Universidad de las Ciencias Informáticas

Facultad 7

Título: Módulo Citas del Sistema de Información Hospitalaria alas HIS

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

Autores: Yosmel Martínez Díaz
Alejandro Tarafa Guzmán
Tutor: Ing. Jublar García Ramos

Ciudad de La Habana, Junio del año 2009
"Año del 50 aniversario del triunfo de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los 24 días del mes de junio del año 2009.

Yosmel Martínez Díaz

Alejandro Tarafa Guzmán

Ing. Jublar García Ramos

DATOS DE CONTACTO

TUTOR: Ing. Jublar García Ramos. (jgarcia@uci.cu)
Profesor graduado en la Universidad de las Ciencias Informáticas (UCI), es profesor de la facultad 7 y actualmente se desempeña como jefe de módulo del proyecto Gestión Hospitalaria (GEHOS).

AGRADECIMIENTOS

A nuestros familiares y especialmente a nuestros padres y abuelos, que han confiado en nosotros y han dado todo de sí y nos han guiado como profesionales.

A nuestro tutor y amigo Jublar que gracias a su apoyo hemos logrado llevar a cabo esta investigación y desarrollo del sistema propuesto.

Agradecemos a todos los amigos que hemos obtenido a lo largo de estos 5 años, amigos como los de todos los grupos que por los que hemos pasado pero en especial al 7501, a Julio, Michael, eL albe, William, Franco, Chichi, Rodney, Soto, Morffe, Isnel, el Geri, Omar, Wilder,.Mamito, Osbey, Fabio, Reynel, Gustabo, Gissel, Arianna, Lorena, el niche, Adnier, Juan Manuel (el poeta), Carlos, y Maykell al cual consideramos como uno más de nosotros.

DEDICATORIA

Le dedico mi tesis a mi familia principal motor impulsor de mis triunfos en la vida, pero en especial a mi papa a mi abuela, a mi tío miguelito, a mi tía Mailing, a mi abuela Teresa, a mi abuelo Juanito, a mi tío Orlandito, a mi abuelita Lucila, a mi abuela Yolanda, a mi abuelo José Miguel, a mi tío Daniel, a mi tía Susana, a mi primos y muy especialmente a mi madre Susel Guzmán, a mi padre Juan Enrique, a mi abuela, y a mi tío Miguelito, a mi tía Mailing, ya que han pasado conmigo los buenos y los malos momentos dándome apoyo y exhortándome a seguir adelante sin importar lo difícil que fuera el camino, a mi novia Elizabeth "mi amor" que me ha brindado su amistad, apoyo y amor, a mi compañero de tesis y hermano el yosme "chamaco todo esfuerzo tiene sus beneficios, al final somos ingenieros", a mi amigos del barrio, y a todos lo que de una forma u otra han hecho posible que haya llegado a ser el ingeniero que siempre quise ser…

Alejandro.

Les dedico este trabajo de diploma a mi familia, especialmente a mi mamá y a mi abuela que son la razón de mi vida y por fin me verán como ingeniero. A mi hermano Assiel que es lo que más quiero en esta vida, a mi novia Lisbey le quisiera agradecer por soportarme durante todos estos años, gracias de corazón; a mis hermanos del barrio Dennis y Mislenis, y a mi hermano El tari que por fin ya somos ingenieros…

Yosmel.

RESUMEN

En el mundo existen aplicaciones informáticas creadas según las necesidades de las instituciones hospitalarias actuales. Estos Sistemas de Información Hospitalaria tienen entre sus componentes principales un módulo capaz de automatizar todo el proceso de planificación y asignación de citas dentro de los hospitales.
El área de citas de las instituciones hospitalarias maneja grandes volúmenes de información que se procesa de forma manual, salvo en algunos casos que emplean aplicaciones aisladas que generalmente son propietarias. Además, permiten la planificación de citas solo para un determinado servicio del hospital. Por estas razones la investigación tiene como objetivo desarrollar el Módulo de Citas, del Sistema de Información Hospitalaria alas HIS, que facilite la gestión de información en esta área de las instituciones hospitalarias.

La aplicación es una solución de software orientada a la gestión de la información generada en las instituciones hospitalarias. Está desarrollada sobre el Eclipse Ganymede, se utilizó Java que es una plataforma libre. Además está guiado por el Proceso Unificado de Desarrollo de Software y es multiplataforma. Como gestor de base de datos se utiliza PostgreSQL y como servidor de aplicaciones el JBoss Server. Para la administración de las reglas y procesos del negocio se utilizan Drools y JBoss jBPM respectivamente.
El desarrollo del Módulo de Citas del Sistema de Información Hospitalaria alas HIS, mejora la calidad del servicio médico. Facilita el trabajo de los planificadores de citas al eliminar en gran parte del trabajo manual y disminuye la posibilidad de que se cometan errores humanos al gestionar la cita.

PALABRAS CLAVE

Sistema de Información Hospitalaria (SIH), Triaje, Charla, Cita primera, Interconsulta, Cita control, Servicio, Central de citas.

ÍNDICE

INTRODUCCIÓN

Desde su surgimiento, el hombre fue evolucionando y creando herramientas de trabajo con el objetivo de hacer más fácil y eficiente su paso por la tierra. Así mismo se fue desarrollando hasta inventar los equipos de cómputo para procesar algoritmos complejos. Hoy en día, el desarrollo de estos equipos ha iniciado una nueva era en el desarrollo de la informática y las comunicaciones. Lo que sin dudas, constituye un fenómeno importante, que ha dotado a la humanidad de medios potentísimos para la expansión del conocimiento, la cultura y la sabiduría.

Como resultado del desarrollo y avance de las Tecnologías de la Informática y las Telecomunicaciones, los países Latinoamericanos y los países subdesarrollados fundamentalmente, para no quedarse excluidos de los beneficios que proporcionan, se han trazado como meta desde hace unos años, la implementación de sistemas computarizados para informatizar la sociedad. Muchos de estos países no cuentan con los recursos apropiados para este fin, aunque han experimentado un incremento moderado en el uso de las tecnologías de la información, que proporcionan diariamente un aumento del nivel y la calidad de vida, así como en la eficiencia y la eficacia de algunos de los procesos que en estos países se ejecutan.

Es en la informatización de los procesos hospitalarios donde se han alcanzado mejores resultados debido a la importancia que tiene el sistema de salud en una sociedad, y más en sociedades capitalistas donde existen dos vertientes en el Sistema de Salud: una privada y otra pública. En la salud pública de los países subdesarrollados tiene un menor nivel de desarrollo debido a la falta de interés y de recursos de algunos gobiernos. La informatización de estas instituciones no llega a cumplir con sus necesidades ya que no pasan de ser soluciones aisladas que no poseen ningún tipo de integración; además de brindar soluciones a problemas muy específicos.

En las instituciones hospitalarias pertenecientes a estos países el retraso en los procesos hospitalarios conduce a una demora en la atención al paciente. La planificación de citas se lleva mayormente en papel salvo algunas aplicaciones aisladas que no resuelven los problemas de esta área. Debido a que la manipulación de esta información es engorrosa y compleja, existe mayor posibilidad de errores humanos en el proceso de asignación y planificación de las citas medicas por parte de los planificadores. Además

no se distribuyen equitativamente las citas entre los médicos sobrecargando de trabajo a unos más que otros. También, se crean largas colas de pacientes en espera de la prestación del servicio, por lo que puede ocurrir que la planificación de citas a los pacientes sea ineficiente en algunos casos. El déficits de información en tiempo real a nivel de departamento, sala, salones quirúrgicos, consultas, nivel de hospital o del país, conlleva a la demora de la información por los distintos niveles de gerencia administrativa por lo tanto una de las prioridades fundamentales en estos países es la de informatizar el Sistema de Salud.

En países latinoamericanos y subdesarrollados según las políticas que rigen y atendiendo a sus factores internos y necesidades, desde hace algún tiempo se comenzaron a desarrollar Sistemas Informáticos Computarizados por la necesidad de administrar y gestionar grandes volúmenes de información. Estos sistemas no eran los más completos y no se integraban en su totalidad. Por esta razón no se solucionaban los problemas de los médicos y el personal de los hospitales públicos, por tanto no garantizaban un buen servicio al paciente.

Los sistemas elaborados para estas instituciones hospitalarias son los denominados Sistemas de Información Hospitalaria, SIH (Hospital Information System, HIS), los cuales tienen como objetivo principal sistematizar y optimizar las actividades que incrementen la calidad de la atención al paciente. Un Sistema de Información Hospitalaria permite la recolección, el almacenamiento, el procesamiento, la recuperación y comunicación de información de atención al paciente y administrativa para todas las actividades relacionadas con el hospital.

Los hospitales como actores principales del sistema sanitario generan un importante volumen de información, pero en la mayoría de los casos esta se encuentra dispersa o no está disponible en tiempo y forma necesarios. El sistema de información hospitalaria es un instrumento que permite recoger y tratar la información de modo que sea útil para la toma de decisiones.

Durante estos últimos años se han desarrollado sistemas para informatizar algunos sectores de la salud. Estas soluciones creadas carecían de integración y de una definición generalizable, además de que dichos sectores no contaban con los recursos, tecnologías necesarias y una planificación previa para su posterior desarrollo.

En el Sistema de Información Hospitalaria, el paciente es el más beneficiado, y los profesionales de la salud encuentran en estos sistemas un recurso idóneo, amigable y flexible que responda a las necesidades de información de la institución hospitalaria o de salud. El SIH cumple varias funciones:

1. Procurarle al paciente el acceso a la información en tiempo y forma oportuna.
2. Darle la posibilidad de actualizar esa información y ejercer su derecho de habeas data.
3. Gestionar el conocimiento, cuando se ha entendido la información y se la aplica a posteriori en la acción.
4. Mejorar la práctica clínica (soporte de decisiones).
5. Apoyar a las actividades de Docencia e Investigación.
6. Tornar más eficiente los planes específicos de la Institución.

Por la importancia de estos sistemas, a la Universidad de las Ciencias Informáticas y en específico al Área Temática de Gestión Hospitalaria que pertenece al Polo de Salud de la facultad 7, se le encomendó la tarea de desarrollar un Sistema de Informatización Hospitalaria (SIH) para facilitar todo el proceso de información dentro de los hospitales.

No se puede desarrollar un SIH único para los sistemas hospitalarios de todos los países ya que existe gran diversidad en cuanto a flujo de información, estructura y dirección. El sistema desarrollado por la Universidad de las Ciencias Informáticas deberá ser adaptable para permitir ser implantado en cualquier institución hospitalaria que lo requiera.

El SIH desarrollado para las instituciones hospitalarias posee módulos integrados entre sí, y uno de estos módulos es el de Planificación de Citas. El cual es el encargado de gestionar las citas asignadas a los pacientes por los médicos o personal autorizado a realizar esta tarea por la administración del hospital. Este módulo gestiona cuatro tipos de citas fundamentales: Cita para Consulta; Cita para Interconsulta; Cita para Triaje y Cita para Charla.

Basado en lo antes expuesto se tiene como **problema a resolver**: ¿Cómo facilitar la gestión de información relacionada con los procesos en el área de Citas de las instituciones hospitalarias?

Objeto de Estudio: Proceso de gestión de información en las instituciones hospitalarias. El **campo de acción** según el objeto de estudio es: Proceso de gestión de la información en el área de Citas de las instituciones hospitalarias.

Para resolver el problema identificado se propone el siguiente **objetivo general**: Desarrollar el Módulo de Citas, del Sistema de Información Hospitalaria alas HIS, que facilite la gestión de información en esta área de las instituciones hospitalarias.

Para cumplir con el objetivo se proponen las siguientes **Tareas de la Investigación:**

1. Analizar los procesos de negocio asociados al área de Citas de las instituciones hospitalarias.
2. Asimilar la arquitectura definida por el Área Temática Gestión Hospitalaria para el desarrollo de sus aplicaciones.
3. Obtener mediante el Proceso Unificado de Desarrollo, los flujos de trabajo de "Modelado de Negocio", "Gestión de Requerimientos", "Diseño" e "Implementación".
4. Implementar los procesos de negocio relacionados con la asignación de cita en la central de citas del módulo de Citas.
5. Implementar los procesos de negocio relacionados con la asignación de cita en el servicio del Módulo de Citas.
6. Evaluar las tendencias actuales en el mundo de los Sistemas de Información Hospitalaria.

En este sentido se puede destacar que el desarrollo del Módulo de Citas del Sistema de Información Hospitalaria alas HIS, proporcionará un grupo de beneficios entre los que pueden ser mencionados los siguientes:

1. Mejorar la calidad del Servicio Médico.

2. Organizar los procesos que se llevan a cabo en el área de citas de las instituciones hospitalarias.
3. Mejora el control de gestión y utilización óptima de recursos de salud.
4. Eliminación parcial de las colas de los pacientes en espera de la prestación de servicios.
5. Lograr una eficiente planificación del tiempo de trabajo del médico.
6. Lograr disminuir el trabajo manual por parte de los planificadores de citas.
7. Disminuir la posibilidad de que se comentan errores humanos en el proceso de gestión de las citas.

El presente documento se encuentra estructurado en cuatro capítulos, el primero de ellos, "**FUNDAMENTACIÓN TEÓRICA**", ubica al lector en el Ambiente de Desarrollo del Componente de Seguridad, justificándose las tendencias, tecnologías, metodologías y herramientas que fueron utilizadas para el desarrollo del mismo. Seguidamente el segundo capítulo, "**CARACTERÍSTICAS DEL SISTEMA**", contiene un marco conceptual asociado a la información que será manipulada por el sistema, descripción de los procesos del negocio, modelo de negocio, especificación de los requisitos de software y definición de los casos de uso.

El tercer capítulo **"ANÁLISIS Y DISEÑO DEL SISTEMA"** se centra en la definición del modelo de análisis, modelo de clases de análisis y diseño. 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 de los proyectos que pertenecen al Área Temática de Gestión Hospitalaria.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se presentan los conceptos básicos relacionados con el proceso de gestión de la información en el área de citas del SIH para lograr una mayor comprensión del problema a resolver. También se realiza un estudio de algunos Sistemas de Información Hospitalaria presentes en la escala mundial y nacional. Además se describen las tecnologías, herramientas y metodologías a utilizar en el desarrollo del trabajo.

1.1 Sistema de Salud Pública

La forma y los métodos que sirven de base para la organización de la atención a la salud en un país determinado, es lo que se conoce como Sistema Nacional de Salud (SNS). La Organización Mundial de la Salud lo define como: «Un complejo de elementos interrelacionados que contribuyen a la salud en los hogares, los lugares de trabajo, los lugares públicos y las comunidades, así como el medio ambiente físico y psicosocial en el sector de salud y otros sectores afines». Además es el conjunto de unidades administrativas, de producción, investigación y servicios, responsabilizado con la atención integral de la salud de una población». (1)

De acuerdo con la complejidad de las acciones preventivas, curativas y de rehabilitación, así como la especialización de los servicios de salud brindados, los diferentes niveles de atención médica se han organizado en: (2)

I. Atención Primaria de Salud (APS): Da solución aproximadamente al 80 % de los problemas de salud de la población y que correspondan con las acciones de promoción y protección de la salud. Aunque sus actividades se realizan en cualquier unidad del SNS, están relacionados fundamentalmente con las que se realizan en clínicas Urbanas o Rurales, Dispensarios y Postas Médicas.

II. Atención Médica Secundaria: Este nivel da cobertura a cerca del 15 % de los problemas de salud, su función fundamental es tratar al hombre ya enfermo, tanto desde el punto de vista individual como colectivo, pero también desempeña funciones de rehabilitación, promoción y prevención de

la salud. Se llevan a cabo acciones de salud más complejas y especializadas (Especialidades). Comprende la atención médica brindada en los distintos Hospitales.

III. Atención Médica Terciaria: El nivel terciario debe abarcar alrededor del 5 % de los problemas de salud, relacionados con secuelas o aumento de las complicaciones de determinadas dolencias. Se brindan servicios de muy alta complejidad, con la óptima utilización de los recursos y medios existentes en los mismos y el desarrollo de la investigación. A este nivel pertenecen los Institutos y Hospitales especializados.

Conceptos básicos relacionados con el dominio del problema

Historia Clínica (HC)

Documento que se crea para almacenar el comportamiento evolutivo de un paciente durante su estancia en el hospital. No se limita a ser una narración o exposición de hechos simplemente, sino que incluye juicios, documentos y procedimientos; es un documento que se va haciendo en el tiempo, documentando fundamentalmente la relación médico-paciente. (3)

Interconsulta

Documento que se emite cuando un paciente de un servicio del hospital necesita ser valorado por un médico de otro servicio del hospital.

Consulta de Triaje

Consulta donde el paciente sin historia clínica lo valora el médico y canaliza la situación.

Consulta de Primera

Consulta que se le hace a un paciente que viene por primera vez a atenderse determinada patología.

Consulta de Control

Consulta que se le hace a un paciente que viene a atenderse por una patología anteriormente diagnosticada en el mismo servicio.

1.2 Sistemas de Información Hospitalaria (SIH)

Toda institución hospitalaria tiene la necesidad de contar con sistemas de información avanzados que les permita cumplir sus objetivos debido a la serie de procedimientos y actividades que tienen que llevar a cabo para lograr una adecuada atención del paciente, es por ello la importancia de lograr la implantación de los Sistemas de Información que no son más que un conjunto de instrucciones organizadas, sistematizadas y lógicas que se relacionan entre sí por medio de un lenguaje informático con el fin de obtener información, analizarla, relacionarla y generar nueva información para satisfacer las necesidades de las áreas administrativas, operativas de una organización en general.

Cada día los procesos de registro, seguimiento y tratamiento del paciente deben mejorarse, innovarse y apoyarse en tecnologías para hacer más eficiente y eficaz las actividades rutinarias del hospital mediante los SIH, no basta con tener datos e información, hay que procesarla, analizarla, interpretarla y utilizarla. Por ello los Sistemas de Información Hospitalaria tienen como propósito permitir la optimización de los recursos humanos y materiales para satisfacer las necesidades de las áreas operativas, administrativas, clínicas y de investigación en las organizaciones de salud.

1.3 Antecedentes

Existe un amplio grupo de empresas especializadas en la concepción, desarrollo e implantación de soluciones de software vinculadas al área de la salud, existiendo un vínculo estrecho con las instituciones hospitalarias. Dentro de dichas instituciones se ha ampliado el marco de desarrollo hacia cada una de las áreas, servicios y departamentos que estas entidades incluyen, específicamente el departamento de citas, en aras de incrementar la calidad de la atención en los servicios de la salud.

1.3.1 Escala Mundial

Uno de los Sistemas de Información Hospitalaria que contribuye hoy en día a hacer más eficiente y eficaz el trabajo, es el denominado **x-HIS**, desarrollado por ISOFT una compañía del grupo IBA Health. El x-HIS es un sistema multiplataforma, soporte para la arquitectura cliente-servidor y/o 3 capas, permite la integración con otros sistemas como RIS, PACs mediante los estándares HL7, CMBD, Codificación ICD-9-CM/ICD 10, SNOMED, entre otros; y es un sistema multilenguaje. Su alcance está enfocado en las Áreas

de Admisión, Urgencias, Consulta Externa, Bloque Quirúrgico, Archivo de Historia Clínica, Gestión de Prescripciones y resultados médicos, Historia Clínica Electrónica, Hoja de Prescripciones, Anatomía Patológica, Radiología, Hemoderivados, Nutrición y Dietética, Fisioterapia. A pesar de ser un sistema fácil, ágil y extensible el sistema es una aplicación de escritorio por lo que se requiere de mayor cantidad de recursos para lograr su distribución y despliegue.

Otro de los SIH utilizados en México es el **SIGHO** (Sistema de Información para la Gerencia Hospitalaria). Es un software basado en la Norma Oficial Mexicana NOM-168-SSA1-1998 referente al resguardo y uso del expediente clínico electrónico para facilitar las actividades de gerencia dentro del hospital y se apoya de estándares internacionales para el diagnóstico de enfermedades y realización de procedimientos tales como el CIE-10 y CIE9MC; el cual consta de 14 módulos fundamentales como son: Agenda, Consulta Externa, Admisión, Hospitalización, Urgencias o Enfermería, Toco-cirugía, Cirugía, Trabajo Social, Imagenología, Laboratorio, Patología, Banco de Sangre, Caja, Relaciones Publicas, Farmacia y Almacén y Tablero de Control. Este sistema no presenta una interfaz amigable, y se requiere de conocimientos informáticos para poder utilizar correctamente el sistema, además de ser una aplicación de escritorio.

Otro de los sistemas utilizados en la escala mundial es el denominado **Kewan – Cosmosalud**, sistema de información hospitalario (HIS), de apoyo a la gestión, que posibilita:

- Procesos administrativos de admisión de pacientes, generación de documentos, imprimir etiquetas, etc.
- Obtención de indicadores de gestión: actividad, demora, entre otros.
- Gestión de áreas básicas: admisión, archivo, facturación, entre otros.
- Gestión de sistemas departamentales: farmacia, rehabilitación, entre otros.

Orientado a todo tipo de organización.

- Público o Privados.
- Atención Primaria o Atención Especializada.

- Centro Único o Conjunto de Centros (multi-centro).

Kewan – Cosmosalud está implantado en más de 68 hospitales y clínicas dentro y fuera de España. De ellos, en más de 37 hospitales públicos y 31 centros privados, además Es multi-centro; multi-idioma, por usuario; multimedia, ya que integra todo tipo de información, ya sean imágenes, voz, video. Su interfaz es Windows completa más interfaz Web en algunos módulos. La interfaz gráfica permite visualizar la ocupación de quirófanos, agendas, entre otros e incorpora las últimas tecnologías como la firma electrónica, reconocimiento de huella, reconocimiento de voz, códigos de barras, facturación electrónica, envío mensajes SMS o e-mail. Automatiza acciones tales como avisos, carga de actos a facturación, entre otros, posee herramientas de personalización: consultas, reportes, formularios y es independiente del gestor de base de datos: Oracle, Informix, SQLServer.

El sistema **Care2x**, desarrollado desde el 2002, rápidamente fue soportado por una comunidad de desarrolladores en el mundo, basado en estándares de código abierto. Integra datos, funciones y flujo de tareas en un entorno de cuidados de la salud. Está conformado por cuatro componentes principales que a su vez pueden funcionar de forma independiente. Soluciona el problema de la dependencia de plataforma y la redundancia e incompatibilidad de datos. Esta implementado en PHP 4.0.4, utilizando como gestor de Base de Datos MySQL 3.2x - 4.0.x y corre sobre un servidor Web Apache 1.X. Cuenta con un módulo de admisión en el cual existen funcionalidades como la inscripción de un paciente nuevo, búsqueda de paciente ingresado y búsqueda de casos archivados.

El sistema **Galenhos(R)** ha sido diseñado con el propósito de apoyar a los establecimientos de salud en el correcto registro de información, clínica o administrativa, y la generación de información gerencial. Su desarrollo nace de la asistencia técnica prestada por la agencia de Estados Unidos para el desarrollo internacional (USAID), a través del proyecto Partners for Health Reform plus (PHRplus), a un hospital de Perú. Como sistema hospitalario para la gestión de su información, el Galenhos cuenta con ventajas apreciables como:

1. Información estandarizada: favorece un registro adecuado, donde se utilizan todos los estándares establecidos por el ministerio de salud.

2. Bases de datos consolidadas y exportables: eliminan los problemas provenientes de la existencia de sistemas paralelos, incompatibles e incomunicados entre sí.

3. Generación de reportes clínicos de uso gerencial. Facilita la evaluación de su capacidad resolutiva y eficiencia en el manejo de los recursos.

4. Altos estándares de seguridad informática: responden a la elevada sensibilidad de la información que se maneja, protegiendo la operación regular del sistema y favoreciendo los controles o auditorias.

5. Diseño modular: considera la ampliación y futuros desarrollos, que se extiendan en toda la complejidad hospitalaria con módulos adicionales.

Fue desarrollado en el lenguaje de programación Visual Basic 6.0 y utiliza el Microsoft SQL server como gestor de base de datos. Su principal desventaja radica en que no es un sistema multiplataforma y se ejecuta mediante una aplicación de escritorio.

1.3.2 Escala Nacional

El sistema **Galen Hospital** es un sistema desarrollado por la empresa SOFTEL cubana que está orientado hacia la informatización de la gestión de pacientes como elemento básico de control para mejorar la atención médica, optimizar el uso del personal, aumentar la calidad de los servicios hospitalarios y disminuir sus costos. Brinda la información requerida para la actividad gerencial a todos los niveles y la elaboración de reportes estadísticos. Este proyecto incluye las áreas de registro de pacientes, hospitalización, gestión de medios diagnósticos y consulta, además que incluye conexión con equipos de diagnóstico y auto-analizadores así como la emisión de informes de resultados y estadísticas.

Incluye módulos que ejecutan los procesos informativos vinculados con el registro, admisión y alta de los pacientes para cada episodio hospitalario y establecen el monitoreo de la información asociada a la estancia de cada paciente en el hospital y efectúan la apertura de las historias clínicas y el control de la ubicación de las mismas realizando las solicitudes de los servicios requeridos por cada paciente y el registro de los servicios recibidos por cada uno.

Según el análisis realizado previamente los Sistemas de Información Hospitalaria antes expuesto son mayormente sistemas propietarios que pueden ser instalados en países que tengan las licencias para comprar el software y hardware que requieran. Estos sistemas son aplicados y construidos según las características de las instituciones hospitalarias y mayormente son aplicaciones de escritorio.

A pesar de la cantidad de SIH que se encuentran en el mercado, sólo unos pocos productos cubren todos los requerimientos de un hospital, o proveen una adecuada integración con las amplias redes de atención en salud. Ninguno de estos sistemas posee un Módulo de Citas de manera independiente dentro del SIH, sino que brindan algunos servicios aislados o forman parte de otros módulos como Consulta Externa; por tanto estos sistemas no podrán planificar citas para otros servicios.

1.4 Objeto de automatización o información manipulada.

El objeto de automatización son los procesos relacionados con la planificación, administración y gestión del área de citas en las instituciones hospitalarias. Las aplicaciones previamente investigadas abordan procesos de citas de forma general, por lo que carecen de un enfoque específico hacia esa área ya que el proceso de planificación, administración y gestión está incluido en otros módulos. Lo que le resta robustez a estos sistemas a la hora de lograr la integración con los restantes módulos del SIH. La automatización de la planificación de citas es un pilar importante dentro de un sistema hospitalario ya que posibilita una mejor planificación del tiempo de trabajo del personal.

1.5 Tendencias y tecnologías actuales a considerar.

La informatización de la sociedad en los últimos años ha sido el motor impulsor del desarrollo de nuevas tecnologías, metodologías y herramientas para facilitar este propósito. En el ámbito mundial existen herramientas que permiten lograr en tiempo record el desarrollo de grandes sistemas que manipulan un amplio volumen de información. Además de posibilitar un alto grado de calidad gracias a las facilidades que implementan dichas herramientas.

En la actualidad para realizar el desarrollo de un software existen ciertos criterios a tener en cuenta como son los lenguajes de programación a utilizar, gestores de bases de datos, patrones arquitectónicos y de

diseño, metodologías, entre otros. A continuación se presenta el resumen realizado durante la investigación, y la propuesta tecnológica realizada por el arquitecto del proyecto.

1.5.1 Sistemas distribuidos. Modelo Cliente Servidor

Esta arquitectura consiste básicamente en que un programa, el cliente informático, realiza peticiones a otro programa, el servidor, que se encarga de darle respuesta. La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico. En ella la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema. (4)

1.5.2 Arquitectura en 3 capas

El objetivo principal de esta arquitectura es la separación de la lógica de negocios de la lógica de diseño. Su ventaja principal radica en que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API (de las siglas Aplication Program Interface) que existe entre niveles. (5)

El diseño más utilizado actualmente es el diseño en tres capas, consiste en:

• Capa de presentación: es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso. Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser fácil de entender y utilizar para el usuario.

• Capa de negocio: es donde residen los programas que se ejecutan. Se denomina capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para

solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

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

1.5.3 Modelo Vista Controlador (MVC)

El patrón MVC separa la aplicación en tres componentes diferentes, lo que proporciona cierta libertad entre los datos la interfaz y el negocio permitiendo tener varias vistas sobre un mismo modelo de datos. El MVC se usa frecuentemente en aplicaciones Web. Los componentes del MVC son: (6)

- **Modelo:** Administra el comportamiento y los datos del dominio de la aplicación, responde a requerimientos de información sobre su estado, usualmente formulados desde la vista, respondiendo a instrucciones de cambio para cambiar el estado de estos datos, habitualmente desde el controlador.

- **Vista:** Presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.

- **Controlador:** Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Ventajas:

- Clara separación entre interfaz, lógica de negocio y de presentación, que además provoca parte de las ventajas siguientes.
    - Sencillez para crear distintas representaciones de los mismos datos.
    - Reutilización de los componentes.
    - Simplicidad en el mantenimiento de los sistemas.

- Facilidad para desarrollar prototipos rápidos.
- Los desarrollos suelen ser más escalables.

- **Adaptación al cambio**: debido que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

- **Varias vistas**: debido a que la vista se encuentra separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente.

Desventajas:

- La distribución de los componentes obliga a crear y mantener mayor cantidad de ficheros.
- **El costo de actualizaciones frecuentes:** Si el modelo es cambiado frecuentemente, se podría desbordar las vistas con una lluvia de requerimientos de actualización. (7)

1.6 Tecnologías utilizadas en el proceso de desarrollo

1.6.1 Java

Java es un lenguaje de programación desarrollado por la compañía Sun Microsystems con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras. Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma. (8)

Ventajas:

- Reduce en un 50% los errores más comunes de programación, reducción del tiempo de desarrollo de aplicaciones, es multiplataforma, las aplicaciones en Java resultan extremadamente seguras, ya que no acceden a zonas delicadas de memoria o de sistema, con lo cual evitan la interacción de ciertos virus, es interpretado y dinámico, entre otras cosas.

1.6.2 Capa de Presentación

1.6.2.1 JSF

JSF (Java Server Faces) es un framework de desarrollo basado en el patrón MVC (Modelo Vista Controlador). JSF trata la vista (el interfaz de usuario) de una forma algo diferente a la acostumbrada en las aplicaciones Web. Sería más similar al estilo de Swing, Visual Basic o Delphi, donde la programación de la interfaz se hace a través de componentes y basada en eventos.JSF es muy flexible. Por ejemplo permite crear componentes propios, o crear los propios "render" para pintar los componentes según convenga.

1.6.2.2 Librería RichFaces

RichFaces es una biblioteca de componentes para JSF y un avanzado framework para la integración de AJAX con facilidad en la capacidad de desarrollo de aplicaciones de negocio. Sus componentes vienen listos para su uso out-of-the-box, por lo que los desarrolladores pueden ahorrar tiempo de inmediato para aprovechar las características de los componentes para crear aplicaciones Web que proporcionan mejoras en gran medida la experiencia del usuario más fiable y más rápidamente. RichFaces también incluye un fuerte apoyo para la skinnability de aplicaciones JSF. Además aprovecha al máximo los beneficios de JSF framework incluyendo, la validación y conversión de instalaciones, junto con la gestión de estática y dinámica los recursos. (9) (10) (11)

Características:

- Conjunto de componentes AJAX prefabricados y la habilidad de añadir capacidad de AJAX a componentes existentes.
- Añade capacidad de AJAX a aplicaciones JSF existentes.
- Componentes skinnable.
- Gran número de componentes.
- Posibilita escribir tus propios componentes con soporte AJAX incorporado.
- Prueba de componentes, acciones y páginas como uno las haya creado. (7)

1.6.3 Capa de Negocio

1.6.3.1 JBoss Seam

Boss Seam es un framework desarrollado por JBoss que combina a los 2 frameworks Enterprise JavaBeans EJB3 y JavaServerFaces JSF. Se puede acceder a cualquier componente EJB desde la capa de presentación refiriéndote a él mediante su nombre de componente Seam.

Seam introduce el concepto de contextos. Cada componente existe dentro de un contexto. El contexto conversacional por ejemplo captura todas las acciones del usuario hasta que éste sale del sistema o cierra el navegador - inclusive puede llevar un control de múltiples pestañas y mantiene un comportamiento consistente cuando se usa el botón de regresar de el navegador. (12) (13) (14)

1.6.3.2 JBoss Server

JBoss Application Server es el servidor de aplicaciones de código abierto más ampliamente desarrollado del mercado. Por ser una plataforma certificada J2EE, soporta todas las funcionalidades de J2EE 1.4, incluyendo servicios adicionales como clustering, caching y persistencia. JBoss es ideal para aplicaciones Java y aplicaciones basadas en la web. También soporta Enterprise Java Beans (EJB) 3.0, y esto hace que el desarrollo de las aplicaciones del empresario sean mucho más simples. (15) (16)

Las características destacadas de JBoss incluyen:

- Producto de licencia de código abierto sin coste adicional.
- Cumple los estándares.
- Confiable a nivel de empresa
- Orientado a arquitectura de servicios.
- Flexibilidad consistente
- Servicios del middleware para cualquier objeto de Java
- Ayuda profesional 24x7 de la fuente
- Soporte completo para JMX

1.6.4 Capa de Datos

1.6.4.1 Hibernate

Hibernate es una herramienta de Mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones además de diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada se podrán generar bases de datos en cualquiera de los entornos soportados: Oracle, DB2, MySQL, entre otros. Además, es Open Source, lo que supone, entre otras cosas, que no se tiene que pagar nada por adquirirlo.

Entre las principales características técnicas que aporta Hibernate están las siguientes:

- Modelo de programación natural. Hibernate es una capa de persistencia objeto/relacional que permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos.
- Gran escalabilidad. Hibernate es muy eficiente, tiene una arquitectura de caché de doble capa y podría ser usado en un clúster. .
- Software libre. Está bajo licencia LGPL[1].
- Persistencia transparente. Ofrece soporte para un amplio conjunto de las colecciones de Java, propiedades del estilo de persistencia de JavaBeans, etc. que abstraen al usuario.
- Mapeado flexible gracias a las asociaciones bidireccionales, la persistencia transitiva, colecciones de tipos básicos, etc. el mapeado resulta mucho más flexible.
- Facilidades en consultas. Debido en parte a que se realizan en un potente lenguaje de consultas orientado a objetos.

Facilidades en metadatos. Soporta el formato del mapeado de XML, diseñado para ser editado a mano y el mapeado basado en anotaciones. Además de validación basada en anotaciones. (17)

1.6.4.2 Postgres SQL 8.3 Servidor de Base de Datos.

[1] Lesser GNU Public License

Es un sistema de gestión de base de datos relacional orientada a objetos de software libre. Es un Sistema de Administración de Bases de Datos de objetos relacionales. Pertenece al movimiento de Software Libre y fue lanzado bajo la licencia BSD[2]. Ofrece una alternativa a los demás Sistema de Administración de Bases de Datos. Al igual que otros proyectos de Software Libre como Apache, GNU Linux y Media Wiki, PostgreSQL no está controlado por una sola compañía, sino que cuenta con comunidad global de desarrolladores y compañías para su evolución. Se conoce además como Postgres, que fue su nombre original y su origen se remonta a que inicialmente fue concebido como post Ingress database (base de datos de post ingreso). Sus autores desarrollaron además el sistema de base de datos Ingress. (18) (19)

1.6.4.3 Java Persistence API (JPA)

Estándar para la persistencia y el ORM o mapeo de objeto relacional para la plataforma Java, que permite utilizar el modelo de dominio Java para la administración de bases de datos. (20)

1.6.4.4 Enterprise JavaBeans (EJB3)

Arquitectura para la plataforma java del lado del servidor, que permite realizar la administración automática de la seguridad, escalabilidad, concurrencia, distribución, transacciones, persistencia de datos y ambientes portables.

1.7 Interfaces de Comunicación

1.7.1 Servicios Web y XML

En el caso del manejo de datos se usa ampliamente lo que se conoce como Servicio Web (en inglés, Web Service) que no es más que una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

[2] Berkeley Software Distribution

Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma pueden utilizar los servicios Web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares. Los servicios Web hacen uso de distintas tecnologías como son XML, SOAP, XSL.

1.7.2 XML

XML[3] es un metalenguaje extensible de etiquetas desarrollado por el W3C[4]. Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos. Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fácil y fiable.

XML, sin embargo permite al diseñador crear sus propias marcas, especificando su sintaxis y funcionalidad de tal forma que las nuevas marcas puedan ser reconocidas e interpretadas por los visualizadores, sirve fundamentalmente para almacenar la información de una forma estructurada, con estructuras que siguen ciertas reglas y que son accesibles de forma manual o automatizada. Proporciona un mecanismo que permite almacenar e intercambiar la información de una forma estructurada y en un formato comprensible por aplicaciones situadas en sistemas heterogéneos. (21)

[3] eXtensible Markup Language por sus siglas en inglés (lenguaje de marcas extensible).

[4] World Wide Web Consortium

1.7.3 HL7

HL7 [5] Es un protocolo para el intercambio de información clínica a través de mensajes.

El estándar HL7 asume que el entorno de comunicaciones proveerá lo siguiente:

- **Trasmisión sin errores:** Las aplicaciones pueden asumir que recibirán correctamente toda cadena de bytes t-trasmitida. Esto implica el chequeo de errores es realizado en un nivel inferior.
- **Conversión de caracteres:** En el caso de que diferentes máquinas utilicen distintas presentaciones de caracteres (ej. ASCII-EBCDIC) será el entorno de comunicaciones el que realice esta tarea.
- **Largo del mensaje:** HL7 no especifica ninguna restricción al largo de un mensaje.

1.7.4 Metodología de Desarrollo RUP

RUP[6] es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Sus principales características se centran en implementar las mejores prácticas de Ingeniería de Software, disciplinar la forma de asignar tareas y responsabilidades, quién hace qué, cuándo y cómo, administrar requisitos, usar arquitectura basada en componentes y controlar cambios y modelado visual del software.

RUP posee tres características fundamentales, la primera de ellas es que su desarrollo es iterativo e incremental por lo que divide el proceso de desarrollo en ciclos, teniendo un producto final al terminar cada ciclo. La segunda es que está guiado por los casos de uso. Un caso de uso será aquello que describe un fragmento de las funcionalidades del sistema que proporciona al usuario un resultado importante. Los casos de uso guían el diseño construcción y prueba del sistema, esto significa que guían el proceso de desarrollo. Por último y no la menos importante RUP está centrada en la arquitectura, lo que le permite a los desarrolladores una mayor visibilidad del sistema, pues la arquitectura es una vista del diseño completo del software con las características más importantes resaltadas, dejando a un lado los detalles. RUP utiliza como lenguaje de modelado UML (Unified Modeling Language).

[5] Health Level Seven

[6] Rational Unified Process

Ventajas

RUP proporciona una serie de ventajas para el desarrollo como son: la mitigación temprana de posibles riesgos altos permitiendo tener un producto con mayor robustez, progreso visible en las primeras etapas tempranas demostrando que el trabajo está avanzando, retroalimentación que se ajuste a las necesidades reales con lo que se obtiene lo que realmente desea el usuario, gestión de la complejidad, además de que el conocimiento adquirido en una iteración puede ser aplicado en el resto de estas, entre otras. (22)

1.7.5 Lenguaje de Modelado UML

Permite modelar, construir y documentar los elementos que forman un sistema software Orientado a Objetos. UML[7] se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa (principalmente Booch, OMT y OOSE). (23)

1.7.6 BPM

Se enfoca en la administración de los procesos del negocio. Metodología empresarial cuyo objetivo es mejorar la eficiencia a través de la gestión sistemática de los procesos de negocio, que se deben modelar, automatizar, integrar, monitorizar y optimizar de forma continua.

1.7.8 Herramienta de Modelado Visual Paradigm

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar

[7] Unified Modeling Language

documentación. La herramienta UML CASE[8], proporciona además abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

1.8 Herramientas

Con los elementos expuestos anteriormente se pueden definir las herramientas que conforman el Ambiente de Desarrollo del Módulo de Hospitalización para el Sistema de Información Hospitalaria alas HIS.

En primer lugar se utilizará el Visual Paradigm como herramienta CASE de modelado utilizando BPM para el modelado del negocio y UML para el resto del modelado, que permite crear los diagramas que se generen como parte de la documentación; posee además una fuerte integración con el IDE[9] Eclipse permitiendo así generar documentación en varios formatos como son HTML, Word, PDF entre otros, además de generar código partiendo de los diagramas creados en el mismo. Para lograr llevar a cabo el desarrollo de la aplicación se utilizará el Eclipse, el cual permite facilitar la programación en distintos lenguajes y es multiplataforma.

Entre las principales características que posee el Eclipse que son de gran utilidad para el desarrollo se destaca un editor de texto, resaltado de sintaxis, compilación en tiempo real, pruebas unitarias con JUnit, integración con Ant, asistentes (wizards) para creación de proyectos, clases, pruebas, entre otros, refactorización y a través de plugins permite añadir control de versiones con SVN (Subversion) e integración con Hibernate.

Eclipse contiene una serie de perspectivas. Cada perspectiva proporciona una serie de funcionalidades para el desarrollo de un tipo específico de tarea. Por ejemplo la perspectiva Java combina un conjunto de vistas que permiten ver información útil cuando se está escribiendo código fuente, mientras que la

[8] Computer-Aided Software Engineering

[9] Integrated Development Environment

perspectiva de depuración contiene vistas que muestran información útil para la depuración de los programas Java.

Eclipse es uno de los proyectos de código abierto más interesantes y de mayor usabilidad para el desarrollo de aplicaciones, entre sus versiones está el Eclipse Europa, con un total de 21 subproyectos que le permiten adaptarse a las necesidades de cualquier programador.

Una de las ventajas de usar la versión de Eclipse "Europa" es que a diferencia de instalar los plugins manualmente de cada uno de los proyectos, en este las dependencias de los paquetes comunes han sido sincronizadas. Otra ventaja la constituye la sencillez con que se instalan los proyectos, ya que existe un repositorio único donde aparecen publicados todos los proyectos que conforman Europa, además de que se eliminan también problemas relacionados con la compatibilidad de versiones. (24)

Conclusiones

En este capítulo se fundamentó la selección de las tecnologías y herramientas que son utilizadas para la modelación, diseño e implementación del sistema propuesto por la presente investigación. Se ha hecho un estudio del estado del arte de los sistemas de gestión de información en el mundo y en Cuba; y se argumentaron las razones por las que se decidió utilizar la metodología de desarrollo del proceso unificado de Rational, conjuntamente con herramientas CASE, para modelar el problema planteado usando el lenguaje UML.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En este capítulo se presenta formalmente el problema científico, el objeto de automatización, punto en el cual se describen los procesos de negocio que son objeto de automatización, así como una descripción de los documentos que se procesan y la información que se manipula. También se hace una descripción general de la propuesta del sistema que propone este trabajo de diploma. Además se exponen el modelo de negocio, la especificación de requerimientos funcionales y no funcionales del sistema y finalmente se muestran los diagramas de casos de uso.

2.1 Flujo actual de los procesos involucrados en el campo de acción.

2.1.1 Asignar cita

Permitir asignar una cita al paciente que solicita asistencia médica en servicios especializados ya sea para charlas, interconsultas, de control o de primera.

2.1.1.1 Solicitar documentos

Al llegar el paciente se puede presentar con una referencia, una orden de apertura de historia clínica, un libro de hipertensión, una orden de cita, una interconsulta o sin un récipe, en este último caso comunicando que necesita asistencia médica especializada. El (la) técnico (a) de registros y estadísticas de salud le solicita el documento y en dependencia de este realiza la acción correspondiente.

2.1.1.2 Buscar cupo para charla

Si el paciente se presenta por primera vez con una referencia o una interconsulta a solicitar una cita para el servicio de hipertensión, el (la) técnico (a) le comunica que antes de la consulta debe asistir a dos charlas de nutrición y busca en el libro de control de charlas según la fecha, un cupo disponible para estas. El (la) técnico (a) de registro también puede buscar un cupo disponible para charlas cuando el paciente no pudo asistir a las charlas programadas.

2.1.1.3 Asignar cupo para charla

El (la) técnico (a) de registros y estadísticas de salud luego de localizar un cupo disponible en el libro de control de charlas registra en este, el número de HC del paciente (en caso de tener historia clínica en el hospital), nombre y apellidos, teléfono, edad y nombre del médico encargado de impartir la charla. Además escribe en la referencia o interconsulta la fecha de la charla para orientación del paciente.

2.1.1.4 Asignar cupo de triaje

Si existe cupo disponible para triaje la enfermera escribe en el registro diario de pacientes los datos personales del paciente y le indica que espere hasta ser atendido por el médico.

2.1.1.5 Buscar cupo para interconsulta

Si el paciente se presenta con una interconsulta a un servicio que no sea hipertensión, el técnico de registros médicos busca el libro de control de citas y localiza un cupo según la disponibilidad existente.

2.1.1.6 Buscar cupo de primera o de control

Si la cita es para determinado médico el técnico de registros médicos busca el libro de control de citas y localiza un cupo disponible teniendo en cuenta el médico, el tipo de consulta (control o primera) y la fecha de la cita en caso de se haya reflejado, Si no tiene el médico especificado solo se tienen en cuenta el tipo de consulta y la fecha de la cita en caso de que se haya reflejado.

2.1.1.7 Asignar cupo

El técnico de registros médicos luego de localizar un cupo disponible en el libro de control de citas registra en este, el número de HC del paciente (en caso de tener historia clínica en el hospital), nombre y apellidos. Además escribe en el libro de hipertensión, la orden de apertura de HC, la interconsulta, la tarjeta de cita o la orden de cita la fecha del día de la consulta.

2.1.1.8 Objeto de automatización.

El siguiente proceso de negocio será objeto fundamental de automatización para que el sistema funcione correctamente.

Asignar Cita: Permitir asignar una cita al paciente que solicita asistencia médica en servicios especializados ya sea para charlas, interconsultas, de control o de primera.

2.2 Información que se maneja.

Existen numerosos documentos que se generan recogiendo y archivando toda la información relacionada con la planificación de citas dentro del sistema. La automatización de estos documentos son claves para el correcto funcionamiento del sistema. A continuación se presentan los documentos que han sido seleccionados para su automatización:

Tarjeta de cita: En la tarjeta se recogen los datos de la cita una vez se haya planificado con anterioridad la cita del paciente, en la tarjeta se muestran datos como el nombre del médico que va atender al paciente, la hora de inicio y fin de la cita, la fecha en que se creó la cita, el servicio al cual pertenece ese paciente y el tipo de cita que se ha planificado.

Libro de cita: Es donde se guardan las citas que tienen planificadas los médicos y el fondo de tiempo, que primeramente se llenan con los días y después se llena con los nombres de los pacientes. En el libro se anotan los días que el médico tiene que trabajar y donde se anotan los pacientes citados en los días planificados.

Estos documentos son los principales a automatizar en el Módulo de Citas para llevar una planificación lo más óptima posible, evitando que se cometan errores humanos para tratar de perfeccionar todo el proceso de planificación y asignación de citas facilitándole el trabajo a la secretaria y a los pacientes que son el principal objetivo.

2.3 Modelo de Negocio.

Uno de los flujos de trabajo que tienen mayor peso durante la fase de Inicio en el desarrollo de software es el modelado del negocio el cual tiene como objetivos fundamentales:

- Comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema.
- Comprender los problemas actuales de la organización e identificar las mejoras potenciales.
- Asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización.
- Derivar los requerimientos del sistema que va a soportar la organización.

Para lograr esos propósitos, el proceso de modelado permite obtener una visión de la organización que permita definir los procesos, roles y responsabilidades de la organización en los modelos de casos de uso del negocio y de objetos.

2.3.1 Actores del Negocio.

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados.

Actores del Negocio	**Descripción**
Técnico de registros médicos	**Encargado de buscar cupos disponibles y asignar citas para consultas de control, interconsultas, de primera y charlas.**
Enfermera	**Se encarga de buscar cupos disponibles para los pacientes que necesitan pasar por una**

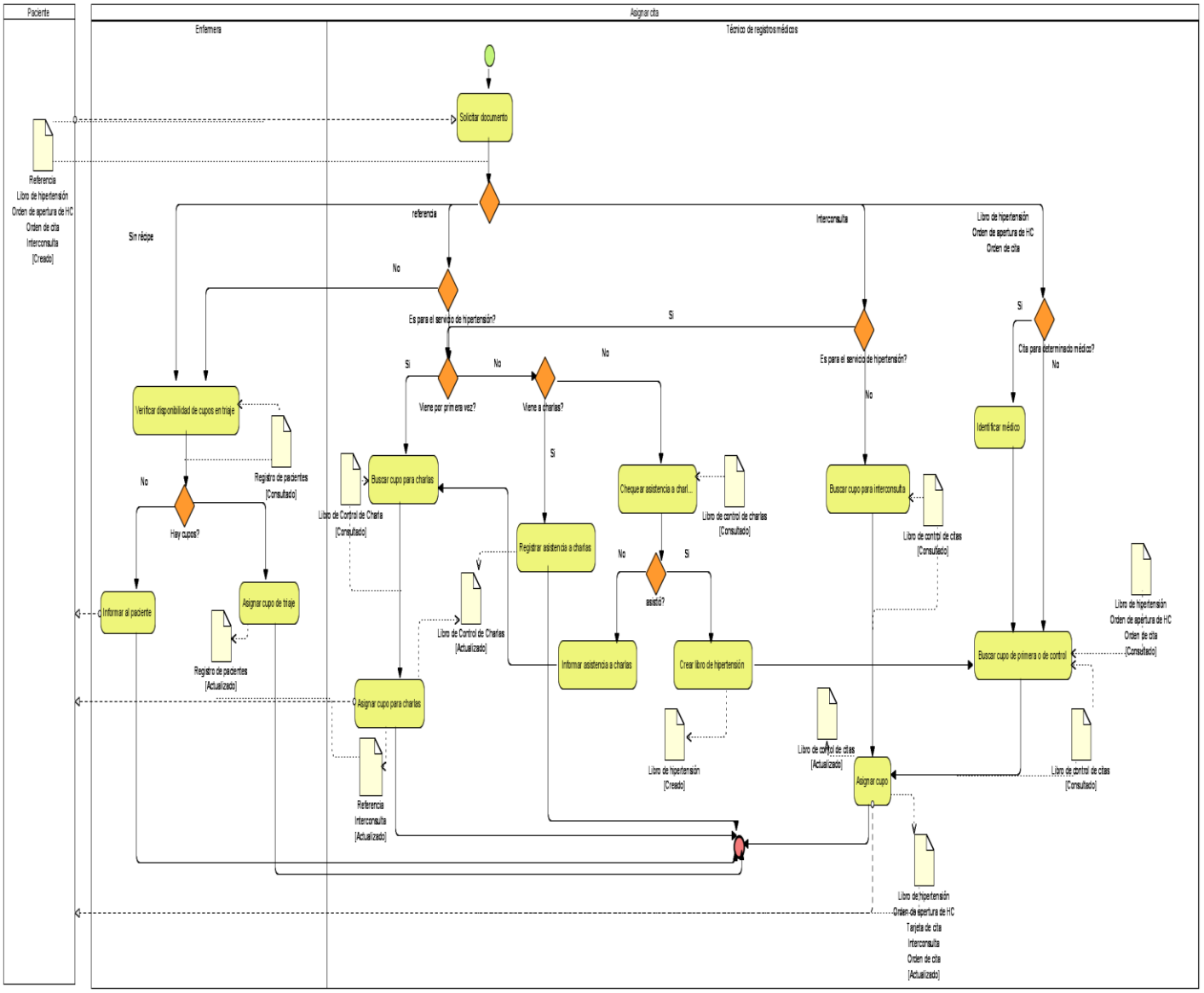
	consulta de triaje.

2.3.2 Trabajadores del Negocio.

Es quién define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.

Trabajadores del Negocio	**Descripción**
Técnico (a) de registros y estadísticas de salud de central de citas	**Encargado (a) de asignar, buscar, modificar y eliminar una cita para cualquier servicio del hospital.**
Técnico (a) de registros y estadísticas de salud de consulta externa	**Encargado (a) de asignar, buscar, modificar y eliminar cita para el servicio al cual pertenece, además registrar asistencia a charla.**

2.3.3 Diagrama de Procesos de Negocio.



2.3.4- Descripción textual de los procesos de negocio

PROCESO:	Asignar cita
Misión:	Permitir asignar una cita al paciente que solicita asistencia médica en servicios especializados ya sea para charlas, interconsultas, de control o de primera.

ACTORES INVOLUCRADOS

Rol	Funciones
Técnico de registros médicos	Encargado de buscar cupos disponibles y asignar citas para consultas de control, interconsultas, de primera y charlas.
Enfermera	Se encarga de buscar cupos disponibles para los pacientes que necesitan pasar por una consulta de triaje.

ACTIVIDADES

Solicitar documentos	

Flujo de Información

Al llegar el paciente se puede presentar con una referencia, una orden de apertura de historia clínica, un libro de hipertensión, una orden de cita, una interconsulta o sin un récipe, en este último caso comunicando que necesita asistencia médica especializada. El técnico de registros médicos le solicita el documento y en dependencia de este realiza la acción correspondiente.

Artefacto	Emisor	Receptor	Formato	Frecuencia
Referencia	Paciente	Técnico de registros médicos	Papel	Cada vez que el paciente necesite ser valorado por un especialista
Orden de apertura de historia	Paciente	Técnico de registros médicos	Papel	Cada vez que el paciente no tenga HC de la especialidad

				y necesite tenerla
Orden de cita	Paciente	Técnico de registros médicos	Papel	Cada vez que el paciente necesite consulta de control
Interconsulta	Paciente	Técnico de registros médicos	Papel	Cada vez que un paciente de un servicio del hospital necesite ser valorado por un médico de otro servicio del hospital

Actividad:	Buscar cupo para charlas

Flujo de Información

Si el paciente se presenta por primera vez con una referencia o una interconsulta a solicitar una cita para el servicio de hipertensión, la técnico le comunica que antes de la consulta debe asistir a dos charlas de nutrición y busca en el libro de control de charlas según la fecha, un cupo disponible para estas.

La técnico de registro también puede buscar un cupo disponible para charlas cuando el paciente no pudo asistir a las charlas programadas.

Artefacto	Emisor	Receptor	Formato	Frecuencia
Libro de control de charla	Técnico de registros médicos	Técnico de registros médicos	Papel	Cada vez que se necesite asignar cita para charla
Interconsulta	Paciente	Técnico de registros médicos	Papel	Cada vez que un paciente de un servicio del hospital necesite una cita con un médico

				de otro servicio del hospital
Referencia	Paciente	Técnico de registros médicos	Papel	Cada vez que el paciente necesite ser valorado por un especialista

Actividad:	Asignar cupo para charlas

Flujo de Información

El técnico de registros médicos luego de localizar un cupo disponible en el libro de control de charlas registra en este, el número de HC del paciente (en caso de tener historia clínica en el hospital), nombre y apellidos, teléfono, edad y nombre del médico encargado de impartir la charla. Además escribe en la referencia o interconsulta la fecha de la charla para orientación del paciente.

Artefacto	Emisor	Receptor	Formato	Frecuencia
Libro de control de charla	Técnico de registros médicos	Técnico de registros médicos	Papel	Cada vez que se necesite asignar cita para charla
Interconsulta	Técnico de registros médicos	Paciente	Papel	Cada vez que el paciente necesite ser valorado por un especialista
Referencia	Técnico de registros médicos	Paciente	Papel	Cada vez que el paciente necesite ser valorado por un especialista

Actividad:	Asignar cupo de triaje

Flujo de Información

Si existe cupo disponible para triaje la enfermera escribe en el registro diario de pacientes los datos personales del paciente y le indica que espere hasta ser atendido por el médico.

Artefacto	Emisor	Receptor	Formato	Frecuencia
Registro de pacientes	Enfermera	Médico	Papel	Cada vez que

				haya consulta de triaje y asista un paciente

Actividad:	Buscar cupo para interconsulta

Flujo de Información

Si el paciente se presenta con una interconsulta a un servicio que no sea hipertensión, la técnico de registros médicos busca el libro de control de citas y localiza un cupo según la disponibilidad existente.

Artefacto	Emisor	Receptor	Formato	Frecuencia
Libro de control de citas	Técnico de registros médicos	Técnico de registros médicos	Papel	Cada que llegue un paciente a solicitar interconsulta

Actividad:	Buscar cupo de primera o de control

Flujo de Información

Si la cita es para determinado médico el técnico de registros médicos busca el libro de control de citas y localiza un cupo disponible teniendo en cuenta el médico, el tipo de consulta (control o primera) y la fecha de la cita en caso de se haya reflejado,
Si no tiene el médico especificado solo se tienen en cuenta el tipo de consulta y al fecha de la cita en caso de que se haya reflejado.

Artefacto	Emisor	Receptor	Formato	Frecuencia
Orden de apertura de historia clínica	Paciente	Técnico de registros médicos	Papel	Cada vez el paciente necesite tener historia del servicio
Orden de cita	Paciente	Técnico de registros médicos	Papel	Cada vez que el paciente necesite una cita
Libro de control de citas	Técnico de registros médicos	Técnico de registros médicos	Papel	Cada que llegue un paciente a solicitar una cita de primera o de control

REGLAS DEL NEGOCIO

Regla	Actividad

Ninguna	Ninguna
REFERENCIAS A PROCEDIMIENTOS	
Procedimiento	**Descripción**
Ninguno	Ninguno

2.4 Propuesta del sistema

De acuerdo con los estudios realizados y luego de materializar un profundo análisis del objeto de estudio, se ha concebido implementar un sistema para gestionar la información generada y manejada en las instituciones hospitalarias de acuerdo a las necesidades. El sistema debe entrar en función desde que el paciente llega a los hospitales, hasta que se retira del mismo, registrando todos los servicios que se le presten al paciente durante su estancia en la institución.

Es imprescindible además que no solamente los pacientes como principales beneficiarios de este sistema, noten una mejoría importante en los procesos del negocio, sino que además para los especialistas y los demás usuarios del software, sea una ventaja su instauración y puesta en práctica. Para lograr esto, los requisitos funcionales se han capturado partiendo de la necesidad real de los procesos de negocio de las instituciones hospitalarias.

2.4.1 Especificación de los requerimientos de software.

Un requerimiento es una condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen. (25)

2.4.1.1 Requerimientos Funcionales.

Los requerimientos funcionales del sistema expresan las operaciones que éste debe implementar para satisfacer a los clientes. El sistema consta de dos servicios, asignar cita en el servicio y en la central de citas donde se crean y se buscan las citas para charla, primera, control, triaje e interconsulta.

1- Asignar cita en el servicio

1.1 Crear cita de primera en el servicio

1.2 Crear cita de control en el servicio

1.3 Crear cita para charla en el servicio

1.4 Crear cita para triaje especializado en el servicio

1.5 Crear cita para interconsulta en el servicio

1.6 Buscar desde el servicio solicitudes de interconsulta

1.7 Buscar cita para consulta desde el servicio

1.8 Buscar cita para triaje especializado desde el servicio

1.9 Buscar cita para charla desde el servicio

1.10 Pasar asistencia a charla

2- Asignar cita en la central de cita

2.1 Crear cita de primera en la central de citas

2.2 Crear cita de control en la central de citas

2.3 Crear cita para charla en la central de citas

2.4 Crear cita para triaje especializado en la central de citas

2.5 Crear cita para interconsulta en la central de citas

2.6 Buscar solicitudes de interconsulta desde la central de citas

2.7 Buscar cita para consulta desde la central de citas

2.8 Buscar cita para triaje especializado desde la central de citas

2.9 Buscar cita para charla desde la central de citas

2.4.1.2 Requerimientos No Funcionales.

- **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

- **Seguridad**

Se mantendrá seguridad y control a nivel de usuario, garantizando el acceso de los mismos sólo a los niveles establecidos de acuerdo a 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.

Se registrarán 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.

El sistema proporcionará un registro de actividades (log) de cada usuario en el sistema.

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.

El sistema permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.

- **Rendimiento**

El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria.

El sistema respetará buenas prácticas de programación para incrementar el rendimiento en operaciones costosas para la máquina virtual como la creación de objetos.

- **Soporte**

Se permitirá la creación de usuarios, otorgamiento de privilegios y roles, asignación de perfiles y activación de permisos por direcciones IP.

Se permitirá 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.

Se permitirá el chequeo de las operaciones y acceso de los usuarios al sistema. Se permitirá establecer parámetros de configuración del sistema y actualización de nomencladores.

- **Hardware**

Estaciones de trabajo

En la solución se incluyen estaciones de trabajo para las consultas del Sistema de Información Hospitalaria alas HIS, las que necesitan capacidad de hardware que soporte un Sistema Operativo que cuente con un navegador actualizado y que siga los estándares Web (se recomienda IE 7 o superior o Firefox 2.x).

Por lo que se escogieron estaciones de trabajo de 256Mb de memoria RAM y un microprocesador de 2.0Hz 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.

- **Software**

El sistema debe correr en los siguientes Sistemas Operativos Windows, Unix y Linux, utilizando la plataforma JAVA (Java virtual machine, JBoss AS y PostgreSql).

El sistema deberá disponer de un navegador web, estos pueden ser IE 7 o superior, Opera 9 superior, Google chrome 1 o superior y Firefox 2 o superior.

- **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.

- **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.

- **Interfaz**

Interfaces de usuario

Las ventanas del sistema contendrán claro y bien estructurados los datos, además de permitir 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 de software

Se interactuará con el sistema ALAS-PACS para realizar solicitudes y obtener resultados de estudios radiológicos e imagenológicos.

Interfaces de comunicación

Para el intercambio electrónico de datos entre aplicaciones se usará el protocolo HL7.

El sistema usará el formato estándar WSDL para la descripción de los servicios web.

El sistema implementará mecanismos de encriptación de datos para el intercambio de información con sistemas externos.

El sistema utilizará mecanismos de compactación de los datos que se intercambiarán con sistemas externos con el objetivo de minimizar el tráfico en la red y economizar el ancho de banda.

- **Portabilidad**

El producto podrá ser utilizado bajo los Sistemas Operativos Linux o Windows.

2.5 Modelo de casos de uso del sistema

Los Casos de Uso no son parte del diseño (cómo), sino parte del análisis (qué). De forma que al ser parte del análisis ayudan a describir qué es lo que es sistema debe hacer. Los Casos de Uso son qué hace el sistema desde el punto de vista del usuario. Es decir, describen un uso del sistema y cómo este interactúa con el usuario.

2.5.1 Definición de actores

- **Técnico (a) de registros y estadísticas de salud de central de citas**

Encargado (a) de asignar, buscar, modificar y eliminar una cita para cualquier servicio del hospital.

- **Técnico (a) de registros y estadísticas de salud de consulta externa**

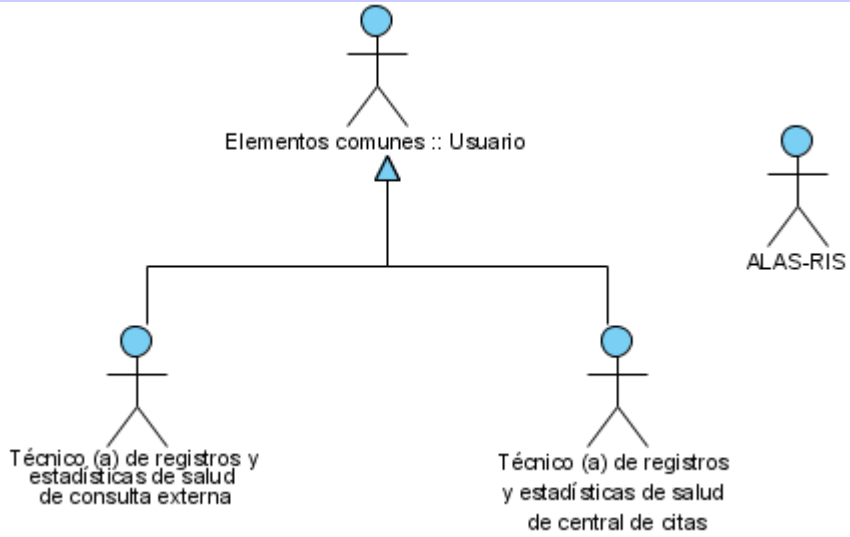
Encargado (a) de asignar, buscar, modificar y eliminar cita para el servicio al cual pertenece, además registrar asistencia a charla.

- **ALAS-RIS**

Sistema externo con el que se comunica el módulo para la gestión de las citas de los estudios radiológicos e imagenológicos.

2.5.1.1 Vista global de actores

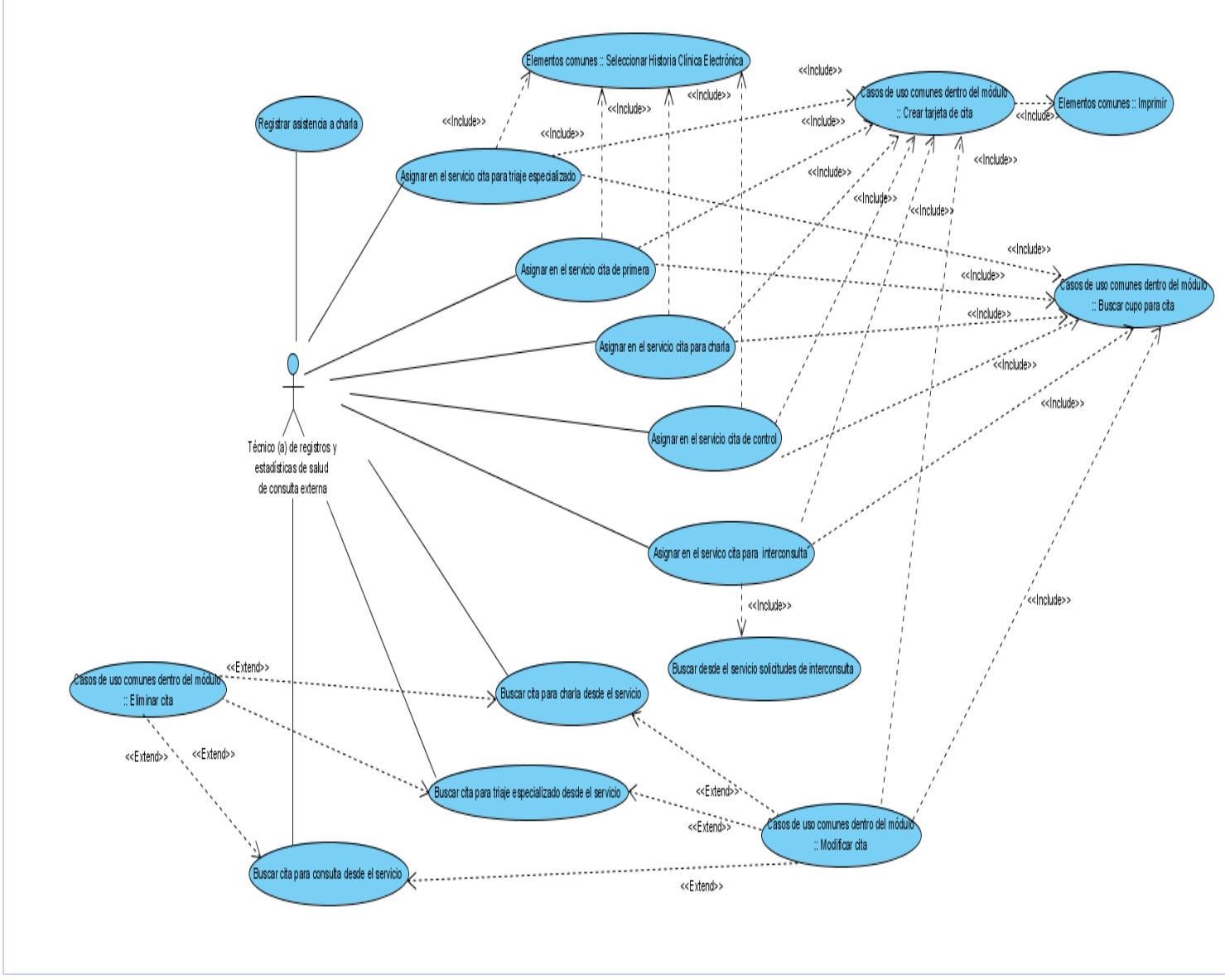
Diagrama de Actores



2.5.1.2 Diagrama de casos de uso

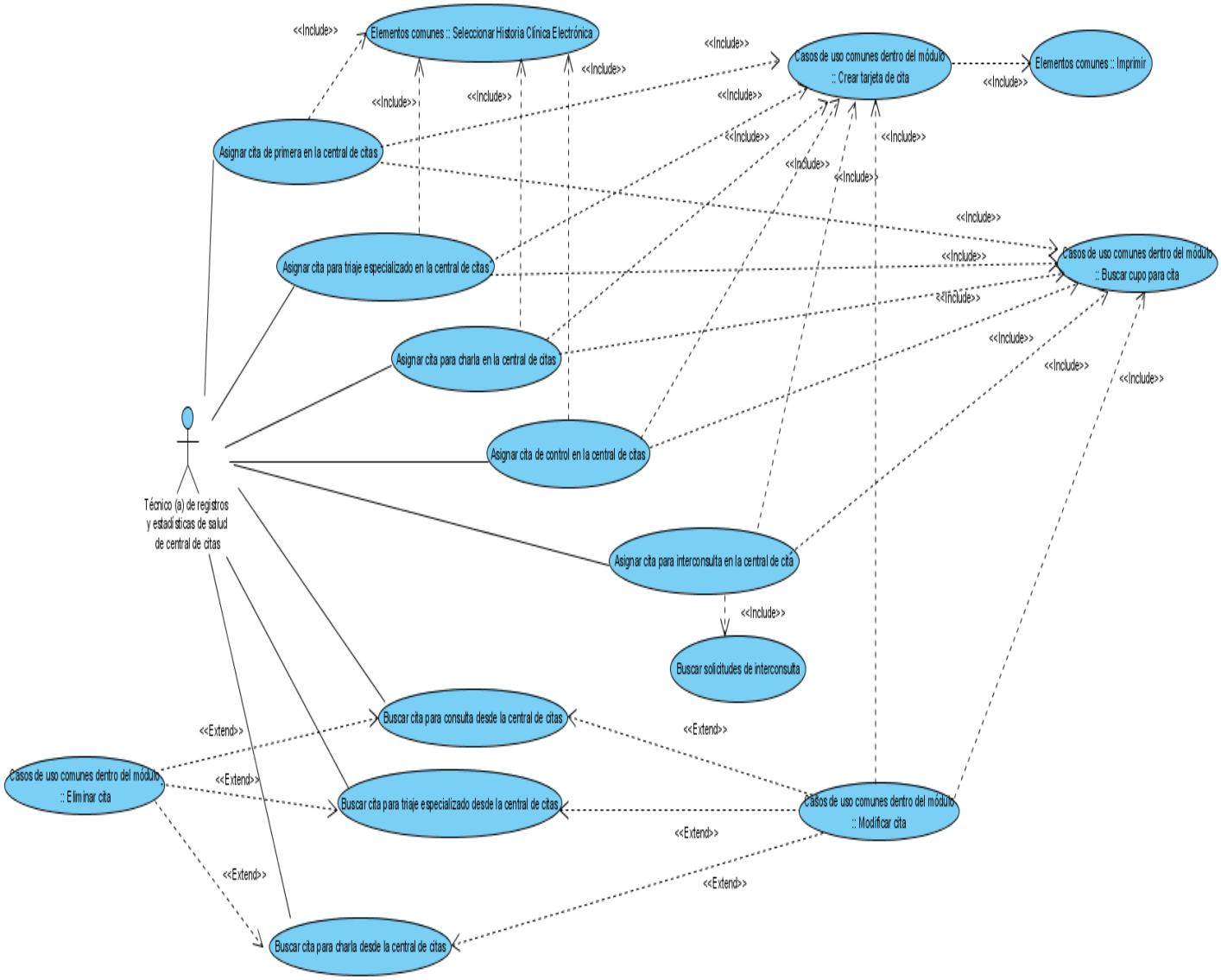
- **Asignar cita en el servicio**

Diagrama de Casos de Uso



- **Asignar cita en la central de citas**

Diagrama de Casos de Uso



2.5.1.3 Descripción textual de los casos de uso

- **Asignar cita en el servicio**

1. Asignar en el servicio cita de primera

CUS_ **Asignar en el servicio cita de primera**	
Propósito:	Asignar una cita de primera en el servicio
Actores:	Técnico (a) de registros y estadísticas de salud de consulta externa
Resumen:	El caso de uso inicia cuando el actor accede a la opción Asignar cita de primera, el sistema brinda la posibilidad de seleccionar especialidad, médico, y fecha para citar al paciente, el actor selecciona la especialidad, el médico, la fecha y confirma la cita, el sistema crea la cita de primera y la tarjeta de cita, el caso de uso termina.
Precondición:	No existe
Poscondición:	Se creó una Cita para consulta
Referencias:	RF-2.3.1

2. Asignar en el servicio cita de control

CUS_ **Asignar en el servicio cita de control**	
Propósito:	Asignar una cita de control en el servicio
Actores:	Técnico (a) de registros y estadísticas de salud de consulta externa
Resumen:	El caso de uso inicia cuando el actor accede a la opción Asignar cita de control, el sistema brinda la posibilidad de seleccionar especialidad, médico y fecha para citar al paciente, el actor selecciona la especialidad, el médico, la fecha y confirma la cita, el sistema crea la cita de control y la tarjeta de cita, el caso de uso termina.
Precondición:	No existe
Poscondición:	Se creó una Cita para consulta.
Referencias:	RF-2.3.2

3. Buscar cita para triaje especializado desde el servicio

***CUS*_ Buscar cita para triaje especializado desde el servicio**	
Propósito:	Buscar cita para triaje especializado desde el servicio
Actores:	Técnico (a) de registros y estadísticas de salud de consulta externa
Resumen:	El caso de uso inicia cuando el actor accede a la opción Buscar Cita para triaje especializado, el sistema brinda la posibilidad de introducir criterios de búsqueda para localizar la cita, el actor introduce los datos que considera como criterios para realizar una búsqueda, el sistema busca y muestra las citas que cumplen con los criterios de búsqueda, el actor puede seleccionar la opción de modificar y eliminar una Cita para triaje especializado, el caso de uso termina.
Precondición:	No existe
Poscondición:	Se buscó Cita para triaje especializado dado criterios.
Referencias:	RF-2.3.8

4. Buscar cita para charla desde el servicio

***CUS*_ Buscar cita para charla desde el servicio**	
Propósito:	Buscar cita para charla desde el servicio
Actores:	Técnico (a) de registros y estadísticas de salud de consulta externa
Resumen:	El caso de uso inicia cuando el actor accede a la opción Buscar Cita para charla, el sistema brinda la posibilidad de introducir criterios de búsqueda para localizar la cita, el actor introduce los datos que considera como criterios para realizar una búsqueda, el sistema busca y muestra las citas que cumplen con los criterios de búsqueda, el actor puede seleccionar la opción de modificar y eliminar una Cita para charla, el caso de uso termina.
Precondición:	No existe
Poscondición:	Se buscó Cita para charla dado criterios.
Referencias:	RF-2.3.9

- **Asignar cita en la central de citas**

1. Asignar cita de primera en la central de citas

CUS_ **Asignar cita de primera en la central de citas**	
Propósito:	Asignar cita de primera en la central de citas
Actores:	Técnico (a) de registros y estadísticas de salud de central de citas
Resumen:	El caso de uso inicia cuando el actor accede a la opción Asignar cita de primera, el sistema brinda la posibilidad de seleccionar servicio, especialidad, médico, y fecha para citar al paciente, el actor selecciona el servicio, la especialidad, el médico, la fecha y confirma la cita, el sistema crea la cita de primera y la Tarjeta de cita, el caso de uso termina.
Precondición:	No existe
Poscondición:	Se creó una Cita para consulta.
Referencias:	RF-2.4.1

2. Buscar cita para triaje especializado desde la central de citas

CUS_ **Buscar cita para triaje especializado desde la central de citas**	
Propósito:	Buscar cita para triaje especializado desde la central de citas
Actores:	Técnico (a) de registros y estadísticas de salud de central de citas
Resumen:	El caso de uso inicia cuando el actor accede a la opción Buscar Cita para triaje especializado, el sistema brinda la posibilidad de introducir criterios de búsqueda para localizar la cita, el actor introduce los datos que considera como criterios para realizar una búsqueda, el sistema busca y muestra las citas que cumplen con los criterios de búsqueda, el actor puede seleccionar la opción de modificar y eliminar una Cita para triaje especializado, el caso de uso termina.
Precondición:	No existe
Poscondición:	Se buscó Cita para triaje especializado dado criterios.
Referencias:	RF-2.4.8

3. Buscar cita para charla desde la central de citas

***CUS*_ Buscar cita para charla desde la central de citas**	
Propósito:	Buscar cita para charla desde la central de citas
Actores:	Técnico (a) de registros y estadísticas de salud de central de citas
Resumen:	El caso de uso inicia cuando el actor accede a la opción Buscar Cita para charla, el sistema brinda la posibilidad de introducir criterios de búsqueda para localizar la cita, el actor introduce los datos que considera como criterios para realizar una búsqueda, el sistema busca y muestra las citas que cumplen con los criterios de búsqueda, el actor puede seleccionar la opción de modificar y eliminar una Cita para charla, el caso de uso termina.
Precondición:	No existe
Poscondición:	Se buscó Cita para charla dado criterios.
Referencias:	RF-2.4.9

Conclusiones

En el presente capítulo se han analizado los procesos de negocio que son objeto de automatización. Se describió el sistema propuesto, se expusieron los modelo del negocio del mismo, los requerimientos funcionales y no funcionales. Además se mostraron los diagramas de casos de uso que se han modelado.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

En el presente capítulo se exponen los diagramas de clases de análisis que participan en la realización de los casos de usos del primer ciclo de desarrollo. Además se representa los distintos diagramas de clases referentes al diseño y los diagramas de secuencia de los casos de usos. Se da una breve descripción de las clases entidades y controladoras usadas en el flujo de trabajo Análisis y Diseño.

3.1 Arquitectura

Entre las arquitecturas más comunes se encuentran la Monolítica, Cliente-servidor , la Arquitectura en tres capas o niveles y perteneciente a ésta, el patrón Modelo Vista Controlador (MVC) es la que se utilizará, debido a que permite aislar la capa de modelo de datos, la lógica del negocio, y la vista o interfaz del sistema. Esto beneficia y permite la creación de grandes sistemas, mejorar la organización del código, distribuir mejor el esfuerzo, una gran reutilización de código, facilita el soporte de las aplicaciones. Todo esto es indispensable debido a la gran complejidad del sistema a implementar .

El patrón MVC se ve reflejado de la siguiente forma:

Vistas o Interfaz

Esta capa incluye todas las páginas interfaces que interactúan con el usuario, dichas páginas están conformadas por componentes JSF y componentes de la librería Richfaces 3.2.0 G, los cuales realizan solicitudes a la capa controladora mediante eventos que envían estos componentes cuando el cliente realiza alguna acción en las mismas, gestionados por un servlet el cual además recibe la respuesta del controlador dado un evento determinado y la refleja en la interfaz. Para lograr la integración con las otras capas se utiliza el framework Seam.

Controlador o Lógica del negocio

En esta capa se encuentra lo referente a la lógica del negocio del sistema estando representada por clases controladoras las cuales son las encargadas de recibir y gestionar los eventos enviados por las interfaces y de acorde al carácter de la petición ejecutar una acción; ejemplo la de modificar los datos del modelo. Para la integración de esta capa con las demás se utiliza Seam.

Modelo de datos

En esta capa se encuentran las entidades persistentes, que no son clases java o Bean que con el uso de la implementación de JPA de Hibernate 3.3 se realiza mapeo de objeto relacional, logrando aislar la implementación del gestor de base de datos además de lograr una forma de conectar mediante Seam estas entidades con la capa lógica del negocio.

3.2 Modelo de diseño.

El objetivo fundamental de este flujo de trabajo es traducir los requisitos funcionales a una descripción de cómo quedará implementado el sistema. Debe ser lo suficientemente robusto para no permitir ambigüedades. El objetivo final de este flujo de trabajo es producir un Modelo Lógico del sistema a implementar.

Propósitos del diseño:

- Adquirir una comprensión en profundidad de los aspectos relacionados con los requisitos no funcionales, restricciones impuestas por el lenguaje de programación, el Sistema Operativo donde se va a ejecutar, el tipo de interfaz, entre otras.

- Punto de partida para la implementación capturando requisitos de las clases de análisis.

- Ser capaz de visualizar y razonar acerca del diseño usando una notación común.

- Crear una abstracción sin costuras de la implementación del sistema, en el sentido de que la implementación es un refinamiento directo del diseño que rellena lo existente sin cambiar la estructura. Esto permite la utilización de tecnologías como la generación de código y la ingeniería de ida y vuelta entre el diseño y la implementación.

3.2.1 Patrones de diseño

Entre los patrones de diseño más utilizados se encuentran los patrones GRASP que son patrones de software para la asignación general de responsabilidades y describen los principios fundamentales de

diseño de objetos para la asignación de responsabilidades. Se pueden destacar como patrones fundamentales:

-Experto: Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo.

-Creador: El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases.

-Alta Cohesión: Plantea que la información que almacena una clase debe de ser coherente y está en la mayor medida de lo posible relacionada con la clase.

-Bajo Acoplamiento: Tratar de independizar las clases entre sí lo menos posible, con el objetivo de que si existiera algún cambio sobre una de ellas no incida directamente sobre las otras.

-Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

3.2.2 Realización de casos de uso del diseño

Una realización de caso de uso del diseño tiene diagramas interacción que muestran la realización de un flujo o escenarios concretos de un caso de uso, la realización de los casos de uso está organizada en paquetes. A continuación se explica la organización por paquetes del sistema y se muestra el diagrama de paquetes.

Descripción de cada paquete:

Repositorio de clases: estarán almacenadas las clases que se definen en el diseño de acuerdo a la tecnología utilizada en la implementación del módulo citas.

- **Entidades (Entity):** Contiene las entidades autogeneradas que son aquellas que se autogeneran de la base de datos, las personalizadas que son modificadas por los desarrolladores y pueden heredar de las autogeneradas y las comunes que son las entidades utilizadas en común.

- **Sesiones (Session):**

  Controladoras autogeneradas: Clases controladoras autogeneradas por el entorno de desarrollo.

  Controladoras personalizadas: Clases controladoras personalizadas que son modificadas.

  Controladoras del proceso: Clases controladoras propias del proceso.

- **Vistas:** contienen todas las vitas del sistema.

Asignar cita desde el servicio y la central de citas: Contiene los paquetes con las realizaciones de los casos de usos del diseño. El nombre de cada paquete corresponde con el nombre del caso de uso correspondiente, donde por cada caso de uso se define el diagrama de clase correspondiente, conjuntamente con el diagrama de secuencia de los casos de uso atómicos. Ver Figura 3.0

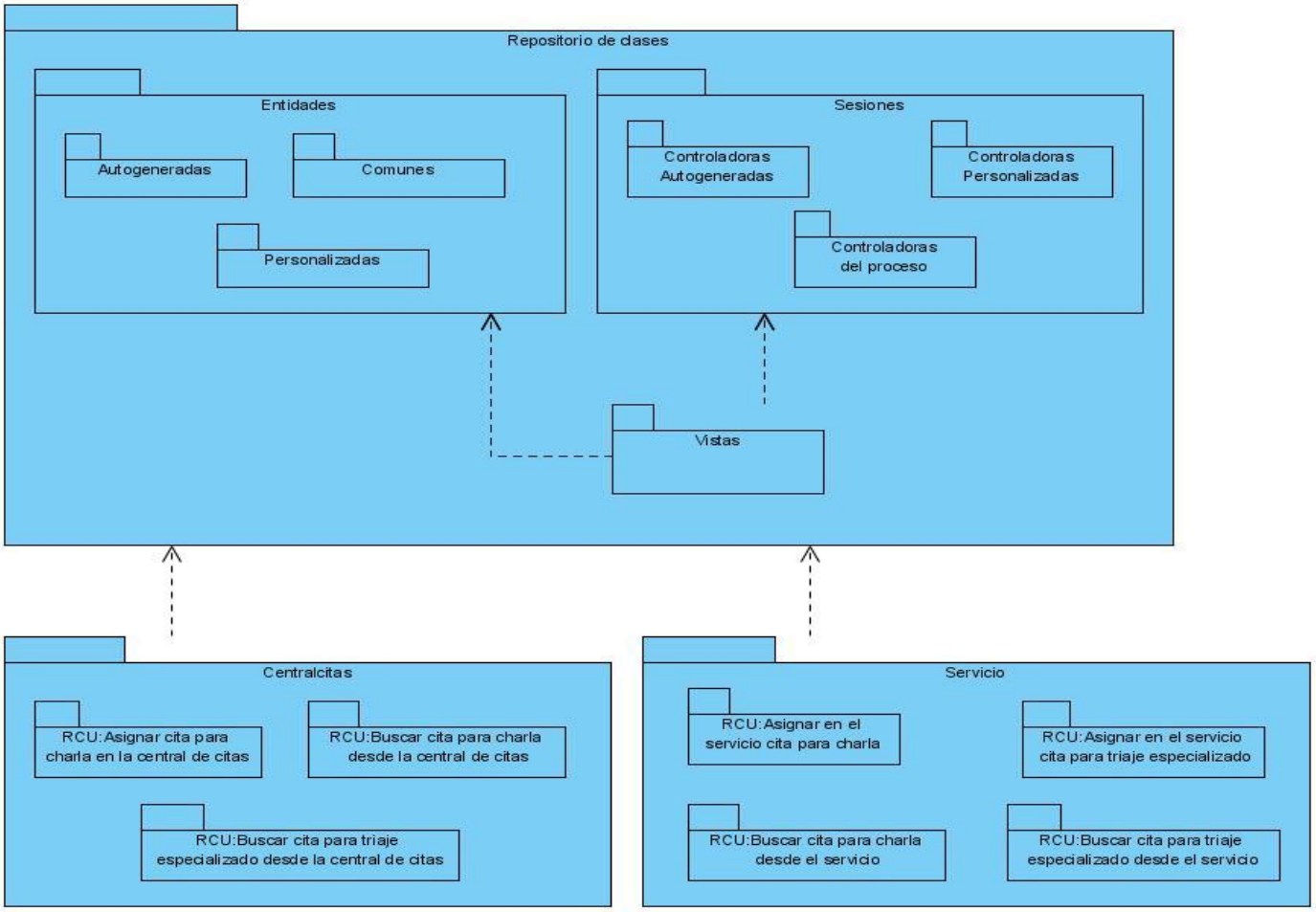


Figura 3.0 Diagrama de paquetes para el caso de uso Asignar cita

Diagramas de Interacción.

El diagrama de interacción, representa la forma en cómo un Cliente (Actor) u Objetos (Clases) se comunican entre sí en petición a un evento. Esto implica recorrer toda la secuencia de llamadas, de donde se obtienen las responsabilidades claramente. Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema. La mayoría de las veces, esto implica modelar instancias concretas o prototípicas de clases, interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento.

Pueden utilizarse para visualizar, especificar, construir y documentar la dinámica de una sociedad particular de objetos, o se pueden utilizar para modelar un flujo de control particular de un caso de uso.

Los diagramas de interacción no son sólo importantes para modelar los aspectos dinámicos de un sistema, sino también para construir sistemas ejecutables por medio de ingeniería directa e inversa.

Diagrama de Secuencia

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en secuencia temporal. Muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por el escenario. En aplicaciones grandes además de los objetos se muestran también los componentes y casos de uso.

El mostrar los componentes tiene sentido ya que se trata de objetos reutilizables, en cuanto a los casos de uso hay que recordar que se implementan como objetos cuyo rol es encapsular lo definido en el caso de uso. Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes. Se forma colocando en primer lugar los objetos que participan en la interacción en la parte superior del diagrama, a lo largo del eje X. Normalmente, se coloca a la izquierda el objeto que inicia la interacción, y los objetos subordinados a la derecha. A continuación, se colocan los mensajes que estos objetos envían y reciben a lo largo del eje Y, en orden de sucesión en el tiempo, desde arriba hasta abajo. Esto ofrece al lector una señal visual clara del flujo de control a lo largo del tiempo.

Los objeto poseen una línea de vida que es la línea discontinua vertical que representa la existencia de un objeto a lo largo de un período de tiempo y un foco de control que es un rectángulo delgado y estrecho que representa el período de tiempo durante el cual un objeto ejecuta una acción, bien sea directamente o a través de un procedimiento subordinado. La parte superior del rectángulo se alinea con el comienzo de la acción; la inferior se alinea con su terminación (y puede marcarse con un mensaje de retorno).

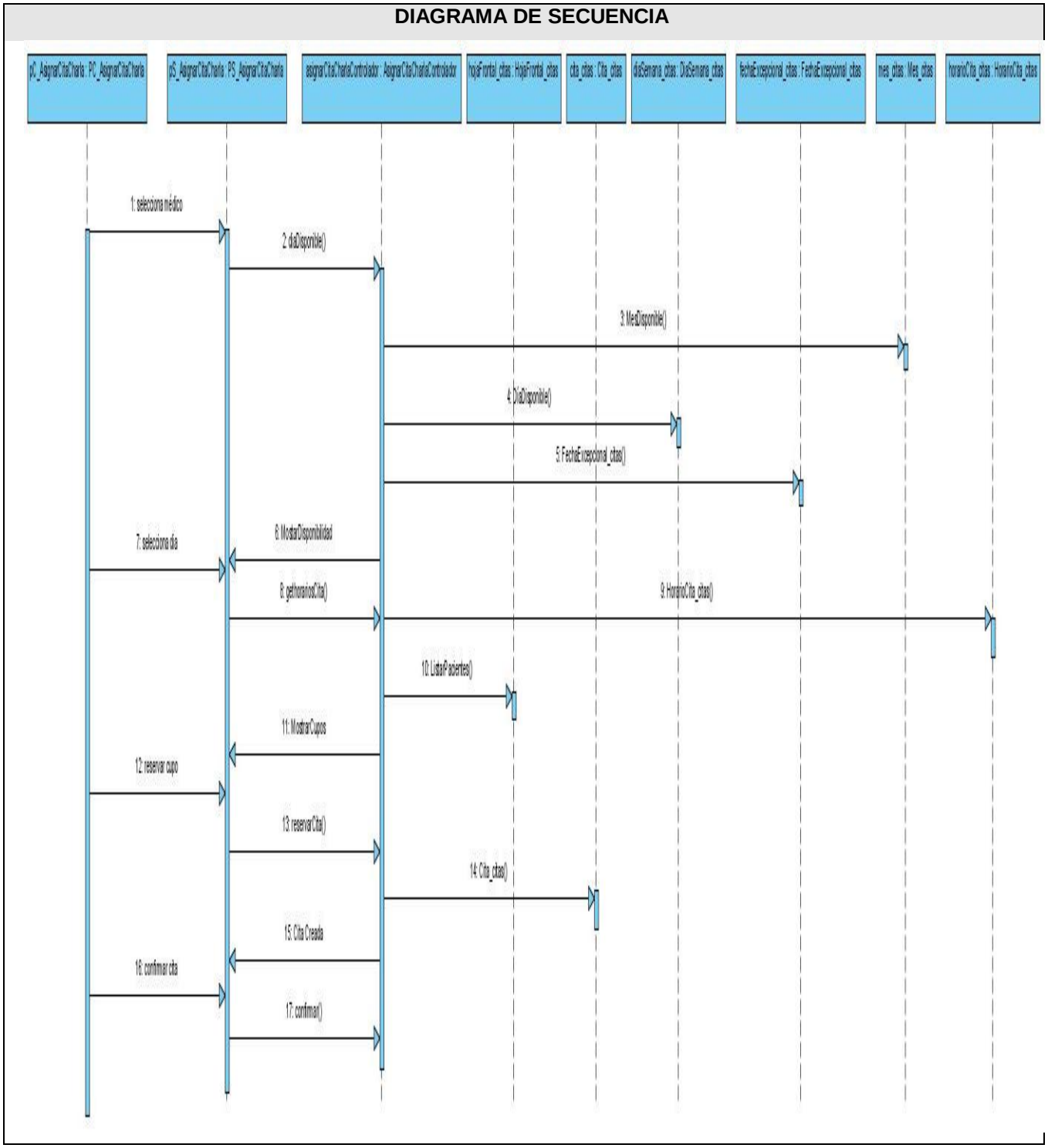


Figura 3.1 DS Asignar Cita de primera desde el servicio.

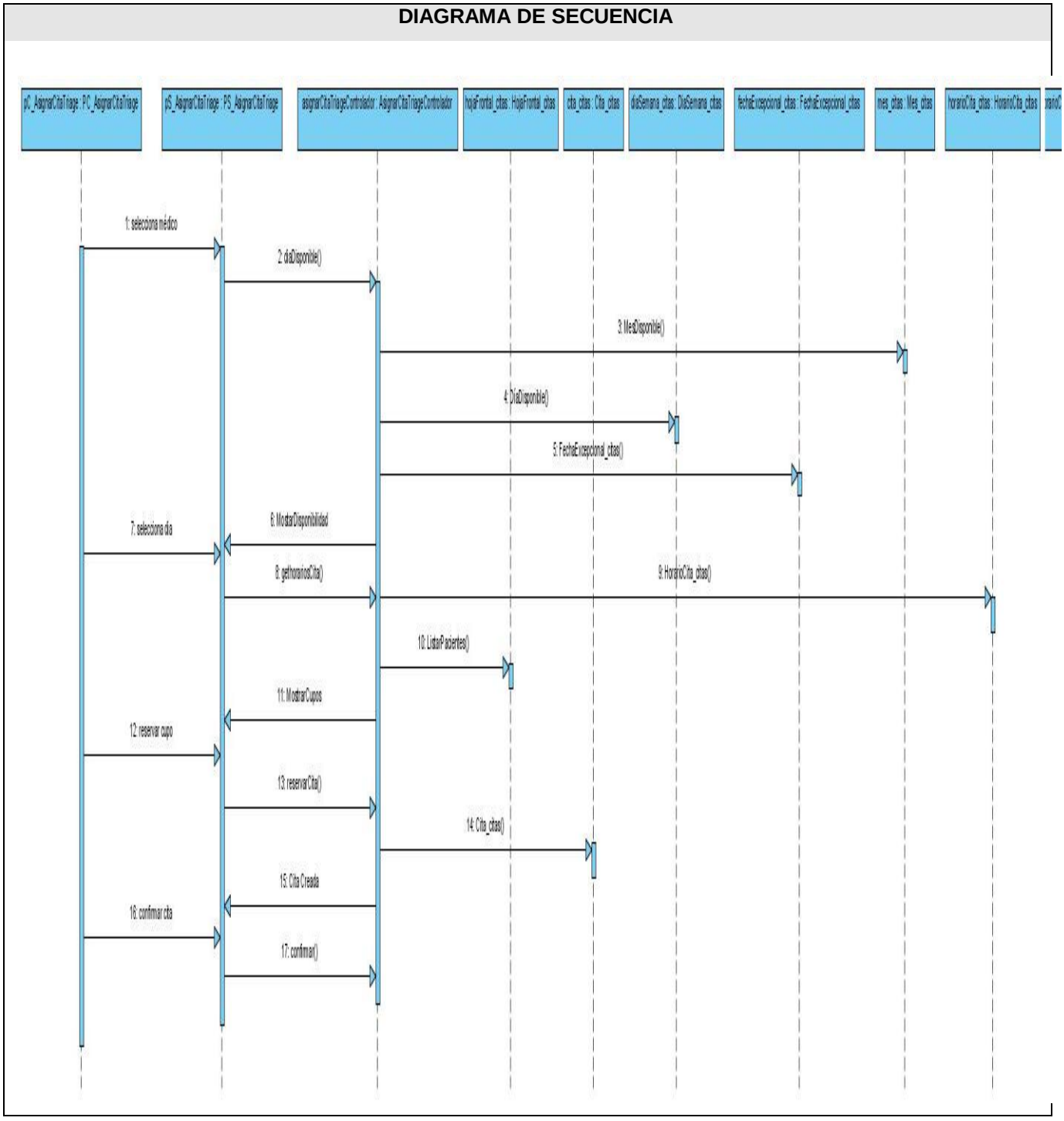


Figura 3.2 DS Asignar Cita de control desde el servicio.

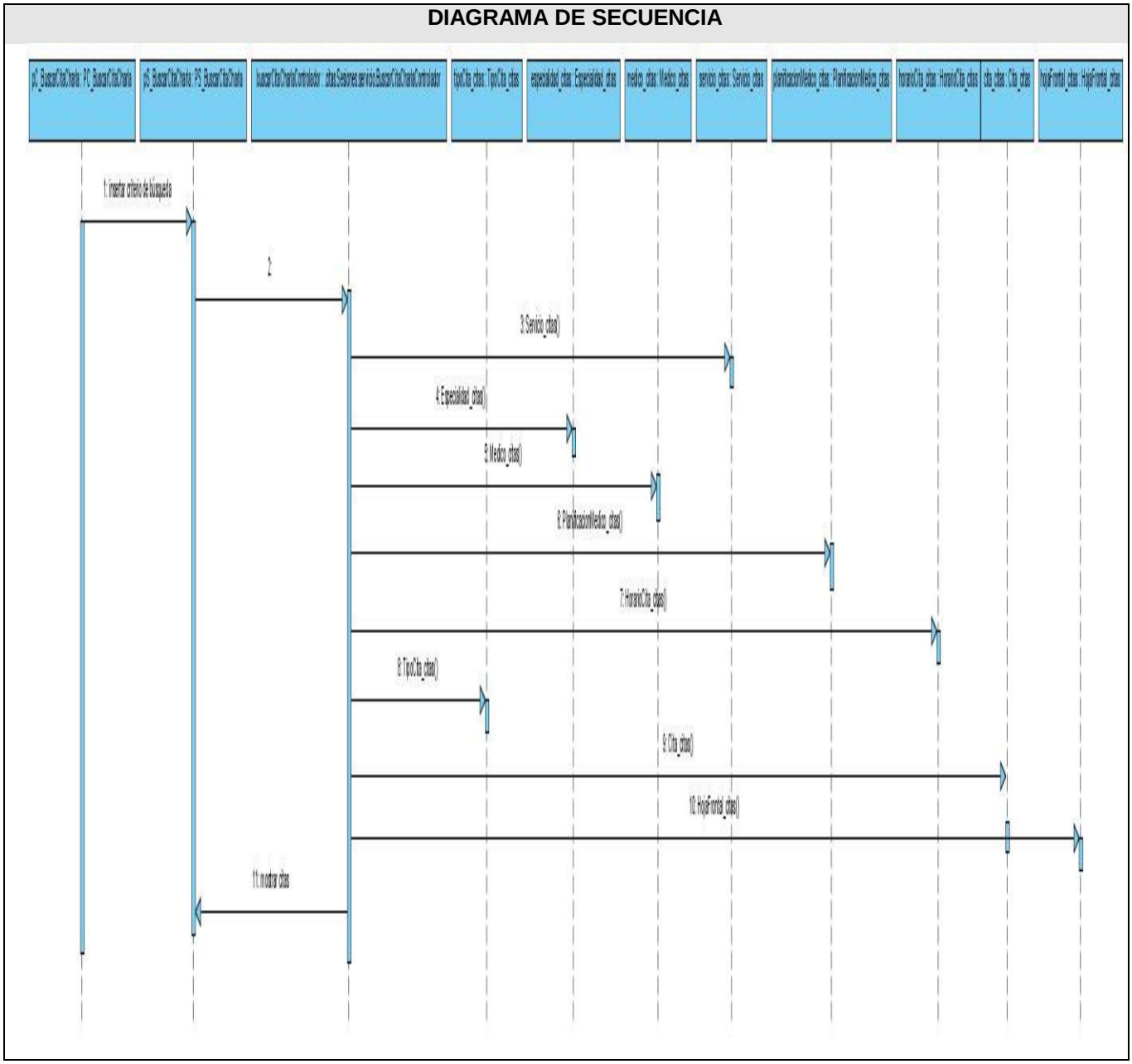


Figura 3.3 DS Buscar Cita para charla desde el servicio.

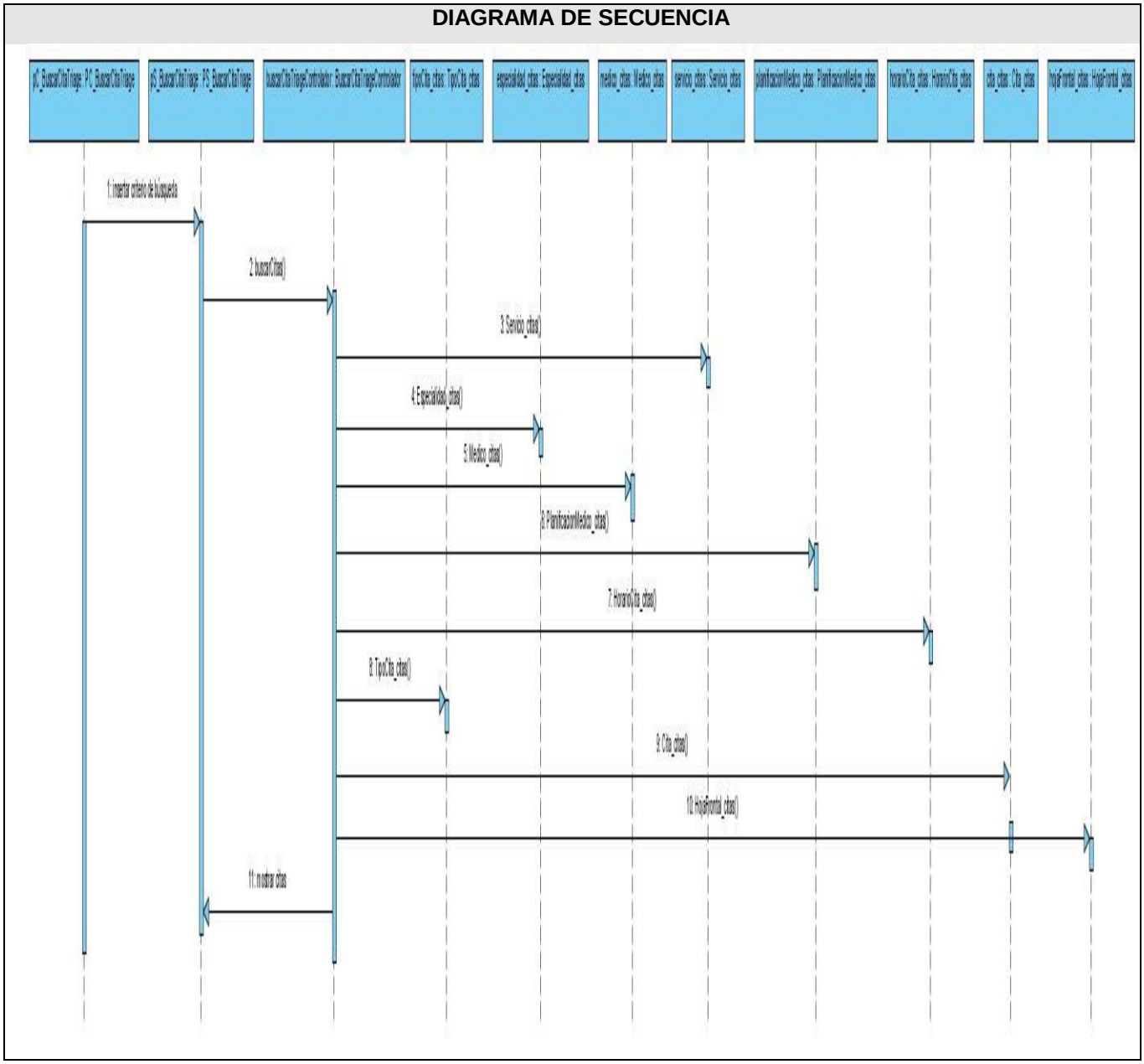


Figura 3.4 DS Buscar Cita para triaje especializado desde el servicio.

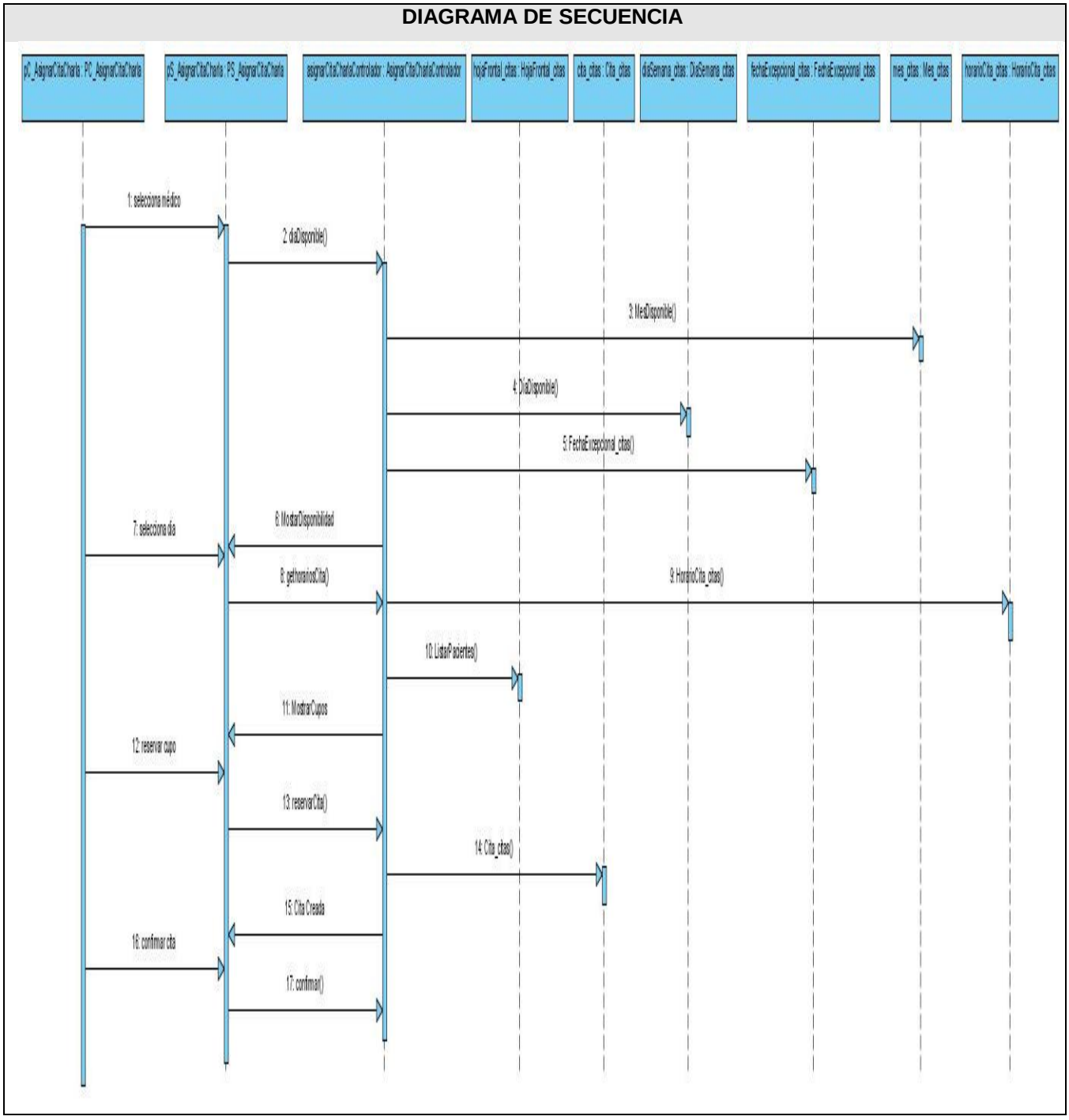


Figura 3.5 DS Asignar Cita de primera desde la central de citas.

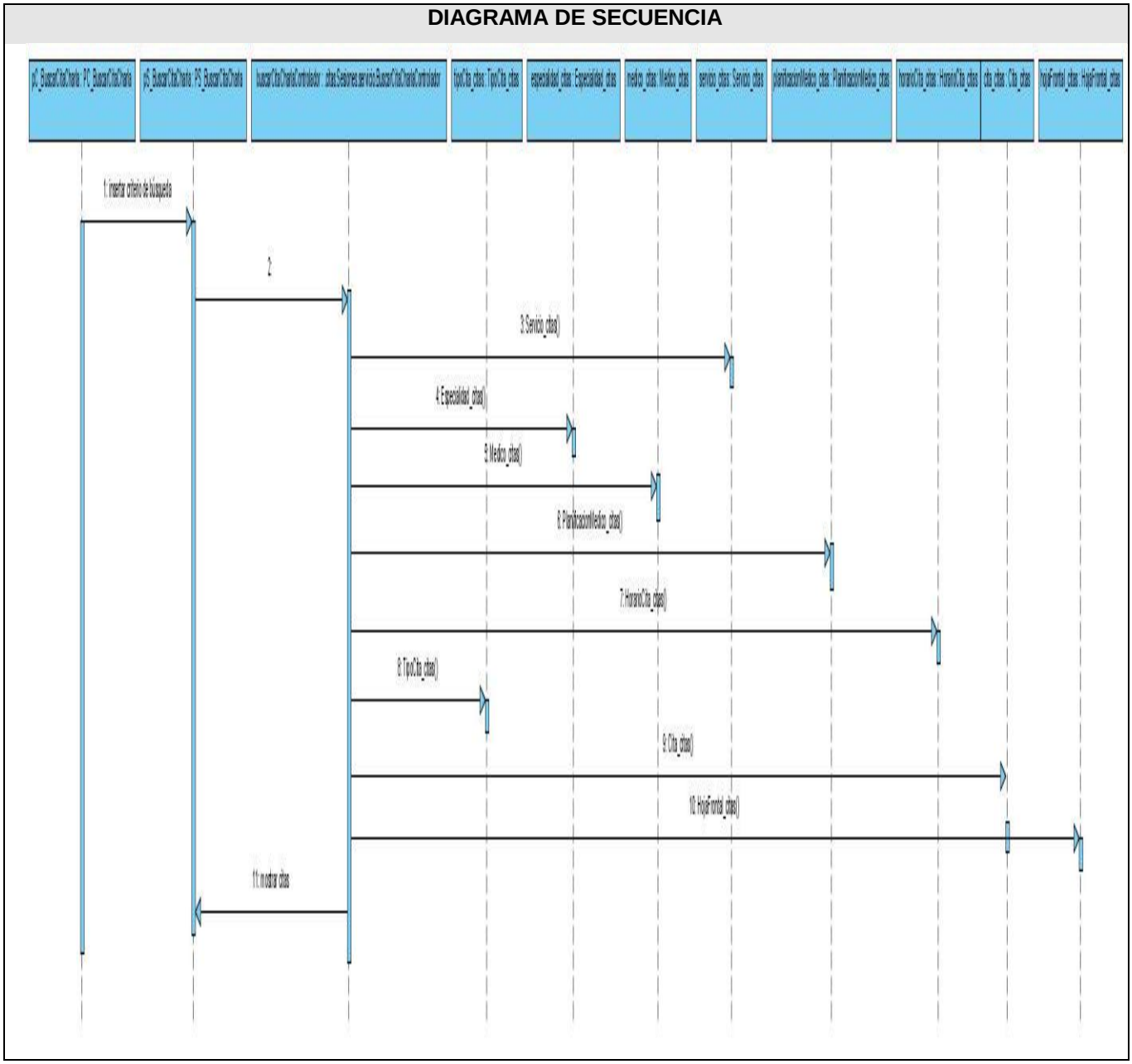


Figura 3.6 DS Buscar Cita para charla desde la central de citas.

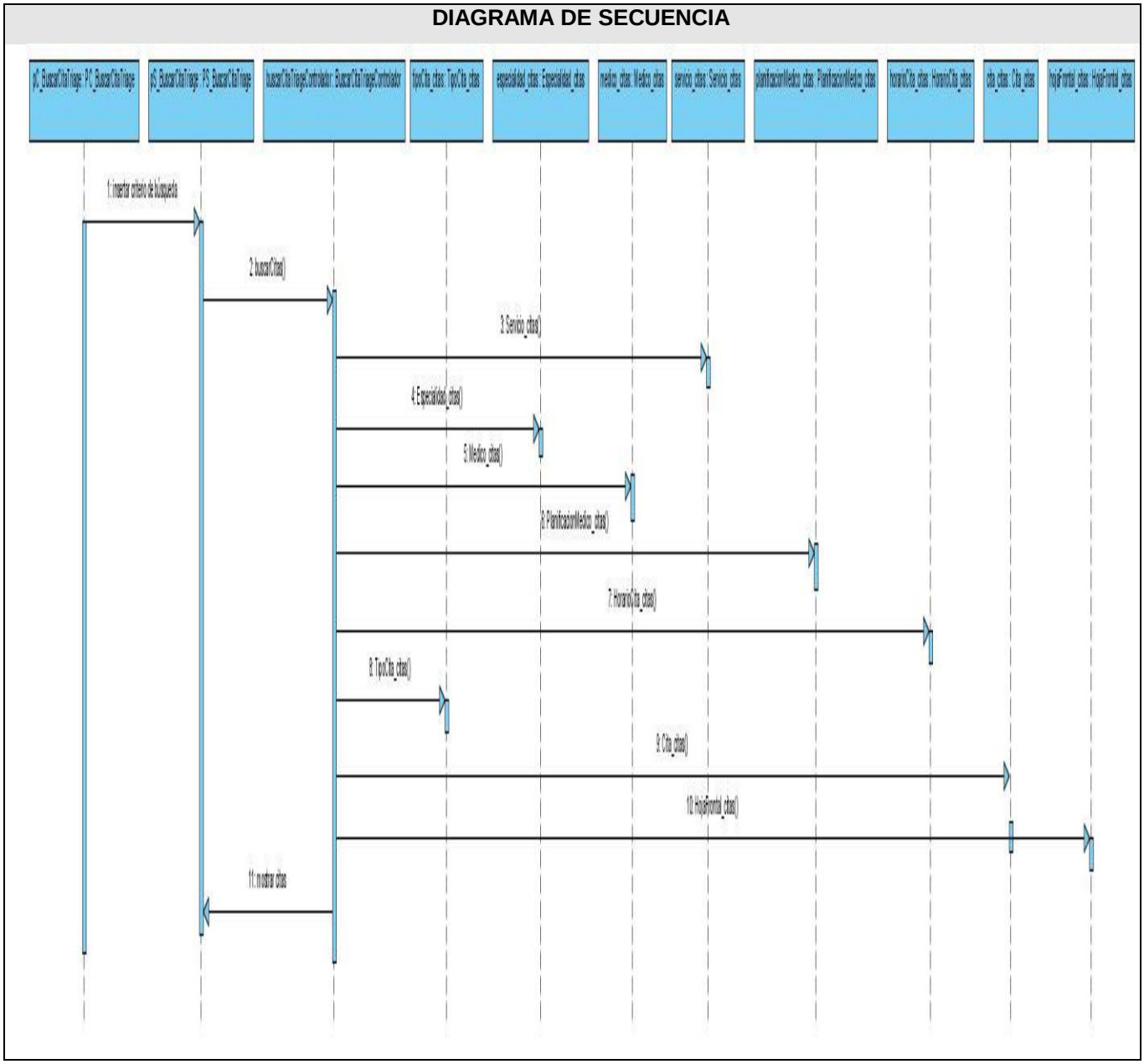


Figura 3.7 DS Buscar Cita para triaje especializado desde la central de citas.

Diagramas de clases del diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

Los diagramas de clases del diseño expuestos anteriormente fueron modelados utilizando estereotipos web.

A continuación se describen los estereotipos web utilizados para modelar los diagramas de clases.

Página servidora: Representa la página Web que tiene código que se ejecuta en el lado del servidor. Las operaciones que se realizan representan las funciones del código y los atributos las variables visibles dentro del alcance de la página.

Página cliente: Es una página Web, con formato XHTML. Son interpretadas por el navegador. Sus atributos son las variables declaradas dentro del script que son accesibles desde cualquier función dentro de la página.

Formulario: Colección de elementos de entrada que son parte de una página cliente. Sus atributos son los elementos de entrada del formulario (input boxes, textarea, radiobuttons, checkboxes y hiddenfields).

Build: Representa una asociación especial que relaciona las páginas cliente con las páginas servidor, es decir, las páginas que se encuentran en el servidor construyen las páginas en el cliente.
Submit: Es la relación que se crea siempre entre una página servidor y un formulario, a través de esta relación el formulario manda los valores de sus campos al servidor, para ser procesados por la página servidor.
Redirect: La página de servidor además de construir una página cliente puede re-direccionar el procesamiento a otra página. Esto se representa con una asociación con el estereotipo <<redirect>>.

Los diagramas de clases del diseño están estructurados por casos de uso, y estos a su vez cuentan con un diagrama de clases, para facilitar su comprensión. Los diagramas de clases de diseño que se presentan en el presente capítulo representan los casos de usos arquitectónicamente significativos.

DIAGRAMA DE CLASES

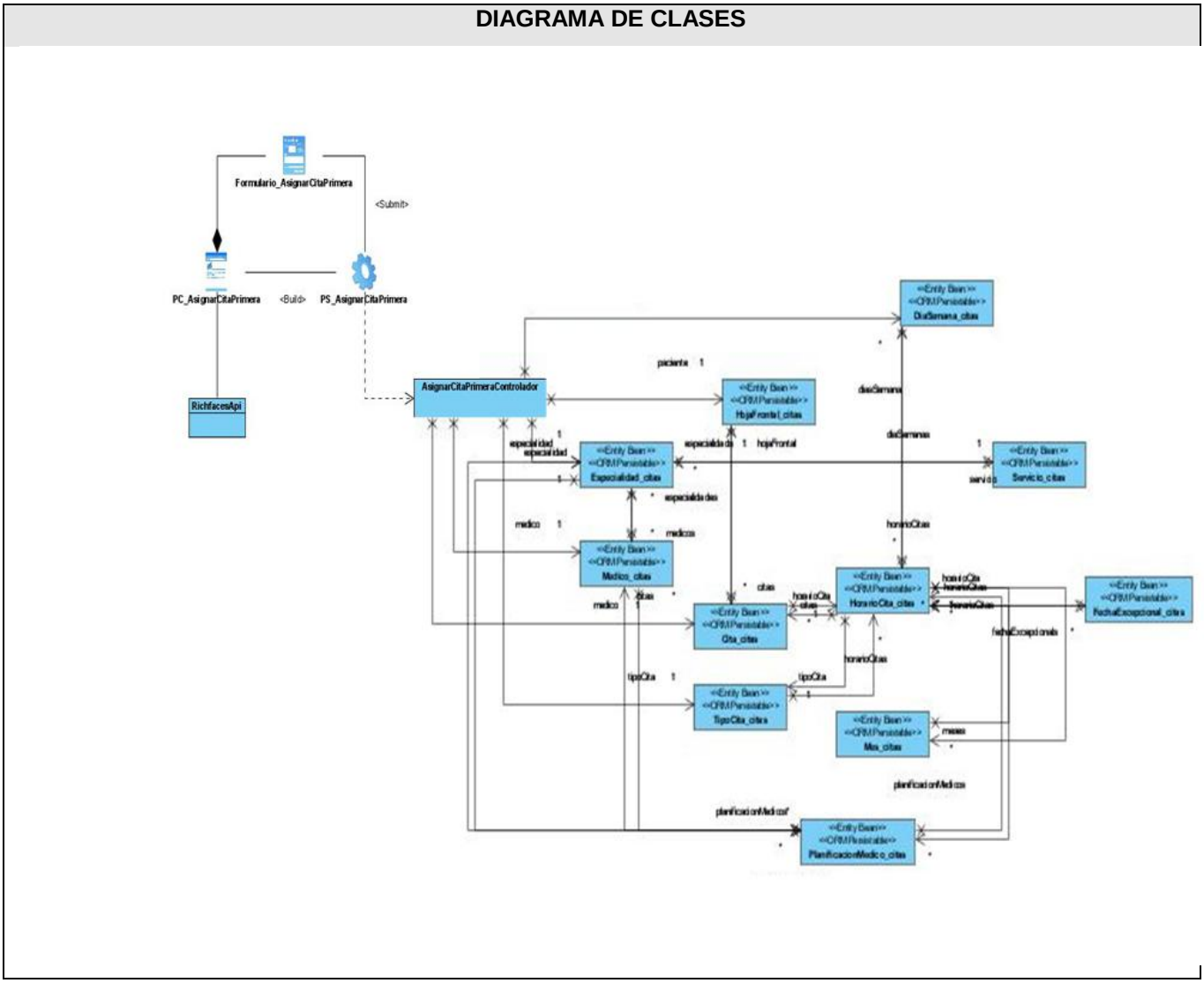


Figura 3.8 DCD Asignar cita primera desde el servicio.

DIAGRAMA DE CLASES

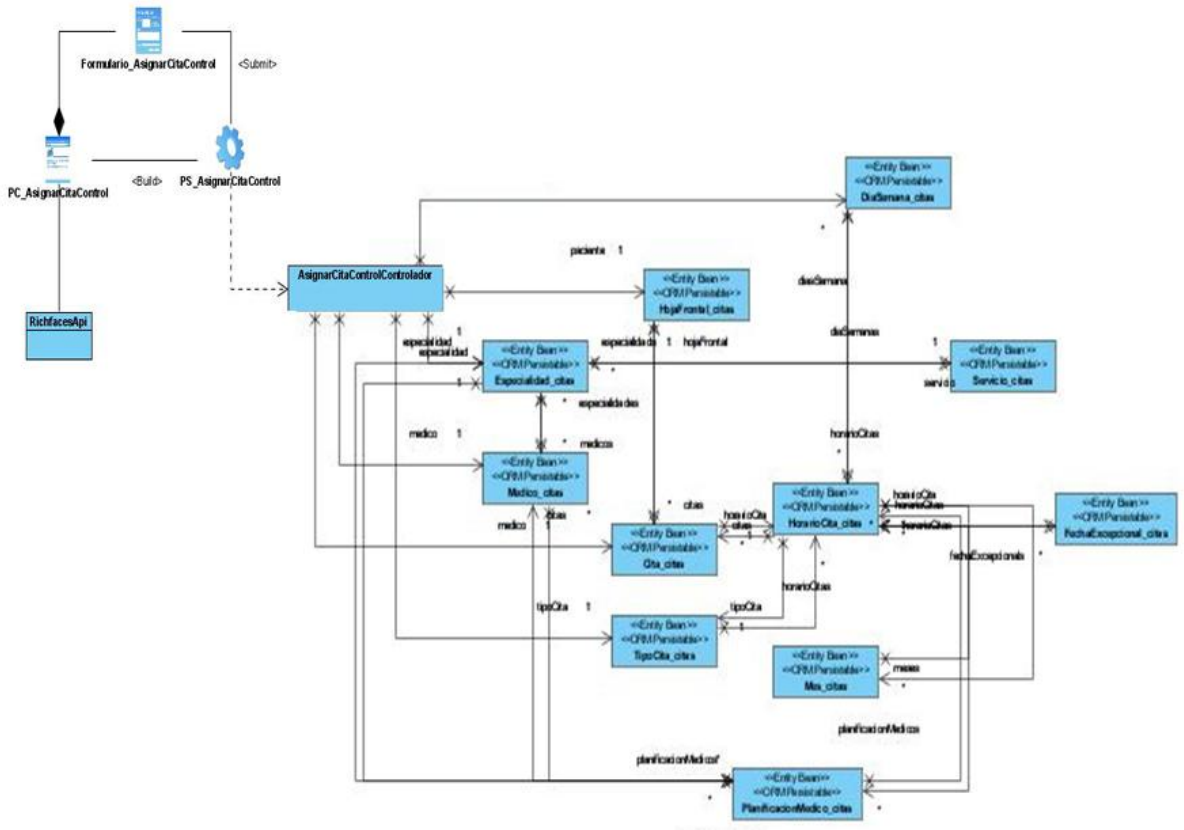


Figura 3.9 DCD Asignar cita control desde el servicio.

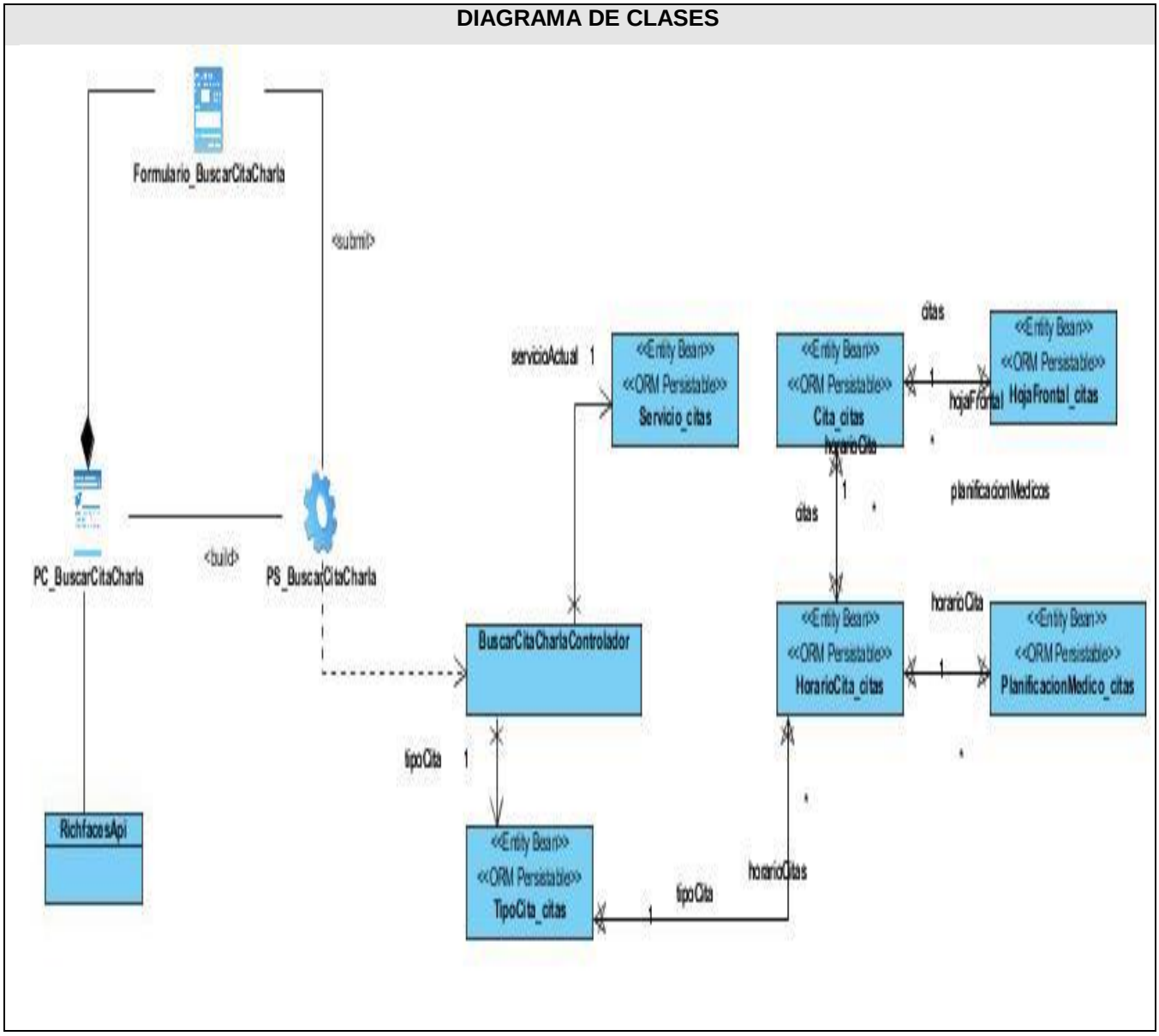


Figura 3.10 DCD Buscar cita charla desde el servicio.

DIAGRAMA DE CLASES

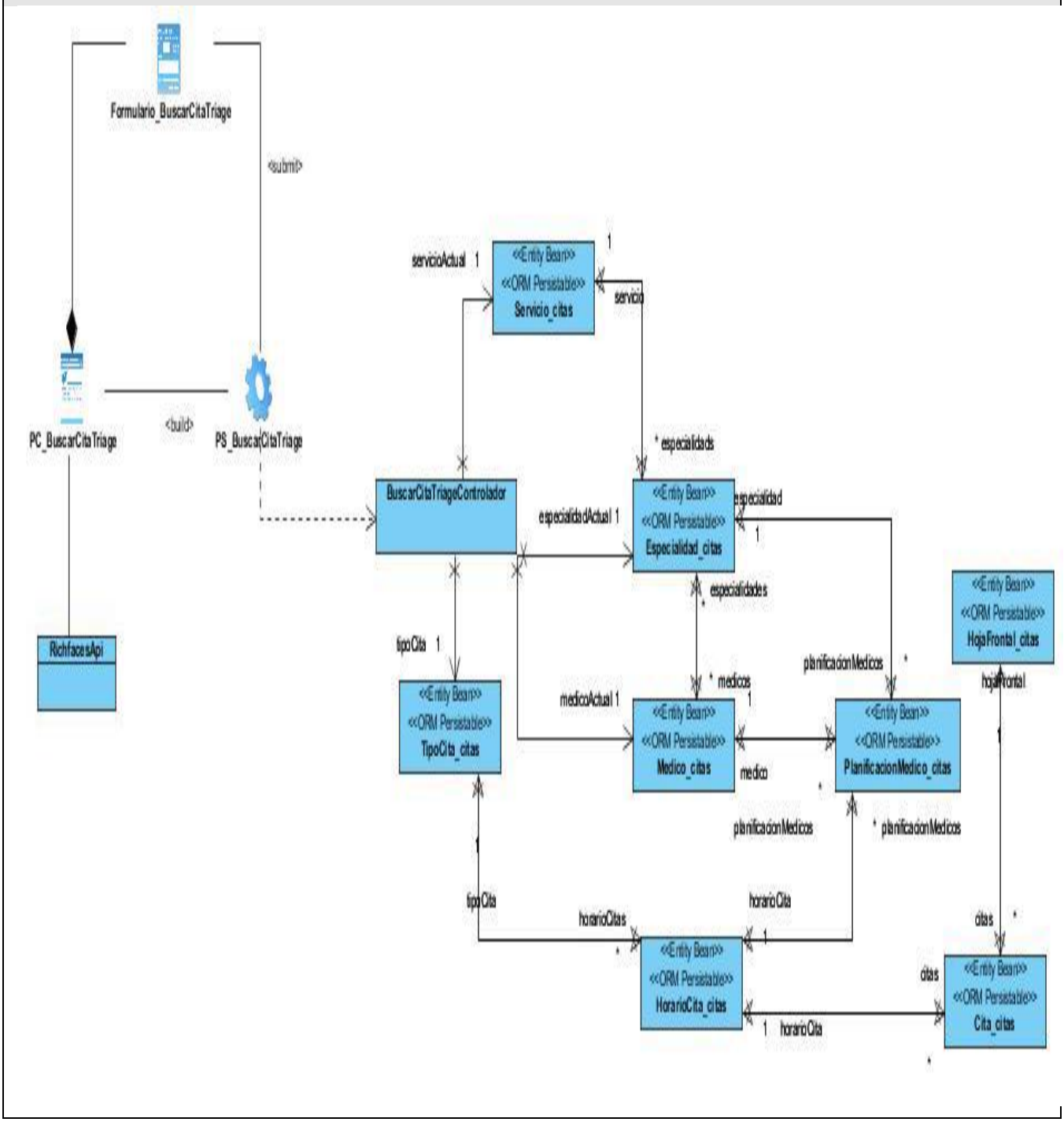


Figura 3.11 DCD Buscar cita triaje desde el servicio.

DIAGRAMA DE CLASES

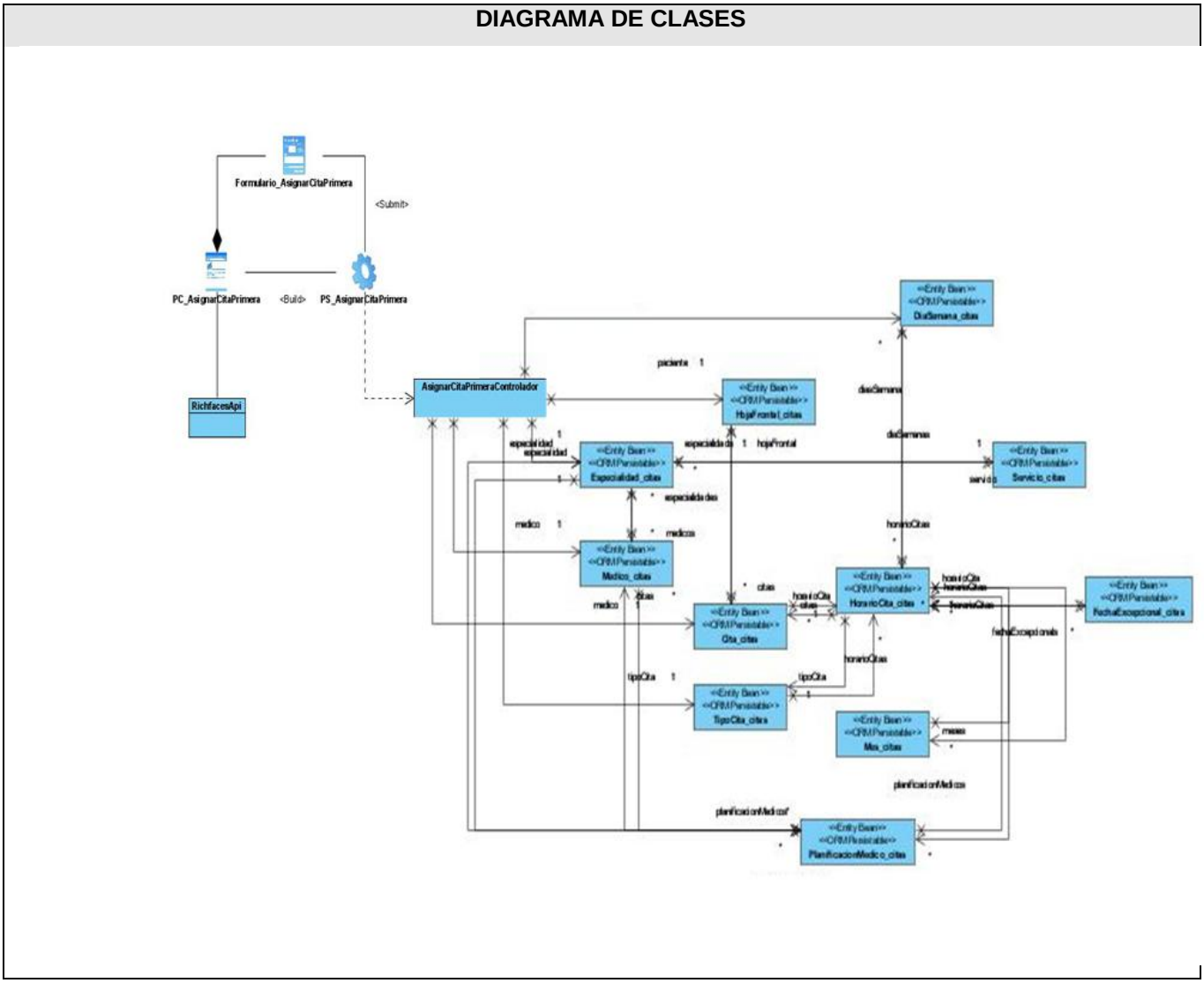


Figura 3.12 DCD Asignar cita primera desde la central de citas.

DIAGRAMA DE CLASES

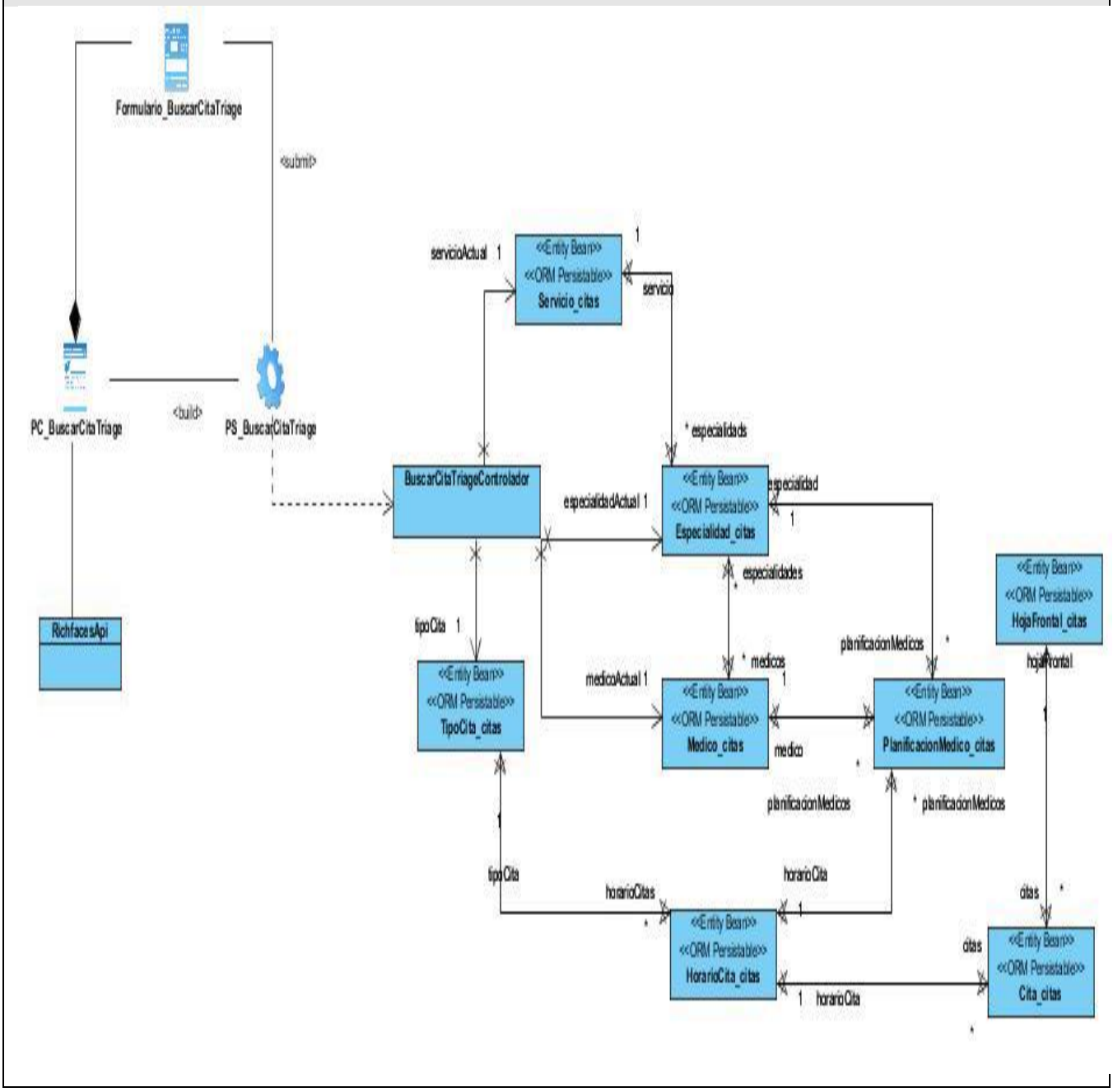


Figura 3.13 DCD Buscar cita triaje desde la central de citas.

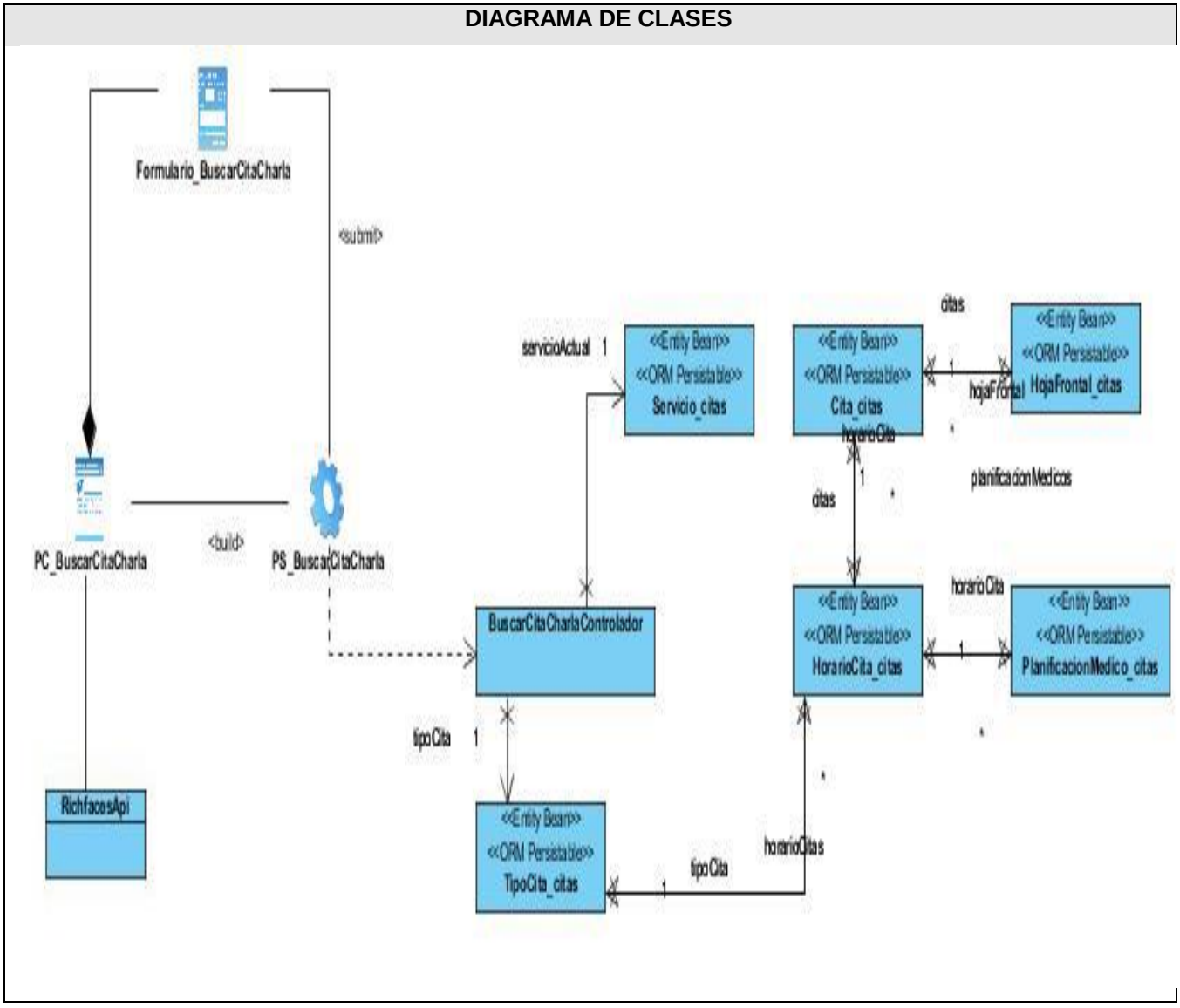


Figura 3.14 DCD Buscar cita charla desde la central de citas.

3.3 Descripción de las clases de diseño

Nombre: AsignarCitaPrimeraControlador	
Tipo de clase :controladora	
Atributo	**Tipo**
cid	Integer
cantTotalMinutos	int
cantTotalPacientes	int
horaInicio	Date
horaFin	Date
confirmar	boolean
Para cada responsabilidad:	
Nombre:	boolean isConfirmar()
Descripción:	Si fue confirmada la operación de asignar cupo
Nombre:	void setConfirmar(boolean confirmar)
Descripción:	Cambiar el estado de la variable confirmar
Nombre:	List<Cita_citas> getCitas()
Descripción:	Deviuelve la lista de citas
Nombre:	void selectPaciente(HojaFrontal_citas selpaciente)
Descripción:	Para seleccionar un paciente
Nombre:	void disableConfirmar()
Descripción:	Desabilita confirmar
Nombre:	void reservarCita(CitaCustom horario)
Descripción:	Reserva una cita dado un horario
Nombre:	void cancelarCita(CitaCustom horario)
Descripción:	Para cancelar una cita
Nombre:	void reservarCupoExtra(HorarioCitaCustom h)
Descripción:	Para reservar un cupo extra dado un horario
Nombre:	void confirmar()
Descripción:	Confirmar la operación de asignar cita
Nombre:	void cancelar()
Descripción:	Confirma la operación cancelar cita
Nombre:	void llenarListaCitas(int cantPacientes, int cantRealPac,int cantMinutos, List<Cita_citas> pacientesCitados, HorarioCita_citas horarioCita,int cantTotalMinutos)
Descripción:	Llena la lista de citas
Nombre:	HojaFrontal_citas getPaciente()
Descripción:	Devuelve el paciente
Nombre:	void selectHorariosMedico()
Descripción:	Busca todos los horarios y los pacientes citados para cada uno de estos horarios
Nombre:	Void selectHorariosOnFechaSelect(
Descripción:	Obtiene todos los horarios de cita para este día, tanto por la mañana, como por la tarde.
Nombre:	void selectHorariosProfesionalSalud()
Descripción:	obtiene los horarios que trabaja el prof de salud,
Nombre:	void OnNodeCollapseExpand(org.richfaces.event.NodeExpandedEvent event)

Descripción:	Para cargar el tree view de la vista
Nombre:	void cambiarCalendario(int mesActual, int anno)
Descripción:	Cambiar el calendario al cambiar el mes y el año
Nombre:	void anteriorMes()
Descripción:	Para actualizar el calendario con el mes anterior
Nombre:	void proximoMes()
Descripción:	Para actualizar el calendario con el mes siguiente
Nombre:	void anteriorAnno()
Descripción:	Para actualizar el calendario con el año anterior
Nombre:	void proximoAnno()
Descripción:	Para actualizar el calendario con el año siguiente
Nombre:	void setMes(int mes)
Descripción:	Cambiar el mes del calendario
Nombre:	int getMes()
Descripción:	Obtener el mes actual del calendario
Nombre:	void setAnno(int anno)
Descripción:	Cambiar el año del calendario
Nombre:	int getAnno()
Descripción:	Obtener el año actual del calendario
Nombre:	Date getFechaSeleccionada()
Descripción:	Obtener la fecha seleccionada
Nombre:	void setFechaSeleccionada(Date fechaSeleccionada)
Descripción:	Cambiar la variable fecha seleccionada
Nombre:	void onFechaSelected(Date d)
Descripción:	Asignar a fecha seleccionada la fecha que se selecciona de la vista
Nombre:	List<HorarioCitaCustom> gethorariosCita()
Descripción:	Retorna la lista de horarios de citas
Nombre:	void sethorariosCita(List<HorarioCitaCustom> horariosCita)
Descripción:	Cambia la lista de horarios citas por una que se le pasa por parámetros
Nombre:	void setPaciente(HojaFrontal_citas paciente)
Descripción:	Cambia el paciente por uno que se le pasa por parámetros
Nombre:	Medico_citas getMedico()
Descripción:	Devuelve la variable médico
Nombre:	boolean diaDisponible(int diaSemana)
Descripción:	Método que dice si un día de la semana está disponible para ese médico
Nombre:	TipoProfesionalSalud_citas getProfesionalSalud()
Descripción:	Devuelve el profesional de la salud
Nombre:	void setProfesionalSalud(TipoProfesionalSalud_citas profesionalSalud)
Descripción:	Cambia el profesional de la salud

Tabla 3.1 DCD Asignar cita primera en el servicio

Nombre: AsignarCitaControlControlador	
Tipo de clase :controladora	
Atributo	**Tipo**
cid	Integer

cantTotalMinutos	int
cantTotalPacientes	int
horaInicio	Date
horaFin	Date
confirmar	boolean
Para cada responsabilidad:	
Nombre:	boolean isConfirmar()
Descripción:	Si fue confirmada la operación de asignar cupo
Nombre:	void setConfirmar(boolean confirmar)
Descripción:	Cambiar el estado de la variable confirmar
Nombre:	List<Cita_citas> getCitas()
Descripción:	Deviuelve la lista de citas
Nombre:	void selectPaciente(HojaFrontal_citas selpaciente)
Descripción:	Para seleccionar un paciente
Nombre:	void disableConfirmar()
Descripción:	Desabilita confirmar
Nombre:	void reservarCita(CitaCustom horario)
Descripción:	Reserva una cita dado un horario
Nombre:	void cancelarCita(CitaCustom horario)
Descripción:	Para cancelar una cita
Nombre:	void reservarCupoExtra(HorarioCitaCustom h)
Descripción:	Para reservar un cupo extra dado un horario
Nombre:	void confirmar()
Descripción:	Confirmar la operación de asignar cita
Nombre:	void cancelar()
Descripción:	Confirma la operación cancelar cita
Nombre:	void llenarListaCitas(int cantPacientes, int cantRealPac,int cantMinutos, List<Cita_citas> pacientesCitados, HorarioCita_citas horarioCita,int cantTotalMinutos)
Descripción:	Llena la lista de citas
Nombre:	HojaFrontal_citas getPaciente()
Descripción:	Devuelve el paciente
Nombre:	void selectHorariosMedico()
Descripción:	Busca todos los horarios y los pacientes citados para cada uno de estos horarios
Nombre:	Void selectHorariosOnFechaSelect(
Descripción:	Obtiene todos los horarios de cita para este día, tanto por la mañana, como por la tarde.
Nombre:	void OnNodeCollapseExpand(org.richfaces.event.NodeExpandedEvent event)
Descripción:	Para cargar el tree view de la vista
Nombre:	void cambiarCalendario(int mesActual, int anno)
Descripción:	Cambiar el calendario al cambiar el mes y el año
Nombre:	void anteriorMes()
Descripción:	Para actualizar el calendario con el mes anterior
Nombre:	void proximoMes()
Descripción:	Para actualizar el calendario con el mes siguiente
Nombre:	void anteriorAnno()
Descripción:	Para actualizar el calendario con el año anterior
Nombre:	void proximoAnno()
Descripción:	Para actualizar el calendario con el año siguiente

Nombre:	void setMes(int mes)
Descripción:	Cambiar el mes del calendario
Nombre:	int getMes()
Descripción:	Obtener el mes actual del calendario
Nombre:	void setAnno(int anno)
Descripción:	Cambiar el año del calendario
Nombre:	int getAnno()
Descripción:	Obtener el año actual del calendario
Nombre:	Date getFechaSeleccionada()
Descripción:	Obtener la fecha seleccionada
Nombre:	void setFechaSeleccionada(Date fechaSeleccionada)
Descripción:	Cambiar la variable fecha seleccionada
Nombre:	void onFechaSelected(Date d)
Descripción:	Asignar a fecha seleccionada la fecha que se selecciona de la vista
Nombre:	List<HorarioCitaCustom> gethorariosCita()
Descripción:	Retorna la lista de horarios de citas
Nombre:	void sethorariosCita(List<HorarioCitaCustom> horariosCita)
Descripción:	Cambia la lista de horarios citas por una que se le pasa por parámetros
Nombre:	void setPaciente(HojaFrontal_citas paciente)
Descripción:	Cambia el paciente por uno que se le pasa por parámetros
Nombre:	Medico_citas getMedico()
Descripción:	Devuelve la variable médico
Nombre:	boolean diaDisponible(int diaSemana)
Descripción:	Metodo que dice si un día de la semana está disponible para ese médico

Tabla 3.2 DCD Asignar cita control en el servicio

Nombre: AsignarCitaPrimeraControlador	
Tipo de clase :controladora	
Atributo	**Tipo**
cid	Integer
cantTotalMinutos	int
cantTotalPacientes	int
horaInicio	Date
horaFin	Date
confirmar	boolean
Para cada responsabilidad:	
Nombre:	boolean isConfirmar()
Descripción:	Si fue confirmada la operación de asignar cupo
Nombre:	void setConfirmar(boolean confirmar)
Descripción:	Cambiar el estado de la variable confirmar
Nombre:	List<Cita_citas> getCitas()
Descripción:	Deviuelve la lista de citas
Nombre:	void selectPaciente(HojaFrontal_citas selpaciente)
Descripción:	Para seleccionar un paciente

Nombre:	void disableConfirmar()
Descripción:	Desabilita confirmar
Nombre:	void reservarCita(CitaCustom horario)
Descripción:	Reserva una cita dado un horario
Nombre:	void cancelarCita(CitaCustom horario)
Descripción:	Para cancelar una cita
Nombre:	void reservarCupoExtra(HorarioCitaCustom h)
Descripción:	Para reservar un cupo extra dado un horario
Nombre:	void confirmar()
Descripción:	Confirmar la operación de asignar cita
Nombre:	void cancelar()
Descripción:	Confirma la operación cancelar cita
Nombre:	void llenarListaCitas(int cantPacientes, int cantRealPac,int cantMinutos, List<Cita_citas> pacientesCitados, HorarioCita_citas horarioCita,int cantTotalMinutos)
Descripción:	Llena la lista de citas
Nombre:	HojaFrontal_citas getPaciente()
Descripción:	Devuelve el paciente
Nombre:	void selectHorariosMedico()
Descripción:	Busca todos los horarios y los pacientes citados para cada uno de estos horarios
Nombre:	Void selectHorariosOnFechaSelect(
Descripción:	Obtiene todos los horarios de cita para este día, tanto por la mañana, como por la tarde.
Nombre:	void selectHorariosProfesionalSalud()
Descripción:	obtiene los horarios que trabaja el prof de salud,
Nombre:	void OnNodeCollapseExpand(org.richfaces.event.NodeExpandedEvent event)
Descripción:	Para cargar el tree view de la vista
Nombre:	void cambiarCalendario(int mesActual, int anno)
Descripción:	Cambiar el calendario al cambiar el mes y el año
Nombre:	void anteriorMes()
Descripción:	Para actualizar el calendario con el mes anterior
Nombre:	void proximoMes()
Descripción:	Para actualizar el calendario con el mes siguiente
Nombre:	void anteriorAnno()
Descripción:	Para actualizar el calendario con el año anterior
Nombre:	void proximoAnno()
Descripción:	Para actualizar el calendario con el año siguiente
Nombre:	void setMes(int mes)
Descripción:	Cambiar el mes del calendario
Nombre:	int getMes()
Descripción:	Obtener el mes actual del calendario
Nombre:	void setAnno(int anno)
Descripción:	Cambiar el año del calendario
Nombre:	int getAnno()
Descripción:	Obtener el año actual del calendario
Nombre:	Date getFechaSeleccionada()
Descripción:	Obtener la fecha seleccionada
Nombre:	void setFechaSeleccionada(Date fechaSeleccionada)
Descripción:	Cambiar la variable fecha seleccionada

Nombre:	void onFechaSelected(Date d)
Descripción:	Asignar a fecha seleccionada la fecha que se selecciona de la vista
Nombre:	List<HorarioCitaCustom> gethorariosCita()
Descripción:	Retorna la lista de horarios de citas
Nombre:	void sethorariosCita(List<HorarioCitaCustom> horariosCita)
Descripción:	Cambia la lista de horarios citas por una que se le pasa por parámetros
Nombre:	void setPaciente(HojaFrontal_citas paciente)
Descripción:	Cambia el paciente por uno que se le pasa por parámetros
Nombre:	Medico_citas getMedico()
Descripción:	Devuelve la variable médico
Nombre:	boolean diaDisponible(int diaSemana)
Descripción:	Metodo que dice si un día de la semana está disponible para ese médico
Nombre:	TipoProfesionalSalud_citas getProfesionalSalud()
Descripción:	Devuelve el profesional de la salud
Nombre:	void setProfesionalSalud(TipoProfesionalSalud_citas profesionalSalud)
Descripción:	Cambia el profesional de la salud

Tabla 3.3 DCD Asignar cita primera en la central de citas

Nombre: BuscarCitaTriageControlador	
Tipo de clase :controladora	
Atributo	**Tipo**
nombrePaciente	String
apellido1Paciente	String
apellido2Paciente	String
noIdentidad	String
idCita;	int
especialidad	String
servicio	String
medico	String
fecha	Date
cantResultado	String
cantResultados	int
paginaActual	int
primeraPagina	boolean
paginaAnterior	boolean
paginaPrimera	boolean
paginaUltima	boolean
paginaSiguiente	boolean
Para cada responsabilidad:	
Nombre:	void create()
Descripción:	Constructor de la clase
Nombre:	void buscarCitas()
Descripción:	Método para buscar todas las citas, acotadas por servicio, especialidad, médico.
Nombre:	Servicio_citas buscarServicioCitas()
Descripción:	Busca y devuelve un objeto de Servicio_citas utilizando el atributo de tipo string del

	nombre servicio.
Nombre:	Medico_citas buscarMedicoCitas()
Descripción:	Busca y devuelve un objeto de Medico_citas utilizando el atributo de tipo string del nombre médico.
Nombre:	Especialidad_citas buscarEspecialidadCitas()
Descripción:	Busca y devuelve un objeto de Especialidad_citas utilizando el atributo de tipo string del nombre especialidad.
Nombre:	List<String> getServiciosMedico()
Descripción:	buscar cada uno de los servicios del usuario actual
Nombre:	List<String> getEspecialidades()
Descripción:	buscar cada uno de las especialidades del usuario dado un servicio actual
Nombre:	List<String> getMedicos()
Descripción:	buscar cada uno de los médicos del usuario dado un servicio y una especialidad actual
Nombre:	void eliminar()
Descripción:	Elimina una cita
Nombre:	void setIdCita(int idCita)
Descripción:	Cambia el id del atributo idcita
Nombre:	void siguientePagina()
Descripción:	Método para el paginado que deveulve el resultado para mostrar en la siguiente página
Nombre:	void anteriorPagina()
Descripción:	Método para el paginado que devuelve el resultado que se mostró en la siguiente anterior
Nombre:	void primeraPagina()
Descripción:	Método para el paginado que devuelve el resultado que se mostró en la primera página
Nombre:	void ultimaPagina()
Descripción:	Método para el paginado que devuelve el resultado a mostrar en la última página
Nombre:	String getNombrePaciente()
Descripción:	Devuelve el valor del atributo nombrePaciente
Nombre:	void setNombrePaciente(String nombrePaciente)
Descripción:	cambia el valor del atributo nombrePaciente
Nombre:	String getApellido1Paciente()
Descripción:	Devuelve el valor del atributo apellido1Paciente
Nombre:	void setApellido1Paciente(String apellido1Paciente)
Descripción:	cambia el valor del atributo apellido1Paciente
Nombre:	String getApellido2Paciente()
Descripción:	Devuelve el valor del atributo apellido2Paciente
Nombre:	void setApellido2Paciente(String apellido2Paciente)
Descripción:	Cambia el valor del atributo apellido2Paciente
Nombre:	String getNoIdentidad()
Descripción:	Devuelve el valor del atributo noIdentidad
Nombre:	void setNoIdentidad(String noIdentidad)
Descripción:	Cambia el valor del atributo noIdentidad
Nombre:	Date getFecha()
Descripción:	Devuelve el valor del atributo fecha
Nombre:	void setFecha(Date fecha)
Descripción:	Cambia el valor del atributo fecha
Nombre:	String getEspecialidad()
Descripción:	Devuelve el valor del atributo especialidad

Nombre:	void setEspecialidad(String especialidad)
Descripción:	Cambia el valor del atributo especialidad
Nombre:	String getMedico()
Descripción:	Devuelve el valor del atributo médico
Nombre:	void setMedico(String medico)
Descripción:	Cambia el valor del atributo médico
Nombre:	int getCantResultados()
Descripción:	Devuelve el valor del atributo cantResultados
Nombre:	void setCantResultados(int cantResultados)
Descripción:	Cambia el valor del atributo cantResultados
Nombre:	boolean isPrimeraPagina()
Descripción:	Devuelve el valor del atributo primeraPagina
Nombre:	void setPrimeraPagina(boolean primeraPagina)
Descripción:	Cambia el valor del atributo primeraPagina
Nombre:	boolean isPaginaAnterior()
Descripción:	Devuelve el valor del atributo paginaAnterior
Nombre:	void setPaginaAnterior(boolean paginaAnterior)
Descripción:	Cambiar el valor del atributo paginaAnterior
Nombre:	boolean isPaginaUltima()
Descripción:	Devuelve el valor del atributo paginaAnterior
Nombre:	void setPaginaUltima(boolean paginaUltima)
Descripción:	Cambiar el valor del atributo paginaUltima
Nombre:	boolean isPaginaSiguiente()
Descripción:	Devuelve el valor del atributo paginaSiguiente
Nombre:	void setPaginaSiguiente(boolean paginaSiguiente)
Descripción:	Cambia el valor del atributo paginaSiguiente
Nombre:	boolean isPaginaPrimera()
Descripción:	Devuelve el valor del atributo paginaPrimera
Nombre:	void setPaginaPrimera(boolean paginaPrimera)
Descripción:	Cambia el valor del atributo paginaPrimera
Nombre:	int getCantPaginas()
Descripción:	Devuelve el valor del atributo cantPaginas
Nombre:	void setCantPaginas(int cantPaginas)
Descripción:	Cambia el valor del atributo cantPaginas
Nombre:	int getPaginaActual()
Descripción:	Devuelve el valor del atributo paginaActual
Nombre:	void setPaginaActual(int paginaActual)
Descripción:	Cambia el valor del atributo paginaActual
Nombre:	String getServicio()
Descripción:	Devuelve el valor del atributo servicio
Nombre:	void setServicio(String servicio)
Descripción:	Cambia el valor del atributo servicio
Nombre:	List<ServicioBuscarCita> getServicios()
Descripción:	Devuelve el valor del atributo servicios
Nombre:	void setServicios(List<ServicioBuscarCita> servicios)
Descripción:	Cambia el valor del atributo servicios
Nombre:	String getCantResultado()

Descripción:	Devuelve el valor del atributo cantResultado
Nombre:	void setCantResultado(String cantResultado)
Descripción:	Cambia el valor del atributo cantResultado

Tabla 3.4 DCD Buscar cita triaje en el servicio.

Nombre: BuscarCitaCharlaControlador	
Tipo de clase :controladora	
Atributo	**Tipo**
nombrePaciente	String
apellido1Paciente	String
apellido2Paciente	String
noIdentidad	String
idCita;	int
servicio	String
fecha	Date
cantResultado	String
cantResultados	int
paginaActual	int
primeraPagina	boolean
paginaAnterior	boolean
paginaPrimera	boolean
paginaUltima	boolean
paginaSiguiente	boolean
Para cada responsabilidad:	
Nombre:	void create()
Descripción:	Constructor de la clase
Nombre:	void buscarCitas()
Descripción:	Método para buscar todas las citas, acotadas por servicio, especialidad, médico.
Nombre:	Servicio_citas buscarServicioCitas()
Descripción:	Busca y devuelve un objeto de Servicio_citas utilizando el atributo de tipo string del nombre servicio.
Nombre:	List<String> getServiciosMedico()
Descripción:	buscar cada uno de los servicios del usuario actual
Nombre:	void eliminar()
Descripción:	Elimina una cita
Nombre:	void setIdCita(int idCita)
Descripción:	Cambia el id del atributo idcita
Nombre:	void siguientePagina()
Descripción:	Método para el paginado que devuelve el resultado para mostrar en la siguiente página
Nombre:	void anteriorPagina()
Descripción:	Método para el paginado que devuelve el resultado que se mostró en la siguiente anterior
Nombre:	void primeraPagina()
Descripción:	Método para el paginado que devuelve el resultado que se mostró en la primera página
Nombre:	void ultimaPagina()
Descripción:	Método para el paginado que devuelve el resultado a mostrar en la última página

Nombre:	String getNombrePaciente()
Descripción:	Devuelve el valor del atributo nombrePaciente
Nombre:	void setNombrePaciente(String nombrePaciente)
Descripción:	cambia el valor del atributo nombrePaciente
Nombre:	String getApellido1Paciente()
Descripción:	Devuelve el valor del atributo apellido1Paciente
Nombre:	void setApellido1Paciente(String apellido1Paciente)
Descripción:	cambia el valor del atributo apellido1Paciente
Nombre:	String getApellido2Paciente()
Descripción:	Devuelve el valor del atributo apellido2Paciente
Nombre:	void setApellido2Paciente(String apellido2Paciente)
Descripción:	Cambia el valor del atributo apellido2Paciente
Nombre:	String getNoIdentidad()
Descripción:	Devuelve el valor del atributo noIdentidad
Nombre:	void setNoIdentidad(String noIdentidad)
Descripción:	Cambia el valor del atributo noIdentidad
Nombre:	Date getFecha()
Descripción:	Devuelve el valor del atributo fecha
Nombre:	void setFecha(Date fecha)
Descripción:	Cambia el valor del atributo fecha
Nombre:	int getCantResultados()
Descripción:	Devuelve el valor del atributo cantResultados
Nombre:	void setCantResultados(int cantResultados)
Descripción:	Cambia el valor del atributo cantResultados
Nombre:	boolean isPrimeraPagina()
Descripción:	Devuelve el valor del atributo primeraPagina
Nombre:	void setPrimeraPagina(boolean primeraPagina)
Descripción:	Cambia el valor del atributo primeraPagina
Nombre:	boolean isPaginaAnterior()
Descripción:	Devuelve el valor del atributo paginaAnterior
Nombre:	void setPaginaAnterior(boolean paginaAnterior)
Descripción:	Cambiar el valor del atributo paginaAnterior
Nombre:	boolean isPaginaUltima()
Descripción:	Devuelve el valor del atributo paginaAnterior
Nombre:	void setPaginaUltima(boolean paginaUltima)
Descripción:	Cambiar el valor del atributo paginaUltima
Nombre:	boolean isPaginaSiguiente()
Descripción:	Devuelve el valor del atributo paginaSiguiente
Nombre:	void setPaginaSiguiente(boolean paginaSiguiente)
Descripción:	Cambia el valor del atributo paginaSiguiente
Nombre:	boolean isPaginaPrimera()
Descripción:	Devuelve el valor del atributo paginaPrimera
Nombre:	void setPaginaPrimera(boolean paginaPrimera)
Descripción:	Cambia el valor del atributo paginaPrimera
Nombre:	int getCantPaginas()
Descripción:	Devuelve el valor del atributo cantPaginas
Nombre:	void setCantPaginas(int cantPaginas)

Descripción:	Cambia el valor del atributo cantPaginas
Nombre:	int getPaginaActual()
Descripción:	Devuelve el valor del atributo paginaActual
Nombre:	void setPaginaActual(int paginaActual)
Descripción:	Cambia el valor del atributo paginaActual
Nombre:	String getServicio()
Descripción:	Devuelve el valor del atributo servicio
Nombre:	void setServicio(String servicio)
Descripción:	Cambia el valor del atributo servicio
Nombre:	List<ServicioBuscarCita> getServicios()
Descripción:	Devuelve el valor del atributo servicios
Nombre:	void setServicios(List<ServicioBuscarCita> servicios)
Descripción:	Cambia el valor del atributo servicios
Nombre:	String getCantResultado()
Descripción:	Devuelve el valor del atributo cantResultado
Nombre:	void setCantResultado(String cantResultado)
Descripción:	Cambia el valor del atributo cantResultado

Tabla 3.5 DCD Buscar cita charla en el servicio.

Nombre: BuscarCitaCharlaControlador	
Tipo de clase :controladora	
Atributo	**Tipo**
nombrePaciente	String
apellido1Paciente	String
apellido2Paciente	String
noIdentidad	String
idCita;	int
servicio	String
fecha	Date
cantResultado	String
cantResultados	int
paginaActual	int
primeraPagina	boolean
paginaAnterior	boolean
paginaPrimera	boolean
paginaUltima	boolean
paginaSiguiente	boolean
Para cada responsabilidad:	
Nombre:	void create()
Descripción:	Constructor de la clase
Nombre:	void buscarCitas()
Descripción:	Método para buscar todas las citas, acotadas por servicio, especialidad, médico.
Nombre:	Servicio_citas buscarServicioCitas()
Descripción:	Busca y devuelve un objeto de Servicio_citas utilizando el atributo de tipo string del

	nombre servicio.
Nombre:	List<String> getServiciosMedico()
Descripción:	buscar todos los servicios
Nombre:	void eliminar()
Descripción:	Elimina una cita
Nombre:	void setIdCita(int idCita)
Descripción:	Cambia el id del atributo idcita
Nombre:	void siguientePagina()
Descripción:	Método para el paginado que devuelve el resultado para mostrar en la siguiente página
Nombre:	void anteriorPagina()
Descripción:	Método para el paginado que devuelve el resultado que se mostró en la siguiente anterior
Nombre:	void primeraPagina()
Descripción:	Método para el paginado que devuelve el resultado que se mostró en la primera página
Nombre:	void ultimaPagina()
Descripción:	Método para el paginado que devuelve el resultado a mostrar en la última página
Nombre:	String getNombrePaciente()
Descripción:	Devuelve el valor del atributo nombrePaciente
Nombre:	void setNombrePaciente(String nombrePaciente)
Descripción:	cambia el valor del atributo nombrePaciente
Nombre:	String getApellido1Paciente()
Descripción:	Devuelve el valor del atributo apellido1Paciente
Nombre:	void setApellido1Paciente(String apellido1Paciente)
Descripción:	cambia el valor del atributo apellido1Paciente
Nombre:	String getApellido2Paciente()
Descripción:	Devuelve el valor del atributo apellido2Paciente
Nombre:	void setApellido2Paciente(String apellido2Paciente)
Descripción:	Cambia el valor del atributo apellido2Paciente
Nombre:	String getNoIdentidad()
Descripción:	Devuelve el valor del atributo noIdentidad
Nombre:	void setNoIdentidad(String noIdentidad)
Descripción:	Cambia el valor del atributo noIdentidad
Nombre:	Date getFecha()
Descripción:	Devuelve el valor del atributo fecha
Nombre:	void setFecha(Date fecha)
Descripción:	Cambia el valor del atributo fecha
Nombre:	void setMedico(String medico)
Descripción:	Cambia el valor del atributo médico
Nombre:	int getCantResultados()
Descripción:	Devuelve el valor del atributo cantResultados
Nombre:	void setCantResultados(int cantResultados)
Descripción:	Cambia el valor del atributo cantResultados
Nombre:	boolean isPrimeraPagina()
Descripción:	Devuelve el valor del atributo primeraPagina
Nombre:	void setPrimeraPagina(boolean primeraPagina)
Descripción:	Cambia el valor del atributo primeraPagina
Nombre:	boolean isPaginaAnterior()
Descripción:	Devuelve el valor del atributo paginaAnterior

Nombre:	void setPaginaAnterior(boolean paginaAnterior)
Descripción:	Cambiar el valor del atributo paginaAnterior
Nombre:	boolean isPaginaUltima()
Descripción:	Devuelve el valor del atributo paginaAnterior
Nombre:	void setPaginaUltima(boolean paginaUltima)
Descripción:	Cambiar el valor del atributo paginaUltima
Nombre:	boolean isPaginaSiguiente()
Descripción:	Devuelve el valor del atributo paginaSiguiente
Nombre:	void setPaginaSiguiente(boolean paginaSiguiente)
Descripción:	Cambia el valor del atributo paginaSiguiente
Nombre:	boolean isPaginaPrimera()
Descripción:	Devuelve el valor del atributo paginaPrimera
Nombre:	void setPaginaPrimera(boolean paginaPrimera)
Descripción:	Cambia el valor del atributo paginaPrimera
Nombre:	int getCantPaginas()
Descripción:	Devuelve el valor del atributo cantPaginas
Nombre:	void setCantPaginas(int cantPaginas)
Descripción:	Cambia el valor del atributo cantPaginas
Nombre:	int getPaginaActual()
Descripción:	Devuelve el valor del atributo paginaActual
Nombre:	void setPaginaActual(int paginaActual)
Descripción:	Cambia el valor del atributo paginaActual
Nombre:	String getServicio()
Descripción:	Devuelve el valor del atributo servicio
Nombre:	void setServicio(String servicio)
Descripción:	Cambia el valor del atributo servicio
Nombre:	List<ServicioBuscarCita> getServicios()
Descripción:	Devuelve el valor del atributo servicios
Nombre:	void setServicios(List<ServicioBuscarCita> servicios)
Descripción:	Cambia el valor del atributo servicios
Nombre:	String getCantResultado()
Descripción:	Devuelve el valor del atributo cantResultado
Nombre:	void setCantResultado(String cantResultado)
Descripción:	Cambia el valor del atributo cantResultado

Tabla 3.6 DCD Buscar cita charla en la central de citas.

Nombre: BuscarCitaTriageControlador	
Tipo de clase :controladora	
Atributo	**Tipo**
nombrePaciente	String
apellido1Paciente	String
apellido2Paciente	String
noIdentidad	String
idCita;	int

especialidad	String
servicio	String
medico	String
fecha	Date
cantResultado	String
cantResultados	int
paginaActual	int
primeraPagina	boolean
paginaAnterior	boolean
paginaPrimera	boolean
paginaUltima	boolean
paginaSiguiente	boolean

Para cada responsabilidad:

Nombre:	void create()
Descripción:	Constructor de la clase
Nombre:	void buscarCitas()
Descripción:	Método para buscar todas las citas, acotadas por servicio, especialidad, médico.
Nombre:	Servicio_citas buscarServicioCitas()
Descripción:	Busca y devuelve un objeto de Servicio_citas utilizando el atributo de tipo string del nombre servicio.
Nombre:	Medico_citas buscarMedicoCitas()
Descripción:	Busca y devuelve un objeto de Medico_citas utilizando el atributo de tipo string del nombre médico.
Nombre:	Especialidad_citas buscarEspecialidadCitas()
Descripción:	Busca y devuelve un objeto de Especialidad_citas utilizando el atributo de tipo string del nombre especialidad.
Nombre:	List<String> getServiciosMedico()
Descripción:	buscar todos los servicios
Nombre:	List<String> getEspecialidades()
Descripción:	buscar cada uno de las especialidades dado un servicio actual
Nombre:	List<String> getMedicos()
Descripción:	buscar cada uno de los médicos dado un servicio y una especialidad actual
Nombre:	void eliminar()
Descripción:	Elimina una cita
Nombre:	void setIdCita(int idCita)
Descripción:	Cambia el id del atributo idcita
Nombre:	void siguientePagina()
Descripción:	Método para el paginado que devuelve el resultado para mostrar en la siguiente página
Nombre:	void anteriorPagina()
Descripción:	Método para el paginado que devuelve el resultado que se mostró en la siguiente anterior
Nombre:	void primeraPagina()
Descripción:	Método para el paginado que devuelve el resultado que se mostró en la primera página
Nombre:	void ultimaPagina()
Descripción:	Método para el paginado que devuelve el resultado a mostrar en la última página
Nombre:	String getNombrePaciente()
Descripción:	Devuelve el valor del atributo nombrePaciente

Nombre:	void setNombrePaciente(String nombrePaciente)
Descripción:	cambia el valor del atributo nombrePaciente
Nombre:	String getApellido1Paciente()
Descripción:	Devuelve el valor del atributo apellido1Paciente
Nombre:	void setApellido1Paciente(String apellido1Paciente)
Descripción:	cambia el valor del atributo apellido1Paciente
Nombre:	String getApellido2Paciente()
Descripción:	Devuelve el valor del atributo apellido2Paciente
Nombre:	void setApellido2Paciente(String apellido2Paciente)
Descripción:	Cambia el valor del atributo apellido2Paciente
Nombre:	String getNoIdentidad()
Descripción:	Devuelve el valor del atributo noIdentidad
Nombre:	void setNoIdentidad(String noIdentidad)
Descripción:	Cambia el valor del atributo noIdentidad
Nombre:	Date getFecha()
Descripción:	Devuelve el valor del atributo fecha
Nombre:	void setFecha(Date fecha)
Descripción:	Cambia el valor del atributo fecha
Nombre:	String getEspecialidad()
Descripción:	Devuelve el valor del atributo especialidad
Nombre:	void setEspecialidad(String especialidad)
Descripción:	Cambia el valor del atributo especialidad
Nombre:	String getMedico()
Descripción:	Devuelve el valor del atributo médico
Nombre:	void setMedico(String medico)
Descripción:	Cambia el valor del atributo médico
Nombre:	int getCantResultados()
Descripción:	Devuelve el valor del atributo cantResultados
Nombre:	void setCantResultados(int cantResultados)
Descripción:	Cambia el valor del atributo cantResultados
Nombre:	boolean isPrimeraPagina()
Descripción:	Devuelve el valor del atributo primeraPagina
Nombre:	void setPrimeraPagina(boolean primeraPagina)
Descripción:	Cambia el valor del atributo primeraPagina
Nombre:	boolean isPaginaAnterior()
Descripción:	Devuelve el valor del atributo paginaAnterior
Nombre:	void setPaginaAnterior(boolean paginaAnterior)
Descripción:	Cambiar el valor del atributo paginaAnterior
Nombre:	boolean isPaginaUltima()
Descripción:	Devuelve el valor del atributo paginaAnterior
Nombre:	void setPaginaUltima(boolean paginaUltima)
Descripción:	Cambiar el valor del atributo paginaUltima
Nombre:	boolean isPaginaSiguiente()
Descripción:	Devuelve el valor del atributo paginaSiguiente
Nombre:	void setPaginaSiguiente(boolean paginaSiguiente)
Descripción:	Cambia el valor del atributo paginaSiguiente
Nombre:	boolean isPaginaPrimera()

Descripción:	Devuelve el valor del atributo paginaPrimera
Nombre:	void setPaginaPrimera(boolean paginaPrimera)
Descripción:	Cambia el valor del atributo paginaPrimera
Nombre:	int getCantPaginas()
Descripción:	Devuelve el valor del atributo cantPaginas
Nombre:	void setCantPaginas(int cantPaginas)
Descripción:	Cambia el valor del atributo cantPaginas
Nombre:	int getPaginaActual()
Descripción:	Devuelve el valor del atributo paginaActual
Nombre:	void setPaginaActual(int paginaActual)
Descripción:	Cambia el valor del atributo paginaActual
Nombre:	String getServicio()
Descripción:	Devuelve el valor del atributo servicio
Nombre:	void setServicio(String servicio)
Descripción:	Cambia el valor del atributo servicio
Nombre:	List<ServicioBuscarCita> getServicios()
Descripción:	Devuelve el valor del atributo servicios
Nombre:	void setServicios(List<ServicioBuscarCita> servicios)
Descripción:	Cambia el valor del atributo servicios
Nombre:	String getCantResultado()
Descripción:	Devuelve el valor del atributo cantResultado
Nombre:	void setCantResultado(String cantResultado)
Descripción:	Cambia el valor del atributo cantResultado

Tabla 3.7 DCD Buscar cita triaje en la central de citas.

Conclusiones

En el presente capítulo se han mostrado los diagramas de clases de análisis y diseño, así como los diagramas de secuencia de cada realización de los casos de uso. Se describieron las clases controladoras para los procesos Asignar Cita en la central de citas y en el servicio, introduciendo y dando paso al flujo de trabajo “Implementación”.

CAPÍTULO 4: IMPLEMENTACIÓN

En este capítulo se expone el modelo de datos, describiendo cada atributo y relación de las tablas de la base de datos, además de explicar brevemente el diagrama de despliegue de la propuesta, en el cual se expone la ubicación física de cada componente, los cuales se exponen más adelante en el diagrama de componentes, exponiendo temas polémicos del sistema como son la seguridad, los estándares de codificación y estilos empleados, así como los tratamientos de errores.

4.1 Modelo de datos

El modelo de datos no es más que la representación de la estructura o descripción física de las tablas de la base de datos. Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos: los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos. A continuación se presenta el modelo de datos del módulo. Ver figura 4.1 (26)

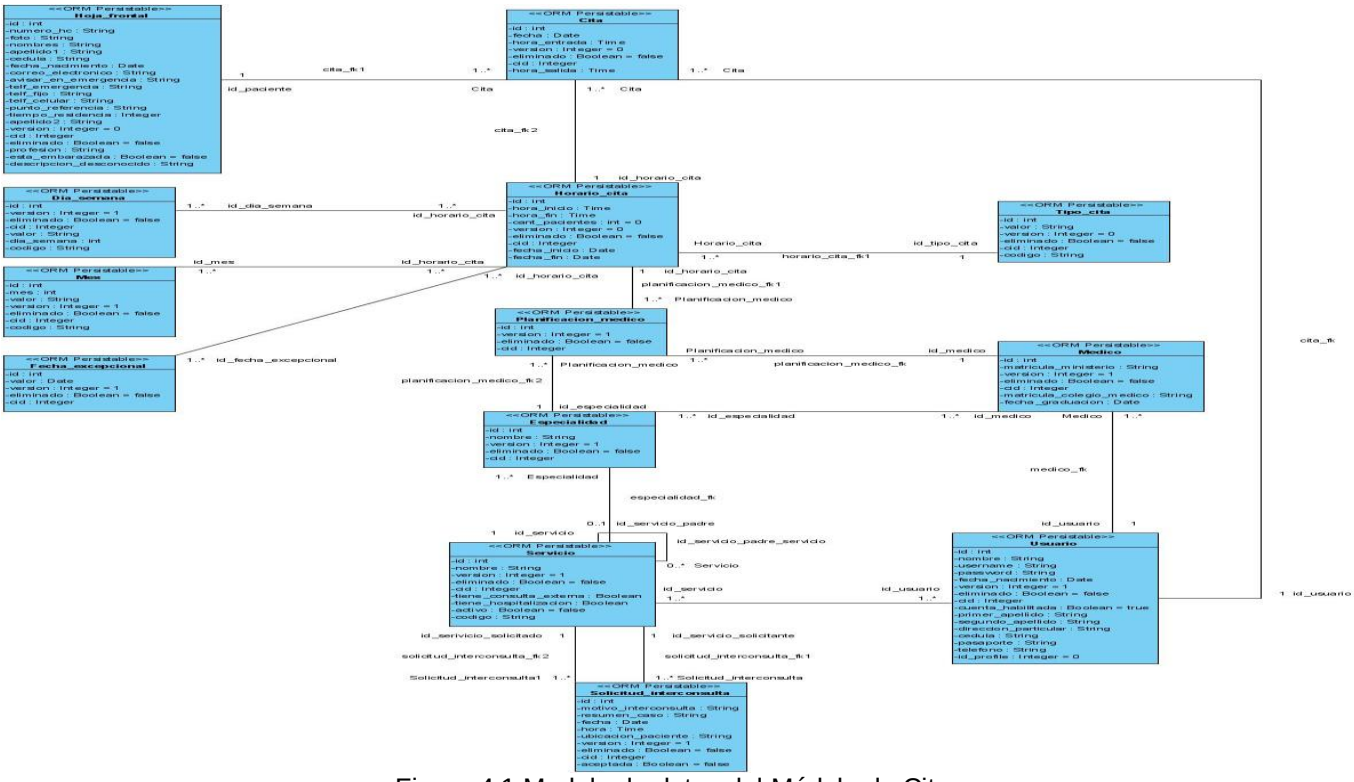


Figura 4.1 Modelo de datos del Módulo de Citas.

4.1.1 Descripción de las tablas de la Base de Datos.

Tabla Solicitud Interconsulta

Nombre: Solicitud_Interconsulta		
Descripción: La tabla se encarga de almacenar las solicitudes de interconsulta		
Atributo	**Tipo**	**Descripción**
id	int	Id de la solicitud
motivo_interconsulta	String	Motivo por la cual se creó la solicitud
resumen_caso	String	Breve resumen del caso
fecha	Date	Fecha de la solicitud de interconsulta
hora	Time	Hora de la solicitud de interconsulta
ubicacion_paciente	String	Ubicación del paciente
eliminado	Boolean	Si es eliminada o no la solicitud
cid	Integer	Para trabajar con la bitácora
aceptada	Boolean	Si es aceptada o no la solicitud

Tabla Servicio

Nombre: Servicio		
Descripción: Almacenar los servicios que presentan las instituciones hospitalarias		
Atributo	**Tipo**	**Descripción**
id	int	Id del servicio
nombre	String	Nombre del servicio
eliminado	Boolean	Si el servicio está eliminado o no
cid	Integer	Para trabajar con la bitácora
tiene_consulta_externa	Boolean	Si el servicio presenta consulta externa o no
tiene_hospitalizacion	Boolean	Si el servicio presenta hospitalización o no
activo	Boolean	Si el servicio aun esta activo en la institución
codigo	String	Código para estandarizar los nombres de los servicios

Tabla Usuario

Nombre: Usuario		
Descripción: Almacenar todos los usuarios del sistema		
Atributo	**Tipo**	**Descripción**
id	int	Id del usuario
nombre	String	Nombre del usuario
username	String	Usuario
password	String	Contraseña
fecha_nacimiento	Date	Fecha de nacimiento
eliminado	Boolean	Si el usuario fue eliminado o no

cid	Integer	Para trabajar con la bitácora
cuenta_habilitada	Boolean	Si la cuenta fue habilitada o no
primer_apellido	String	Primer apellido
segundo_apellido	String	Segundo apellido
dirección_particular	String	Dirección particular
cedula	String	Número de identidad
pasaporte	String	Pasaporte
telefono	String	Teléfono

Tabla Especialidad

Nombre: Especialidad		
Descripción: Almacenar las especialidades que presentan las instituciones hospitalarias		
Atributo	**Tipo**	**Descripción**
id	int	Id de la especialidad
nombre	String	Nombre de la especialidad
eliminado	Boolean	Si la especialidad fue eliminada o no
cid	Integer	Para trabajar con la bitácora

Tabla Médico

Nombre: Médico		
Descripción: Almacenar los médicos que trabajan en la institución hospitalaria		
Atributo	**Tipo**	**Descripción**
id	int	Id del médico
matricula_ministerio	String	Matrícula que lo identifica en el ministerio
eliminado	Boolean	Si el médico fue eliminado o no
cid	Integer	Para trabajar con la bitácora
matricula_colegio_medico	String	Matrícula que lo identifica en el colegio
fecha_graduacion	Date	Fecha de graduación

Tabla Planificación Médico

Nombre: Planificacion_medico		
Descripción: Almacenar la planificación de citas de los médicos de las instituciones hospitalarias		
Atributo	**Tipo**	**Descripción**
id	int	Id de la planificación
eliminado	Boolean	Si la planificación fue eliminada o no
cid	Integer	Para trabajar con la bitácora

Tabla Horario Cita

Nombre: Horario_cita		
Descripción: Almacenar los horarios de la planificación de citas de los médicos		
Atributo	**Tipo**	**Descripción**
id	int	Id del horario
hora_inicio	Time	Hora de inicio del horario
hora_fin	Time	Hora de fin del horario
cant_pacientes	int	Cantidad de pacientes que almacena el horario
eliminado	Boolean	Si el horario fue eliminado o no
cid	Integer	Para trabajar con la bitácora
fecha_inicio	Date	Fecha de inicio del horario
fecha_fin	Date	Fecha fin del horario

Tabla Tipo de Cita

Nombre: Tipo_cita		
Descripción: Almacenar los tipos de citas existentes a los cuales se les hará la planificación		
Atributo	**Tipo**	**Descripción**
id	int	Id del tipo de cita
valor	String	Nombre del tipo de cita
eliminado	Boolean	Si el tipo de cita es eliminado o no
cid	Integer	Para trabajar con la bitácora
codigo	String	Código para estandarizar los nombres de los tipos de citas

Tabla Cita

Nombre: Cita		
Descripción: Almacenar las citas existentes que han sido planificadas a los médicos		
Atributo	**Tipo**	**Descripción**
id	int	Id de la cita
fecha	Date	Fecha en que se crea la cita
hora_entrada	Time	Hora de entrada de la cita
eliminado	Boolean	Si la cita está eliminada o no
cid	Integer	Para trabajar con la bitácora
hora_salida	Time	Hora de salida de la cita

Tabla Hoja Frontal

Nombre: Hoja_frontal		
Descripción: Almacenar los pacientes de las instituciones hospitalarias		
Atributo	**Tipo**	**Descripción**
id	int	Id del paciente
numero_hc	String	Número de la historia clínica
foto	String	Foto del paciente
nombres	String	Nombre(s) del paciente
apellido1	String	Primer apellido del paciente
cedula	String	Número de identidad
fecha_nacimiento	Date	Fecha de nacimiento
correo_electronico	String	Correo electrónico del paciente
avisar_en_emergencia	String	Avisar en caso de emergencia
telf_emergencia	String	Teléfono en caso de emergencia
telf_fijo	String	Teléfono fijo
telf_celular	String	Teléfono celular
punto_referencia	String	Punto de referencia
tiempo_residencia	Integer	Tiempo en la residencia
apellido2	String	Segundo apellido del paciente
cid	Integer	Para trabajar con la bitácora
eliminado	Boolean	Si el paciente es eliminado o no
profesion	String	Profesión que ocupa el paciente
esta_embarazada	Boolean	Si el paciente está embarazada o no
descripcion_desconocido	String	Descripción en caso de ser desconocido

4.2 Modelo de Despliegue

El modelo de despliegue no es más que la representación física de los nodos, es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Es una colección de nodos y arcos; donde cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar. (27)

A continuación se representa la estructura de estos nodos para el Módulo de Citas. Ver figura 4.2.

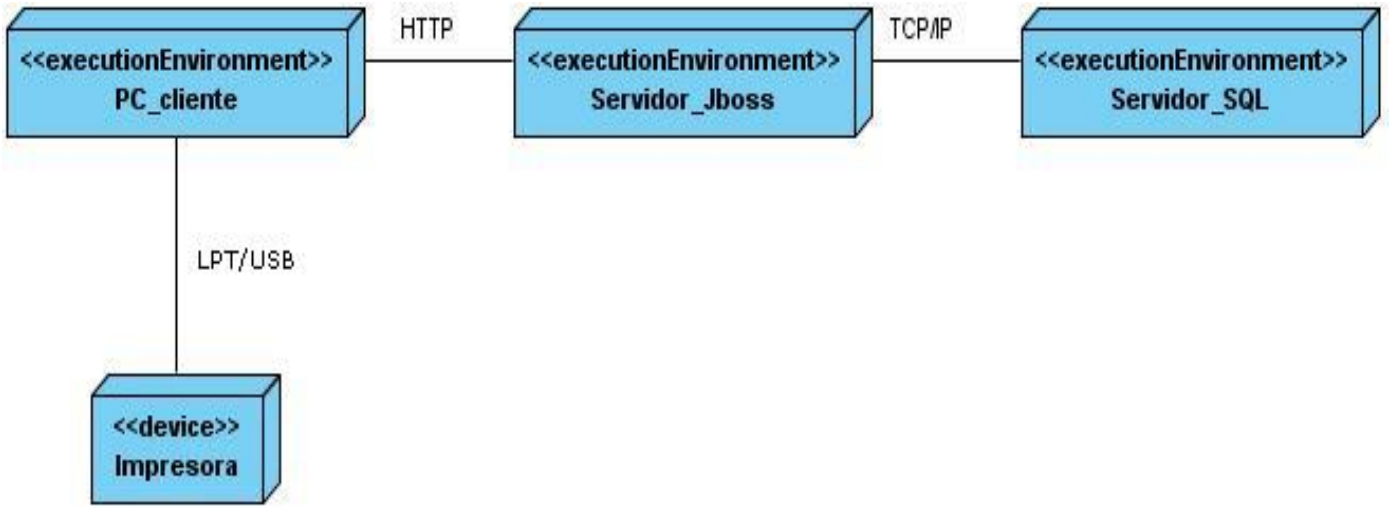


Figura 4.2 Modelo de Despliegue

4.3 Diagrama de Componentes

El diagrama de componentes define como las clases, artefactos y otros elementos de bajo nivel, se unen para formar componentes de alto nivel y las conexiones entre ellos. Los componentes son artefactos de software compilados que trabajan acoplados para brindar el comportamiento requerido dentro de las restricciones definidas en el proceso de captura de requisitos. A continuación se presenta como quedan estructurados estos componentes en el módulo Citas. Ver figura 4.3

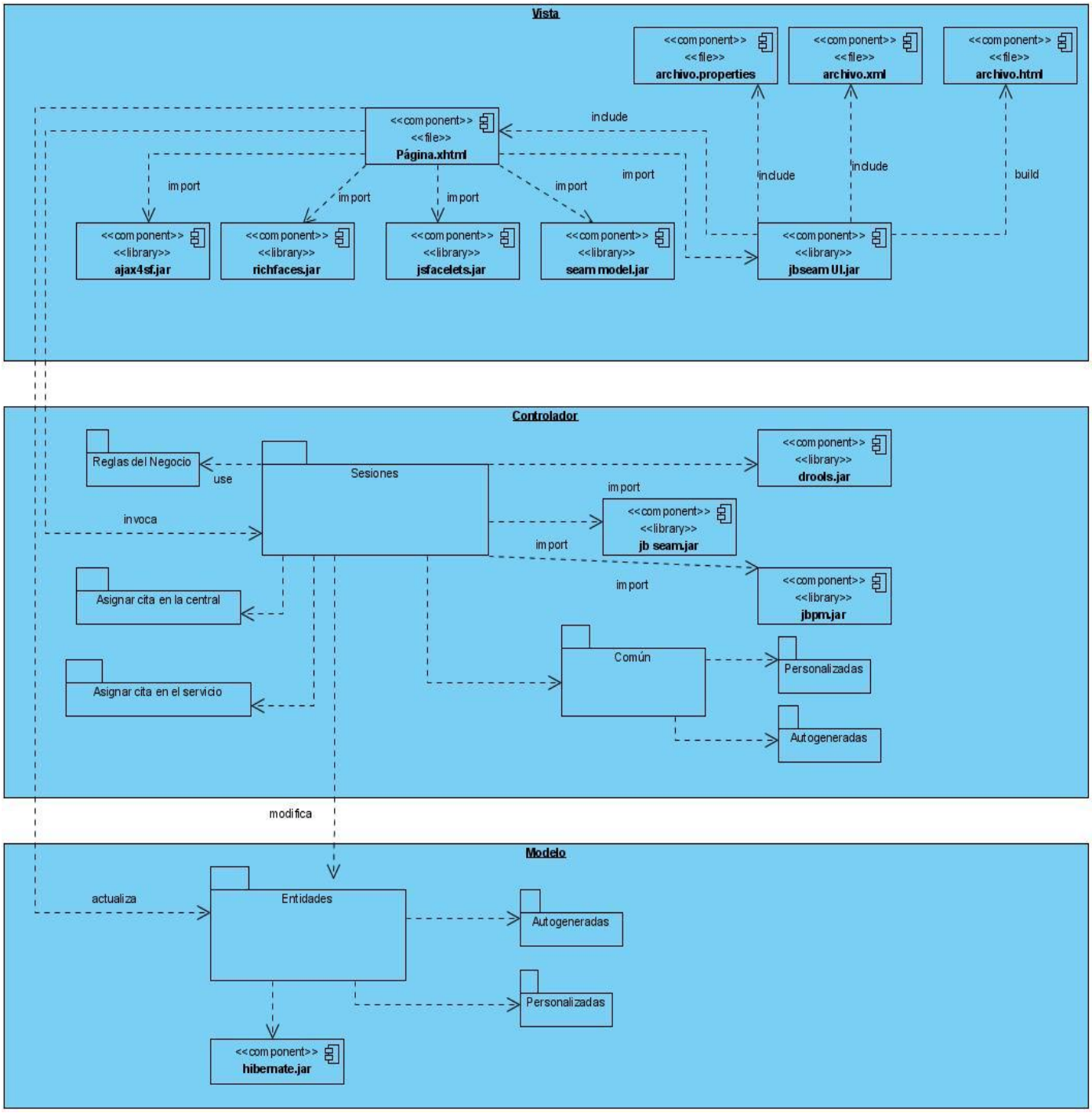


Figura 4.3 Diagrama de componentes.

4.4 Tratamiento de errores

Las excepciones no son más que eventos o sucesos inesperados que aparecen en un programa en tiempo de ejecución, y que pueden llevar a interrumpir el correcto funcionamiento del mismo. Para mitigar estos sucesos se lleva a cabo el tratamiento de excepciones el cual permite tomar el control de la respuesta del programa ante un error o evento inesperado, lo cual posibilita la continuidad y el correcto funcionamiento del programa.

Para lograr un sistema estable, confiable y libre de errores, se lleva el manejo de excepciones en todas las porciones de código fuente donde pueda ocurrir un error, especialmente en donde se llevan a cabo transferencias desde y hacia la base de datos debido a que estas sentencias son las más propensas a error, además que hay dependencia de los datos de la base de datos. También en la interfaz de usuario se lleva una validación de sus elementos.

Para darle a informar al usuario sobre los errores que pueden ocurrir, el sistema cuenta con un archivo llamado page.xml que contiene la configuración de todos los mensajes que se muestran por cada error. Este archivo contiene además a donde va a ser dirigido el usuario si ocurriese algún suceso sorpresivo de tipo error. Por otra parte se hace uso del componente Sean FacesMessages, para capturar cualquier mensaje de error o notificación proveniente de los controladores y posteriormente mostrarlos en las vistas o interfaces de usuario.

4.5 Seguridad

Para lograr un sistema seguro se sostendrá un control a nivel de usuarios y contraseñas, permitiendo el acceso por tipo de usuario logrando así la visibilidad sólo a las áreas establecidas de acorde a la función que realizan. Las contraseñas solo podrán ser cambiadas por el usuario o por el administrador del sistema.

Otra cuestión es lograr la fiabilidad en las estaciones de trabajo, para lograr esto se define un segundo nivel de seguridad a nivel de estaciones de trabajo lo que posibilita la ejecución sólo de las aplicaciones que hayan sido definidas para la estación en cuestión.

El sistema además permitirá llevar una traza de todas las operaciones llevadas a cabo por cada usuario mediante un registro de actividades por usuario en todo momento.

Para lograr la fidelidad de los datos, todo el intercambio entre el sistema y otros sistemas que soliciten información desde cualquier hospital, se realizará de forma cifrada eliminando posibilidades de acceso o modificación de la misma.

4.6 Estrategias de codificación. Estándares y estilos a utilizar.

Para definir una robusta estructura y organización del código, se deben definir algunos estándares para su posterior entendimiento y cumplir con las buenas prácticas establecidas en la Ingeniería de Software. A continuación se resumen algunas de las convenciones tomadas en relación a estos aspectos.

Idioma: Se debe utilizar como idioma el español, las palabras no se acentuarán.

Identación: Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento.

Se recomienda dejar dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}. Lo mismo sucede para el caso de las instrucciones if, else, for, while, do while, switch, foreach.

Comentarios, separadores, líneas, espacios en blanco y márgenes: Establecer un modo común para comentar el código de forma tal que sea comprensible con sólo leerlo una vez.

Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de dato, y objetivo del parámetro) entre otras cosas.

Se recomienda dejar una línea en blanco antes y después de la declaración de una clase o de una estructura y de la implementación de una función.

Se recomienda usar espacios en blanco entre los operadores para lograr una mayor legibilidad en el código. Ejemplo: producto = nomproducto.

Se debe evitar comentar cada línea de código. Cuando el comentario se aplica a un grupo de instrucciones debe estar seguido de una línea en blanco. En caso de que se necesite comentar una sola instrucción se suprime la línea en blanco o se escribe a continuación de la instrucción.

No se debe usar espacio en blanco:

- Después del corchete abierto y antes del cerrado de un arreglo.
- Después del paréntesis abierto y antes del cerrado.
- Antes de un punto y coma.

Variables y constantes

El nombre que se le da a las variables debe comenzar con la primera letra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamellCasing. El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.

Ejemplo: nombrePaciente

Clases y Objetos: El objetivo fundamental es nombrar las clases e instancias de forma estándar para todas las aplicaciones.

Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing. Ejemplo: MiClase(). Para el caso de las instancias se comenzara con un prefijo que identificara el tipo de dato, este se escribirá en minúscula.

Para nombrar las funciones se debe tratar de utilizar verbos que denoten la acción que hace la función. Se empleará notación PascalCasing*. Ejemplo: function BuscarUnidad(). Si son funciones que obtienen un dato se emplea el prefijo get y si fijan algún valor se emplea el prefijo set.

El nombre empleado para las clases, objetos, atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos.

Conclusiones

En el presente capítulo se detallaron los diagramas de despliegue y componentes que describen la correspondencia entre la arquitectura de software definida y la del sistema; la dependencia entre los nodos físicos en los que funcionará el sistema propuesto, así como los componentes a construir y su organización. Además, se describieron los principios de diseño seguidos, concepción del tratamiento de errores y principios de codificación. Se mostraron los resultados de la etapa de implementación del sistema propuesto.

CONCLUSIONES GENERALES

El ansia de desarrollo en la informática y las comunicaciones de muchos países subdesarrollados fundamentalmente, la baja calidad de los servicios prestados por el área de citas de las instituciones hospitalarias de dichos países, el no existir una herramienta que automatiza el proceso de gestión de la información en el área de citas de dichas instituciones , la falta en el mundo de un sistema desarrollado sobre tecnologías libres, que este basado sobre la web, que sea multiplataforma ,que contengan un modulo de citas independiente y que gestione este proceso, ha sido el punto de partida en el presente trabajo.

La descripción de los procesos que tienen lugar en las instituciones hospitalarias, ha permitido obtener una mejor comprensión del problema e identificar las principales necesidades a resolver. La posterior definición de los requerimientos, ha sido punto de partida en el proceso de desarrollo del sistema propuesto; al lograr un entendimiento común y efectivo con el cliente, sobre las funcionalidades que la aplicación debe brindar.

Como resultado de las etapas de diseño e implementación desarrolladas, se ha concebido un sistema, basado en la gestión de base de datos utilizando tecnologías Web según el modelo cliente/servidor aplicando el patrón MVC. De manera que su valor fundamental se expresa en la contribución a simplificar el trabajo y la demora que produce el procesamiento manual de la información, mejorar el sistema de planificación y distribución de citas.

Con el presente trabajo se cumplió con el objetivo propuesto, se desarrolló el módulo de Citas, del Sistema de Información Hospitalaria alas HIS. El mismo facilita la gestión de información en esta área de las instituciones hospitalarias reafirmando una vez más que el empleo de las tecnologías informáticas sirve de gran apoyo al sistema de salud de cualquier país.

RECOMENDACIONES

Se recomienda:

- Enriquecer el sistema con nuevas funcionalidades que puedan surgir, pues los procesos hospitalarios no se desarrollan de la misma manera en todos los países.
- Poner el sistema a prueba durante un período significativo para comprobar que sus funcionalidades se correspondan con las actividades que están gestionando.
- Proponer su utilización y generalización en Cuba y en los países que lo requieran.
- El estudio de las posibilidades de integración con los subsistemas desarrollados por otras empresas o grupos de desarrollo de la UCI, involucrados en el proceso de informatización de la Salud.

REFERENCIAS BLIOGRÁFICAS

1. **Díaz, Miguel E. Marín.** *Fundamentos del Sistema de Salud Pública en Cuba para estudiantes de Informática.* La Habana, Cuba : s.n., 2006.
2. **Dr.Abelardo Ramírez Márquez, Dr. Pastor Castell-Florit Serrate,Dr. Guillermo Mesa.** *El Sistema Nacional de Salud de Cuba. Escuela Nacional de Salud Pública (ENSAP).* La Habana, Cuba : s.n., 2003-2004.
3. **ALBET.** *IH-SW-DE-024 ALAS-HIS_Consulta Externa y Citas_Glosario de términos.* La Habana, Cuba : s.n., 2008.
4. **Valle, José Guillermo.** Arquitectura Cliente-Servidor. [Online] 2005. http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml.
5. **Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal John Wiley & Son Ltd.** *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns.* August 8, 1996.
6. **Reynoso, Carlos and Kicillof, Nicolás.** Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* [Online] Universidad de Buenos Aires, 2004. [Cited: febrero 22, 2009.] http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp#10.
7. Model-View-Controller. MSDN. 2009. *Microsoft patterns and practices.* [Online] 2008. [Cited: enero 30, 2009.] http://msdn.microsoft.com/en-us/library/ms978748.aspx.
8. **Internet, Factoria de.** Manual de Java. [Online] 2003-2008. http://www.webtaller.com/manual-java/caracteristicas-java.php.
9. **Seguel, Frank.** Introducción a Richfaces. [Online] 2008. http://frankseguel.blogspot.com/2008/05/introduccin-richfaces.html.
10. **Max, Katz.** *Practical Richfaces.* s.l. : Apress, 2008.
11. **Hat, Red.** Richfaces Developer Guide. *Richfaces Developer Guide.* [Online] 2008. [Cited: febrero 10, 2009.] http://www.jboss.org/jbossrichfaces/docs/.
12. Seam - Contextual Components. Introduction to JBoss Seam. *Seam - Contextual Components. Introduction to JBoss Seam.* [Online] 2008. [Cited: febrero 25, 2009.] http://www.seamframework.org/Documentation.
13. **Dan, Allen.** *Seam in action. Manning Early Access Program.* s.l. : Manning Publications Co., 2008.
14. **ALBET.** *Documento de arquitectura de Software alas HIS.* La Habana, Cuba : s.n., 2008.
15. **Michael, Dr. Juntao Yuan.** On the road to simplicity. *On the road to simplicity.* [Online] 2008. [Cited: marzo 15, 2009.] http://www.javaworld.com/javaworld/jw-02-2005/jw-0221-jboss4.html.
16. **Hat, Red.** JBoss Server Manager Reference Guide. *JBoss Community.* [Online] 2008. [Cited: marzo 20, 2009.] http://www.jboss.org/jbossas/docs/.
17. **Bauer Christian, King Gavin.** *Java Persistence with Hibernate.* s.l. : Manning Publications Co., 2007.
18. PostgreSQL About. *PostgreSQL About.* [Online] 2008. [Cited: enero 9, 2009.] http://www.postgresql.org/about.
19. PostgreSQL Awards. *PostgreSQL Awards.* [Online] 2008. [Cited: enero 10, 2009.] http://www.postgresql.org/about/awards.

20. **Microsystems, Sun.** Java Persistence API FAQ. *Java Persistence API FAQ.* [Online] 2008. [Cited: enero 2, 2009.] http://java.sun.com/javaee/overview/faq/persistence.jsp.
21. **O'Reilly Media, Inc.** XML.com. [Online] 2009. http://www.xml.com/.
22. **Molpeceres, Alberto.** Proceso de Desarrollo. [Online] 2003-2004. http://www.willydev.net/descargas/Articulos/General/cualxpfddrup.PDF.
23. **García, Joaquín.** UML: Casos de Uso. Desarrollo de software orientado a objetos. [Online] septiembre 27, 2003. http://www.ingenierosoftware.com/analisisydiseno/casosdeuso.php.
24. **Fuentes, Sacha.** Eclipse Europa, nueva versión del entorno de programación. [Online] julio 2, 2007. http://www.genbeta.com/2007/07/02-eclipse-europa-nueva-version-del-entorno-de-programacion.
25. **Chaves, Michael Arias.** La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. [Online] julio 7, 2006. http://www.intersedes.ucr.ac.cr/10-art_11.html.
26. **Marqués, María Mercedes.** Modelo de Datos. [Online] febrero 12, 2003-2004. http://www3.uji.es/~mmarques/f47/apun/node32.html.
27. **I. Jacobson, G. Booch, J. Rumbaugh.** *El Proceso Unificado de Desarrollo de software.* s.l. : Addison-Wesley, 2000.

BILIOGRÁFIA

1. **Díaz, Miguel E. Marín.** *Fundamentos del Sistema de Salud Pública en Cuba para estudiantes de Informática.* La Habana, Cuba : s.n., 2006.
2. **Dr.Abelardo Ramírez Márquez, Dr. Pastor Castell-Florit Serrate,Dr. Guillermo Mesa.** *El Sistema Nacional de Salud de Cuba. Escuela Nacional de Salud Pública (ENSAP).* La Habana, Cuba : s.n., 2003-2004.
3. **ALBET.** *IH-SW-DE-024 ALAS-HIS_Consulta Externa y Citas_Glosario de términos.* La Habana, Cuba : s.n., 2008.
4. **Valle, José Guillermo.** Arquitectura Cliente-Servidor. [Online] 2005. http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml.
5. **Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal John Wiley & Son Ltd.** *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns.* August 8, 1996.
6. **Reynoso, Carlos and Kicillof, Nicolás.** Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* [Online] Universidad de Buenos Aires, 2004. [Cited: febrero 22, 2009.] http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp#10.
7. Model-View-Controller. MSDN. 2009. *Microsoft patterns and practices.* [Online] 2008. [Cited: enero 30, 2009.] http://msdn.microsoft.com/en-us/library/ms978748.aspx.
8. **Internet, Factoria de.** Manual de Java. [Online] 2003-2008. http://www.webtaller.com/manual-java/caracteristicas-java.php.
9. **Seguel, Frank.** Introducción a Richfaces. [Online] 2008. http://frankseguel.blogspot.com/2008/05/introduccin-richfaces.html.
10. **Max, Katz.** *Practical Richfaces.* s.l. : Apress, 2008.
11. **Hat, Red.** Richfaces Developer Guide. *Richfaces Developer Guide.* [Online] 2008. [Cited: febrero 10, 2009.] http://www.jboss.org/jbossrichfaces/docs/.
12. Seam - Contextual Components. Introduction to JBoss Seam. *Seam - Contextual Components. Introduction to JBoss Seam.* [Online] 2008. [Cited: febrero 25, 2009.] http://www.seamframework.org/Documentation.
13. **Dan, Allen.** *Seam in action. Manning Early Access Program.* s.l. : Manning Publications Co., 2008.
14. **ALBET.** *Documento de arquitectura de Software alas HIS.* La Habana, Cuba : s.n., 2008.
15. **Michael, Dr. Juntao Yuan.** On the road to simplicity. *On the road to simplicity.* [Online] 2008. [Cited: marzo 15, 2009.] http://www.javaworld.com/javaworld/jw-02-2005/jw-0221-jboss4.html.
16. **Hat, Red.** JBoss Server Manager Reference Guide. *JBoss Community.* [Online] 2008. [Cited: marzo 20, 2009.] http://www.jboss.org/jbossas/docs/.
17. **Bauer Christian, King Gavin.** *Java Persistence with Hibernate.* s.l. : Manning Publications Co., 2007.
18. PostgreSQL About. *PostgreSQL About.* [Online] 2008. [Cited: enero 9, 2009.] http://www.postgresql.org/about.
19. PostgreSQL Awards. *PostgreSQL Awards.* [Online] 2008. [Cited: enero 10, 2009.] http://www.postgresql.org/about/awards.

20. **Microsystems, Sun.** Java Persistence API FAQ. *Java Persistence API FAQ.* [Online] 2008. [Cited: enero 2, 2009.] http://java.sun.com/javaee/overview/faq/persistence.jsp.
21. **O'Reilly Media, Inc.** XML.com. [Online] 2009. http://www.xml.com/.
22. **Molpeceres, Alberto.** Proceso de Desarrollo. [Online] 2003-2004. http://www.willydev.net/descargas/Articulos/General/cualxpfddrup.PDF.
23. **García, Joaquín.** UML: Casos de Uso. Desarrollo de software orientado a objetos. [Online] septiembre 27, 2003. http://www.ingenierosoftware.com/analisisydiseno/casosdeuso.php.
24. **Fuentes, Sacha.** Eclipse Europa, nueva versión del entorno de programación. [Online] julio 2, 2007. http://www.genbeta.com/2007/07/02-eclipse-europa-nueva-version-del-entorno-de-programacion.
25. **Chaves, Michael Arias.** La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. [Online] julio 7, 2006. http://www.intersedes.ucr.ac.cr/10-art_11.html.
26. **Marqués, María Mercedes.** Modelo de Datos. [Online] febrero 12, 2003-2004. http://www3.uji.es/~mmarques/f47/apun/node32.html.
27. **I. Jacobson, G. Booch, J. Rumbaugh.** *El Proceso Unificado de Desarrollo de software.* s.l. : Addison-Wesley, 2000.
28. **Bro, Joseph D.** *Handbook of Biomedical Engineering.2 nd ed.* 2004.
29. **2005, MundoDigital [Monografia].** Historia y Funcionamiento de Internet. [Online] 2005. http://www.wikilearning.com/monografia/historia_y_funcionamiento_de_internet-historia_de_internet/3443-2.
30. **Hat, Red.** Richfaces Developer Guide. [Online] 2008-2009. http://www.jboss.org/file-access/default/members/jbossrichfaces/freezone/docs/devguide/en/html_single/index.html.
31. —. Jboss Enterprise Middleware. [Online] 2007. http://www.latam.redhat.com/pdf/jboss/JBoss_enterprise_300507_esp.pdf.
32. **Hispano, Asociación java.** Java Hispano. [Online] 2002-2007. http://www.javahispano.org.
33. **Lovelle, Juan Manuel Cueva.** *Introducción a UML.* Universidad de Oviedo, España : s.n., 2003-2004.
34. **Valencia, Maria Eugenia.** Diagrama de Clases de Diseño. *Escuela de Ingeniería de Sistemas y Computación.* [Online] 2009. [Cited: enero 15, 2009.] http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/DISCLASES_A12.pdf.
35. **Medicina, D. R. Facultad de.** Sistema de Información Hospitalaria. *Universidad Nacional Autónoma de México.* [Online] 2003. [Cited: enero 23, 2009.] http://www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf.
36. **Bauer Christian, King Gavin.** *Java Persistence with Hibernate.* s.l. : Manning Publications Co., 2007.

GLOSARIO DE TÉRMINOS

Consulta de Triaje: Consulta donde el paciente sin historia clínica lo valora el médico y canaliza la situación.

Consulta de Primera: Consulta que se le hace a un paciente que viene por primera vez a atenderse determinada patología.

Consulta de Control: Consulta que se le hace a un paciente que viene a atenderse por una patología anteriormente diagnosticada en el mismo servicio.

HTML (HyperText Markup Language): Es el lenguaje para la representación de la información en la web (ver Web).

HL7: Es una especificación para un estándar de intercambio de datos electrónicos en el área de la salud, con especial énfasis en las comunicaciones intra–hospitalarias en el área de la información clínica y administrativa.

Habeas data: Acción constitucional o legal que tiene cualquier persona que figura en un registro o banco de datos, de acceder a tal registro para conocer qué información existe sobre su persona, y de solicitar la corrección de esa información si le causara algún perjuicio.

Historia Clínica (HC): Documento que se crea para almacenar el comportamiento evolutivo de un paciente durante su estancia en el hospital. No se limita a ser una narración o exposición de hechos simplemente, sino que incluye juicios, documentos y procedimientos; es un documento que se va haciendo en el tiempo, documentando fundamentalmente la relación médico-paciente.

Interconsulta: Documento que se emite cuando un paciente de un servicio del hospital necesita ser valorado por un médico de otro servicio del hospital.

PACS: Sistema para el almacenamiento y comunicación de imágenes médicas.

PDF (Portable Document Format): Es un formato de fichero que simula un documento impreso como una imagen electrónica que se puede leer, crear, navegar, imprimir, ente otros.

Sistema de Información Hospitalaria (SIH): Son sistemas software que trabajan acoplados y permiten la informatización de los distintos servicios de las instituciones de salud.

Sistema de Información Radiológica (RIS-Radiological Information System): Es un sistema encargado de la gestión de la información generada y manipulada como resultado de los procesos de negocio de carácter radiológico (imagenológico).

SNS: Sistema Nacional de Salud.

XML o Extensible Markup Language: Es un metalenguaje extensible para la especificación de datos en documentos.

XHTML o EXtensible HyperText Markup Language: Es una reformulación de HTML escrito en XML. Puede ser considerado como una versión más estricta y limpia de HTML.