# Universidad de las Ciencias Informáticas

**Facultad 7**


# Título: Desarrollo del Módulo Configuración del Sistema de Información Hospitalaria alas HIS

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

**Autores:** David Crespo Pérez
Nelson Francisco Fernández Pérez

**Tutores:** Ing. Alejandro Mario Velázquez Carralero
Ing. Reynier Soto Góngora

Ciudad de La Habana, Julio de 2010
"Año 52 de la Revolución"

## Resumen

El objetivo de este trabajo de diploma consiste en la necesidad de la creación de un módulo de configuración encargado de gestionar como su nombre lo indica la configuración del sistema de forma centralizada.

El módulo controla cada aspecto de la configuración para todos los módulos que conforman al HIS. Esto evita la gestión de forma independiente en cada módulo, lo que conllevaría a un conjunto de problemas como: multiplicidad de la implementación de funcionalidades por cada módulo, la creación de usuarios con diferentes roles que al final responden a una misma persona, así como la ausencia de este módulo cuyo fin sea contener funcionalidades generales y administrativas como las citadas anteriormente.

El desarrollo del sistema está guiado por el Proceso Unificado de Desarrollo y se basa en tecnologías libres, multiplataforma y sobre una arquitectura en capas. Se utiliza además Java como lenguaje de programación y se implementa el patrón de arquitectura Modelo Vista Controlador. Como Sistema de Gestión de Bases de Datos se hace uso de PostgreSQL y como servidor de aplicaciones el JBoss Server. Para la administración de las reglas se utiliza Drools y JasperReport para la generación de reportes.

El sistema posibilita gestionar la seguridad centralizada mediante la asignación de privilegios a usuarios del sistema. Además la administración de entidades, módulos y procesos se pudiera gestionar de forma dinámica en tiempo de ejecución. Alcanza una mayor integración y compatibilidad con sistemas externos existentes y la posibilidad de generar reportes de información estadística actualizada de las entidades.

**Palabras claves:** Configuración, gestión de la información, seguridad, módulos, funcionalidades.

# Tabla de contenidos

# Índice de tablas

# Índice de figuras

# Introducción

En la actualidad la situación económica y el desarrollo de la ciencia y la tecnología imponen nuevos retos al mundo. La superación profesional, la rapidez, eficacia y eficiencia que se desarrollen para tomar decisiones, así como la optimización de los servicios, constituyen los principales objetivos a lograr en los diferentes sectores de la sociedad. El proceso de informatización de cada uno de estos campos contribuiría a implementar nuevas vías, menos complejas y capaces de brindar una mejor respuesta a los problemas existentes en las esferas de la sociedad.

La informática ha desempeñado un papel primordial en el desarrollo científico y técnico de todas las esferas sociales. Hoy en día se han implementado nuevas vías, las cuales son menos complejas pero capaces de brindar una mejor solución a los problemas existentes en sectores como la salud. La creación de los Sistemas de Información Hospitalaria ha sido uno de los frentes fundamentales en la búsqueda de la informatización de procesos y servicios, que redunden, a su vez, en una mejor atención a las personas.

La información clínica que se genera diariamente, en paralelo al crecimiento poblacional a nivel mundial, determina ineficiente crecimiento en el consumo de variados recursos. Entre estas ineficiencias se cita el costoso papel, que inevitablemente, al discurrir el tiempo, se deteriora, se torna ilegible la información y se pierde completamente. Por esta y otras razones cada día se buscan nuevas soluciones informáticas con el objetivo de lograr mayor rendimiento y aprovechamiento de los recursos utilizados, así como su perdurabilidad y con esta la posibilidad de extender el potencial de uso de la información recopilada.

Con el objetivo de solucionar esta problemática se han desarrollado numerosos sistemas conocidos comúnmente como sistemas de información hospitalaria (HIS, por sus siglas en inglés). Estos han ido evolucionando, atendiendo a mejoras de funcionalidades de procesos médicos y también a la estructura presentada por las instituciones hospitalarias en las cuales se encuentran. Su impacto en las instituciones de salud es fuerte, ya que busca elevar la calidad de la atención del paciente y de los servicios brindados, así como aplicar la información obtenida a las áreas de investigación, clínica, docencia y administración con el propósito adicional de abatir costos y elevar la productividad.

Un HIS está conformado por módulos que responden a cada una de las áreas que se pueden encontrar en un hospital. Teniendo como característica principal la interacción, de forma tal que la información generada por cualquiera de las áreas pueda almacenarse, y al mismo tiempo fluir hasta el resto de los módulos que se encuentran interactuando. Entre los módulos que podemos encontrar en un HIS están: Hospitalización, Emergencia, Admisión, Laboratorio, y otros. Cada una de las funcionalidades que poseen, como el flujo de datos, es controlada por la administración central del hospital.

Cada uno de los módulos de un HIS trabaja usando como condición necesaria la información proporcionada por el personal que interactúa con el sistema. Este último gestiona toda la información referente a los módulos atendiendo a la configuración realizada previamente. Dicha configuración se puede reflejar en la gestión de datos referentes a medios físicos y de capital humano que garanticen, desde la seguridad hasta el correcto uso de los recursos tangibles existentes en el hospital.

Un sistema informático de gestión de la información ante todo debe contar con la adecuada seguridad. Para esto crean usuarios como medio de acceso a la aplicación, para cada trabajador y roles, e identifica el nivel que tenga, el cual se diseña atendiendo a las responsabilidades específicas. La gestión de los usuarios es una funcionalidad general, común. Cada persona que necesite acceder al sistema debe contar con un usuario sin importar sobre que módulo trabaje o la responsabilidad que desempeñe. Es por ello que esta funcionalidad no debe de ser implementada en un módulo, como Banco de Sangre, Hospitalización, Epidemiología, entre otros, que responden a un área específica del hospital y sus funciones bien definidas.

La gestión de camas en los hospitales se realiza de forma manual, controlada por un empleado, subordinado a la Administración. Este personal lleva el control de las camas en un documento donde consta, además, el estado físico de ellas. Este proceso manual está sujeto a errores en la adecuada actualización de los datos, lo que puede conllevar a que se asignen camas en mal estado a pacientes, o que estas simplemente ya no existan, originando demoras y conflictos en los procesos de hospitalización en entidades que dependen o utilizan temporalmente este recurso físico.

Un hospital generalmente no cuenta con todos los servicios médicos que puede necesitar un paciente en un momento determinado. Por ello existen las referencias a otras instituciones por servicios no disponibles en la entidad, es decir, una remisión. La información relacionada con las posibles remisiones a realizar forma parte de la configuración de la propia entidad en cuestión y no depende de un área

específica del hospital. Por esta razón surge la necesidad de automatizar la gestión de estos datos, para prevenir que, por falta de conocimiento del personal médico, un paciente sea erróneamente remitido a un hospital de referencia, que tal vez ya no brinde el tipo de servicio que necesita la persona.

Actualmente no existe un proceso automatizado para mitigar cada uno de los errores presentados anteriormente, o llegar a soluciones en un tiempo racional. Por ello la gestión de una herramienta informática que atienda la automatización administrativa, la de recursos y funcionalidades de los hospitales, viabilizaría en gran medida sus procedimientos.

Evitaría problemas existentes en las aplicaciones de gestión de la información de los sistemas de salud como pueden ser: inaccesibilidad, no automatización de funciones necesarias para la configuración inicial, la no actualización de la información en tiempo real disponible para todo el personal médico de la institución. Por lo que se define de forma general la ausencia de controles de gestión en el sistema.

Atendiendo a todo lo antes citado, llegamos al ***Problema a resolver*****:** ¿Cómo facilitar la centralización de las configuraciones necesarias para garantizar una correcta gestión de la información en el Sistema de Información Hospitalaria alas HIS?

El ***Objeto de estudio*** está constituido por el análisis de los procesos de gestión de la información manejada por los módulos de los Sistemas de Información Hospitalaria, y enmarcado en el ***Campo de acción*** asociado a los procesos de gestión de la configuración del Sistema de Información Hospitalaria alas HIS.

Para solucionar el problema identificado se propone el siguiente ***Objetivo general:*** Desarrollar el Módulo Configuración del Sistema de Información Hospitalaria alas HIS.

Para dar solución al objetivo planteado se definen las siguientes tareas a desarrollar:

1. Evaluar las tendencias actuales en el mundo de los Sistemas de Información Hospitalaria.
2. Identificar los procesos de negocios asociados a la gestión de la administración en las instituciones hospitalarias.
3. Asimilar la arquitectura y pautas definidas por el Departamento de Sistemas de Gestión Hospitalaria para el desarrollo de sus aplicaciones.

4. Obtener los artefactos correspondientes a los flujos de trabajo de “Modelado de Negocio”, “Gestión de Requerimientos”, “Análisis y Diseño” e “Implementación”.

De esta forma se puede destacar que el desarrollo del Módulo Configuración del Sistema de Información Hospitalaria, brindará los siguientes beneficios a partir de su puesta en práctica:

1. Configuración centralizada del Sistema de Información Hospitalaria alas HIS en todo su conjunto, agilizando así el proceso de puesta en ejecución del mismo.
2. Administración de roles, usuarios, entidades, seguridad entre otros recursos disponibles en las entidades, así como la configuración de reglas del negocio y el flujo de la información para cada uno de los módulos.
3. Adaptabilidad a heterogéneos entornos de despliegue, así como a los procesos existentes en las estaciones hospitalarias mediante la utilización de jBPM y Drools.
4. Flexibilidad para configuración personal de cada usuario por roles de trabajo.

El presente documento se encuentra estructurado en cuatro capítulos. El primero de ellos, “**FUNDAMENTACIÓN TEÓRICA**”, contiene un estudio del estado del arte así como el ambiente de desarrollo del Módulo Configuración. Justificándose además las tendencias, tecnologías, metodologías y herramientas que fueron utilizadas para su desarrollo. Seguidamente el capítulo, “**CARACTERÍSTICAS DEL SISTEMA**”, contiene un marco conceptual asociado a la información que será manipulada por el sistema. En este se llega a un acuerdo sobre las funcionalidades, requerimientos deseados y el objeto de automatización, quedando explícitamente descritos mediante casos de uso del sistema.

El tercer capítulo **“DISEÑO DEL SISTEMA”** se centra en la modelación detallada y la construcción de la estructura de la aplicación. En el cuarto y último, **“IMPLEMENTACIÓN”,** se implementan las clases y subsistemas en términos de componentes. Se presenta la propuesta de solución para lograr una gestión más eficiente de los requerimientos de seguridad y configuración centralizada para el Sistema de Información Hospitalaria alas HIS.

# Capítulo 1: Fundamentación Teórica

Este capítulo está dedicado a realizar un estudio minucioso de los principales conceptos y características que constituyen las bases para el desarrollo del Módulo Configuración del Sistema de Información Hospitalaria alas HIS. Se presenta un conjunto de temas relacionados con la configuración general de los Sistemas de Información Hospitalaria así como tecnologías, metodologías y herramientas de software definidas por el Departamento de Sistemas de Gestión Hospitalaria con las que se llevará a cabo el proceso de desarrollo.

## 1.1 Sistemas de Información Hospitalaria

Los sistemas de información están orientados a satisfacer las necesidades de generación de información para almacenar, procesar y reinterpretar datos médico-administrativos de cualquier institución hospitalaria. Permiten la optimización de los recursos humanos y materiales minimizando a su vez los inconvenientes burocráticos que pudieran afrontar los pacientes en el proceso de atención médica. Su principal función es la de apoyar las actividades en los niveles operativos, tácticos y estratégicos dentro de un Hospital, haciendo uso de las computadoras para recabar, almacenar, procesar y comunicar información clínica y administrativa. (1) Brindan una solución estructurada en módulos, representando de forma independiente cada una de las áreas de una institución hospitalaria de manera que se integren al paciente, al personal médico y administrativo.

## 1.2 Descripción específica del módulo

El objetivo principal del Módulo Configuración es gestionar la información general necesaria para trabajar con el Sistema de Información Hospitalaria alas HIS. Entre sus funcionalidades se encuentra la gestión de la información relacionada con la administración de Usuarios. Dicha administración implica la creación de cuentas de usuarios que utilizarán el sistema, clave de acceso y privilegios sobre las opciones y operaciones del Sistema. Otras de sus funcionalidades son la actualización de la información referente a los servicios y departamentos con que cuenta el hospital, las posibles referencias a otras instituciones por

servicios no disponibles y el control de las camas y ubicaciones; permitiendo así una mayor flexibilidad ante cambios funcionales en el sistema y ofreciendo una visión general de la entidad.

### 1.3 Sistemas automatizados existentes vinculados al campo de acción

Se ha realizado un análisis de los diferentes sistemas informáticos vinculados al sector de la salud los formarán parte del estado del arte del trabajo. Haciendo énfasis en los aspectos relacionados con la configuración general, con el objetivo de determinar soluciones eficaces capaces de brindar respuesta a los problemas existentes en la gestión de la información necesaria para usar el Sistema. Para ello se describen las ventajas y desventajas que presenta cada uno, así como la descripción de las tecnologías y herramientas utilizadas para el desarrollo y posterior funcionamiento en las instituciones hospitalarias.

Los principales sistemas informáticos analizados son los siguientes:

**SIGHO**

SIGHO es un Sistema de Información para la Gerencia Hospitalaria. Ha sido implementado para brindar apoyo a la gerencia de todos los Hospitales del sector salud en México. Se desarrolló basado en la Norma Oficial Mexicana NOM-168-SSA1-1998, permitiendo realizar registros individuales alrededor de la historia clínica electrónica en cada uno de los módulos que componen al SIGHO relacionados con la atención al paciente.(2)

Se compone de 14 módulos (2 administrativos y 12 relacionados con la atención al paciente). Incluye el Módulo Configuración, que contiene las opciones de configuración de parámetros necesarios para el correcto funcionamiento de los módulos del SIGHO. Este módulo también permite la administración de Usuarios que implica la creación de cuentas de los usuarios que utilizarán el sistema. Su gestión incluye la asignación de clave de acceso y la asignación de privilegios sobre las opciones y operaciones del Sistema a cada una de las cuentas creadas. El objetivo principal de este módulo es configurar la información general necesaria para trabajar con el SIGHO. (3)

**HIS CNT Pacientes**

La empresa CNT Sistemas de Información, establecida en Colombia con 25 años de experiencia, especializada para realizar soluciones integradas en el área de la salud. Presenta como su principal

producto el HIS CNT Pacientes. Dicho sistema está diseñado para integrar todo el ciclo de atención del paciente frente a la prestación de servicios médicos, terapéuticos y de diagnósticos, cuyo eje funcional es la Historia Clínica del paciente. (4) Constituye una herramienta ideal para administrar la esencia del negocio de Clínicas y Hospitales de alta complejidad. A partir de ésta se generan distintas órdenes médicas electrónicas OME, que en línea alimentan todos los procesos clínicos y administrativos que interactúan con las diferentes áreas de la institución. Altamente parametrizable, ajustable a necesidades puntuales de cada organización. Cuenta con el módulo de: Central de Urgencias y/o Emergencias, Administración de salas de Cirugías, Control de Cuentas Glosadas u Objetadas y Administración de Adscritos. (5)

**XHosp**

Es un sistema de información hospitalario para el apoyo operacional y el control administrativo integral de un hospital o clínica. Posibilita llevar el control administrativo de pacientes hospitalizados durante su internamiento. Hace posible llevar el control de las transferencias o movimientos de los pacientes entre las diferentes áreas de hospitalización. Fue diseñado desde su inicio como un sistema modular y escalable, que puede ser instalado en la mayoría de las instituciones hospitalarias, desde clínicas regionales pequeñas hasta los grandes centros médicos institucionales o privados.

Aprovecha todas las ventajas de una aplicación diseñada y desarrollada para Windows: interfaz de usuario gráfica y amigable, capacidad de compartir la información con otras aplicaciones de Windows como Word, Excel o Access, entre otras. Permite utilizar prácticamente cualquier base de datos para el almacenamiento de la información de pacientes y resultados. (6)

Posee un módulo de Administración, donde se gestiona:

- Seguridad de usuarios a 3 niveles.
- Aumento automático de precios.
- Actualización de catálogos.
- Configuración del sistema.
- Reportes administrativos y estadísticas médicas.

**Hosix-V**

Es un Sistema de Información Hospitalaria flexible, integrado y modular. Abarca todas las áreas de actividad de un Hospital y pretende, a través de una utilización fácil, rentabilizar los recursos existentes para organizar el trabajo desarrollado diariamente. Hosix-V es uno de los sistemas más flexibles porque está constituido por módulos orientados a la gestión específica de cada servicio o departamento, algunos de los cuales pueden funcionar autónomamente, y se adaptan fácilmente a los diversos tipos de organización. Es un sistema integrado porque relaciona directamente las Áreas principales de gestión de un hospital (Asistencial, Clínica, Económica, Servicios y Control de Gestión). Ha sido desarrollado utilizando como lenguaje de programación Microsoft Visual Basic y herramientas que son propiedades de la compañía Microsoft, como Microsoft Office 2003 Professional Edition, Microsoft Office 2003 Small Business Edition, Microsoft Office 2003 Standard Edition. (7)

**SELENE de SIEMENS**

Es una solución clínica integral que gestiona el proceso asistencial al paciente de forma completa. Las posibilidades de configuración y personalización pueden ser llevadas hasta el nivel de usuario final. Ello posibilita la adaptación de los objetos clínicos a casi todos los flujos de trabajo. Además también permite la generación del entorno adecuado para cada usuario en cada uno de los ámbitos en los que realiza su actividad.

Entre sus funcionalidades básicas se encuentran la:

Herramienta asistencial: empleada por el personal médico. Incluye todas las posibilidades de almacenamiento y acceso a la información del paciente. Está organizada en procesos asistenciales completos y la definición de protocolos médicos y enfermeros como soporte al flujo de trabajo asistencial.

Herramienta de gestión clínica: Entre sus funcionalidades se encuentra la gestión de recursos en un centro. Posibilita la gestión compartida de recursos entre centros con independencia de la herramienta de gestión propia de cada uno de ellos. (8)

**Valoración de los sistemas analizados**

Existe un grupo de características con las que deben contar los productos, tales como:

- Desarrollado sobre software libre.

- Ser una aplicación web implementando un modelo Cliente-Servidor para que la utilización de un SO determinado no resulte un inconveniente en su utilización.
- Constituir un sistema flexible, independiente desde el punto de vista modular-funcional pero al mismo tiempo integrado.

Por las características mencionadas se plantea que las soluciones anteriormente analizadas presentan un gran conjunto de desventajas. Una de estas desventajas está dada porque ninguna de ellas gestiona de forma centralizada toda la información necesaria para comenzar a trabajar con el Sistema. Particularmente el caso del sistema SIGHO presenta un módulo de configuración que posee un gran número de funcionalidades administrativas, pero aún así no llega a alcanzar una configuración detallada y rigurosa de los recursos de la entidad a la que responde.

Otros de los sistemas implementan la gestión de la información necesaria de forma independiente, separada, sin lograr una centralización de la gestión de funcionalidades comunes. La mayoría de los sistemas están desarrollados con herramientas y tecnologías de carácter propietarias (también llamadas privativas o de código cerrado). De esta forma el funcionamiento de estos sistemas se convierte en un secreto que guarda celosamente la compañía que los produce, donde es imposible encontrar la causa de un resultado erróneo, producido por un componente cuyo funcionamiento se desconoce. Además, algunos son aplicaciones de escritorio carentes de multifuncionalidad con otros sistemas operativos (SO).

## 1.4 Herramientas y tecnologías de desarrollo a tener en cuenta

Durante el proceso de desarrollo del software el principal objetivo es lograr la comunicación entre los integrantes del grupo de desarrollo y el cliente, y de los entes que intervienen en la gestión de un producto. Esta relación hace a la aplicación más robusta, portable, capaz de asimilar cambios estructurales de la forma más flexible posible, sin tener que llegar a transformaciones severas. Los patrones arquitectónicos proveen a los desarrolladores de gran ayuda y arquitectura claramente definida.

Existen grandes compilaciones de definiciones alternativas de la arquitectura del software donde una de las más reconocidas es la de Clements, que plantea:

> "…es una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las

formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema…"(9)

Una vez analizadas las características necesarias del producto informático que se desea realizar se define un conjunto de herramientas y tecnologías a utilizar para su creación. Además se hace preciso alcanzar el deseado nivel de capacitación y experiencia del equipo de desarrollo con que se cuenta. De esta forma se garantiza la creación de un producto robusto, costeable y eficiente, en el menor tiempo posible, pero de alta calidad.

### 1.4.1 Java EE

Java Enterprise Edition (Java EE), es una plataforma de programación distribuida para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java. Soporta una arquitectura de N niveles a ejecutar sobre un servidor de aplicaciones basadas en un completo grupo de componentes modulares de software. Posibilita el manejo de diversos detalles mediante una programación simple. Se rige por un estándar que la asemeja a otras de la comunidad de procesos de Java. (10)

### 1.4.2 Seam

Seam es un marco de trabajo desarrollado para Java EE, creada para la realización de aplicaciones Web 2.0 de forma fácil, soportando una arquitectura unificada de componentes. Permite la integración de tecnologías como AJAX, JavaServer Faces (JSF), Enterprise Java Beans (EJB3), java Portlets, Business Process Management (BPM), Drools, Hibernate y JPA en una única solución.

Elimina en gran medida la complejidad existente desde el nivel de arquitectura hasta el nivel API. Permite el desarrollo de aplicaciones web basadas en Plain Old Java Objects (POJO), componentes de UI y la menor cantidad de XML. Se integra con librerías de controles de código abierto basadas en JSF como RichFaces e ICEFaces. (11)

### 1.4.3 JBoss AS como servidor de aplicaciones

JBoss Application Server es el servidor de aplicaciones de código abierto. Cuenta con licencia LGPL, por lo que puede ser distribuido o redistribuido y usarse en la implementación de cualquier aplicación comercial sin la implicación de costo alguno. Soporta todas las especificaciones correspondientes, incluyendo servicios adicionales como clusterizar, carga en memoria caché y persistencia, por ser una

plataforma con certificación JEE 5. Es ideal como servidor de aplicaciones Java y aplicaciones Web. También soporta Enterprise Java Beans (EJB) 3.0. Cuenta con un conjunto de componentes claves como son: JBoss AS 4.2, Hibernate 3.2.4, Seam 2.0. (12)

### 1.4.4 JBoss Tools

JBoss Tools es un conjunto de plug-ins de Eclipse que tiene como objetivo ayudar a los desarrolladores a crear aplicaciones webs de forma rápida y sencilla.

Los módulos de JBoss Tools son:

- RichFaces VE: Editor visual proporcionado por Exadel. Brinda el apoyo para la edición visual de páginas HTML, JSF, JSP y Facelets. También incluye soporte visual para las librerías de componentes JSF incluyendo JBoss RichFaces.
- Seam Tools: Incluye soporte para seam-gen, RichFaces VE.
- Hibernate Tools: Soporta el mapeo de archivos, anotaciones y JPA con la ingeniería inversa, completamiento de código, asistentes de proyecto, refactorización, ejecución interactiva de HQL/JPA-QL/Criteria.
- JBoss AS Tools: Fácil de iniciar, detener y analizar ejecución paso a paso al estar integrado con Eclipse. También incluye funciones para el despliegue eficaz de cualquier tipo de proyecto en el IDE.
- Drools IDE: Editor de ficheros de reglas, análisis de ejecución paso a paso e inspección de reglas.
- JBPM Tools: Edición del flujo de trabajo del JBPM, motor de procesos BPM.
- JBossWS Tools: Desarrollo, invocación, inspección y pruebas de servicios web sobre http con la adición y soporte de características JBossWS.

### 1.4.5 JSF

JSF es una librería de interfaz de usuario para aplicaciones Web implementadas con Java. Diseñado para aliviar la carga de desarrollo y mantenimiento de aplicaciones que se ejecutan en Servidores de aplicaciones Java y prestar sus interfaces de usuario a un cliente objetivo, JSF aprovecha

las interfaces de usuario ya existentes, estándares y conceptos de Web. JSF usa Facelets como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL. (13)

### 1.4.6 Rich Faces 3.2 como librería de componentes JSF

Rich Faces es una librería de código abierto que añade los componentes de Ajax (Ajax4jsf) en aplicaciones existentes de JSF sin recurrir a JavaScript. Rich aprovecha la librería de JavaServer Faces para incluir validaciones, instalaciones de conversión y la gestión de los recursos estáticos y dinámicos. De esta forma permite a los desarrolladores ahorrar tiempo y aprovechar las características de los componentes para crear aplicaciones Web, con mejor apariencia visual. (14)

### 1.4.7 Ajax4jsf

Ajax4jsf es una librería de código abierto que añade la capacidad de AJAX en las aplicaciones existentes de JSF sin recurrir a la JavaScript. Ajax4jsf se integra totalmente en la implementación JS Faces como librería que permite: la validación, las instalaciones de conversión y gestión de los recursos estáticos y dinámicos. Mediante esta librería se puede variar el ciclo de vida de una petición JSF, recargar determinados componentes de la página sin necesidad de recargar por completo, realizar peticiones automáticas al servidor, control de cualquier evento de usuario, etc. Ajax4jsf es función de soporte de AJAX y altamente personalizable (look-and-feel) que pueden ser fácilmente incorporados en aplicaciones JSF. (15)

### 1.4.8 PosgtreSQL 8.3 como servidor de base de datos

PosgtreSQL es un sistema gestor de bases de datos objeto-relacional (ORDBMS) basado en Postgres, versión 4.2, desarrollado en la Universidad de California en Berkeley Computer Science Department. Postgres fue pionera en muchos conceptos que sólo estuvo disponible en algunos sistemas de bases de datos comerciales mucho después (PostgreSQL Global Development Group, 2005).

PostgreSQL es un descendiente de código fuente abierto del código original de Berkeley. Soporta una gran parte del estándar SQL y ofrece muchas características modernas:

- Consultas complejas.

- Claves foráneas.
- Desencadenantes o disparadores.
- Vistas.
- La integridad transaccional.
- Control de concurrencia multiversión.

El tamaño máximo de la base de datos es ilimitado; el de una tabla asciende a 32 TB, el de una fila a 1.6 TB y el de un campo de datos a 1 GB. De igual forma el número de filas en una tabla es ilimitado, pero no el de columnas, que oscila entre 250 y 1 600 columnas por tabla. El número de índices por tabla es también ilimitado.

Además, PostgreSQL puede ser ampliado por el usuario en muchos aspectos, por ejemplo, la adición de nuevos:

- Tipos de datos.
- Funciones.
- Operadores.
- Las funciones de indexado.
- Métodos de índice.
- Lenguajes procedurales.

Debido a la licencia libre, PostgreSQL puede ser utilizado, modificado y distribuido por todo el mundo de forma gratuita para cualquier propósito, sea comercial privado, o académico. (16)

### 1.4.9 Framework Hibernate para acceso a los datos

Hibernate es un producto de código abierto (publicado bajo licencia LPGL). Proporcionar la persistencia y el mapeo objeto-relacional de objetos Java, mediante archivos declarativos (XML) que permiten establecer estas relaciones. Tiene soporte para más de 30 dialectos diferentes (controladores de bases de datos diferentes).

Proporciona un lenguaje de consulta rica para acceder a objetos, proporcionar almacenamiento en caché y el apoyo JMX. El es destinado a proporcionar la persistencia de alto rendimiento con bajos recursos. (17)

### 1.4.10 JPA (Java Persistence API)

Java Persistence API (JPA) proporciona un modelo de persistencia basado en POJO's para mapear bases de datos relacionales en Java. El Java Persistence API fue desarrollado por el grupo de expertos de EJB 3.0 como parte de JSR 220, aunque su uso no se limita a los componentes software EJB. También puede utilizarse directamente en aplicaciones web y aplicaciones clientes; incluso fuera de la plataforma Java EE, por ejemplo, en aplicaciones Java SE.

En su definición, se han combinado ideas y conceptos de las principales librerías de persistencia como Hibernate, Toplink y JDO, y de las versiones anteriores de EJB. Todos estos cuentan actualmente con una implementación JPA. (18)

## 1.5 Lenguajes de programación

### 1.5.1 Java

Java es un lenguaje originalmente desarrollado por un grupo de ingenieros de Sun Microsystems. Es utilizado posteriormente por Netscape como base para Javascript. Su uso se destaca en la Web, sirviendo para crear todo tipo de aplicaciones (locales, intranet o internet).

Java es un lenguaje:

- de objetos.
- independiente de la plataforma.

Algunas características notables:

- robusto.
- gestiona la memoria automáticamente.
- multi-hilo.

- cliente-servidor.
- mecanismos de seguridad incorporados.
- herramientas de documentación incorporadas.

El lenguaje mismo se inspira en la sintaxis de C++, pero su funcionamiento es más similar al de Smalltalk que a éste.

**Orientado a objetos**

Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos se agrupan en estructuras encapsuladas, tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red. (19)

**Interpretado y compilado a la vez**

Java es un lenguaje que presenta un conjunto de características significativas. Entre ellas se destaca que es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador. Además es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina la cual porte el intérprete y el sistema de ejecución en tiempo real (run-time). (19)

**Distribuido**

Java proporciona una colección de clases para su uso en aplicaciones de red. Ejemplos de su puesta en práctica lo constituyen, la posibilidad de abrir sockets, así como establecer y aceptar conexiones con servidores o clientes remotos. De esta forma se facilita la creación de aplicaciones distribuidas. (19)

**Robusto**

Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), al prescindir por completo de los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria. (19)

**Seguro**

Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia. Para eso se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real. Su objetivo, es que no se pueda ejecutar en el ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas. (19)

**Indiferente a la arquitectura**

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows NT, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñada para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de los problemas son solucionados por el intérprete de Java. (19)

**Portable**

La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas. Estas dos últimas características se conocen como la Máquina Virtual Java (JVM). (19)

**Alto rendimiento**

**Multihebra**

Hoy se ven limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario, un tercero presenta una animación en pantalla y el cuarto realiza cálculos. (19)

**Dinámico**

El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código de baja demanda, procedente de fuentes muy variadas, incluso desde la Red. (19)

## 1.6 Sistemas distribuidos

### 1.6.1 Modelo Cliente-Servidor

El esquema cliente-servidor es un modelo de computación. En él, el procesamiento requerido para ejecutar una aplicación o conjunto de aplicaciones relacionadas, se divide entre dos o más procesos que cooperan entre sí. Usualmente la mayoría del trabajo pesado se hace en el proceso llamado servidor y el (los) proceso(s) cliente(s) sólo se ocupa de la interacción con el usuario (aunque esto puede variar).

Los principales componentes del esquema cliente-servidor son los Clientes, los Servidores y la infraestructura de comunicaciones.

Los Clientes interactúan con el usuario, usualmente en forma gráfica. Frecuentemente se comunican con procesos auxiliares que se encargan de establecer conexión con el servidor, enviar el pedido, recibir la respuesta, manejar los fallos y realizar actividades de sincronización y de seguridad.

Los Servidores proporcionan un servicio al cliente y devuelven los resultados. En algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente. Además, deben manejar los interbloqueos, la recuperación ante fallos, y otros aspectos afines.

Por las razones anteriores, la plataforma computacional asociada con los servidores es más poderosa que la de los clientes. Por esta razón se utilizan PCs poderosos, estaciones de trabajo, minicomputadores o sistemas grandes. Además, deben manejar servicios como administración de la red, mensajes, control y administración de la entrada al sistema ("login"), auditoría y recuperación y contabilidad. Usualmente en los servidores existe algún tipo de servicio de bases de datos. (20)

### 1.6.1 Estilo arquitectónico

El estilo arquitectónico o variante arquitectónica define a una familia de sistemas informáticos en términos de su organización estructural. Describe componentes y las relaciones entre ellos con las restricciones de su aplicación, la composición asociada y el diseño para su construcción.

### 1.6.2 Modelo Vista Controlador (MVC)

La arquitectura MVC (Model/View/Controller) fue introducida como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk. Diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizado de los mismos datos. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas. Esto permite que cualquier cambio producido en el Modelo, se refleje automáticamente en cada una de las Vista, las cuales a su vez cuentan con su correspondiente controlador.

**Definición por partes:**

El Modelo es el objeto que representa los datos del programa, a los que maneja, y controla todas sus transformaciones. No tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Interactúa con este a través de una referencia al mismo. Además genera una representación visual de la estructura de datos mostrándolos al usuario.

El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, dado por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo. (21)

Este modelo de arquitectura presenta varias ventajas:

- Presenta una clara separación entre los componentes de un programa; que permite implementarlos por separado.

- Cuanta con un API muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.
- La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

## 1.7 Metodologías de desarrollo

### 1.8.1 RUP (Racional Unified Process). Proceso Unificado Racional

RUP es una metodología adaptable al contexto y necesidades de cada organización cuyo fin es entregar un producto de software. Durante su puesta en práctica se estructuran todos los procesos y se mide la eficiencia de la organización. Además es un proceso de desarrollo de software el cual utiliza el lenguaje unificado de modelado UML. Constituye así la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos**.**

Principales características de RUP:

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo.
- Administración de requisitos.
- Uso de arquitectura basada en componentes.
- Control de cambios.
- Modelado visual del software.
- Verificación de la calidad del software.

Dado las características anteriormente planteadas Racional Unified Process es capaz de implementar:

Desarrollo iterativo del software:

- Permite comprender los requerimientos que hacen crecer el sistema.

- Sigue un modelo que busca las tareas más riesgosas, reduciendo así los riesgos del proyecto.

Administración de requerimientos:

- Describir como se obtienen, organizan, documentan los requerimientos.
- Captar y comunicar los requerimientos de la organización.
- Documentar las decisiones.

Uso de arquitecturas basadas en componentes:

- Se basa en diseñar una arquitectura que sea flexible, fácil de modificar, comprensible y que se fundamenta en la reutilización de sus componentes.

Modelado visual del software:

- Modela visualmente la organización.
- Permite analizar la consistencia entre los componentes, el diseño y su implementación.
- Verificar calidad del software.
- Controlar cambios.

RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al culminar cada uno de ellos, que a la vez se dividen en fases y donde se debe tomar una decisión importante:

Concepción: se hace un plan de fases, se identifican los principales casos de uso y los riesgos.

Elaboración: se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.

Construcción: se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.

Transición: se Instala el producto en el cliente y se entrena a los usuarios. Surgen nuevos requisitos para ser analizados.

Mantenimiento: una vez instalado el producto, el usuario realiza requerimientos de ajuste. Esto se realiza de acuerdo con solicitudes generadas como consecuencia de interactuar con el producto.

**Flujos de trabajo a desarrollar**

En Racional Unified Process se definen nueve flujos de trabajo distintos, separados en dos grupos. Los flujos de trabajo de ingeniería son los siguientes:

- Modelado del negocio.
- Requisitos.
- Análisis y diseño.
- Implementación.
- Test.
- Despliegue.

Los flujos de trabajo de apoyo son:

- Administración del proyecto.
- Configuración y control de cambios.
- Entorno.

Aunque los nombres de los flujos de trabajo de ingeniería recuerden a las etapas de una metodología en cascada, en RUP las fases son distintas, y estos flujos de trabajo serán visitados una y otra vez a lo largo de todo el proceso. De forma general se plantea que RUP se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. (22)

En el desarrollo del proyecto se hace mayor énfasis en los flujos de trabajo de “Modelado de Negocio”, “Gestión de Requerimientos”, “Análisis y Diseño” e “Implementación” obteniendo los artefactos correspondientes a cada uno de estos flujos.

**Modelado del negocio**

Con este flujo de trabajo se pretende llegar a un mejor entendimiento de la organización donde se va a implantar el producto. Los principales motivos para esto son los siguientes: asegurarse que el producto será algo útil, no un obstáculo; lograr el mayor porciento de integración posible en la organización; y tener

un marco común para los desarrolladores, los clientes y los usuarios finales. Para modelar el negocio se usan las mismas técnicas que para modelar software. De esta forma se logra que ambas partes entiendan los modelos. En general se cuentan casos de uso de negocio, actores de negocio, entre otros, elementos todos que ayudan a describir desde el punto de vista informático el problema a resolver.

**Gestión de requerimientos**

Este es uno de los flujos de trabajo más importantes, porque en él se establece qué es lo que tiene que hacer exactamente el sistema que se desarrolla. En esta línea los requisitos son el contrato que se debe cumplir. De esta forma los usuarios finales deben que comprender y aceptar los requisitos especificados. Estos requerimientos se dividen en dos grupos, los funcionales y los no funcionales. Los funcionales representan todo lo que hace la aplicación, es decir, sus funcionalidades y los no funcionales son aquellos atributos que debe exhibir el sistema, pero que no son una funcionalidad específica. En este flujo de trabajo, y como parte de los requisitos de usabilidad, se diseña la interfaz gráfica de usuario.

**Análisis y diseño**

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema. El análisis consiste en obtener una visión del sistema que se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales. El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. Los resultados finales más importantes de este flujo de trabajo serán: el modelo de diseño constituido por colaboraciones de clases, que pueden ser agregadas en paquetes y subsistemas, y la documentación de la arquitectura software en la que se capturan varias visiones arquitectónicas del sistema.

**Implementación**

En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. Además, se deben hacer las pruebas de unidad: cada implementador es responsable de probar las unidades que produzca. El resultado final de este flujo de trabajo es un sistema ejecutable.

### 1.8.2 UML(Unifed Modeling Languaje)

El lenguaje para modelado unificado (UML), es utilizado para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. Fue originalmente concebido por la Corporación Rational Software y tres de los más prominentes metodológicos en la industria de la tecnología y sistemas de información: Grady Booch, James Rumbaugh, e Ivar Jacobson (“The Three Amigos”).

UML prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objeto y describe la semántica esencial de lo que estos diagramas y símbolos significan. UML supone una abstracción de un sistema para llegar a construirlo en términos concretos. Permite describir un sistema en diferentes niveles de abstracción, simplificando la complejidad sin perder información, para que los usuarios y desarrolladores comprendan las características de la aplicación.

Entre más complejo es el sistema que se desea crear más beneficios presenta el uso de UML, por las siguientes razones:

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.

- Diagramas de Casos de Uso para modelar los procesos “business”.
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- Diagramas de Colaboración para modelar interacciones entre objetos.
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de Componentes para modelar componentes.
- Diagramas de Implementación para modelar la distribución del sistema.

Los principales beneficios de UML son:

- Mejores tiempos totales de desarrollo (de 50 % o más).
- Modelar sistemas (y no sólo de software) utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado por humanos y por máquinas.
- Mejor soporte a la planificación y al control de proyectos.
- Alta reutilización y minimización de costos.

## 1.9 Herramientas de desarrollo

### 1.10.1 Eclipse como herramienta de desarrollo

Se define como un IDE para todo y nada en particular. Eclipse es, en el fondo, únicamente un armazón (workbench) sobre el que se pueden montar herramientas de desarrollo para cualquier lenguaje, mediante la implementación de los plug-ins adecuados. El IDE Eclipse es, únicamente, una de las herramientas que se engloban bajo el denominado Proyecto Eclipse. El Proyecto Eclipse aúna tanto el desarrollo del IDE Eclipse como de algunos de los plug-ins mas importantes (como el JDT, plug-ins para el lenguaje Java, o el CDT, plug-ins para el lenguaje C/C++).

La arquitectura de plug-ins de Eclipse permite, además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, etc.

Todas las versiones de Eclipse necesitan tener instalado en el sistema una máquina virtual Java (JVM), preferiblemente JRE (Java Runtime Environment) o JDK (Java Developer Kit) de Sun.

Eclipse se distribuye bajo licencia EPL (Eclipse Public License). Esta licencia es considerada como libre por la FSF y por la OSI. La licencia EPL permite usar, modificar, copiar y distribuir nuevas versiones del

producto licenciado. El antecesor de EPL es CPL (Common Public Licencse) escrita por IBM. (23)

### 1.10.2 PgAdmin como aplicación cliente para manejar la Base de datos

PgAdmin III es una aplicación gráfica para trabajar con el gestor de bases de datos PostgresSQL. Es la más completa y popular con licencia Open Source. Está escrita en C++. Usa la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usan en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como Pervasive Postgres, EnterpriseDB, Mammoth Replicator y SRA PowerGres.

PgAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, soporte para el motor de replicación Slony-I y mucho más. La conexión al servidor puede hacerse mediante conexión TCP/IP o Unix Domain Sockets (en plataformas Unix), y puede encriptarse mediante SSL para mayor seguridad. (24)

### 1.10.3 Visual Paradigm como herramienta de modelado

Visual Paradigm 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 contribuye a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Visual Paradigm para UML incluye los objetos más recientes de UML, además de diagramas de casos de uso, diagramas de clase, diagramas de componentes, reversa instantánea para Java, C++, DotNet Exe/dll, XML, XML Schema, y Corba IDL, ofrece soporte para Rational Rose, integración con Microsoft Visio, además permite generar reportes y documentación en HTML/PDF.

Provee soporte para la generación de código, tiene integración con diversos IDE’s como NetBeans (de Sun Microsystems), JDeveloper (de Oracle), Eclipse (de IBM), JBuilder (de Borland), así como la

posibilidad de realizarse la ingeniería inversa para aplicaciones realizadas en JAVA, .NET, XML e Hibernate.

Es una aplicación portable. Su diseño se centra en casos de uso y se enfoca al negocio que genera un software de mayor calidad. También tiene disponibilidad en múltiples plataformas. Soporta BPMN (Business Process Modeling Notation), modelado colaborativo con CVS y Subversion.

### 1.10.4 JRE

Java Runtime Environment (JRE) es un conjunto de herramientas provisto por la empresa creadora del lenguaje JAVA, Sun Microsystems, para la ejecución de código. Este paquete provee la máquina virtual de JAVA ó JVM dado sus siglas en inglés, para que el código sea interpretado, independientemente del hardware del ordenador. Además, contiene todo el conjunto de APIs indispensables para que todo código JAVA pueda ejecutarse. El avance de esta tecnología ha permitido a quienes acceden a la Red disponer de un sinnúmero de utilidades, como juegos, clientes P2P, aplicaciones para móviles, herramientas de sistema.

Incorpora plug-ins para los principales navegadores (Internet Explorer, Firefox, Opera, Zafari, Netscape), para explorar Internet disfrutando de todo el potencial de los sitios y aplicaciones web, así como sus agregados elaborados en JAVA, como aplicaciones de chat y otras interactivas.

Un usuario sólo necesita el JRE para ejecutar las aplicaciones desarrolladas en lenguaje Java, mientras que para desarrollar nuevas aplicaciones en dicho lenguaje es necesario un entorno de desarrollo, denominado JDK, que además del JRE (mínimo imprescindible) incluye, entre otros, un compilador para Java.

### Conclusiones

En el capítulo, después de realizar un estudio de varios Sistemas de Información Hospitalaria que existen a nivel internacional, se logra determinar los principales problemas en la configuración general de dichas aplicaciones y acumular información para la implementación de la configuración del Sistema de Información Hospitalaria alas HIS. Es por ello que a partir de las experiencias obtenidas se decide implementar el Módulo Configuración para el Sistema de Información Hospitalaria alas HIS.

De esta forma se agruparían todas las funcionalidades administrativas que se encuentren en cada uno de los módulos del sistema, logrando con ello una mejor centralización, organización y flexibilidad de su configuración. Finalmente, como aporte de la caracterización de las herramientas y las tecnologías que son utilizadas para el desarrollo de la solución propuesta, se logra mostrar en gran medida la robustez, potencialidad y portabilidad que brinda el software libre para el desarrollo de proyectos a gran escala.

# Capítulo 2: Características del sistema

En el presente capítulo se realiza la descripción de los objetivos estratégicos de la organización, el flujo actual de los procesos involucrados en el campo de acción, y se describen los procesos que son objeto de automatización. Debido a la no existencia de una definición clara de los procesos de negocio en los que tiene lugar la configuración del sistema, se decide desarrollar un Modelo de Dominio, el cual abarca las definiciones asociadas a los conceptos encontrados en el entorno donde está enmarcado el sistema, así como las relaciones existentes entre ellos. Se plantean además los requerimientos funcionales y no funcionales y se explica de forma detallada la solución propuesta en términos de casos de uso.

## 2.1 Modelo de Dominio

El Modelo de Dominio, o Modelo Conceptual, captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará embebido el sistema. Es construido con las reglas de UML durante la fase de concepción, en la tarea de construcción del modelo de dominio, presentado como uno o más diagramas de clases. Se utiliza para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. Contiene conceptos que estarán asociados tanto a su definición natural como al papel que juegan desde el punto de vista informático.

## 2.2 Conceptos fundamentales del dominio

Para brindar una mejor comprensión del Diagrama del Modelo de Dominio a continuación se realiza una breve descripción de los conceptos encontrados en el problema.

**2.2.1** **Usuario:** Persona que por medio de un ordenador puede acceder a los recursos y servicios que ofrece el sistema de acuerdo con los privilegios, permisos y roles asignados.

**2.2.2** **Servicio:** Constituirá un conjunto organizado de recursos humanos y materiales. Su objetivo será prestar atención especializada a pacientes en consulta ambulatoria referida, en hospitalización y en

atención de emergencia, mediante la aplicación de los procedimientos de diagnóstico y terapéutica correspondientes a su área específica.

**2.2.3 Entidad:** Es el establecimiento destinado al diagnóstico y tratamiento de los individuos que padecen una determinada enfermedad y acuden a él para recibir un diagnóstico y un posterior tratamiento de su afección.

**2.2.4 Médico:** Es el usuario o persona con ciertos privilegios y accesos a funcionalidades especificas del sistema que practica la medicina, intenta mantener, prevenir y recuperar la salud humana mediante el estudio, el diagnóstico y tratamiento de la enfermedad o lesión del paciente.

**2.2.5 Enfermero:** Es el usuario o cualificado profesional que cuida y proporciona la atención individualizada de los enfermos y se centra en las condiciones de los pacientes.

**2.2.6 Perfil:** Es toda la información obtenida del usuario con sus preferencias y configuraciones del sistema aplicada en su entorno de trabajo.

**2.2.7 Rol:** Son los permisos o niveles de seguridad asignados a un usuario de la aplicación. En dependencia del rol que tenga asignado podrá disponer de la información y funcionalidades del sistema.

**2.2.8 Población del área de influencia:** Es la población que se encuentra en el área donde la institución hospitalaria puede reflejar sus servicios médicos en bienestar de sus pacientes.

**2.2.9 Tipo entidad:** Son las clasificaciones de las instituciones hospitalarias según los niveles de salud.

**2.2.10 Tipo ubicación:** Son las distintas clasificaciones que puede tomar las ubicaciones dependiendo del lugar físico del hospital, dígase: ala, habitación privada, habitación pública.

**2.2.11 Departamento:** Unidad estructural hospitalaria que se ocupa de un determinado servicio o servicios afines.

## 2.3 Diagrama del modelo de dominio

Diagrama del modelo

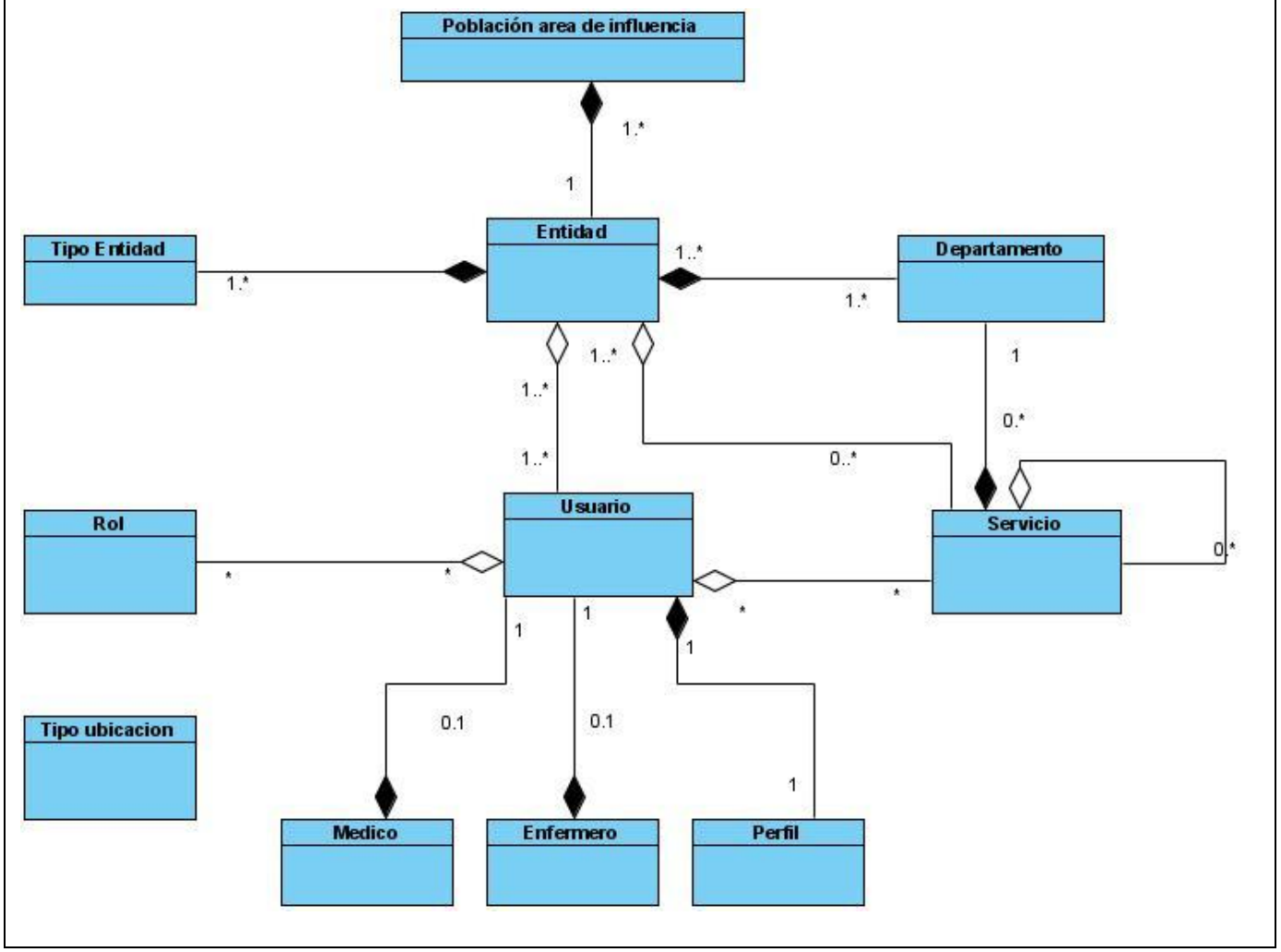


Figura 2.1. Diagrama del Modelo de Dominio

## 2.4 Propuesta del sistema

La aplicación que se propone tiene como principal objetivo lograr una gestión centralizada de la información previa y necesaria para la puesta en práctica del Sistema de Información Hospitalaria alas HIS. Además, gestionar la configuración y las funcionalidades globales de dicho sistema. El software a desarrollar será una aplicación web, la cual reportará un conjunto de beneficios significativos a todos sus usuarios. Entre estos beneficios se citan: la gestión central de usuarios y roles, la asignación de privilegios de seguridad a los usuarios atendiendo al cargo que desempeñan profesionalmente (médico, enfermera, etc.), la configuración del perfil personal a la hora de interactuar con el sistema, así como la gestión de departamentos y servicios con que cuentan las entidades (software multientidad) a las que responde el sistema, característica que se logra mediante la configuración que provee el Módulo Configuración.

Se garantiza una confidencialidad e integridad de la información manejada, persistida en la base de datos localizada en un servidor central. Dicha información se encuentra asegurada por mecanismos de alta fidelidad debido al nivel de integridad de la aplicación con el resto de los módulos. Estos interactúan simultáneamente sobre el servidor de base de datos e implícitamente sobre la información almacenada.

## 2.5 Especificación de los requisitos del software

Un requisito de software podría definirse como una condición o capacidad que es necesitada por los clientes y debe encontrarse en un sistema o componente para satisfacer un contrato, norma, especificación u otro documento impuesto formalmente. El conjunto de todas las necesidades es el fundamento para el consiguiente desarrollo del sistema o componente.

Los requisitos del software tienen varias clasificaciones entre las que se encuentran, los Requisitos funcionales y los Requisitos no funcionales.

**Requisitos funcionales**

Describen la funcionalidad o los servicios que se espera que el sistema proveerá, sus entradas, salidas y excepciones.

**Requisitos no funcionales**

Se refieren a las propiedades emergentes del sistema como la fiabilidad, el tiempo de respuesta, la capacidad de almacenamiento, la capacidad de los dispositivos de entrada/salida, y la representación de datos que se utiliza en las interfaces del sistema.

### 2.5.1 Requisitos funcionales

El sistema deberá permitir:

| Requerimientos funcionales | |
|---|---|
| RF1 - Crear usuario. | RF33 - Crear departamento no clínico. |
| RF2 - Modificar usuario. | RF34 - Modificar departamento no clínico. |
| RF3 - Ver datos del usuario. | RF35 - Ver datos del departamento no clínico. |
| RF4 - Buscar usuario. | RF36 - Buscar departamento no clínico. |
| RF5 - Eliminar usuario. | RF37 - Eliminar departamento no clínico. |
| RF6 - Crear rol. | RF38 - Crear servicio clínico. |
| RF7 - Modificar rol. | RF39 - Modificar servicio clínico. |
| RF8 - Ver datos del rol. | RF40 - Ver datos del servicio clínico. |
| RF9 - Buscar rol. | RF41 - Buscar servicio clínico. |
| RF10 - Eliminar rol. | RF42 - Eliminar servicio clínico. |
| RF11 - Crear perfil. | RF43 - Crear servicio no clínico. |
| RF12 - Modificar perfil. | RF44 - Modificar servicio no clínico. |
| RF13 - Ver datos del perfil. | RF45 - Ver datos del no clínico. |
| RF14 - Buscar perfil. | RF46 - Buscar servicio no clínico. |

| | |
|---|---|
| RF15 - Eliminar perfil. | RF47 - Eliminar servicio no clínico. |
| RF16 - Crear médico. | RF48 - Crear tipo de ubicación. |
| RF17 - Asociar usuario a médico. | RF49 - Modificar tipo de ubicación. |
| RF18 - Modificar médico. | RF50 - Ver datos del tipo de ubicación. |
| RF19 - Ver datos del médico. | RF51 - Buscar tipo de ubicación. |
| RF20 - Buscar médico. | RF52 - Eliminar tipo de ubicación. |
| RF21 - Eliminar médico. | RF53 - Crear tipo de cama. |
| RF22 - Crear enfermera. | RF54 - Modificar tipo de cama. |
| RF23 - Asociar usuario a enfermera. | RF55 - Ver datos del tipo de cama. |
| RF24 - Ver datos de la enfermera | RF56 - Buscar tipo de cama. |
| RF25 - Modificar enfermera. | RF57 - Eliminar tipo de cama. |
| RF26 - Buscar enfermera. | RF58 - Crear entidad. |
| RF27 - Eliminar enfermera. | RF59 - Modificar entidad. |
| RF28 - Crear departamento clínico. | RF60 - Ver datos de la entidad. |
| RF29 - Modificar departamento clínico. | RF61 - Buscar entidad. |
| RF30 - Ver datos del departamento clínico. | RF62 - Eliminar entidad. |
| RF31 - Buscar departamento clínico. | RF63 - Ver datos de la ficha hospitalaria. |
| RF32 - Eliminar departamento clínico. | |

### 2.5.2 Requisitos no funcionales

#### 2.5.2.1 Usabilidad

El sistema estará diseñado de manera que los usuarios adquieran las habilidades necesarias para explotarlo en un tiempo reducido:

- Usuarios normales: 20 días
- Usuarios avanzados: 30 días

La estructura concebida para la organización de la información agiliza el entendimiento del sistema por parte del usuario. Los usuarios serán capaces de alcanzar sus objetivos con un mínimo esfuerzo y obteniendo los resultados máximos.

El sistema será capaz de solucionar un error cometido por el usuario o sugerir las posibles soluciones, indicándole al usuario las acciones pertinentes a seguir. Además brindará comodidad a la hora de acceder a las diferentes funcionalidades que proporciona la aplicación mediante teclas de acceso rápido. Serán reutilizados diseños de aplicaciones comúnmente usada por los usuarios finales con vistas a aprovechar la experiencia de usuario.

#### 2.5.2.2 Fiabilidad

Las informaciones médicas relacionadas con los pacientes y que vayan a ser intercambiadas con otros hospitales por la red pública, viajarán cifradas para evitar accesos o modificaciones no autorizadas.

Se mantendrá seguridad y control a nivel de usuario, garantizando el acceso de estos sólo a los niveles establecidos, de acuerdo con la función que realizan. Las contraseñas podrán cambiarse solo por el propio usuario o por el administrador del sistema.

Se mantendrá un segundo nivel de seguridad a nivel de estaciones de trabajo, garantizando sólo la ejecución de las aplicaciones que hayan sido definidas para la estación en cuestión. Además de un registro de todas las acciones que se realizan, llevando el control de las actividades de cada usuario en todo momento.

Se establecerán mecanismos de control y verificación para los procesos susceptibles de fraude. Los mecanismos serán capaces de informar al personal autorizado sobre posibles irregularidades que den indicios sobre la introducción de información falseada.

El sistema implementará un mecanismo de auditoría para el registro de todos los accesos efectuados por los usuarios, proporcionando un registro de actividades (log) de cada usuario en el sistema. Este soportará el uso de firmas digitales para la transferencia de información cuya certificación sea imprescindible para validar su uso.

La aplicación implementará un control de cambios a determinados campos de información (seleccionados por su importancia), de forma tal que sea posible determinar cuáles han sido las actualizaciones que se le han realizado. Por otro lado, ninguna información que se haya ingresado en el sistema será eliminada físicamente de la BD, independientemente de que para el sistema este elemento ya no exista. Por lo que el sistema permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.

#### 2.5.2.3 Eficiencia

El Centro de Datos permitirá agregar recursos para aumentar el poder de procesamiento y almacenamiento sin afectar los sistemas, garantizando expansiones motivadas por futuros requerimientos.

El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria. Para ello se potenciará como regla guardar en la memoria caché datos y recursos de alta demanda.

#### 2.5.2.4 Rendimiento

El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria y respetará buenas prácticas de programación para incrementar el rendimiento en operaciones costosas para la máquina virtual como la creación de objetos.

#### 2.5.2.5 Soporte

Se permitirá la creación de usuarios, otorgamiento de privilegios y roles, asignación de perfiles y activación de permisos por direcciones IP además de la administración remota, monitoreo del funcionamiento del sistema en los centros hospitalarios y detección de fallas de comunicación.

Se permitirá realizar copias de seguridad de la base de datos hacia otro dispositivo de almacenamiento externo, además de recuperar la base de datos a partir de los respaldos realizados y el chequeo de las operaciones y acceso de los usuarios al sistema. Así como establecer parámetros de configuración del sistema y actualización de nomencladores.

#### 2.5.2.6 Requerimientos de hardware

- Estaciones de trabajo.

  En la solución se incluyen estaciones de trabajo para las consultas del Sistema de Información Hospitalaria alas HIS. Dichas estaciones necesitan capacidad de hardware que soporte un sistema operativo el cual cuente con un navegador actualizado que siga los estándares web. Por lo que se escogieron estaciones de trabajo de 256 Mb de memoria RAM y un microprocesador de 2.0 Hz con sistema operativo Linux.

- Servidores.

  La solución estará conformada, fundamentalmente, por servidores de alta capacidad de procesamiento y redundancia, que permitan garantizar movilidad y residencia de la información y las aplicaciones bajo esquemas seguros y confiables.

  - ✓ Servidores de Base de datos: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 4GB de memoria y 2x72GB de disco y sistema operativo Linux.
  - ✓ Servidores de Aplicaciones: 2 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 4GB de memoria y 2x72GB de disco y sistema operativo Linux.
  - ✓ Servidores de Intercambio: 1 DL380 G5, Procesador Intel® Xeon® 5140 Dual - Core 2 GB de memoria y 2x72GB de disco y sistema operativo Linux.

#### 2.5.2.7 Requerimientos de software

La aplicación debe correr en sistemas operativos Windows, Unix y Linux, utilizando la plataforma JAVA (Java Virtual Machine, JBoss AS y PostgreSQL). Deberá disponer de un navegador web, estos pueden ser IE 7, Opera 9, Google chrome 1 y Firefox 2 o versiones superiores de estos.

#### 2.5.2.8 Restricciones de diseño

La capa de presentación contendrá todas las vistas y la lógica de la presentación. El flujo web se manejará de forma declarativa y basándose en definiciones de procesos del negocio. La capa del negocio mantendrá el estado de las conversaciones y procesos del negocio que concurrentemente pueden estar siendo ejecutados por cada usuario. La capa de acceso a datos contendrá las entidades y los objetos de acceso a datos correspondientes a las mismas. El acceso a datos está basado en el estándar JPA y particularmente en la implementación del motor de persistencia Hibernate.

#### 2.5.2.9 Requisitos para la documentación de usuarios en línea y ayuda del sistema

Se posibilitará el uso de ayudas dinámicas y tutoriales en línea sobre el funcionamiento del sistema.

#### 2.5.2.10 Interfaz

- **Interfaces de usuario**

  Las ventanas del sistema contendrán los datos claros y bien estructurados. Además permitirán la interpretación correcta de la información. La interfaz contará con teclas de función y menús desplegables que faciliten y aceleren su utilización. La entrada de datos incorrecta será detectada claramente e informada al usuario. Todos los textos y mensajes en pantalla aparecerán en idioma español.

- **Interfaces software**

  Se interactuará con el sistema alas RIS para realizar solicitudes y obtener resultados de estudios radiológicos e imagenológicos.

#### 2.5.2.11 Portabilidad

El producto podrá ser utilizado bajo los sistemas operativos Linux o Windows.

### 2.5.3 Modelo de casos de uso del sistema

#### 2.5.3.1 Definición de actores

| Actor | Descripción |
|---|---|
| Usuario | Usuario global que permite la autenticación en el sistema y que valida al mismo dándole un rol. |
| Administrador del sistema | Es el encargado de realizar la configuración general del sistema. Maneja los datos con que cuentan las entidades para el correcto funcionamiento de sus actividades en las distintas áreas que la componen. Gestiona los cada uno de los conceptos encontrados en el negocio como: usuarios, roles, departamentos y servicios clínicos o no, entre otros conceptos. |

Tabla 2.1. Actores del sistema

#### 2.5.3.2 Vista global de actores del sistema

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



Figura 2.2. Actores del sistema.

### 2.5.3.3 Diagramas de casos de uso

**Diagrama de caso de uso**

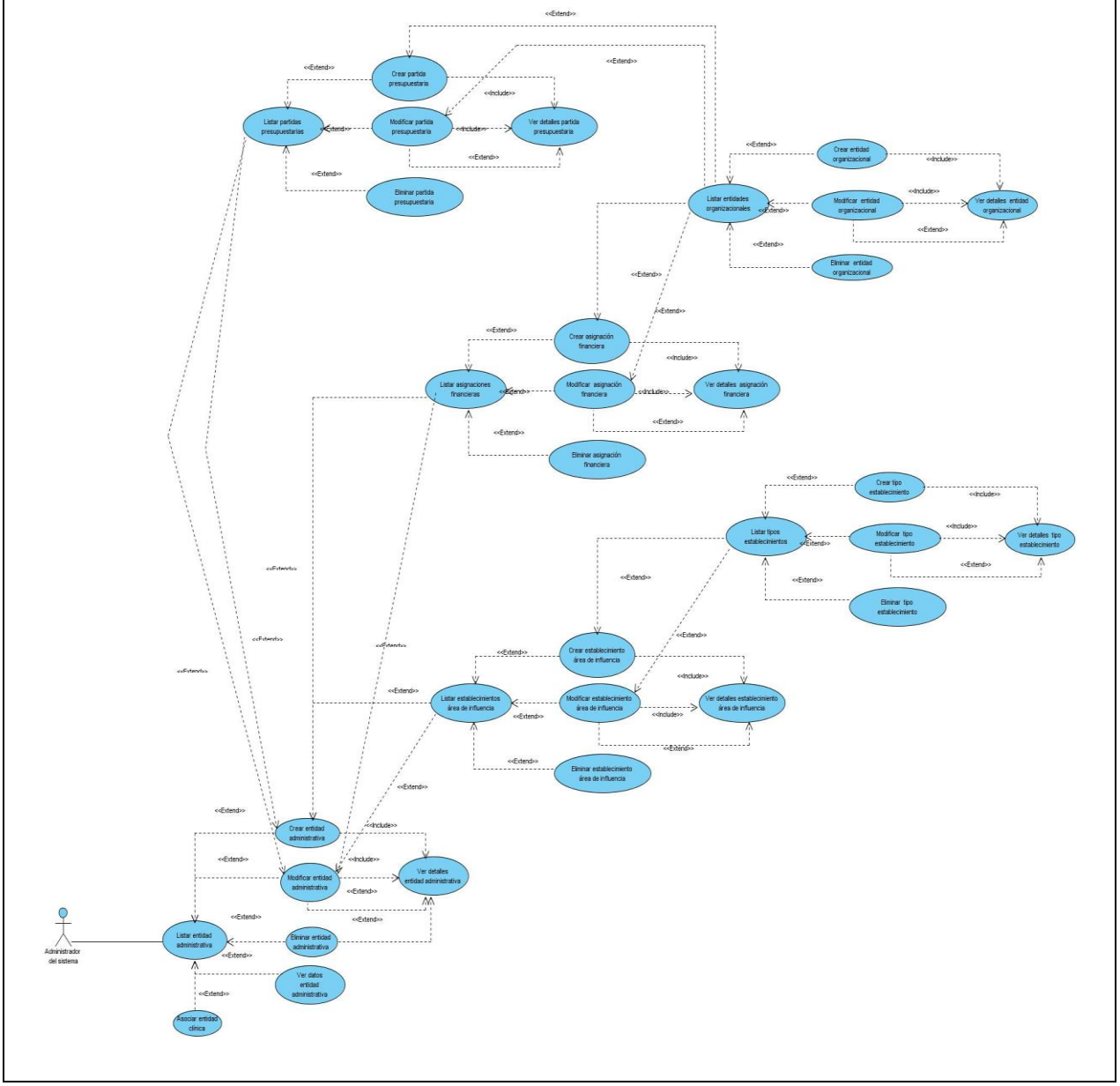


Figura 2.3. Diagrama de caso de uso del sistema – Gestionar entidad.

**Diagrama de caso de uso**

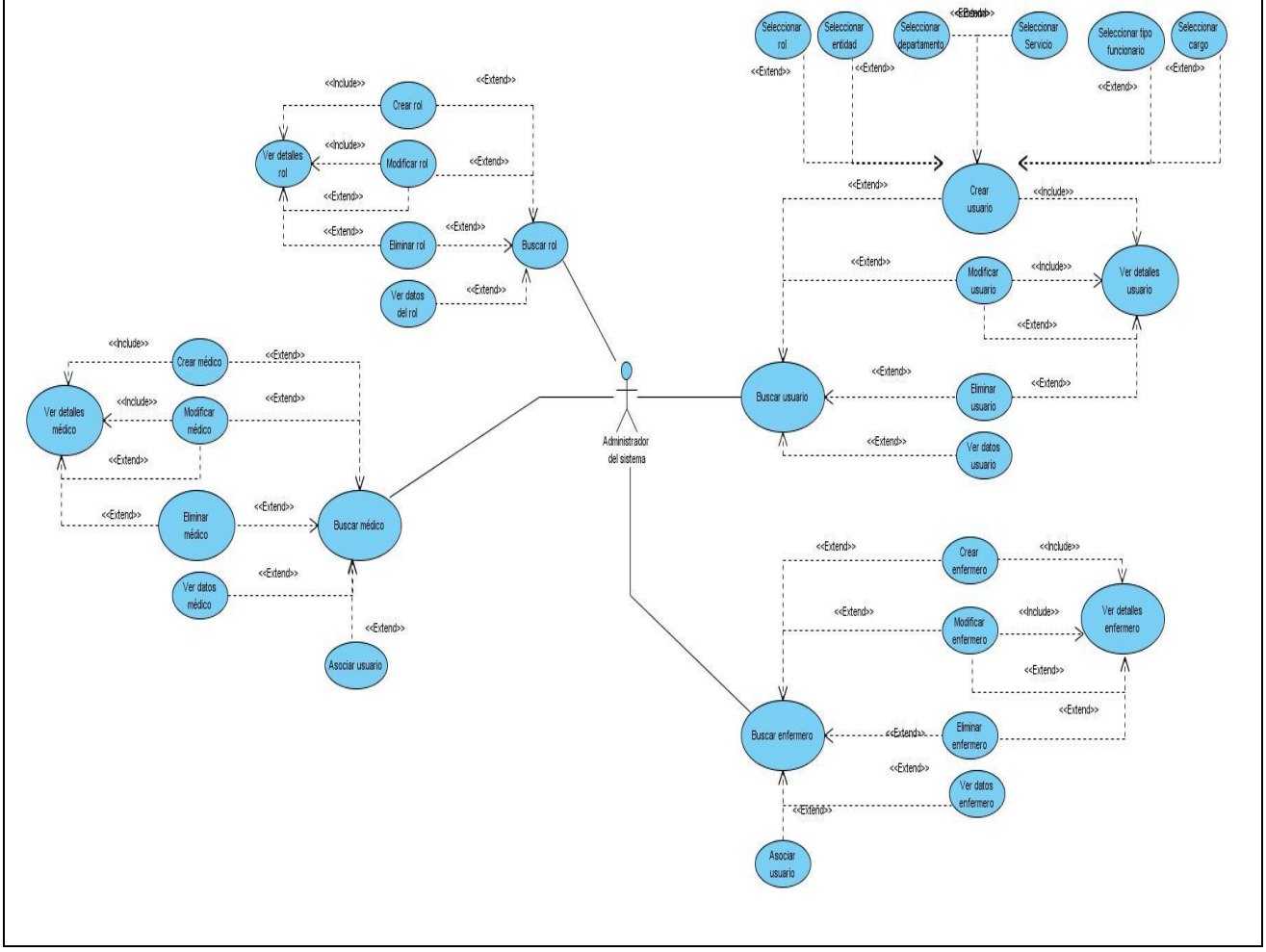


Figura 2.4. Diagrama de caso de uso del sistema – Gestionar usuarios y roles.

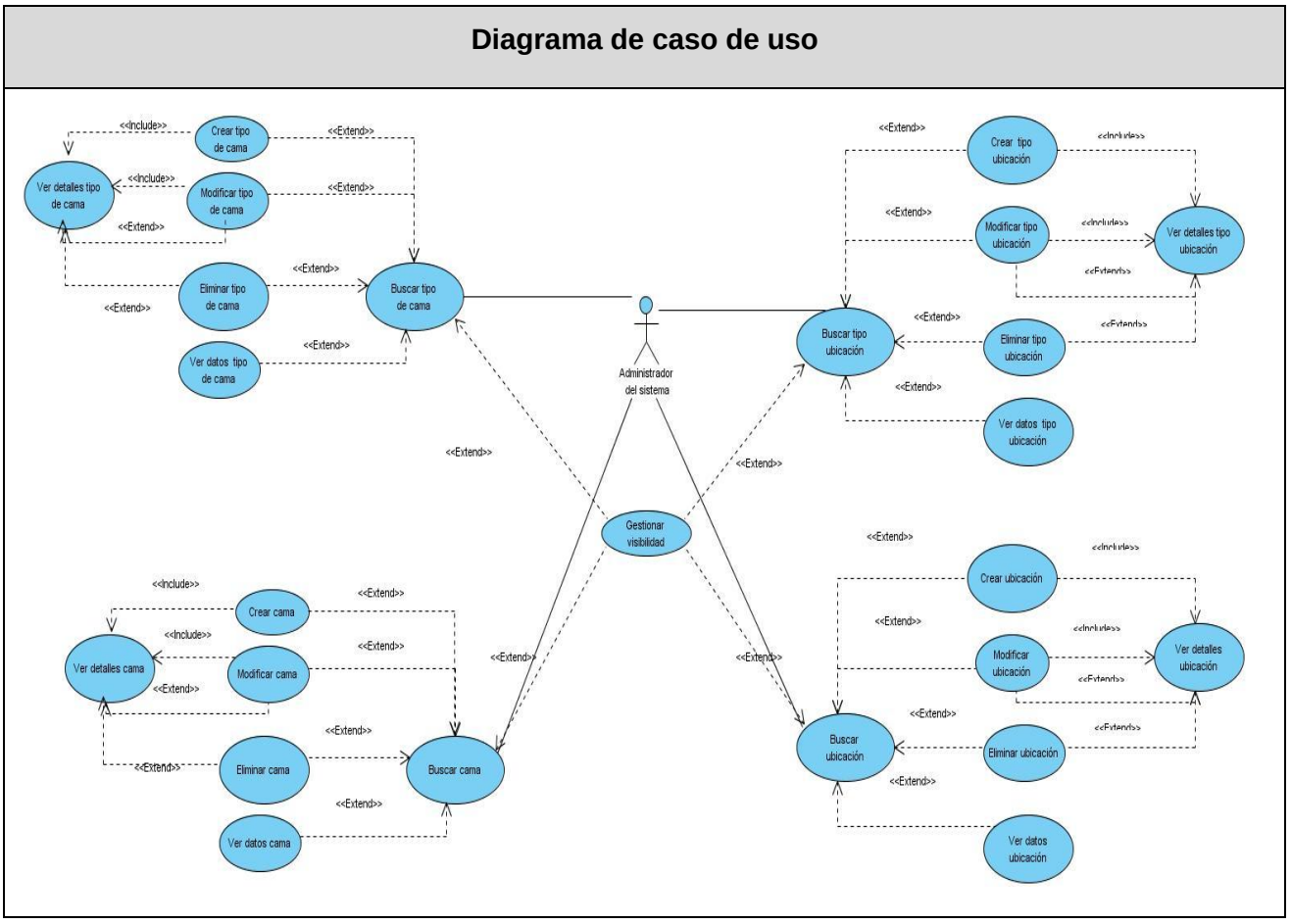


Figura 2.5. Diagrama de caso de uso del sistema – Ubicación, tipo de ubicación, cama y tipo de cama.

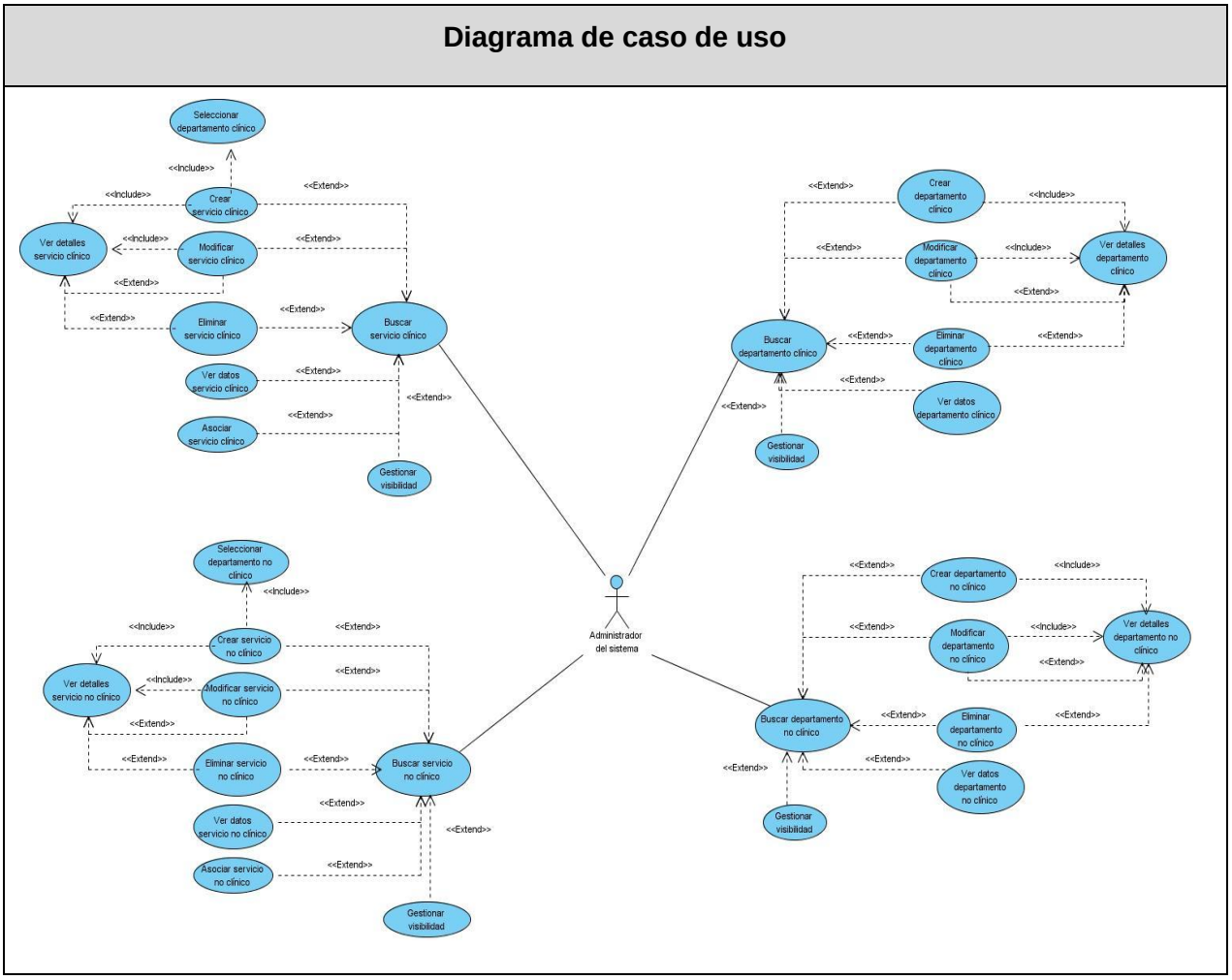


Figura 2.6. Diagrama de caso de uso del sistema – Gestionar servicios y departamentos.

#### 2.5.3.4 Descripción textual de los casos de uso

| | |
|---|---|
| **CASO DE USO:** | **Crear usuario** |
| **Propósito:** | Registrar un usuario en el sistema. |
| **Actores:** | Administrador del sistema |
| **Resumen:** | El caso de uso inicia cuando el actor accede a la opción Crear usuario el sistema brinda la posibilidad de introducir los datos para crear el usuario, el actor introduce los datos del usuario, el sistema crea el usuario, el caso de uso termina. |
| **Precondiciones:** | No existen |
| **Poscondiciones:** | Se creó un usuario. |
| **Referencia:** | RF – 1 |

Tabla 2.2. Registrar un usuario en el sistema

**Conclusiones**

Al finalizar el capítulo se logra detallar las funcionalidades que abarca el Módulo Configuración del Sistema de Información Hospitalaria alas HIS. Se hace una descripción de los principales problemas y la definición de los conceptos existentes. Como resultado de su análisis se define un modelo de dominio donde se representa la interacción entre los conceptos encontrados, además de establecer cada uno de los requisitos funcionales y no funcionales con los que debe contar el sistema para lograr una configuración centralizada.

# Capítulo 3: Análisis y diseño del sistema

En el presente capítulo se realiza una descripción minuciosa del sistema propuesto. Además se explica la arquitectura definida por el Departamento de Sistemas de Gestión Hospitalaria. Finalmente se expone la modelación de los artefactos necesarios que contribuyen a la implementación del sistema, mostrando para ello los principales diagramas de clases y secuencia del modelo de diseño.

## 3.1 Descripción de la arquitectura

Las técnicas metodológicas desarrolladas con el fin de facilitar la programación se engloban dentro de la llamada Arquitectura de Software o Arquitectura lógica. Se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarnos en el desarrollo de software dentro de un sistema informático. Dichos patrones serán capaces de aportar elementos idóneos para tomar buenas decisiones. Además, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto.

Para el desarrollo del sistema y teniendo en cuenta las herramientas, tecnologías y metodologías propuestas, se define como parte de la línea base de la Arquitectura la implementación del patrón de diseño Modelo Vista Controlador. Este patrón es muy usado en aplicaciones web. Permite la separación de los datos de una aplicación, la interfaz de usuario y la lógica de control, en tres componentes distintos: el modelo, donde se encuentran los datos y las reglas del negocio; la vista, que muestra la información del modelo al usuario; y el controlador, que gestiona las entradas del usuario.

Con este patrón se logra realizar un diseño que desacople la vista del modelo y permita la reusabilidad de los componentes. De esta forma, las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos. Esto posibilita que los componentes del módulo se agrupen en tres capas fundamentales: presentación, negocio y acceso a datos. Brinda mejor organización según la función que realizan, permitiendo que en un momento determinado un elemento de una capa pueda ser modificado o sustituido completamente causando el mínimo de alteraciones en otro elemento que lo utilice.

La capa de presentación está conformada principalmente por páginas XHTML. Estas están compuestas por formularios que mediante controles JSF, Seam UI y RichFaces obtienen y validan los datos que el

usuario provee en cada una de las operaciones que realiza. El uso de estos componentes enriquece el diseño de la interfaz de usuario. Además al realizar el envío y carga de datos mediante los componentes ajax4sf se logra un efecto más agradable y natural al interactuar con el sistema.

La capa de negocio está constituida por clases controladoras que se encargan de definir la lógica del negocio del módulo, así como del manejo y validación de los datos capturados en la capa de presentación. A estas clases, mediante anotaciones que provee Seam como marco de trabajo, se les puede especificar el contexto en que se encuentran, ya sea conversacional, evento, página, entre otros, los que definen el estado de los datos y las entidades que manejan. En esta capa se manejan también las reglas del negocio, haciendo uso de Drools para proporcionarle mayor dinamismo y funcionalidad al sistema.

Finalmente, la capa de acceso a datos es la encargada de cargar, modificar, eliminar y persistir la información existente en la base de datos una vez validada. Todo este proceso tiene lugar gracias al uso de componentes Hibernate por los que se encuentra constituida la capa. Dichos componentes logran abstraer al desarrollador del gestor de base de datos utilizado a través del mapeo de tablas. Esto permite llevar las consultas a un lenguaje de objetos. (25)

## 3.2 Análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados. Estrategias de integración

El Módulo Configuración no constituye un servicio o responde a un área específica de las instituciones hospitalarias como puede ser admisión, hospitalización o consulta externa. Este se encarga de gestionar todos los datos que son utilizados en la entidad y manejados por cada uno de los módulos que sí responden a áreas específicas como las citadas anteriormente. Además tiene la responsabilidad de controlar la gestión de funcionalidades generales como la seguridad del sistema, entre otras.

Fue diseñado para satisfacer las necesidades funcionales y de gestión del sistema en general, trabajando ante todo de forma independiente al resto de los módulos. No se cuenta con procesos definidos pero si con un considerable número de operaciones, las cuales van desde la seguridad de la aplicación hasta gestionar cada uno de los nomencladores existentes en la base de datos, los que

responden a valores estándares establecidos en la esfera de la salud. Actualmente no existe una versión estable del sistema mediante la cual se pudieran obtener problemas y soluciones de estos.

Como resultado de una completa gestión de la configuración se propone un módulo capaz de brindar un conjunto de recursos reutilizables. Ejemplos de estos recursos lo constituyen reportes del estado general de las entidades a las que responda el sistema, donde sus usuarios pueden detallar una vista completa de cada valor circulante. Se brindan a otros módulos un conjunto de funcionalidades como es la Bitácora para llevar a cabo el registro de las acciones que realiza el como son: actualizar, modificar, ver, liberar, aceptar y crear. Funcionalidades de seguridad de forma central, pero logrando un nivel alto de especificación, ya sea por el rol que tenga el usuario, por módulos a los que puede acceder, por funcionalidades a mostrar o recursos disponibles para estos.

## 3.3 Modelo de diseño

Es un modelo de objetos que describe la realización física de los casos de uso. Se centra en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. (26) En el diseño se modela el sistema de forma tal que sea capaz de soportar todos los requisitos, tanto los funcionales como los no funcionales, así como las restricciones que se le suponen. Entre los artefactos del Modelo de diseño se encuentran: descripción de la arquitectura y realización de casos de uso.

## 3.4 Patrones de diseño

> *"Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma"*
>
> *Christopher Alexander*

El patrón es un esquema de solución que se aplica a un tipo de problema, esta aplicación del patrón no es mecánica, sino que requiere de adaptación y matices. Por ello, plantea Alexander Christopher que los numerosos usos de un patrón no se repiten dos veces de la misma forma. Con la aplicación de los

patrones de diseños se logra ahorrar tiempo y mejorar el software haciéndolo más eficiente, dinámico y seguro.

En el proceso de implementación se hace necesario el uso de patrones que permitan aumentar la reutilización de código así como el correcto uso de sus instancias. Con el uso de Hibernate se logra una abstracción del gestor de bases de datos utilizados y la persistencia de las entidades del sistema obtenidas mediante el proceso de mapeo.

**Abstract Factory** es un patrón de diseño que se encuentra en Hibernate y es implementado por el componente *EntityManagerFactory,* el cual figura entre los principales de la Arquitectura JPA (Java Persistence API). Permite el acceso a la base de datos y es el encargado de crear objetos *EntityManager* cuando es inyectado en algún contexto de la aplicación. Los objetos EntityManager constituyen la interfaz principal de JPA utilizada para la persistencia de las aplicaciones. Cada instancia puede realizar operaciones como inserción, lectura, modificación y eliminación (CRUD por sus siglas en inglés) sobre un conjunto de objetos persistentes.

Hibernate implementa también el patrón Active Record el cual da lugar a una clase a partir de una fila de la base de datos, creando una asociación entre filas únicas de la base de datos con objetos del lenguaje de programación del que se esté haciendo uso. También implementa los patrones Identity field, Foreign Key Mapping y Association Table Mapping**.** El primero se relaciona con la forma de cargar los distintos objetos en memoria, consiste básicamente en la generación de un único ID de forma tal que se defina un valor por el cual se identifique una entidad. El segundo es utilizado para el mapeo de relaciones uno a muchos, y el tercero para mapear las relaciones muchos a muchos.

Hibernate implementa además los patrones de comportamiento Identity map y Lazy Load. El primero es utilizado para evitar tener en memoria dos representaciones distintas del mismo objeto en una transacción de negocio; funcionando como una caché de objetos de negocio mientas que el segundo resuelve problemas de carga desmesurada y dependencias circulares, es decir, mantiene en memoria solo los datos de los objetos que se invoquen en cada momento.

Uno de los patrones más importantes que implementa *Hibernate* es el patrón *Query object. Este está a* encargo de interpretar la estructura de objetos y traducirlos a consultas (querys) SQL. Haciendo uso del mismo se pueden crear consultas referenciando a las clases y sus campos, en lugar de tablas y columnas independientemente del esquema en que puedan estar.

Se hace uso de los patrones *GRASP* que describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Se le asignó una responsabilidad a cada clase que tiene la información necesaria para el completo cumplimiento de la asignación, así como la creación de objetos (instancias) de otras clases, atendiendo a la responsabilidad otorgada, dando lugar a los patrones *Experto* y *Creador***.**

Otro de los patrones utilizados es el de *Bajo acoplamiento* y *Alta cohesión*. Su aplicación se pone de manifiesto donde la asignación de responsabilidades se realiza permitiendo la colaboración entre las clases, sin afectar el proceso de reutilización. También lo constituyen ejemplos de su uso la creación de clases controladoras, que posibilitó realizar las operaciones del sistema las cuales reflejan los procesos del negocio o del dominio, factibles de manejar en otras capas, como la de interfaz o de presentación.

## 3.5 Diagramas de clases del diseño

El diseño de Software representa un papel importante en el desarrollo de aplicaciones informáticas y permite producir varios modelos del sistema o producto que se va a diseñar. En el diseño se modela el sistema de forma tal que sea capaz de soportar todos los requisitos, ya sean los no funcionales como los funcionales. Al unísono se va definiendo arquitectura. En este modelo, las clases del diseño y sus objetos dan lugar a los casos de uso, mediante los que produce el diagrama de clases del diseño.

Una *clase de diseño* es aquella suficientemente detallada que sirve como base para generar código fuente o lo que es lo mismo, es aquella cuya especificación es completa hasta un nivel que se pueda implementar. (27)

En los *diagramas de clases de diseño* se expone un conjunto de interfaces, colaboraciones y sus relaciones. Se utilizan para modelar la vista de diseño estática de un sistema. Estos son de gran importancia, ya que permiten visualizar, especificar y documentar modelos estructurales. Los diagramas de clases de diseño forman parte de las realizaciones de casos de usos.

Uno de los factores esenciales para la realización de los casos de uso son los diagramas de interacción que muestran cómo interactúan conjuntos de objetos y sus relaciones, incluyendo los mensajes que puedan ser realizados entre ellos. Son importantes para modelar los aspectos dinámicos de un sistema y

para construir sistemas ejecutables a través de ingeniería hacia adelante e ingeniería inversa. Los diagramas de interacción están conformados por los diagramas de secuencia y los diagramas de colaboración, donde generalmente se recomienda utilizar los de colaboración en el análisis y el de secuencia para el diseño. Un diagrama de secuencia enfatiza el orden de tiempo de los mensajes.

Con la definición de los principales aspectos a tener en cuenta para la realización del modelo de diseño, se establece una estructura de paquetes dividida en fragmentos manejables para su posterior implementación. Existe una relación entre los paquetes mediante los que se establecen dependencias entre las distintas clases que lo contienen. El proceso de empaquetamiento se realiza teniendo en cuenta un solo criterio y es el referente a los tipos de entidades que se pueden encontrar.

A continuación se representa el diagrama de paquetes del sistema propuesto:

Diagrama de paquetes del sistema

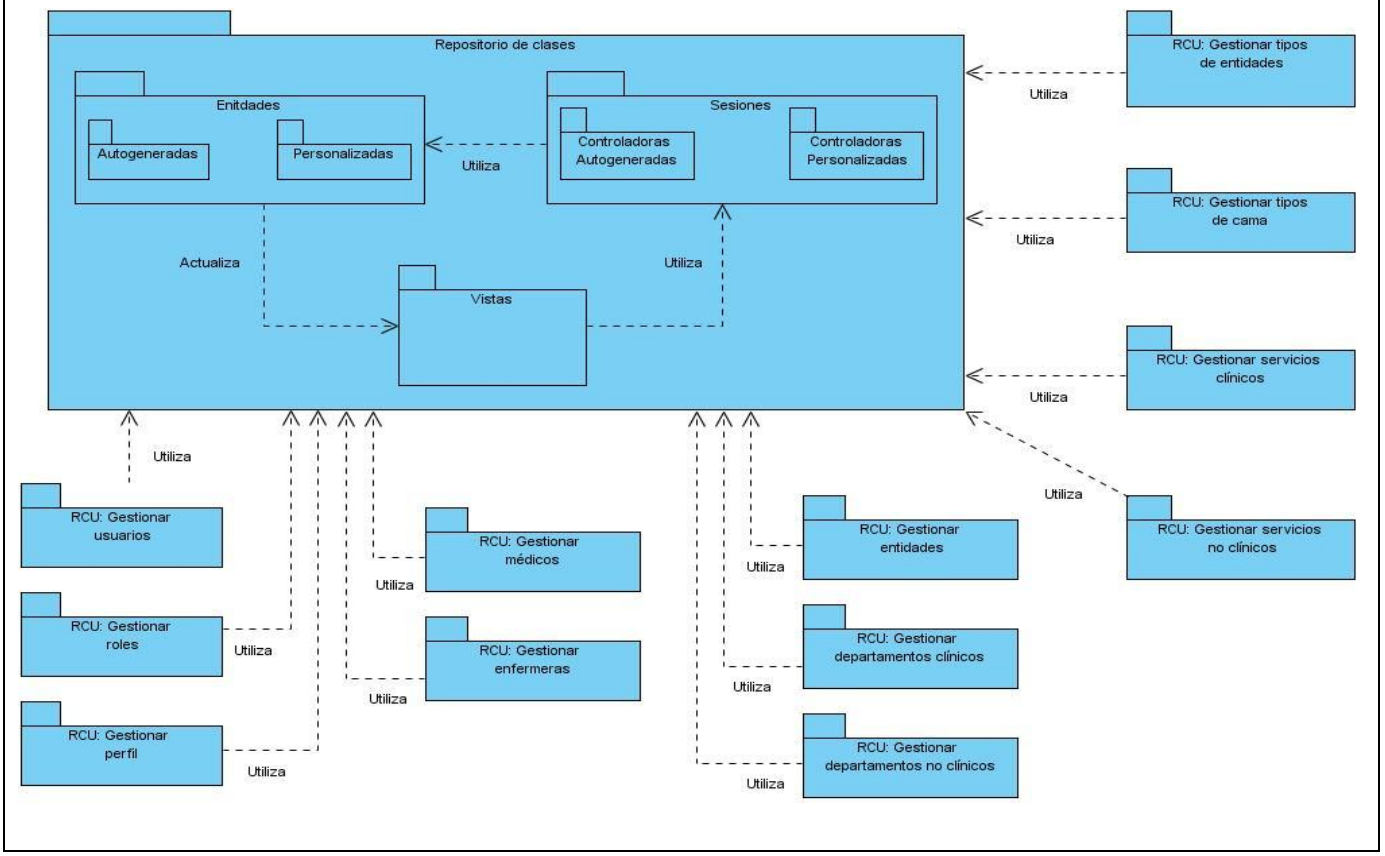


Figura 3.1. Modelo de diseño

En el modelo de diseño el paquete *Repositorio de clases,* está compuesto por las clases del diseño de acuerdo con la tecnología que se usará en la implementación de la aplicación. Este engloba dos paquetes *Entidades y Sesiones.* En el primero se aprecia el paquete de *Autogeneradas*, el cual contiene a todas las entidades que se generaron desde la base de datos mediante el proceso de mapeo, mientras que el paquete *Personalizadas*, cuenta con las especializaciones que se le realizarán a las clases autogeneradas, ya sea haciendo uso de relaciones de composición o herencia. El segundo cuenta con las clases *Controladoras autogeneradas,* clases generadas también mediante el mapeo de la base de datos y las clases *Controladoras personalizadas,* de igual forma con las especializaciones de las clases autogeneradas haciendo uso de relaciones de composición o herencia.

## 3.6 Diagramas de clases de diseño

Diagrama de clase de diseño

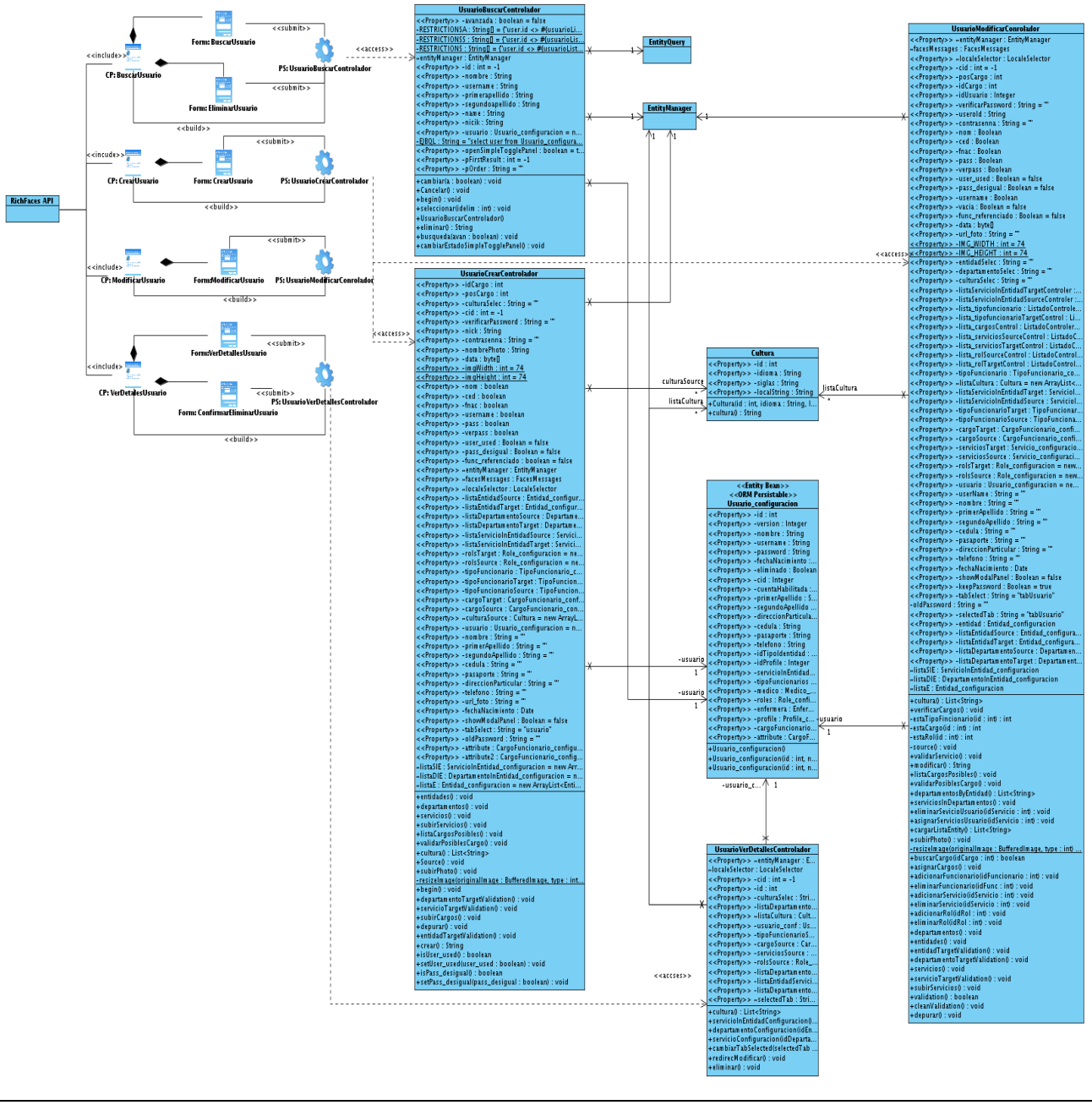


Figura 3.2. DCD Gestionar usuario

## 3.7 Diagramas de secuencia

Diagrama de secuencia

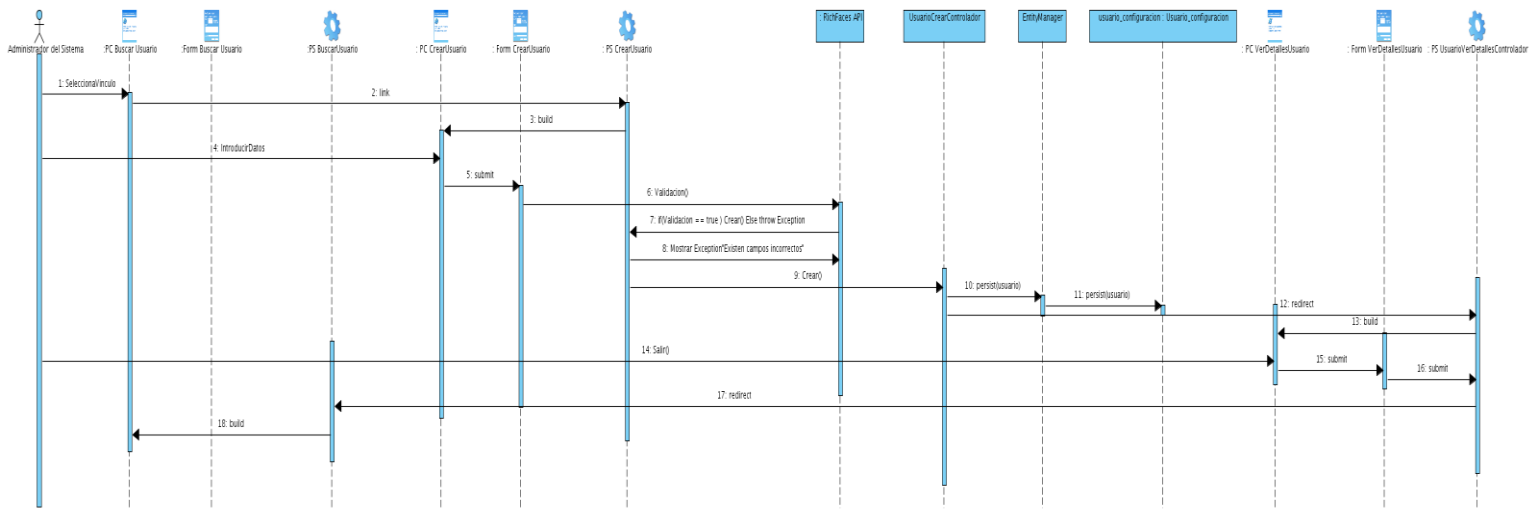


Figura 3.3. DS Crear usuario

### 3.8 Descripción de las clases y sus atributos

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

| **Nombre:** | UsuarioCrearControlador |
|---|---|
| **Tipo de clase:** | Controladora |
| **Atributo** | **Tipo** |
| entityManager | EntityManager |
| facesMessages; | FacesMessages |
| localeSelector | LocaleSelector |
| usuario | Usuario_configuracion |
| rolsSource | List<Role_configuracion> |
| tipoFuncionarioSource | List<TipoFuncionario_configuracion> |
| cargoSource | List<CargoFuncionario_configuracion> |
| listaServicioInEntidadSource | List<ServicioInEntidad_configuracion> |
| listaDepartamentoSource | List<DepartamentoInEntidad_configuracion> |
| listaServicioInEntidadSource | List<ServicioInEntidad_configuracion> |
| culturaSource | List<Cultura> |
| perfil | Profile_configuracion |
| **Para cada responsabilidad**: | |

| | |
|---|---|
| Nombre: | begin() |
| Descripción: | Crea una conversación. |
| Nombre: | subirPhoto() |
| Descripción: | Copiar en el servidor el fichero de la foto del usuario desde la ubicación dada. |
| Nombre: | entidades() |
| Descripción: | Carga la lista de entidades donde interactuará el usuario. |
| Nombre: | departamentos() |
| Descripción: | Carga la lista de departamentos existentes en las entidades seleccionadas. |
| Nombre: | servicios() |
| Descripción: | Carga la lista de servicios de los departamentos ya seleccionados que pueden ser asignados al usuario. |
| Nombre: | cultura() |
| Descripción: | Devuelve la lista de culturas a definidas en la aplicación. |
| Nombre: | listaCargosPosibles() |
| Descripción: | Mediante un llamado al método se actualizan los valores de los servicios seleccionados. |
| Nombre: | crear() |
| Descripción: | Persiste en la base de datos todos los valores entrados sobre el usuario a crear. |

Tabla 3.1. Descripción textual clase UsuarioCrearControlador

**Conclusiones**

Una vez finalizado el estudio del análisis y diseño del sistema a desarrollar se define una arquitectura a utilizar. Además se identifican las principales clases controladoras que deben ser definidas para un correcto funcionamiento de la aplicación. De cada una de estas clases se realiza una descripción de los atributos y los métodos que deben contener. Con ello se logra brindar al desarrollador una idea más clara de lo que se debe implementar, haciendo uso de los diagramas de clases del diseño y los diagramas de secuencia por escenarios pertenecientes a las principales funcionalidades.

# Capítulo 4: Implementación

Este capítulo contiene las principales características de la implementación del Sistema. Muestra los principales componentes y sus relaciones, haciendo uso del Diagrama de Componentes. También se exponen la estrategia de seguridad, los estándares de codificación así como los tratamientos de errores aplicados.

## 4.1 Modelo de datos

Los modelos de datos aportan la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos. De igual forma brindan la base formal para las herramientas y técnicas empleadas en el desarrollo y uso de sistemas de información. En general un modelo de datos es la estructura o representación física de las tablas de la base de datos (Figura 4.1).

Modelo de datos

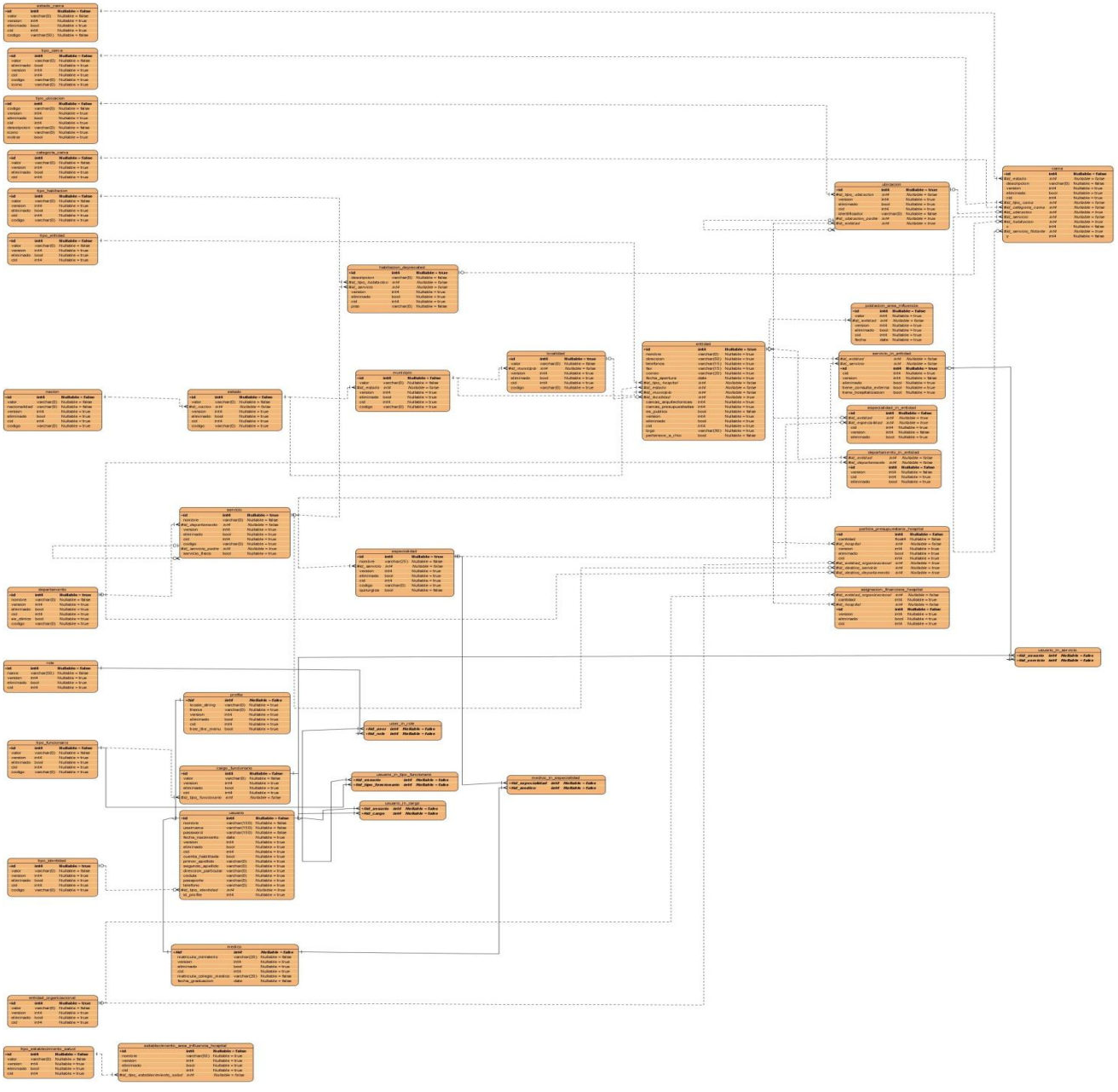

Figura 4.1. Modelo de datos

### 4.1.1. Descripción de las tablas

| Nombre | Documentación |
|---|---|
| entidad | Almacena los datos de las entidades hospitalarias, ya sean las controladas por el sistema como las del área de influencia poblacional. |
| usuario | Almacena todos los usuarios que interactuarán con el sistema. |

Tabla 4.1. Tablas del modelo de datos

**Campos comunes**

| Nombre | Tipo de dato | PK/FK | Nulo | Documentación |
|---|---|---|---|---|
| id | integer | PK | No | Identificador. |
| version | integer | | Si | Campo para controlar la concurrencia optimista. |
| eliminado | booelan | | No | Campo para eliminación lógica. |
| cid | integer | | Si | Campo para tracking de los cambios en la bitácora. |

Tabla 4.2. Campos comunes para todas las tablas

**entidad**

| Nombre | Valor |
|---|---|
| Clase mapeada | Entidad |

| | |
|---|---|
| Esquema | publico |
| Nombre llave primaria | entidad_pkey |

Tabla 4.3. Tabla entidad

**Campos**

| Nombre | Tipo de dato | Nulo | Documentación |
|---|---|---|---|
| nombre | varchar | Si | Nombre de la entidad. |
| direccion | varchar(50) | No | Dirección de la entidad. |
| telefonos | varchar(15) | No | Teléfonos de la entidad. |
| fax | varchar(15) | No | Fax de la entidad. |
| correo | varchar(20) | No | Correo electrónico de la entidad. |
| fecha_apertura | date | No | Fecha de apertura de la entidad. |
| id_tipo_hospital | integer | Si | Identificador del tipo de entidad. |
| id_estado | integer | Si | Identificador del estado donde se encuentra la entidad. |
| id_municipio | integer | Si | Identificador del municipio donde se encuentra la entidad. |
| id_localidad | integer | No | Identificador de la localidad donde se encuentra la entidad. |

| | | | |
|---|---|---|---|
| camas_arquitectonicas | integer | No | Camas arquitectónicas que tiene la entidad. |
| camas_presupuestadas | integer | No | Camas presupuestadas que tiene la entidad. |
| es_publico | boolean | No | Establece si el hospital es público o no. |
| logo | varchar(40) | No | Nombre del logotipo de la entidad. |
| pertenece_a_rhio | boolean | Si | Establece si la entidad es asociada o no al sistema. |
| id_categoria_entidad | integer | No | Identificador de categoría de la entidad. |

Tabla 4.4. Campos de la tabla entidad

**usuario**

| Nombre | Valor |
|---|---|
| Clase mapeada | Usuario |
| Esquema | publico |
| Nombre llave primaria | usuario_pkey |

Tabla 4.5. Tabla usuario

**Campos**

| Nombre | Tipo de dato | Nulo | Documentación |
|---|---|---|---|
| nombre | varchar(150) | Si | Nombre del usuario. |

| | | | |
|---|---|---|---|
| username | varchar(150) | Si | Nombre de usuario para acceder al sistema. |
| password | varchar(159) | Si | Contraseña del usuario para acceder al sistema. |
| fecha_nacimiento | date | No | Fecha de nacimiento del usuario. |
| cuenta_habilitada | boolean | Si | Establece si el usuario está habilitado en el sistema. |
| primer_apellido | varchar | No | Primer apellido del sistema. |
| segundo_apellido | varchar | No | Segundo apellido del sistema. |
| dirección_particular | varchar | No | Dirección particular del usuario. |
| cedula | varchar | No | Cédula del usuario. |
| pasaporte | varchar | No | Pasaporte del usuario. |
| telefono | varchar | No | Teléfono del usuario. |
| id_tipo_entidad | Integer | No | Identificador del tipo de entidad. |
| id_profile | Integer | No | Identificador del perfil del usuario. |

Tabla 4.6. Campos de la tabla usuario

## 4.2 Modelo de clases persistentes

El Modelo de clases persistentes no es más que el mapeo de las tablas de la base de datos. Dicho mapeo una vez creadas las entidades que representan las tablas en la base de datos permite manejar la lógica del negocio a partir de la lógica del Modelo de diseño. (Figura 4.2)

Modelo de datos persistentes

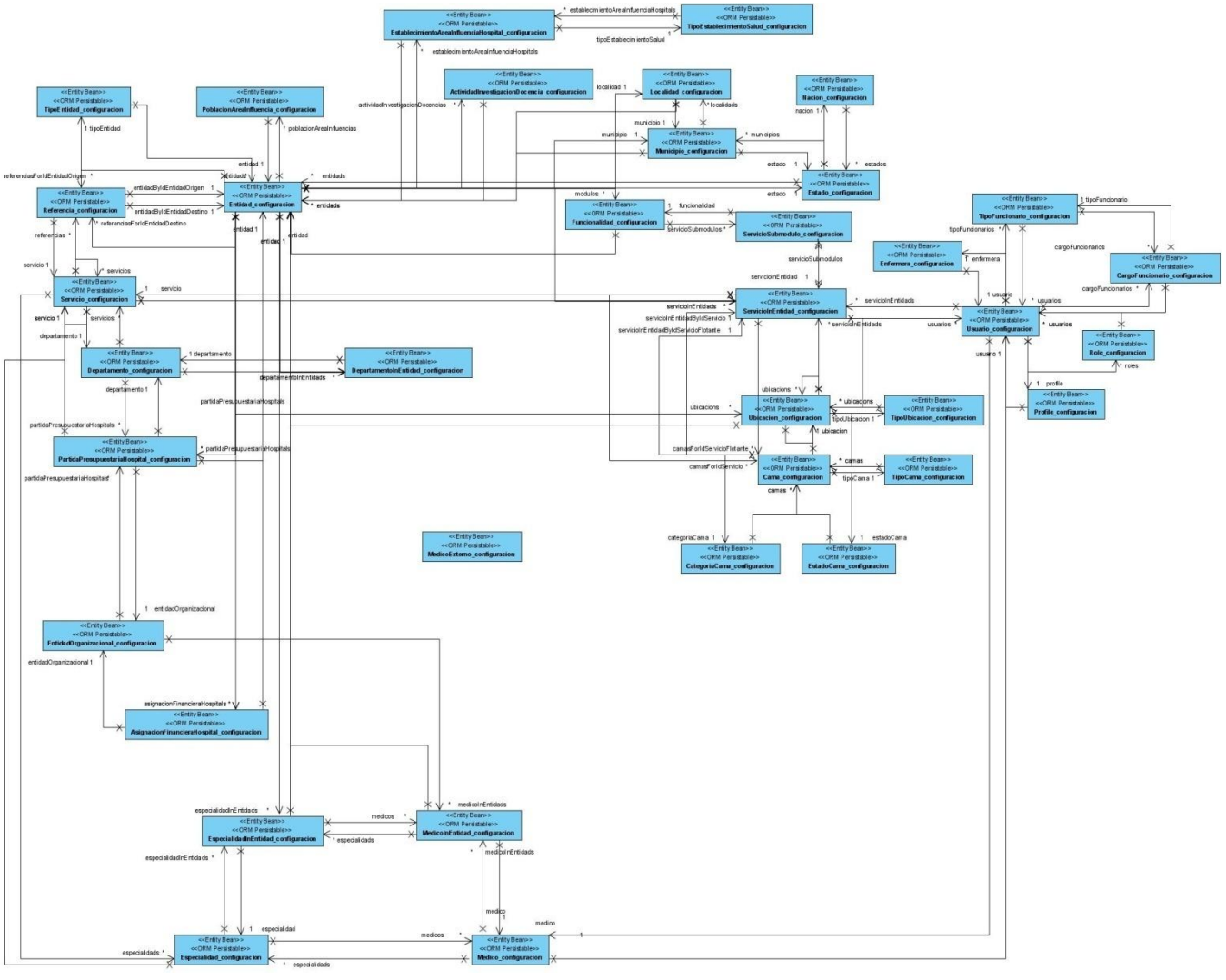

Figura 4.2. Modelo de datos persistentes

### 4.3 Modelo de despliegue

El Modelo de despliegue es utilizado para definir la estructuración, descripción y relaciones físicas del hardware concebido para el despliegue del sistema. (Figura 4.3)

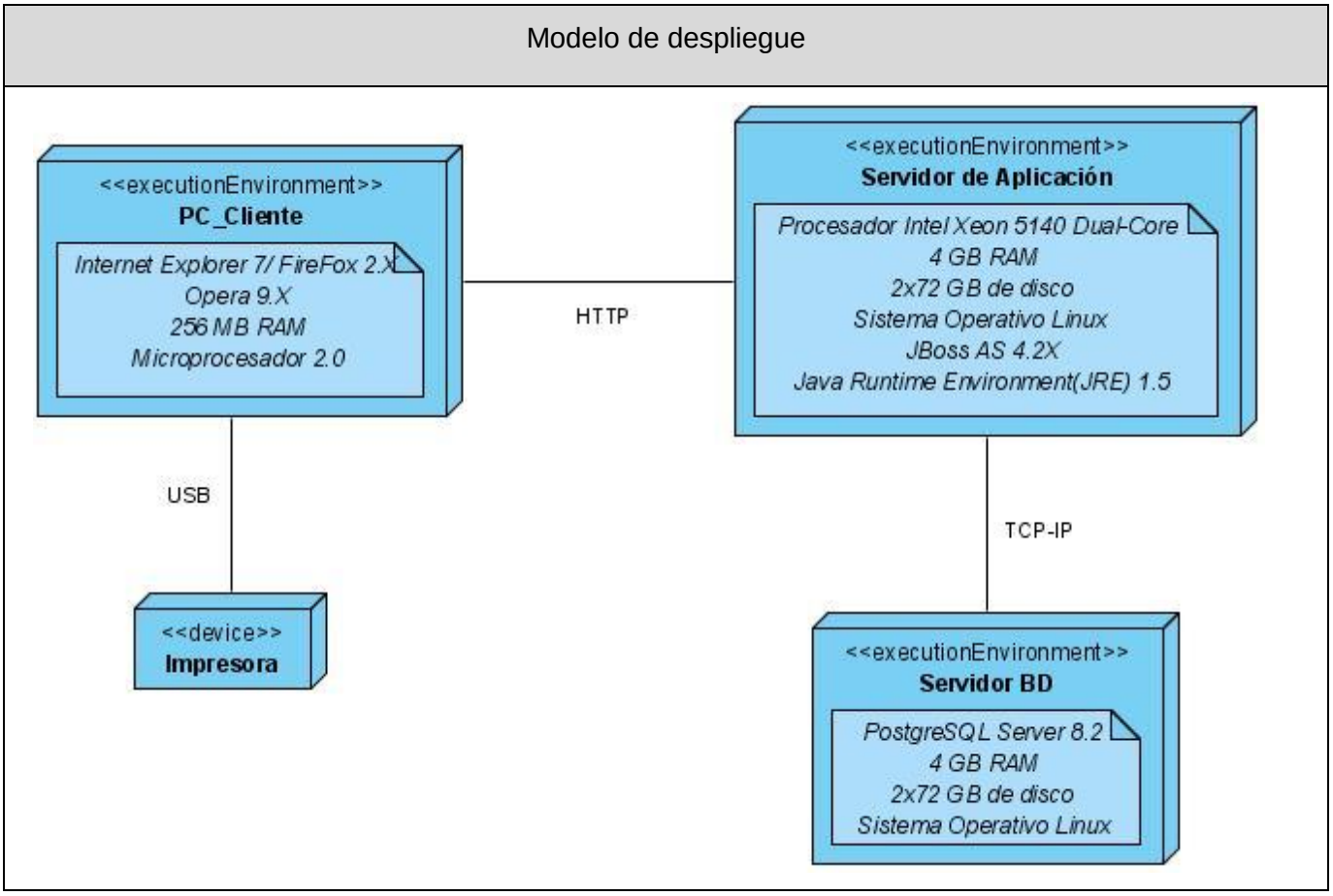


Figura 4.3. Modelo de despliegue

### 4.4 Diagrama de componentes

Este diagrama describe la estructura de los componentes del sistema agrupados por paquetes lógicos.

Diagrama de componentes

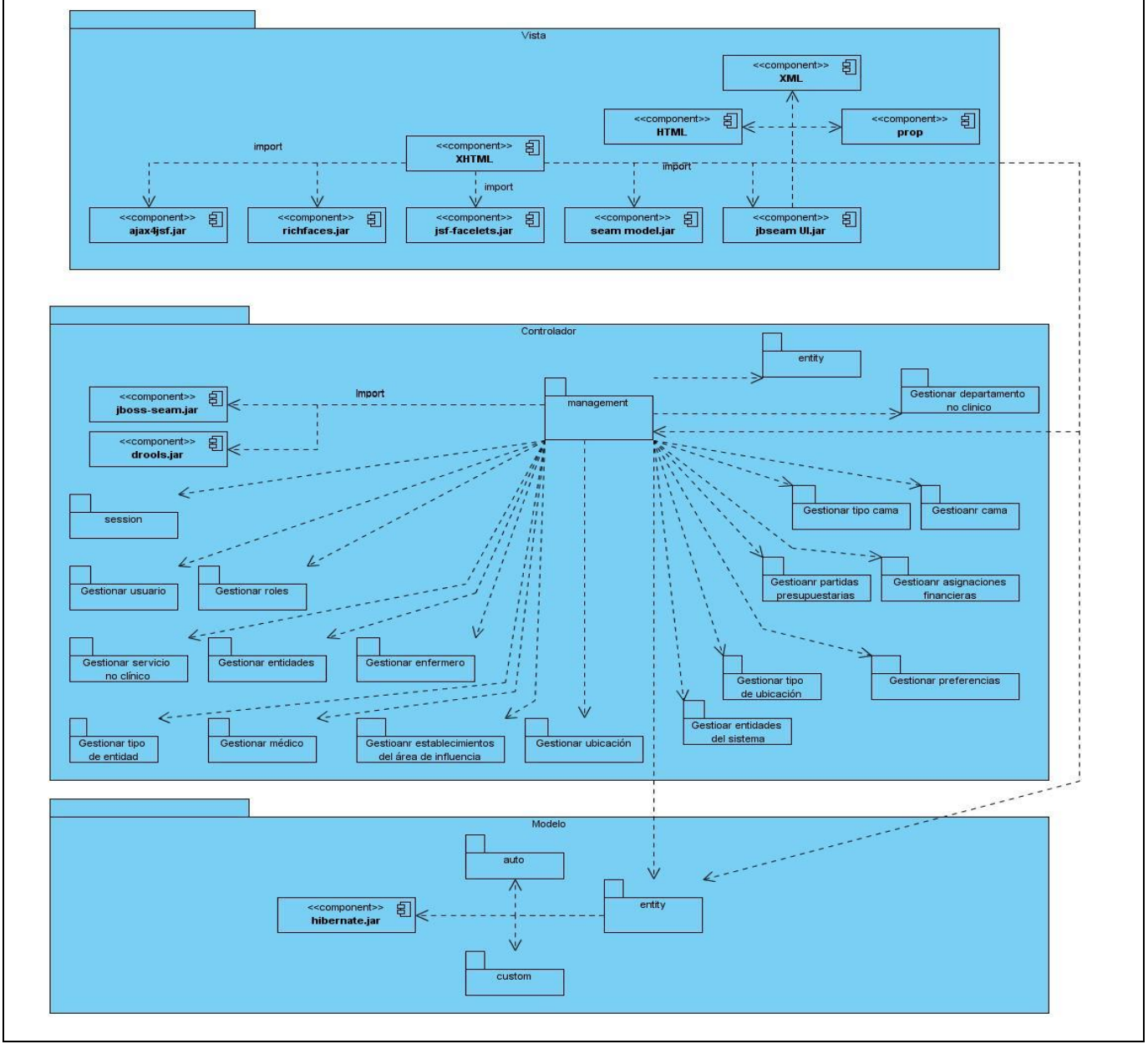


Figura 4.4. Diagrama de componentes

### 4.5 Tratamiento de errores

Una excepción es un evento que ocurre durante la ejecución del programa que interrumpe el flujo normal de las sentencias. Las excepciones son el mecanismo recomendado para la propagación de errores que se produzcan durante la ejecución de las aplicaciones. Cuando ocurre “un error” dentro de un método Java, automáticamente se crea un objeto ‘Exception’ el cual es tratado en el sistema de ejecución. Este objeto contiene información sobre la excepción, incluyendo su tipo y el estado del programa.

En el HIS se propone el tratamiento de excepciones principalmente en las regiones críticas de código, o sea, donde los datos son insertados o modificados en la base de datos, así como en el proceso de validación. El control de la navegación, en caso de ocurrir una excepción que implique una redirección, se maneja mediante los ‘.pages.xml’, los mismos encargan de capturar globalmente las excepciones y ejecutar las instrucciones determinadas. Para el control de las demás excepciones es utilizado el componente FacesMessages del framework Seam el cual se encarga de mostrar los mensajes que se manejan a través del objeto facesMessages inyectado en las clases controladoras. El mismo trata los mensajes por tipo (error, alerta y notificación).

### 4.6 Seguridad

A la hora de desarrollar cualquier Sistema de Gestión de Información la seguridad es un tema delicado el cual toma mayor relevancia cuando se gestiona información relacionada con la salud de las personas.

Para garantizar la seguridad del Módulo Configuración toda la autorización está basada en reglas que restringen el acceso a directorios, páginas, controles, opciones del menú y servicios del negocio. Cada regla es creada en el mismo módulo atendiendo a su función en el sistema (Gestión de la configuración). A su vez ninguna de estas están contenidas en el código de la aplicación, sino en ficheros (no compilados) dotando a la aplicación de mayor dinamismo al no requerir una recompilación en caso que cambie alguna. Esto puede llevarse a cabo por la posibilidad de integración con el motor de reglas JBoss Rules, que brinda el Framework de Seguridad de JBoss Seam.

Se realiza además el control a nivel de usuarios y contraseñas, garantizando el acceso sólo a los niveles establecidos de acuerdo con la función que realiza cada usuario. Las contraseñas sólo pueden ser

cambiadas por el propio usuario o por el administrador del sistema. También se validan todos los ids o llaves enviadas por URL a fin de asegurar que sean correctos.

Todas las actividades realizadas por los usuarios quedan registradas a cada momento en una especie de bitácora, almacenándose la fecha, la hora, el usuario y la actividad que realizó el mismo.

**Conclusiones**

Una vez concluido este último capítulo se pudo realizar la implementación del sistema en términos de componentes. Por lo que se proporciona una solución a los requisitos funcionales y no funcionales especificados anteriormente. Para ello se realizó la descripción de la estructura de tablas sobre las que se establece el modelo de datos del sistema, así como sus atributos y relaciones. Además se estructuran las clases de diseño en paquetes y subsistemas de implementación.

# Conclusiones

Con el desarrollo del Módulo Configuración del Sistema de Información Hospitalaria alas HIS se arribó a las siguientes conclusiones:

- De los HIS analizados a nivel mundial solo uno cuenta con funcionalidades para las configuraciones generales del sistema y otro con la gestión de procesos multientidad, pero estos no cumplen e integran el resto de los requerimientos deseados para el Sistema de Información Hospitalaria alas HIS.
- La identificación de los principales conceptos asociados al dominio de la investigación, así como el establecimiento de las interdependencias entre estos, posibilitó un mejor entendimiento del mismo y constituyó el punto de partida para una adecuada identificación de los requerimientos funcionales del sistema a desarrollar.
- Las tecnologías, herramientas y patrones utilizados, permitieron la construcción de un sistema robusto y flexible, adaptable a diferentes entornos de despliegue.
- La utilización de pautas para el proceso de desarrollo, garantizó uniformidad y homogeneidad en los artefactos obtenidos a partir de este.
- La implementación de la gestión centralizada de los requerimientos de seguridad y administración del Sistema de Información Hospitalaria alas HIS, evita la multiplicidad de esfuerzos individuales en cada uno de los módulos, optimizando y perfeccionando su proceso de configuración general.

## Recomendaciones

Los autores recomiendan:

- Implementar una nueva versión de la bitácora del sistema que permita un mayor nivel de análisis de estos datos y búsquedas de información, así como la generación de reportes estadísticos a partir de las trazas almacenadas en esta.
- Implementar funcionalidades para mantenimiento de los datos almacenados así como para el monitoreo de estado del servidor desde el Módulo Configuración.

## Referencias bibliográficas

1. Florina, Gatica Lara y Fernando J., Fernández Puerto. *Sistema de Información Hospitalaria.* s.l. : UNAM - Facultad de Medicina.

2. *Manuales. SIGHO.* Céspedes, Paúl y Armando, Jesús. 2007.

3. SIGHO. *SIGHO.* [En línea] http://www.sigho.gob.mx/quees.htm.

4. CNT PACIENTES. *CNT PACIENTES.* [En línea] Hospital Information System Edition, 2009. http://www.cnt.com.co/pagina/index.asp?id=178.

5. CNT PACIENTES. *CNT PACIENTES.* [En línea] www.cnt.com.co.

6. XHosp Sistema integral para la administración hospitalaria . *XHosp Sistema integral para la administración hospitalaria .* [En línea] www.virtus.com.mx/xhosp/index.html.

7. Hosix –V. *Hosix –V.* [En línea] https://solutionfinder.microsoft.com.

8. SELENE de SIEMENS. *SELENE de SIEMENS.* [En línea] http://revistas.um.es/eglobal/article/view/458/429.

9. *Introducción a la Arquitectura de Software.* Billy Reynoso, Carlos. s.l. : Universidad de Buenos Aires.

10. Java Enterprise Edition(Versión Empresarial de Java). *Java Enterprise Edition(Versión Empresarial de Java).* [En línea] http://java.sun.com/javaee/, http://java.sun.com/j2ee/overview.html.

11. Allen, Dan**.** *Seam in Action.* 2008.

12. Jamae David y Johnson, Peter**.** *JBoss in Action.* 2009. 1933988029.

13. *UI Develoment with JaveServer Faces.*

14. Hat, Red. *RichFaces developer Guide.* 2007.

15. *Ajax4jsf Developer Guide.* 2007.

16. PostgreSQL. *PostgreSQL.* [En línea] http://www.postgresql.org/.

17. Bauer, Cristian y King, Gavin. *Java Persistence with Hibernate.* 2005.

18. Java Persistence API. *Java Persistence API.* [En línea] JPA. http://luchorondon.blogspot.com/2009/04/jpa-java- persistence-api.html.

19. Lenguaje Java. *Lenguaje Java.* [En línea] http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html.

20. Modelo cliente servidor. *Modelo cliente servidor.* [En línea] http://agamenon.uniandes.edu.co/~revista/articulos/cliser.html.

21. *Aplicaciones para la Web II Arquitectura MVC.*

22. *Ingeniería de Software II.* Jiménez Garzón, Darwin.

23. Breve guía. *Breve guía.* [En línea] Eclipse. http://www.slideshare.net/Benedeti/ide-eclipse-breve-gua-201399.

24. PgAdmin III. *PgAdmin III.* [En línea] Guía de Ubuntu. http://www.guia-ubuntu.org/index.php?title=PgAdmin_III.

25. *Alas-His_Documento de Arquitectura del Sistema.* Velázquez Carralero, Alejandro Mario. 2008. IH-SW-DR-091.

26. Clases de diseño. *Clases de diseño.* [En línea] Mayo de 2010 . http://www.taringa.net/posts/info/1492028/Workflow-De-Dise%C3%B1o,-Clases-de-Dise%C3%B1o,-Interfaces,-Diagrama.html.

27. Ochoa Ornelas, Raquel. Simulador gráfico de algoritmos de programación para computadoras. *Simulador gráfico de algoritmos de programación para computadoras.* [En línea] http://digeset.ucol.mx/tesis_posgrado/Pdf/Raquel%20Ochoa%20Ornel.

## Bibliografía

Lara., Florina Gatica. Sistema de Información Hospitalaria – UNAM – Facultad de Medicina.

*Manuales. SIGHO.* Céspedes, Paúl y Armando, Jesús. 2007.

SIGHO. *SIGHO.* [En línea] http://www.sigho.gob.mx/quees.htm.

PACIENTES. *CNT PACIENTES.* [En línea] Hospital Information System Edition, 2009. http://www.cnt.com.co/pagina/index.asp?id=178.

CNT PACIENTES. *CNT PACIENTES.* [En línea] www.cnt.com.co.

XHosp Sistema integral para la administración hospitalaria . *XHosp Sistema integral para la administración hospitalaria .* [En línea] www.virtus.com.mx/xhosp/index.html.

Hosix –V. *Hosix –V.* [En línea] https://solutionfinder.microsoft.com.

SELENE de SIEMENS. *SELENE de SIEMENS.* [En línea] http://revistas.um.es/eglobal/article/view/458/429.

*Introducción a la Arquitectura de Software.* Billy Reynoso, Carlos. s.l. : Universidad de Buenos Aires.

Billy, Reynoso, Carlos. Introducción a la Arquitectura de Software – Universidad de Buenos Aires.

Java Enterprise Edition. (Java Enterprise Edition) [En línea]. http://java.sun.com/javaee/, http://java.sun.com/j2ee/overview.html. , .

Allen, Dan. Seam in Action. 2008.

Jamae David, Peter Johnson. JBoss in Action. 2009. 1933988029.

—. UI Develoment with JaveServer Faces.

Red Hat. RichFaces developer Guide. 2007.

—. Ajax4jsf Developer Guide. 2007.

Cristian Bauer, Gavin King. Java Persistence with Hibernate. 2005. 1-932394-88-5.

—. Aplicaciones para la Web II Arquitectura MVC

Java Persistence API. *Java Persistence API.* [En línea] JPA. http://luchorondon.blogspot.com/2009/04/jpa-java- persistence-api.html.

Lenguaje Java. *Lenguaje Java.* [En línea] http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html.

Modelo cliente servidor. *Modelo cliente servidor.* [En línea] http://agamenon.uniandes.edu.co/~revista/articulos/cliser.html.

*Aplicaciones para la Web II Arquitectura MVC.*

Jiménez Garzón Darwin. Ingeniería de Software II

Jiménez Garzón Darwin. Ingeniería de Software II

PgAdmin III. *PgAdmin III.* [En línea] Guía de Ubuntu. http://www.guia-ubuntu.org/index.php?title=PgAdmin_III.

*Alas-His_Documento de Arquitectura del Sistema.* Velázquez Carralero, Alejandro Mario. 2008. IH-SW-DR-091.

[En línea] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/

CC40B - Análisis y Diseño Orientado a Objetos [En línea] http://www.dcc.uchile.cl/~luguerre/cc40b/clase4.html

Clases de diseño. *Clases de diseño.* [En línea] Mayo de 2010 . http://www.taringa.net/posts/info/1492028/Workflow-De-Dise%C3%B1o,-Clases-de-Dise%C3%B1o,-Interfaces,-Diagrama.html.

Ochoa Ornelas, Raquel. Simulador gráfico de algoritmos de programación para computadoras. *Simulador gráfico de algoritmos de programación para computadoras.* [En línea] http://digeset.ucol.mx/tesis_posgrado/Pdf/Raquel%20Ochoa%20Ornel.

Conferencia 5 Ingeniería de Software – Dpto. Ing. Software. 2009

Martínez Alejandro, Martínez Raúl. Guía a Rational Unified Process – Escuela Politécnica Superior de Albacete – Universidad de Castilla la Mancha.

Soto Góngora, Ing. Reinier. Módulo Farmacia del Sistema de Información Hospitalaria alas HIS. Junio 2009.