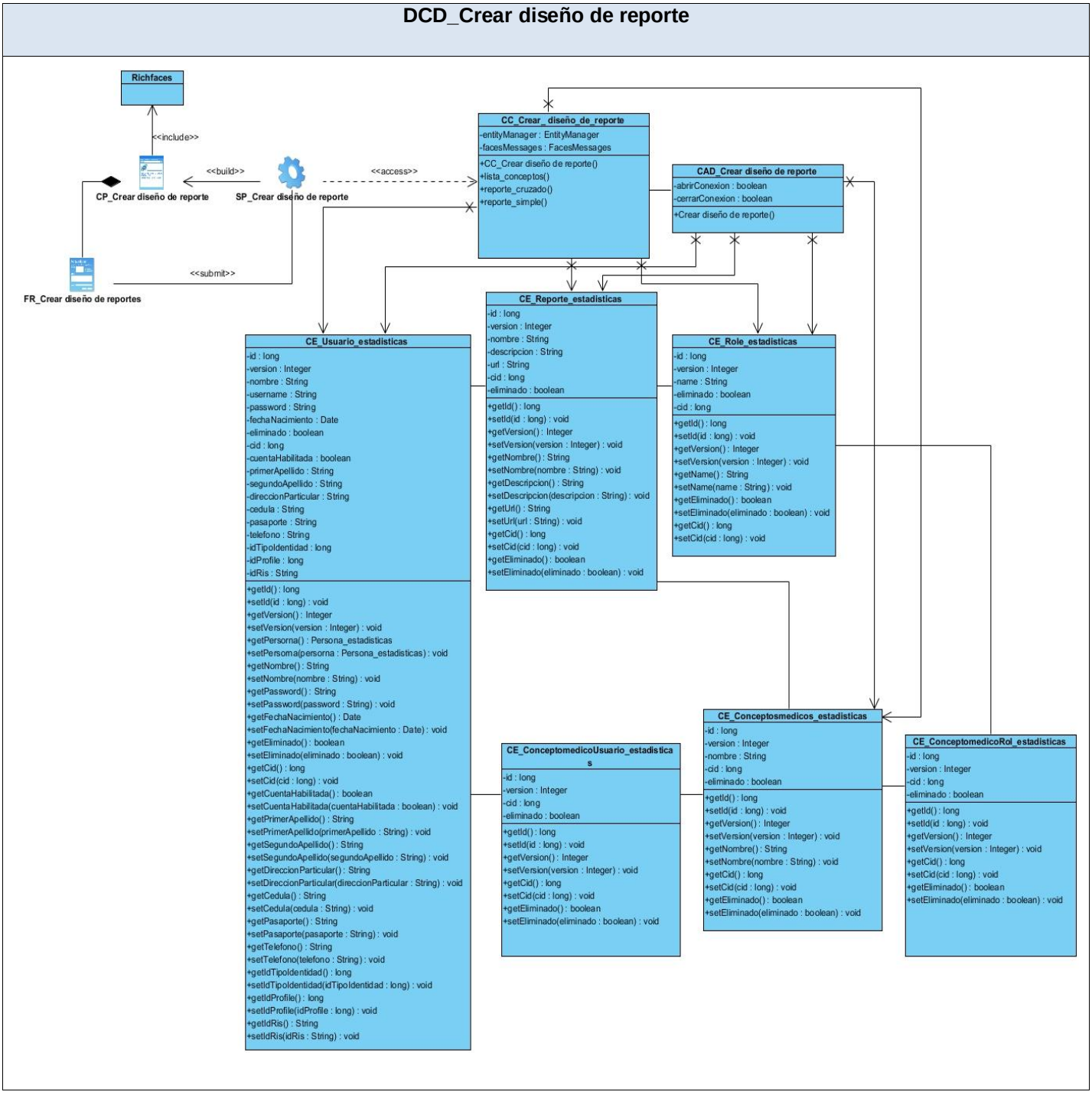



Figura 3.3 Diagrama de Clases del Diseño: *Crear diseño de reporte*

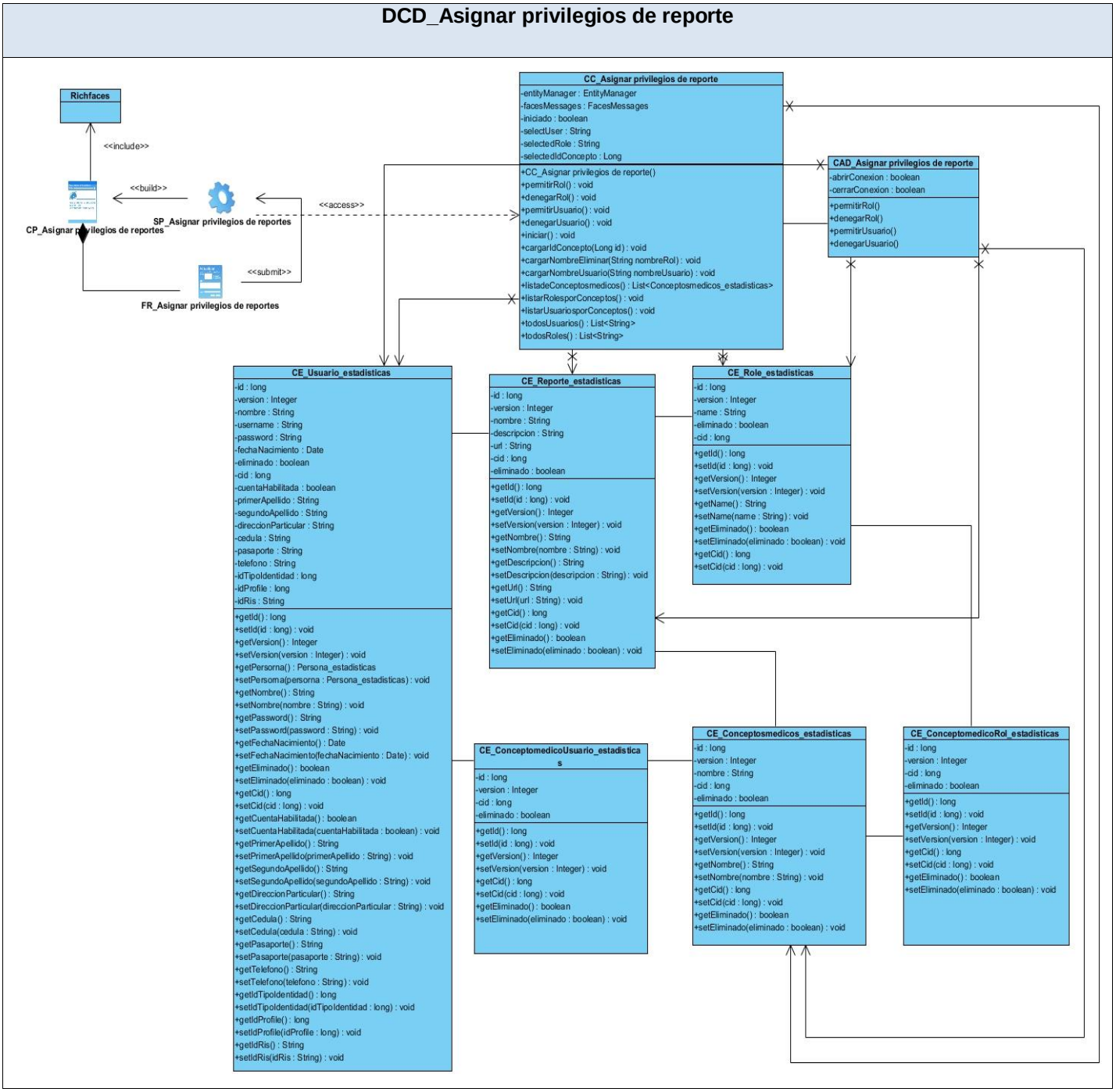



Figura 3.4 Diagrama de Clases del Diseño: Crear diseño de reporte *Asignar privilegios de reporte*

**DCD_Generar reporte**

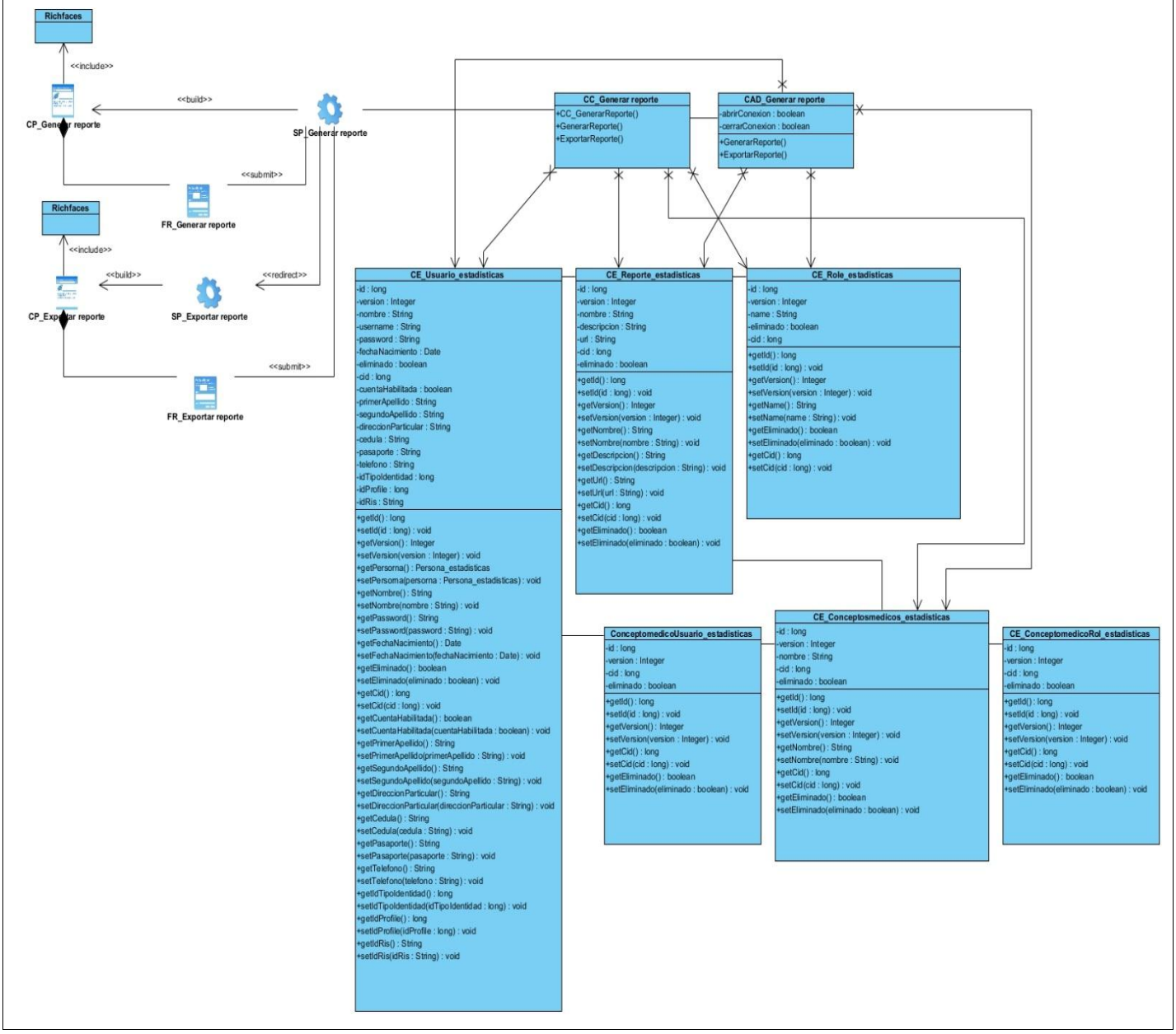



Figura 3.5 Diagrama de Clases del Diseño*: Generar reporte*

**DCD_Listar diseño de reporte**

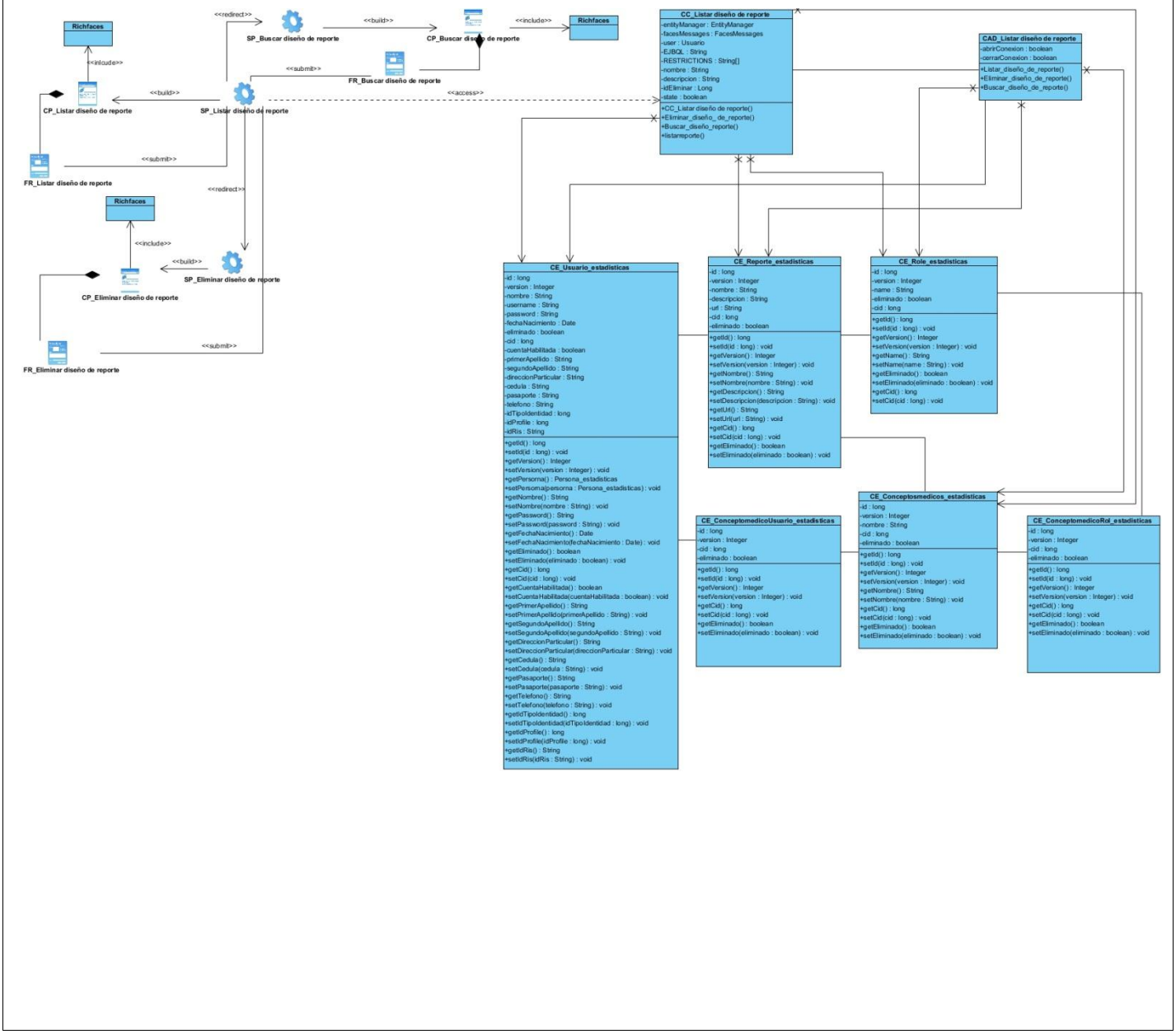



Figura 3.6 Diagrama de Clases del Diseño*: Listar diseño de reporte*

### 3.3.2 Descripción de las clases y sus atributos

A continuación se realiza la descripción de las clases que han sido identificadas en el diseño para su futura implementación, con el objetivo de lograr una mejor comprensión del sistema.

| **Nombre: Asignar privilegios de reporte** | |
|---|---|
| **Tipo de clase: Controladora.** | |
| **Atributo** | **Tipo** |
| entityManager | EntityManager |
| facesMessages | FacesMessages |
| iniciado | boolean |
| selectUser | String |
| selectRole | String |
| selectIdConcepto | Long |
| **Para cada responsabilidad**: | |
| Nombre: | permitirRol(): void |
| Descripción: | Permite dar acceso a un rol, de un concepto seleccionado. |
| Nombre: | denegarRol():void |
| Descripción: | Deniega el acceso a un rol, de un concepto seleccionado. |
| Nombre: | permitirUsuario():void |
| Descripción: | Permite dar acceso a un usuario, de un concepto seleccionado. |
| Nombre: | denegarUsuario():void |
| Descripción: | Deniega el acceso a un usuario, de un concepto seleccionado. |
| Nombre: | inicia():void |
| Descripción: | Se inicializan las listas Usuarios y Roles. |
| Nombre: | cargarIdConcepto(id:Long):void |
| Descripción: | Carga el identificador del concepto seleccionado. |
| Nombre: | cargarNombreEliminar(nombreRol :String):void |
| Descripción: | Carga el nombre del rol que se desea eliminar. |
| Nombre: | cargarNombreUsuario(nombreUsuario :String):void |

| | |
|---|---|
| Descripción: | Carga el nombre del usuario que se desea eliminar. |
| Nombre: | listadeConceptosmedicos():List<Conceptosmedicos_estadisticas> |
| Descripción: | Lista todos los conceptos médicos. |
| Nombre: | listarRolesporConceptos ():void |
| Descripción: | Lista todos los roles que tienen acceso al concepto seleccionado. |
| Nombre: | listarUsuariosporConceptos ():void |
| Descripción: | Lista todos los usuarios que tienen acceso al concepto seleccionado. |
| Nombre: | todosUsuarios ():void |
| Descripción: | Lista todos los usuarios del sistema. |
| Nombre: | todosRoles ():void |
| Descripción: | Lista todos los roles del sistema. |

Tabla 3.1 Descripción de la clase controladora: *Asignar privilegios de reporte*.

| **Nombre: Crear diseño de reporte** | |
|---|---|
| **Tipo de clase: Controladora.** | |
| **Atributo** | **Tipo** |
| entityManager | EntityManager |
| facesMessages | FacesMessages |
| user | Usuario |
| **Para cada responsabilidad**: | |
| Nombre: | listar_conceptos(): Conceptos_estadísticas |
| Descripción: | Devuelve una la lista de conceptos con que va ser creado el reporte. |
| Nombre: | reporte_cruzado:void |
| Descripción: | Devuelve el reporte de forma cruzada con los conceptos y los cálculos realizados por las funciones de agregación seleccionadas para su confección. |
| Nombre: | reporte_simple: void |
| Descripción: | Devuelve el reporte de forma simple con los conceptos y los cálculos realizados por las funciones de agregación seleccionadas para su confección. |
| Nombre: | getUser(): Usuario |

| | |
|---|---|
| Descripción: | Permite conocer cuál es el usuario que está operando con el módulo para guardar dicho reporte. |

Tabla 3.2 Descripción de la clase controladora: *Crear diseño de reporte.*

## Conclusiones

En el presente capítulo, se profundizó en el estudio de la arquitectura para el desarrollo del componente de generación de reportes dinámicos. Se identificaron las clases fundamentales: interfaz, control y entidad, que se deben definir para el correcto funcionamiento del sistema. Se realizó el modelado, en términos del diseño, de los diagramas de clases y de paquete y se explica la nomenclatura utilizada, como parte de la solución propuesta. Además, se realizó un análisis de las posibles implementaciones, módulos o componentes que serán reutilizados en el desarrollo del sistema.

# Capítulo 4. Implementación del Componente de Generación de reportes dinámicos basado en conceptos

En el presente capítulo se muestran las principales características de la implementación del sistema, además de los componentes y sus relaciones, mostrados a través del Diagrama de Componentes y el Diagrama de Despliegue. Se especifica la estrategia de codificación a seguir, así como elementos de seguridad y captura de errores.

## 4.1 Modelo de Datos

El Dr. Dittrich (41) en 1994 definió el Modelo de Datos como el "Conjunto de herramientas conceptuales para describir la representación de la información en términos de datos. Los modelos de datos comprenden aspectos relacionados con estructuras y tipos de datos, operaciones y restricciones".

El Modelo de Datos describe la representación lógica y física de los datos persistentes, este puede ser generado a partir del modelo de objetos y viceversa, lo que brinda la posibilidad de ahorrar esfuerzo y tiempo de trabajo. El modelado de datos y el desarrollo de un prototipo arquitectónico que permita que el rendimiento de la base de datos se evalúe, son esenciales para alcanzar una arquitectura estable. Como los casos de uso arquitectónicamente significativos se detallan y analizan en cada iteración, los elementos del Modelo de Datos se definen basándose en el desarrollo de los diseños de clases persistentes a partir de los casos de uso. Una ventaja que ofrece este modelo es que posibilita un conjunto de operaciones básicas para realizar consultas y actualizar datos.

El Modelo de Datos está compuesto por entidades, atributos y sus relaciones. Las entidades son objetos de los que el sistema necesita guardar información. Los atributos son las características asociadas a una entidad, los mismos pueden ser clasificados en obligatorios, opcionales, claves foráneas y claves primarias (las cuales se dividen en simples y compuestas).

Las relaciones, muestran la forma en que dos entidades se asocian. Se representan mediante una línea que une a las dos entidades implicadas y manifiestan dos características principales: la cardinalidad y la obligatoriedad.

A continuación se presenta el Diagrama de Modelo de Datos:

**Modelo de Datos**

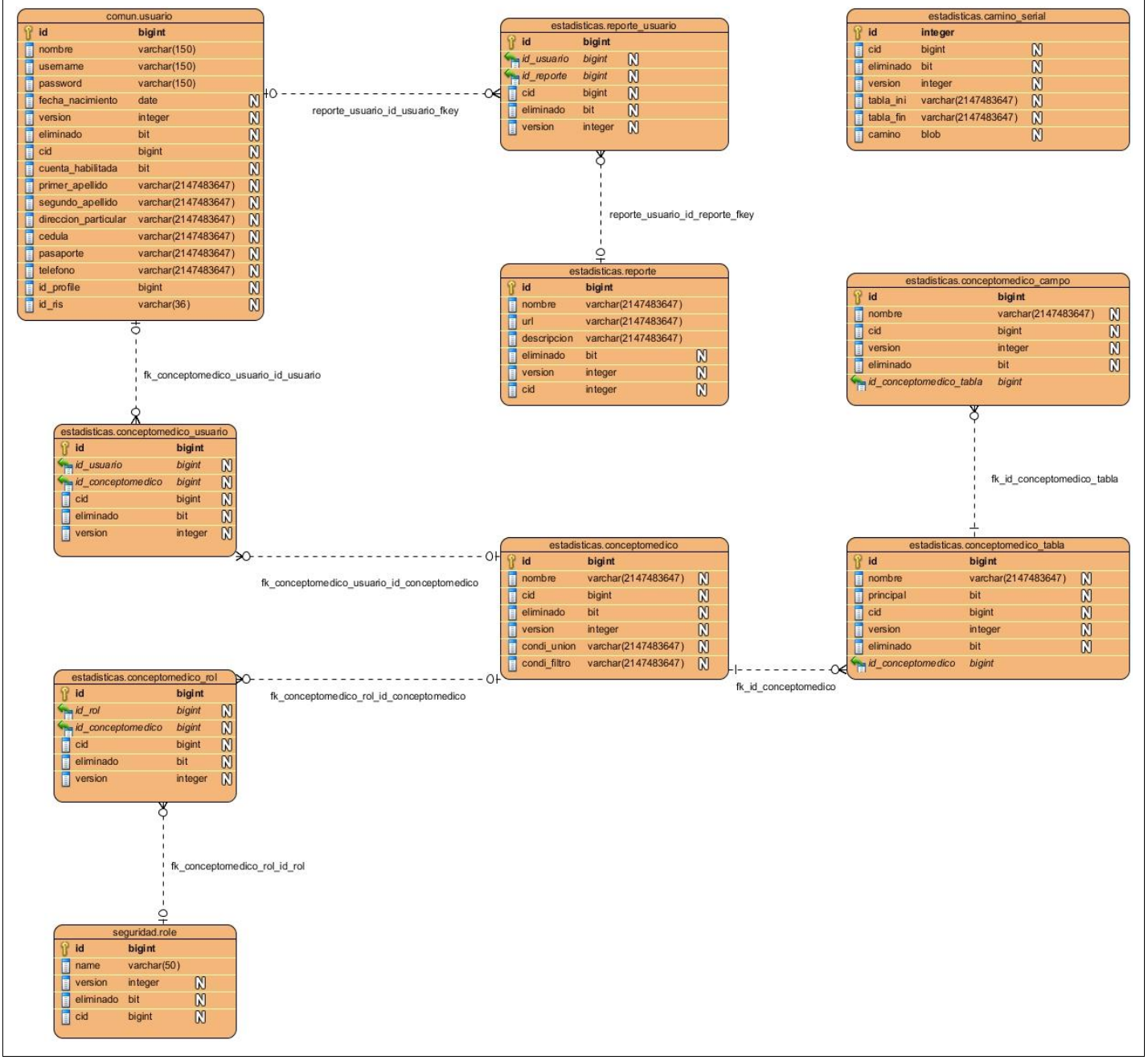



Figura 4.1 Modelo de Datos

### 4.1.1 Descripción de las tablas de la base de datos

A continuación se describen las entidades más representativas del modelo anterior.

Los siguientes atributos son comunes a todas las entidades ya que fueron agregados con el objetivo de facilitar la implementación de algunas funcionalidades del componente.

| Atributo | Tipo | Descripción |
|---|---|---|
| Id | long | Identificador necesario en cada entidad para las referencias en las relaciones entre tablas. |
| version | integer | Indica con qué versión de la entidad se está trabajando. Es usado para garantizar que se está trabajando con la versión de la entidad más actualizada que existe en la base de datos. |
| eliminado | boolean | Indica que la información fue borrada si el valor está en verdadero (true). En caso contrario, se mantiene en falso (false). |
| cid | long | Permite identificar quién realiza alguna acción sobre la entidad. |

Tabla 4.1 Descripción de los atributos comunes

| reporte | | |
|---|---|---|
| La tabla contiene los datos referentes al reporte que está registrado en la base de datos. | | |
| **Atributo** | **Tipo** | **Descripción** |
| nombre | varchar | Nombre del reporte que está registrado en la base de datos. Ejemplo: Reporte_Estadíticas#1 |
| descripcion | varchar | Campo descriptivo de un reporte realizado teniendo en cuenta los conceptos del mismo. Ejemplo: Costo de |

| | | |
|---|---|---|
| | | Hospitalización, Fecundidad. |
| url | varchar | Dirección del diseño del reporte que se encuentra registrado en la base de datos. Ejemplo: C:\Users\Administrador\reportes\Reporte_Estadíticas#1 |

Tabla 4.2 Descripción de la tabla: *reporte*

| **conceptomedico** | | |
|---|---|---|
| La tabla contiene los datos referentes al concepto médico que está registrado en la base de datos. | | |
| **Atributo** | **Tipo** | **Descripción** |
| nombre | varchar | Nombre del concepto médico que está registrado en la base de datos. Ejemplo: Reporte_Estadíticas#1 |
| condi_union | varchar | Almacena los atributos que unen a las tablas que componen el concepto médico. |
| condi_filtro | varchar | Condiciones de filtrado para los atributos a visualizar en el reporte. |

Tabla 4.3 Descripción de la tabla: *conceptomedico*

| **conceptomedico_tabla** | | |
|---|---|---|
| Representa el conjunto de tablas que pertenecen a un concepto. | | |
| **Atributo** | **Tipo** | **Descripción** |
| nombre | varchar | Nombre de la tabla. |
| principal | bit | Identifica a la tabla central del concepto, siendo esta la intermediaria para la unión con otro concepto. |

Tabla 4.4 Descripción de la tabla: *conceptomedico_tabla*

| **conceptomedico_campo** | | |
|---|---|---|
| Representa el conjunto de atributos que pertenecen a un concepto. | | |
| **Atributo** | **Tipo** | **Descripción** |
| nombre | varchar | Nombre del atributo. |

Tabla 4.5 Descripción de la tabla: *conceptomedico_campo*

| **usuario** | | |
|---|---|---|
| La tabla contiene los datos referentes al usuario que está registrado en el sistema. | | |
| **Atributo** | **Tipo** | **Descripción** |
| nombre | varchar | Nombre del usuario que está registrado en el sistema. |
| username | varchar | Usuario que está operando en el sistema. |
| password | varchar | Contraseña del usuario que está operando en el sistema. |

Tabla 4.6 Descripción de la tabla: *usuario*

| **camino_serial** | | |
|---|---|---|
| Almacena todos los caminos desde una tabla inicial a una tabla final. | | |
| **Atributo** | **Tipo** | **Descripción** |
| tabla_ini | varchar | Nombre de la tabla inicial. |
| tabla_fin | varchar | Nombre de la tabla final. |

| | | |
|---|---|---|
| camino | byte[ ] | Almacena el objeto camino serializado. |

Tabla 4.7 Descripción de la tabla: *camino_serial*

## 4.2 Modelo de Implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del Modelo de Diseño a subsistemas y componentes físicos. Este artefacto describe cómo se implementan los componentes, congregándolos en subsistemas organizados en capas, por jerarquías y señala las dependencias entre los mismos. Para representar los diagramas del Modelo de Implementación se puede emplear el diagrama de Componentes.

### 4.2.1 Diagrama de Componentes

Un componente es una parte física de un sistema (módulo, base de datos o programa ejecutable). Se puede decir que un componente es la materialización de una o más clases, puesto que una abstracción con atributos y métodos pueden ser implementados en los componentes. Se pueden agrupar en paquetes así como los objetos en clases, además puede existir entre ellos relaciones de dependencia como generalización, asociación y agregación.

El Diagrama de Componentes provee una vista arquitectónica de alto nivel del sistema, ayuda a los desarrolladores a visualizar el camino de la implementación y permite tomar decisiones respecto a las tareas de implementación. A continuación se presenta el Diagrama de Componentes:

**Diagrama de Componentes**

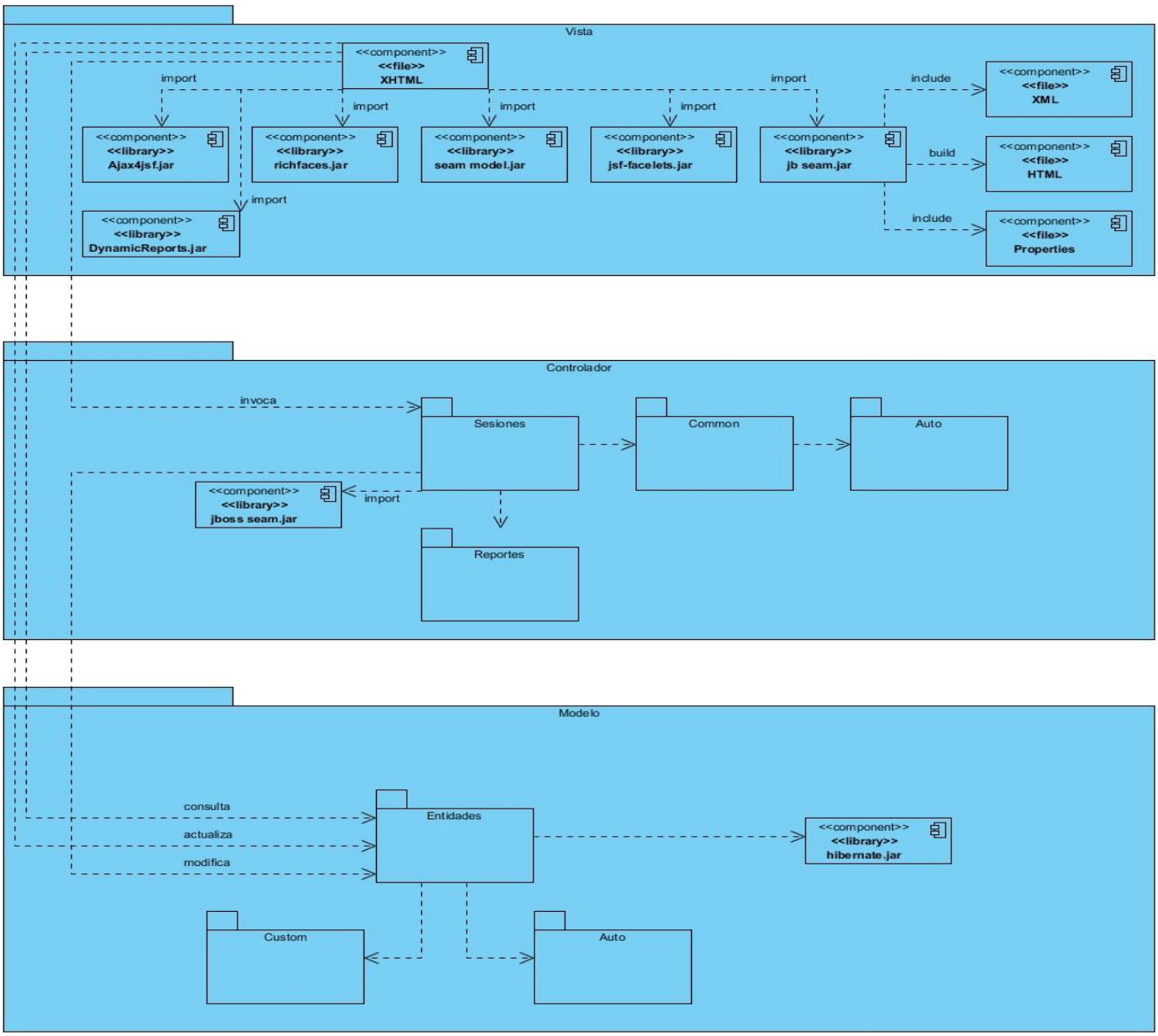



Figura 4.2 Diagrama de Componentes

El diagrama anterior muestra cómo se encuentra estructurado el sistema en componentes. La Vista está formada por las páginas XHTML que importan diversas librerías para su construcción, entre ellas se

encuentra DynamicReports.jar que permite, mediante un conjunto de clases, la generación dinámica de reportes. En el paquete Sesiones se encuentra el jboss_seam.jar y las clases controladoras que son las encargadas de llevar a cabo los requisitos funcionales del sistema. En el Modelo están presentes las entidades autogeneradas y las personalizadas, contenidas todas en el paquete Entidades, el cual utiliza la librería hibernate.jar. Estos paquetes se relacionan entre sí, las vistas consultan y actualizan las entidades e invocan a las controladoras y estas a su vez modifican las entidades.

### 4.2.2 Diagrama de Despliegue

Los Diagramas de Despliegue son los complementos de los Diagramas de Componentes que, unidos, proveen la vista de implementación del sistema. Describen la topología del mismo y la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos.

Teniendo en cuenta las características del sistema, el Diagrama de Despliegue quedó estructurado de la siguiente manera:

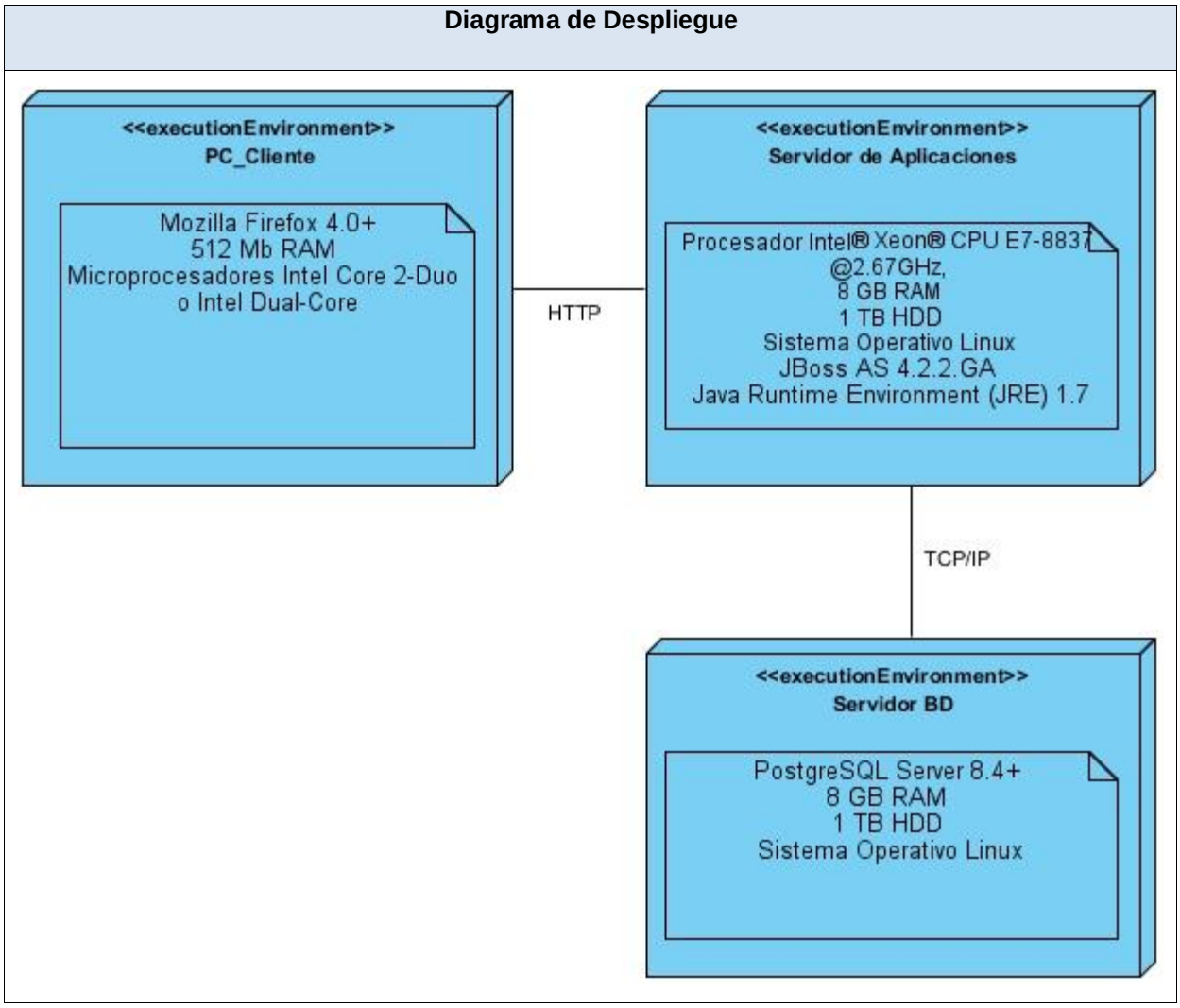



Figura 4.3 Diagrama de Despliegue

Para hacer uso del sistema, el usuario debe conectarse al mismo mediante una computadora cliente, haciendo uso del navegador web Mozilla Firefox 4.0 o cualquier versión superior. Las peticiones por el protocolo HTTP serán procesadas por el servidor de aplicaciones y emitirá peticiones por el protocolo TCP / IP (Transfer Control Protocol / Internet Protocol) hacia el servidor de base de datos. El servidor de aplicaciones enviará una respuesta a la PC cliente, la cual se encontrará conectada a una impresora haciendo uso de los puertos USB (serial) o LPT (paralelo), de esta forma, en caso de ser necesario, el usuario podrá imprimir reportes.

## 4.3 Tratamiento de errores

La fiabilidad del software se define como la medida del éxito con que el sistema cumple las especificaciones fijadas para su comportamiento, cuando el mismo se desvía del especificado, se dice que se produce un error, por lo que es necesaria la recuperación frente a errores del sistema. Para lograr tratar los errores producidos se valida la información, lo que garantiza la corrección y precisión de todos los valores introducidos en la aplicación, además de lograr elevar la calidad del producto informático.

A toda porción de código, donde pueda surgir un evento que durante la ejecución del programa interrumpa el flujo normal de las sentencias, se le realiza el control de las excepciones, principalmente donde se ejecutan códigos que manipulan datos que viajan desde y hacia la base de datos.

En caso de ocurrir una excepción que implique una redirección, se maneja mediante los archivos “Nombre_de_la_clase.pages.xml”, los mismos se encargan de capturar globalmente las excepciones y ejecutar las instrucciones determinadas. Para la depuración y manejo de errores se utiliza el componente FacesMessages del framework Seam, que se encarga de mostrar los mensajes que se manejan a través del objeto facesMessages inyectado en las clases controladoras.

## 4.4 Seguridad

Los diseños de reportes deben respetar la confidencialidad de la información, no se debe construir un diseño que contenga datos de un concepto al cual no tiene acceso. El usuario que accede al sistema solo puede visualizar la lista de reportes que han sido construidos por su entidad. En aras de garantizar una correcta protección de la información que se maneja, se propone un control de acceso a nivel de usuario mediante la clase Usuario donde se conoce el operador informático que trabaja en el sistema, con el objetivo de obtener el principio de mínimo privilegio, lo que garantiza el acceso de cada usuario únicamente a lo que tiene permiso.

## 4.5 Estrategias de codificación. Estándares y estilos a utilizar

El propósito fundamental de los estándares de codificación es lograr que el sistema resulte fácil de entender y facilitar el futuro mantenimiento del software por desarrolladores ajenos al proyecto. Los estándares de codificación son un complemento a la programación, no solo es importante tener un estándar, sino tener un buen estándar de codificación, que permita que los programas se acerquen lo mejor posible al lenguaje natural e incorporar un buen estilo de programación, que aporte a la eficiencia del proceso de desarrollo.

Existen un conjunto de nomenclaturas, pero la de excelencia en el mundo del lenguaje java, es Camel Case, que consiste en utilizar las mayúsculas como separadores de palabras. Si comienza con mayúscula, se denomina *UpperCamelCase*, de lo contrario, *lowerCamelCase*.

### 4.5.1 Lower Camel Case

Similar al Camel Case sólo que la primera letra, de la primera palabra, es también en minúscula. Por ejemplo, la frase "generar reporte" pasada a *lowerCamelCase* sería "generarReporte".

### 4.5.2 Upper Camel Case

Similar al Camel Case solo que la primera letra, de la primera palabra, se escribe en mayúscula. Por ejemplo, la frase "generar reporte" pasada a *UpperCamelCase* sería "GenerarReporte".

### 4.5.3 Identación

La indentación o sangría debe realizarse mediante tabulaciones y no se deben insertar espacios. Se recomienda que el tamaño de las tabulaciones sea de cuatro espacios aunque esto queda a opción del programador.

### 4.5.4 Variables

El nombre empleado para las variables y constantes, debe permitir que con sólo leerlo se conozca el propósito de la misma.

El nombre que se le da a las variables debe comenzar con la primera letra en minúscula e identificará el tipo de dato al que se refiere. En caso de que sea un nombre compuesto, la segunda palabra comenzará con letra inicial mayúscula.

Ejemplo: seleccionarUsuario.

### 4.5.5 Constantes

Las constantes dentro de una clase se deben declarar todas en mayúsculas y las diferentes palabras que la compongan, separadas por barras bajas (_).

Ejemplo:String EJBQL = "select report from Reporte_estadisticas report join report.reporteUsuarios ru";

### 4.5.6 Métodos y Funciones

Los métodos de las clases, al igual que las funciones normales, deben estar escritos en *lowerCamelCase*. Los delimitadores del cuerpo del método serán llaves { }, la de apertura en la misma línea que la declaración del método separado por un espacio, aunque se permite que esté en la siguiente línea. La llave de cierre debe estar en una línea diferente y no debe existir más código que la propia llave de cierre. Hay que incorporarles su modificador de acceso aunque sea privada.

Ejemplo:private String getNombre() {

//código

}

### 4.5.7 Clases

Los nombres de las clases deben estar escritos en *UpperCamelCase*. La indentación de las mismas debe ser de una tabulación con respecto al margen izquierdo, mientras que los campos y los métodos deben tener un sangría con respecto a la clase.

Ejemplo:public class ListarReporte {

//código

}

## 4.6 Validación de la implementación propuesta

Con vista a verificar el correcto funcionamiento de la solución implementada el algoritmo de búsqueda de caminos desarrollado se ejecutó con la base de datos del Sistema de Información Hospitalaria del Centro de Informática Médica que cuanta con alrededor de 3000 tablas, generaron más de 96 millones de rutas solamente desde un esquema de la base de datos hacia el resto, utilizando más de 20 gb de espacio en disco. El proceso se interrumpió por falta de espacio en el servidor de prueba, por lo que se decidió

trabajar con otro esquema que contuviera menos tablas para comprobar su funcionalidad. El resultado arrojado mostró que el algoritmo aplica de forma correcta la teoría sobre la búsqueda de caminos entre un conjunto de tablas.

A partir de esto se implementó y probó en el módulo de Estadísticas del HIS el conjunto de funcionalidades de generación de reportes y asignación de privilegios a conceptos de manera satisfactoria.

## Conclusiones

En el presente capítulo se cumplió con los requisitos funcionales y no funcionales especificados en el Capítulo 2. Se realizó un análisis sobre los aspectos de mayor importancia de la implementación del sistema. Se construyó el Modelo de Datos, el Diagrama de Componentes y Diagrama de Despliegue; además de describirse las tablas de la base datos, que son vitales para la aplicación. Durante el proceso de desarrollo del componente de generación dinámica de reportes se hizo uso de los estándares de codificación y estilos planteados en el capítulo, lo que permitió que el sistema resulte fácil de entender y facilitar el futuro mantenimiento del software por desarrolladores ajenos al proyecto.

# CONCLUSIONES

Con el desarrollo del Generador de reportes dinámicos basados en conceptos para el Sistema de Información Hospitalaria del CESIM, se concluye lo siguiente:

1. El análisis de los sistemas relacionados con el campo de acción evidenció que los mismos no cumplen con todas las funcionalidades deseadas ni permiten su integración al Sistema de Información Hospitalaria del CESIM.
2. Se realizó una selección de indicadores de gestión en instituciones de atención médica para dotar la aplicación de conceptos médicos previamente definidos y que los mismos puedan ser utilizados por los usuarios.
3. Luego de analizadas las aplicaciones, herramientas y bibliotecas de clases que generan reportes, se identificó la biblioteca de clases DynamicReports como la vía más idónea para el desarrollo del componente, debido a que permite la generación de reportes de alta complejidad con el mínimo de esfuerzo.
4. Para el desarrollo del componente se asimiló la arquitectura definida por el departamento Sistemas de Gestión Hospitalaria para que la documentación generada cumpla con las pautas definidas por el departamento y el componente desarrollado sea totalmente integrable a la solución existente.
5. Se desarrolló un componente de generación de reportes dinámicos basados en conceptos para el Sistema de Información Hospitalaria del CESIM lo que posibilita la construcción de los mismos de acuerdo a las necesidades del personal médico y de forma rápida e intuitiva.

# RECOMENDACIONES

Sobre la presente investigación los autores recomiendan:

1. Implementar un algoritmo de poda que permita reducir o controlar el conjunto de tablas implicadas en la búsqueda de caminos para la unión entre conceptos.
2. Incorporar las funcionalidades que brinden la posibilidad de insertar, modificar y eliminar conceptos médicos.

# REFERENCIAS BIBLIOGRÁFICAS

1. **Instituto de Información Científica y Tecnológica (IDICT).** Tecnologías de la Información y las Comunicaciones (TIC). *EcuRed.* [En línea] 2008. http://www.ecured.cu/index.php/Tecnolog%C3%ADas_de_la_informaci%C3%B3n_y_las_comunicaciones.

2. **Francisco J. Fernández Puerto, Florina Gatica Lara.** *Sistema de Información Hospitalaria.* México : s.n., 2003.

3. **García Serven, José R.** *Indicadores de Gestión para Establecimientos de Atención Médica.* Caracas : DISINLIMED, CA., 1993.

4. **Idem 3.**

5. **Organización Mundial de la Salud (OMS).** *Constitución de la Organización Mundial de la Salud.* Nueva York : s.n., 1946.

6. **Ramírez Marquéz, Dr. Abelardo, Castell-Florit Serrate, Dr. Pastor y Mesa, Dr. Guillermo.** *El Sistema Nacional de Salud de Cuba.* Ciudad de la Habana : s.n., 2003.

7. **Moya de Madrigal.** *Introducción a la estadística en salud.* Costa Rica : s.n., 1989.

8. **Idem 2.**

9. **Softel.** *Propuesta para la Informatización de Centros de alención Médica. Galen Hospital.* La Habana : s.n., 2007.

10. **Comlogik.** Comlogik. [En línea] 25 de 2 de 2014. http://www.comlogik.com/Products.aspx#products.

11. **3M.** 3M Health Information Systems. [En línea] 2014. http://solutions.3m.com/wps/portal/3M/en_US/Health-Information-Systems/HIS/Products-and-Services.

12. **Holzinger, Andreas, Burgsteiner, Harald y Maresch, Helfrid.** *Experiences with the Practical Use of Care2x.* Austria : s.n., 2002.

13. **Abreu Medina, Aldis Joan.** *Generador Dinámico de Reportes.* La Habana : s.n., 2012.

14. **Pentaho Corporation.** Pentaho. [En línea] 2005. http://www.pentaho.com/.

15. **Jaspersoft.** *Jaspersoft OLAP User Guide.* U.S.A : s.n., 2013.

16. **Mariaca, Ricardo.** DynamicReports. [En línea] 3 de 6 de 2007. [Citado el: 15 de 12 de 2013.] http://www.dynamicreports.org/.

17. **Juan Manuel Alvarez, Martin Gorostegui, Mariano Simone, Alejandro Gomez, Ricardo Mariaca.** DynamicJasper. [En línea] 2012. [Citado el: 15 de 12 de 2013.] http://dynamicjasper.com/.

18. **Alfred V. Aho, John E. Hopcroft, Jefrey D. Ullman con la colaboración de Guillermo Levine Gutiérrez.** *Estructura de datos y algoritmos.* s.l. : México, DF , 1988. pág. 207.

19. **Dijkstra, E.W.** *A Note on Two Problems in Connexion with Graphs.* 1959. págs. 269-271.

20. **Reynoso, Carlos y Kicillof, Nicolás.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. BUENOS AIRES : UNIVERSIDAD DE BUENOS AIRES, 2004.

21. **García de Jalón, Javier, y otros, y otros.** *Aprenda Java como si estuviera en primero.* España. : s.n., 2000.

22. **Idem 19.**

23. **Oracle.** The Java EE 5 Tutorial. [En línea] Septiembre de 2007. http://docs.oracle.com/javaee/5/tutorial/doc/bnaph.html.

24. **JBoss community.** *RichFaces Developer Guide.*

25. **Idem 20.**

26. **Sánchez Rosas, Juan Eladio.** Desarrollo en Web. *Facelets y JSF – Uso de Templates.* [En línea] 17 de Diciembre de 2008. http://blogs.antartec.com/desarrolloweb/2008/12/facelets-y-jsf-uso-de-templates/.

27. **Red Hat.** *JBoss Enterprise Application Platform 4.2 Seam Reference Guide.* 2010.

28. **Desarrolloweb.com.** Manual de HTML. [En línea] 2013. http://www.desarrolloweb.com/manuales/21/.

29. **Idem 19.**

30. **Oracle.** The Java EE 5 Tutorial . [En línea] 2010. http://docs.oracle.com/javaee/5/tutorial/doc/bnbpz.html.

31. **Bernard, Emmanuel.** *Hibernate Annotations.* 20012.

32. **Itera.** Rational Unified Process (RUP). [En línea] 2008. http://www.iteraprocess.com/index.php?option=com_content&task=view&id=18&Itemid=42.

33. **S. Pressman, Roger.** *Ingeniería de Software.* New York : McGraw-Hill, 2001.

34. PostgreSQL About. [En línea] http://www.postgresql.org/about.

35. **Red Hat.** Red Hat JBoss Developer Studio 5.0. [En línea] https://devstudio.jboss.com/earlyaccess/5.0.0.GA.html.

36. **Craig Larman.** *UML y Patrones. Modelo de Dominio.* s.l. : Prentice Hall, 2003.

37. **O.P, Giraldo.** *Ingeniería de Requisitos. Volumen, 13.* 2007.

38. **López, Ing. Leydis Hidalgo.** *ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE DEL SISTEMA HIS – Elementos Comunes.* Habana : s.n., 07/04/2014.

39. **S. Pressman, Roger.** *Software Engineering.* New York : McGraw-Hill, 2010.

40. **Morgade, Gerardo Donato.** *Proyecto Implantación del Sistema Integral de Salud en clínicas y hospitales de PDVSA. Fase II.ARQUITECTURA DE SOFTWARE DEL SISTEMA HIS.* Universidad de Ciencias Informáticas. Habana : s.n. pág. 18.

41. **Mercedes, Avda Reina.** *Bases de Datos Tema 3 Modelo de Datos.* Sevilla : Departamento de Lenguajes y Sistemas Informáticos E.T.S.Ingeniería Informática, 205.

# ANEXOS

Anexo 1: Interfaz de usuario para la asignación de privilegios del reporte. Componente Generador Dinámico de Reportes basado en conceptos.

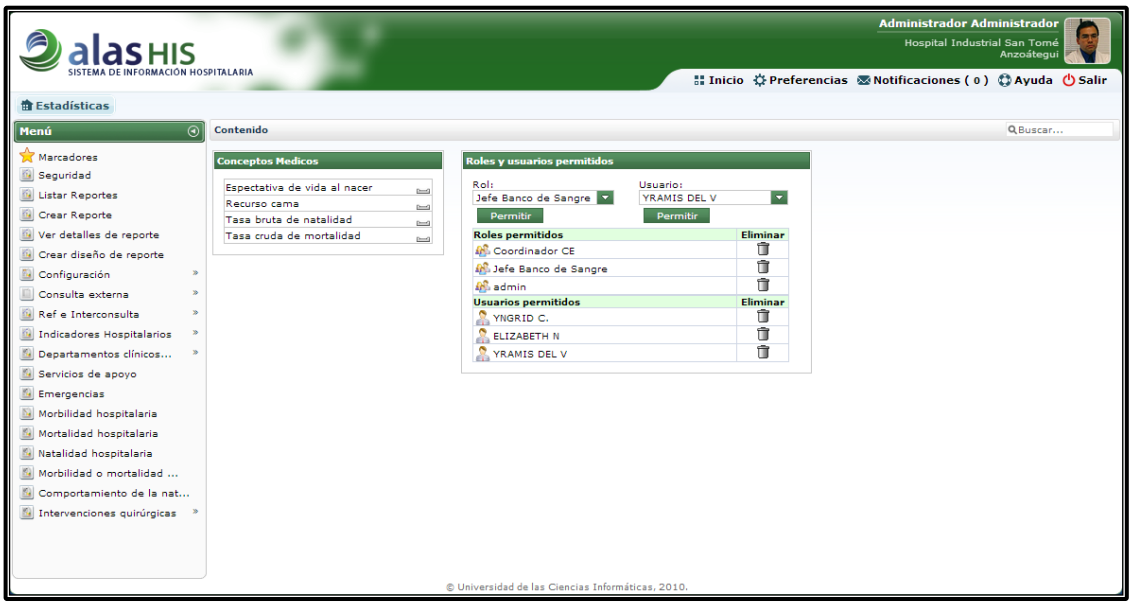



Figura A1: Asignar privilegios de reporte

Anexo 2: Interfaz de usuario para listar los reportes creados. Componente Generador Dinámico de Reportes basado en conceptos.

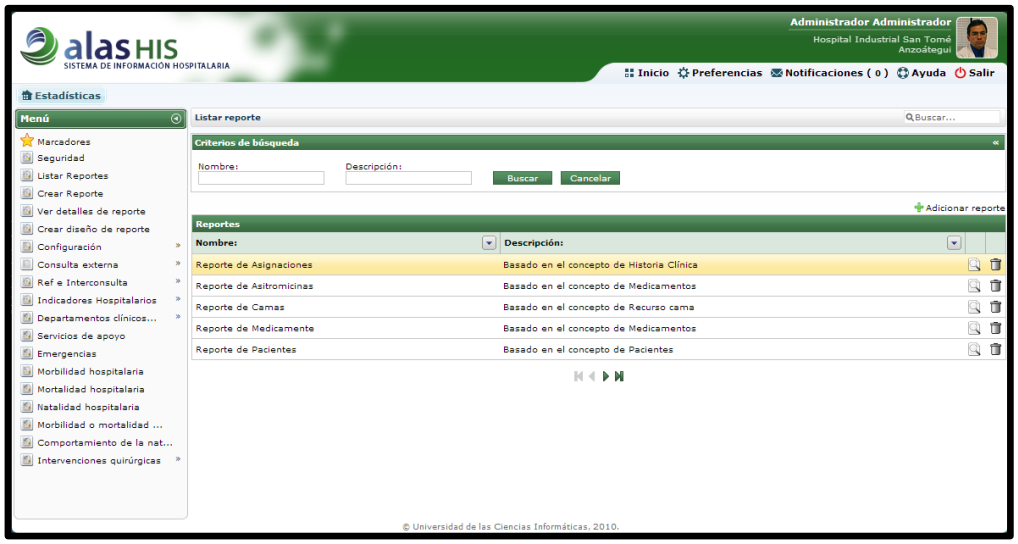



Figura A2: Listar diseño de reporte

Anexo 3: Interfaz de usuario para crear el diseño del reporte. Componente Generador Dinámico de Reportes basado en conceptos.

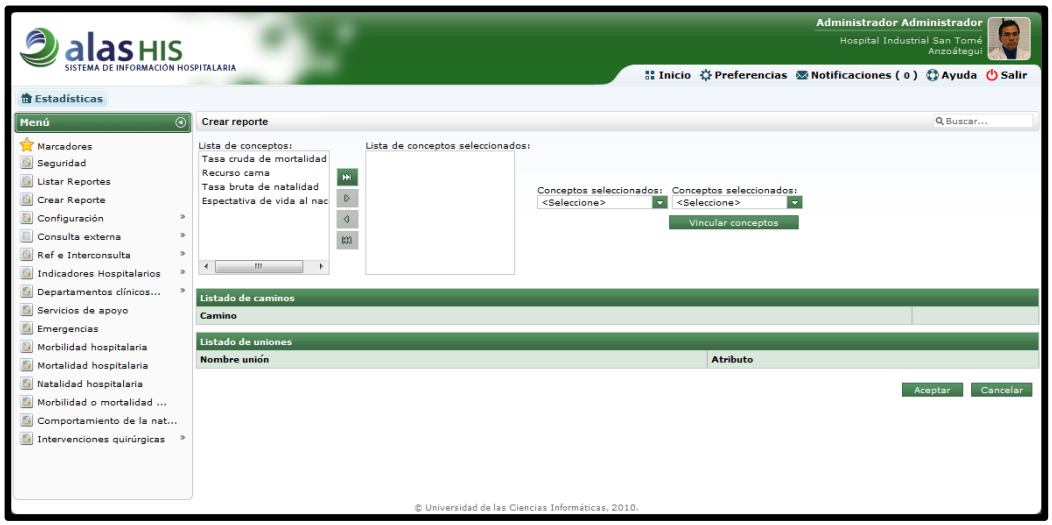



Figura A3: Crear diseño de reporte

# GLOSARIO DE TÉRMINOS

**CIE-10:** es el acrónimo de la Clasificación Internacional de Enfermedades**,** décima versión, determina la clasificación y codificación de las enfermedades y una amplia variedad de signos, síntomas, hallazgos anormales, denuncias, circunstancias sociales y causas externas de daños y/o enfermedad.

**Java Development Kit (JDK):** es un software que provee herramientas de desarrollo para la creación de programas en Java.

**Java Database Conectivity (JDBC):** es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede.

**World Wide Web Consortium (W3C):** es una comunidad internacional donde las organizaciones miembros, un equipo de trabajo a tiempo completo, y el público, trabajan de conjunto para desarrollar estándares web.

**HTTP:** protocolo de transferencia de hipertexto, usado en la World Wide Web. Este protocolo define cómo los mensajes son formateados y transmitidos, además de cuáles acciones deben tomar los servidores web y navegadores en respuesta a varios comandos.

**TCP / IP:** protocolo de control de transmisión (TCP) y Protocolo de Internet (IP), conjunto de protocolos de red en los que se basa Internet y que permiten la transmisión de datos entre redes de computadoras.

**USB/LTP:** significa "Terminal de impresión". Esto se debe a que el puerto LPT o USB se utiliza generalmente para conectar una impresora a su computadora.

**ALBET S.A:** es una empresa cubana, cuyo origen y desarrollo se vincula estrechamente a la Universidad de Ciencias Informáticas (UCI). Posee los derechos comerciales de todos los productos y servicios que desarrolla la UCI y mediante la alianza con otras prestigiosas entidades ofrece soluciones integrales en la esfera de las tecnologías de la información y las comunicaciones.