I. EL PROYECTO

I.1 PRESENTACION DEL PROYECTO

I.1.1 Titulo

Mejoramiento del Control de la Informacion de los Pacientes Hospitalizados en la caja nacional de salud- Regional Tarija

I.1.2 Área del proyecto

Mejorar el control de la información de los pacientes hospitalizados en la Caja nacional de salud- Regional Tarija

I.1.3 Responsable del Proyecto

Carrera de Ingeniería Informática - Taller III – grupo 5

I.1.4 Entidades asociadas

Caja nacional de salud- Regional Tarija

I.1.5 Compromiso del director de Proyecto

Yo, Rodrigo Garcia Zambrana director del proyecto "Mejoramiento del Control de la Informacion de los Pacientes Hospitalizados en la caja nacional de salud- Regional Tarija", acepto la responsabilidad de cumplir los compromisos de ejecución del proyecto "Mejorar el control de la información de los pacientes hospitalizados en la Caja nacional de salud- Regional Tarija ", en caso de aprobarse.	
Rodrigo Garcia Zambrana **Nombre de Director**	**Firma de director**

Tabla 1. Compromiso del Director de Proyecto

I.1.6 Duración del Proyecto

El proyecto tiene una duración de 8 meses.

I.1.7 Director responsable del Proyecto

NOMBRE	TALLER/ GRUPO
Rodrigo Garcia Zambrana	Taller III grupo 5
Email	**Telefono**
Jhoncena_conelfu@hotmail.com	75121180

Tabla 2. Director responsable del proyecto

I.2 DESCRIPCION DEL PROYECTO

I.2.1 Resumen Ejecutivo del Proyecto

La Caja Nacional de Salud (CNS) en sus más de 50 años de vida no ha podido sumarse de manera consistente al avance tecnológico referido al uso de los Sistemas Informáticos basados en computadora. Si bien se cuenta con más de 70 equipos de computación, con sistemas de información y sus redes locales (Hospital Obrero, Oficina Regional, Administración y policlínico); ello no ha tenido el impacto organizativo que busca la institución.

El gran crecimiento vegetativo de la población asegurada en la ciudad de Tarija, ha generado grandes cantidades de información que tiene que ser correctamente administrado por los distintos departamentos. Toda la información que se encuentra registrada en depósitos de hojas impresas y kardex podría ser cargada en un sistema de almacenamiento digital capaz de optimizar el espacio físico, facilitar el registro y el control, además de mejorar la utilización de los recursos humanos.

La CNS-Regional Tarija tiene como objetivo principal brindar atenciones en salud con calidad a la población asegurada con la implementación de planes, programas y control de calidad.

Para coadyuvar a cumplir con los objetivos de la CNS- Regional Tarija y los problemas existentes se necesita administrar la información de los pacientes hospitalizados, asegurados y no asegurados de forma eficiente y eficaz. Para tal efecto se propone el proyecto **"Mejoramiento del Control de la Informacion de los Pacientes Hospitalizados en la caja nacional de salud- Regional Tarija "**

El proyecto está dirigido a mejorar la administración de la información de los asegurados, beneficiarios, así también de los no asegurados como el SUMI, SPAM y otros que reciben atención en medicina interna.

El proyecto se centra en administrar la información de relevancia de los pacientes, historia clínica informatizada, para así de esta manera hacer un seguimiento del estado de salud de los mismos. Además de reducir las barreras de acceso a la información.

Entre las principales estrategias (Productos) tenemos lo siguiente:

- **Un sistema de gestión de información para tener un mejor control de la información de pacientes hospitalizados en la CNS-Regional Tarija.**

Con la elaboración del sistema de gestión, se pretende manejar de manera eficaz y eficiente la información relacionada con los pacientes de la CNS-Regional Tarija, de esta manera se beneficiara a la parte administrativa, asegurados y/o familiares y población en general.

- **Capacitación del personal administrativo de la CNS-Regional Tarija en el manejo del sistema informático**

Con esta estrategia se busca, capacitar al personal administrativo de la CNS-Regional Tarija acerca del manejo adecuado del sistema informático y de qué manera contribuye al mejor manejo de sus necesidades administrativas

I.2.2 Descripción, Fundamentación y Justificación del Proyecto

La CNS- Regional Tarija en los últimos ha tenido y tendrá un importante presupuesto destinado a la adquisición de equipos de computación, fibra óptica, intranet, hosting, dominios y otros, así de esta manera poder mejorar el manejo y acceso a la información por parte de la administración, como de la población asegurada.

La parte administrativa del policlínico como del hospital obrero, pero especialmente la parte de sistema, ha visto la necesidad de realizar una reingeniería en los softwares que se usaban para los procesos administrativos y desarrollar otros nuevos. Dichos softwares no cumplía con los requisitos actuales, además de estar creados con tecnologías que están en desuso.

Un problema que se tiene en la parte administrativa, es lo referente a la información que se maneja y brinda, tanto a la parte administrativa (reportes, cuadros estadísticos y otros) como así también asegurados y no asegurados internados en el hospital obrero de Tarija.

Tal información es morosa e incorrecta, porque el personal de cada área de medicina interna, tales como medicina mujeres, cirugía y otras no realizan el correcto registro de los pacientes, porque muchas veces existe un llenado incorrecto. Esto lleva a tener una historia clínica, estadísticas e informes no correctos.

El proyecto "Mejoramiento del Control de la Informacion de los Pacientes Hospitalizados en la caja nacional de salud- Regional Tarija" tiene por finalidad mejorar la administración de la información de manera eficiente y eficaz, de los asegurados y sus beneficiarios, como así también de los no asegurados tales como SUMI (Seguro Universal Materno Infantil), SPAM (), y toda la población en general.

Por lo tanto el proyecto ayudara a llevar un seguimiento correcto de los pacientes, para la toma de decisiones por parte de la administración y así poder asignar más recursos para brindar un mejor servicio.

I.3 Objetivos

I.3 .1 Objetivo General

Mejorar el control de la información de manera oportuna y precisa sobre los pacientes hospitalizados en la CNS-Regional Tarija

I.3.2 Objetivos Específicos

1.- Desarrollar un sistema informático de control de la información sobre los pacientes hospitalizados en la CNS-Regional Tarija

2.- Capacitar al personal administrativo de la CNS- Regional Tarija en las TIC y el manejo del sistema informático

I.3.3 Metodología

Metodología para el desarrollo de las aplicaciones

Por las características del presente proyecto a realizar, la metodología de trabajo se divide en dos partes:

- **Metodología para el desarrollo del Sistema Informático**
- **Metodología para la capacitación del personal administrativo y otros.**

- **Metodología para el desarrollo del Sistema Informático**

Para el desarrollo del Sistema Informático se utilizará la metodología RUP (Proceso Unificado de Rational) y UML (Lenguaje Unificado de Modelado), que es un lenguaje gráfico que utiliza diagramas ya definidos para especificar o describir métodos o procesos y definir un sistema. En RUP se siguen cuatro fases para el desarrollo del software, al final de las cuales, y tras una serie de iteraciones, se establece objetivos precisos a conseguir:

Inicio.- En esta fase se establece los requisitos de negocio que cubrirá el sistema, se obtendrá la especificación de requerimientos.

Elaboración.- En esta fase el problema se analiza y comprende desde el punto de vista del equipo de desarrollo. Al final de la fase se tiene definida la arquitectura y el modelo de requisitos del sistema empleando los diagramas de casos de uso especificados en lenguaje UML.

Construcción. - En esta fase se profundiza en el diseño de los componentes del sistema y de manera iterativa se van añadiendo las funcionalidades al software a medida que se construyen y prueban, permitiendo a la vez que se puedan ir incorporando cambios.

Al final de esta fase se obtiene un sistema completamente operativo y la documentación (diagrama de clases, de secuencia, modelo entidad-relación, modelo de dominio, manual de instalación, manual de usuario, manual de seguridad y otros) para entregar a los usuarios.

Transición.- La fase final del RUP se ocupa del traslado del software desde los entornos de desarrollo a los entornos de producción, en los que el usuario final hará uso del sistema.

Por otra parte se utilizará las siguientes herramientas para el desarrollo del software:

Se desarrolla el sistema en el lenguaje de programación Java (Servlets, templates, mapeadores, framework spring y otros) porque es una de las tecnologías más seguras para el desarrollo de programas, es multiplataforma, de fuente abierta y utiliza el paradigma orientado a objetos, que debido al enfoque y complejidad del problema en este proyecto nos proporciona la mejor solución al permitir crear programas modulares, visuales de fácil manejo para el usuario y facilitar el mantenimiento del software.

La base de datos se desarrolla en Postgres ya que es un sistema de gestión de base de datos relacional y multiusuario, al ser una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo, esto permite velocidad y Flexibilidad.

-**Metodología para la capacitación del personal administrativo y otros.**

Para llevar a cabo las capacitaciones, se utilizará toda la tecnología moderna posible (computadoras, data show, pizarras acrílicas, Manuales de usuario, Manuales del sistema y otros) necesaria para llegar con gran facilidad, agrado y detalle de comprensión a los usuarios finales.

Para la capacitación, la metodología a utilizar se hará de la siguiente manera:

Se tomara en cuenta el nivel de conocimiento del personal sobre la TIC para impartir la capacitación básica y la capacitación del sistema.

Se realiza el estilo participativo, con una duración de las sesiones de una hora por día, durante los días programados, las sesiones se dividen en módulos, uno básico sobre la TIC y los demás para el uso del sistema.

I.3.4 Bibliografía Consultada para la realización del Perfil de Proyecto

James A. Senn Análisis y Diseño de Sistemas Informáticos

KENDALL Y KENDALL Análisis y diseño de sistemas.

BURCH Y GRUDNITSKI Diseño de sistemas de información.

PRESSMAN Ingeniería de software.

FAIRLEY Ingeniería de software.

MATIN, J Y ODELL, J. Análisis y diseño orientado a objetos.

SOMERVILLE Software Engineering.

www.desarrolloweb.com Sitio de análisis y diseño de sistemas.

I.3.5 Resultados esperados

- Implementar un Sistema informático de información sobre los pacientes asegurados a CNS-Regional Tarija

  Sistema web desarrollado con tecnologías libres. El sistema genera reportes gráficos y estadísticos para la parte administrativa, como así también información relevante acerca de los pacientes y su estado de salud

Resultado: En fecha 30 de Julio de 2011 se concluye en un 80% la etapa de análisis y diseño, el 30 de Septiembre de 2011 se concluye en un 80% la etapa de construcción del Sistema Informático.

- **Capacitar el personal administrativo de la CNS Tarija en las TIC y el manejo del sistema informático**

Se capacitara previamente en el uso de las Tecnologías de la Información y Comunicación(TIC) para posteriormente capacitar en el uso del sistema web

Resultado: En el mes de diciembre del 2011 al menos el 80% de los miembros del área administrativa involucrada son capacitados.

I.3.6 Transferencia de resultados

I.3.6.1 a) Medios y estrategias para la transferencia de resultados.

Presentación final del sistema informático a las autoridades universitarias y personas involucradas en el proyecto

Entrega de instaladores del sistema informático y la documentación desarrollada en el proyecto.

Socialización del producto final con la institución involucrada.

I.3.6.2 b) Grupo de beneficiarios de los resultados

Los grupos beneficiarios son:

Parte administrativa

Encargado de sistemas

Enfermeras

Pacientes hospitalizados y Familiares

I.3.7 Cronograma de Actividades

N°	Actividad	N° dias	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
	COMPONENTE 1											
1	Especificación de requerimientos	16	X									
2	Elaboración de los diagramas UML del Sistema.	24		X								
3	Diseño de la base de datos	18		X								
4	Diseño del Prototipo	25			X							
5	Programación del Sistema.	110				X	X	X	X			
6	Elaboración del Informe final sobre el Desarrollo del Sistema.	14							X	X		
7	Realización de pruebas.	14								X		
8	Instalación de la base de datos.	1								X		
9	Instalación del sistema Informático.	1								X		
	COMPONENTE 2											
10	Definición de metodologías de enseñanza de los cursos de capacitación.	1								X		
11	Realizar la formación al personal en el uso del sistema	2								X		

Tabla 3. cronograma de actividades

I.3.8 Marco Lógico del Proyecto

Resumen Narrativo del Proyecto	Indicadores	Medios de Verificación	Supuestos
Fin Brindar un servicio de calidad para los pacientes de la caja nacional de salud –Regional Tarija	Al segundo año después de la ejecución del proyecto, la administración de la CNS Tarija tiene información detallada y precisa sobre los pacientes asegurados hospitalizados.	Informe del director de la CNS Tarija sobre el manejo eficiente de la información de los pacientes asegurados	La Sociedad tarijeña se beneficia de los servicios que brinda el proyecto.
Objetivo General (Propósito) Control de la información oportuna y precisa sobre los pacientes hospitalizados de la CNS-Regional Tarija mejorada	***Eficacia*** Al final del 2013 se tiene actualizada y completa la información acerca de la salud de los nuevos pacientes que fueron registrados en noviembre del 2012, en un 50%. **Formula matemática** Lista de nuevos pacientes hospitalizados registrados	Informe por parte de la administración sobre el estado de salud de los nuevos pacientes hospitalizados	Presupuesto suficiente para la implementación del sistema informático

Objetivos Específicos (Componentes)			
1.- Sistema informático de información sobre los pacientes hospitalizados de la CNS-Regional Tarija, desarrollado	1.1 En fecha 30 de Julio de 2011 se concluye en un 80% la etapa de análisis y diseño del Sistema, de acuerdo a los requisitos sujetos a la norma 830 de la IEEE. 1.2. En fecha 30 de Septiembre de 2011 se concluye en un 80% la etapa de construcción del Sistema Informático.	1.- Documento de Análisis y diseño, como de la etapa de construcción del Proyecto presentado al Docente de Taller III.	1.- Se cuenta con recursos Humanos suficientes en el área de administración
2.- Capacitado el personal administrativo de la CNS-Regional Tarija en las TIC y el manejo del sistema informático	2.1 En el mes de octubre del 2012 al menos el 80% de los miembros del área administrativa involucrada son capacitados, con 2 horas. El contenido será uso de la TIC y manejo del sistema informático	2.1 Carta de presentación y aceptación del sistema por parte de las autoridades correspondientes de la CNS. Lista de participantes de la capacitación	2.- Los miembros del personal administrativo han reconocido la importancia de la capacitación.

Actividades C1. Sistema Informático 1.-Especificación de requerimientos Elaboración de los diagramas UML del Sistema. 2.- Diseño de la base de datos del Sistema. 3.- Diseño del Prototipo del Sistema. 4.- Programación del Sistema. 5.- Validación del Software. 6.- Elaboración del Informe final sobre el Desarrollo del Sistema	C1.- Componente Partida 12100 Bs. 9000 Partida 21100 Bs. 120 Partida 21400 Bs. 300 Partida 25200 Bs 1560 Partida 39500 Bs 20 **Total** **Bs 11000**	Informe del presupuesto de gastos.	C1.- - Existe disponibilidad por parte del personal de la institución, para brindar información sobre el manejo de los registros. - Se cuenta con la disponibilidad de los recursos económicos.

C2. Capacitación al Personal Definición de metodologías de enseñanza y planificación del contenido temático de los cursos de capacitación. Realizar la formación al personal en el uso de las TIC para el manejo del producto final.	C2.-Componente 2 22100 Pasajes Bs. 200 22200 Viáticos Bs. 400 25500 publicidad Bs 40 25600 Imprenta Bs 250 **Total** **890 Bs**		C2.- Los miembros del área administración han asignado su tiempo para capacitarse Ambiente de Capacitación adecuado. Disponibilidad del personal para asistir a la capacitación.

Tabla 4. Matriz de Marco Logico

I.4 Presupuesto / Justificación

ITEM	RUBROS	Aporte Universidad	Otro Aporte	TOTAL (Bs)
20000	SERVICIOS NO PERSONALES			100
	21000. Servicios Básicos			
	22000. Servicios de transporte			1140
	23000. Alquileres			540
	24000. Mantenimiento y reparación			
	25000. Servicios Profesionales y Comerciales			8380
	1.1.1 Sub total rubro			**10160**
30000	MATERIALES Y SUMINISTROS			
	31000. Alimentos y Productos Forestales			2400
	32000. Productos de Papel, Cartón e Impresos			1194
	33000. Textiles y Vestuario.			
	34000. Productos Químicos, Combustibles y Lubricantes			
	39000. Productos Varios.			303

	1.1.2 Sub total rubro			**3897**
40000	**ACTIVOS REALES**			
	43000. Maquinaria y Equipo.			1000
	46000. Descripción de estudios y proyectos para inversión			
	49000. Otros Activos			
	1.1.3 Sub total rubro			**1000**
	1.1.4 TOTAL			15057

1) GRUPO 20000. SERVICIOS NO PERSONALES

b) SUB GRUPO 21000. Descripción de los gastos de servicios básicos

Partida	Tipo de servicio básico *	Costo	Tiempo mes	Costo Total
21100	Comunicación	100		100
21200	Energía Eléctrica			
21300	Agua			
21400	Servicios Telefónicos			
Total				100

* Se refiere principalmente a los gastos por servicios; como: servicio de correo, radiogramas, servicio telefónico, fax, Internet.

c) SUB GRUPO 22000. Descripción de los gastos de viajes y transporte

Partida	Personal	Lugar	Nº de viajes	Costo unitario*	Costo total
22100	Pasajes		6	30	180
Total					180

* En el caso de pasajes debe indicarse el costo de ida y vuelta (costo unitario), indicando el número de viajes.

Partida	Personal	Lugar	Duración (días)	Costo unitario*	Costo total
22200	Viáticos				
22300	Fletes y Almacenamientos				
22600	Transporte de Personal	Dpto. de Investigación Ciencias y Tecnología	480	2	960
Total					960
Total sub grupo 22000					1140

* En el caso de los viáticos, debe considerarse la escala establecida por la UAJMS.

d) SUB GRUPO 23000. Descripción de los gastos por concepto de alquileres de equipos y maquinarias

Partida	Alquiler de equipo y maquinaria	Costo/U	Tiempo/m	Costo total
23100	Alquiler de Edificios			
23200	Alquiler de Equipos y Maquinaria			

	➢ 30 equipos de Computación	6	3 días	540
23300	Alquiler de Tierras y Terrenos			
Total				540

* Se refiere principalmente a los gastos por el uso de edificios y equipos y maquinaria en general

e) SUB GRUPO 24000. Descripción mantenimiento y reparación

Partida	**Mantenimiento y reparación de equipo y maquinaria**	**Costo unitario**	**Tiempo mes**	**Costo total**
24100	Mantenimiento y Reparación de Edificios y Equipos			
24300	Otros Gastos por Mantenimiento y Reparación			
Total				

* Se refiere principalmente a los gastos por el mantenimiento y reparación de edificios y equipos y maquinaria en general

f) SUB GRUPO 25000. Descripción de los gastos en servicios profesionales y comerciales

Partida	**Tipo de servicio profesional y comercial ***	**Cantidad**	**Costo unitario**	**Tiempo mes**	**Costo total**
25500	Publicidad				
25600	Imprenta ➢ Certificado de Participación	60	8		480

	➢ Empastados	9	100		900
25700	Capacitación de Personal				
25800	Estudios e Investigaciones Para Proyectos de Inversión				
25810	Consultores por Producto ➢ Consultor Informático en Análisis de Sistema ➢ Consultor de Diseño Grafico	1 1	500 3000	8 1	4000 3000
25820	Consultores en Línea ➢ Director	1		0	0
Total					8380

* Se refiere a gastos por servicios profesionales de asesoramiento especializado, se incluyen, estudios, investigaciones, publicidad, imprenta, fotocopias, capacitación de personal y otros ejecutados por terceros.

2) GRUPO 30000. MATERIALES Y SUMINISTROS

g) SUB GRUPO 31000. Descripción de los gastos Alimentos y Productos Agroforestales

Partida	Tipo de material *	Cantidad	Costo/Unitario	Total
31110	Refrigerios y Gastos Administrativos ➢ Varios	960	2.5	2400
31200	Alimento para Animales			
31300	Productos Agroforestales y Pecuarios			
Total				2400

* Se refiere a la adquisición de materiales y bienes como: alimentos y productos agroforestales, alimentos y bebidas para personas (indicar el total de refrigerios), alimentos para animales, productos pecuarios.

h) SUB GRUPO 32000. Descripción del gasto de Productos de Papel, Cartón e Impresos

Partida	Tipo de material *	Cantidad	Costo/Unitario	Total
32100	Papel de Escritorio ➢ Resma de Hojas Bond	15	40	600
32200	Productos de Artes Graficas, Papel y Cartón ➢ Tinta Negra ➢ Tinta de Color	6 9	45 36	270 324
32300	Libros y Revistas			

32400	Textos de Enseñanza			
Total				1194

* Se refiere a la adquisición de; papel y cartón en sus diversas formas y clases, impresos y publicaciones, periódicos, revistas, libros, fotocopias, etc.

i) SUB GRUPO 33000. Descripción del gasto en textiles y vestuario

Partida	**Productos textiles y vestuarios**	**Cantidad**	**Costo/Unitario**	**Total**
33100	Hilados y Telas			
33200	Confecciones Textiles			
33300	Prendas de vestir			
33400	Calzados			
Total				

* Se refiere principalmente a los gastos por vestuario uniformes, ropa de trabajo

j) SUB GRUPO 34000. Combustibles, Productos Químicos, Farmacéuticos y Otros

Partida	Combustibles, Productos Químicos, Farmacéuticos y Otros	**Cantidad**	**Costo/Unitario**	**Total**
34110	Combustibles y Lubricantes para Consumo			
34200	Productos químicos y Farmacéuticos			
34400	Productos de Cuero y Caucho			

34500	Productos de Minerales no Metálicos y Plásticos			
34600	Productos Metálicos			
34700	Minerales			
34800	Herramientas Menores			
Total				

* Se refiere a gastos de combustibles, químicos, productos farmacéuticos, llantas etc.

k) SUB GRUPO 39000. Descripción del gasto en productos varios

Partida	**Productos de cuero y caucho**	**Cantidad**	**Costo/Unitario**	**Total**
39100	Material de Limpieza			
39400	Instrumental Menor Médico - Quirúrgico			
39500	Útiles de Escritorio y de Oficina ➢ Agenda Personal ➢ Memoria Flash ➢ Bolígrafos, Lapiceras ➢ Cd's, DVD's	 3 2 6 10	 25 90 3 3	 75 180 18 30
39700	Útiles y Materiales Eléctricos			
39800	Otros Repuestos y Accesorios			
Total				303

*Se refiere principalmente a los gastos por productos de limpieza, todo le referente a la funcionamiento de la oficina en material de escritorio.

3) GRUPO 40000. ACTIVOS REALES

l) SUB GRUPO 43000. Descripción del gasto de Maquinaria y Equipo

Partida	Tipos de productos	Cantidad	Costo/Unitario	Total
43100	Equipo de Oficina y Muebles ➢ Cámara digital	1	1000	1000
43200	Maquinaria y Equipo de Producción			
43300	Equipos de Transporte, Tracción y Elevación			
43400	Equipo Medico y de Laboratorio			
43700	Otra Maquinaria y Equipo			
Total				1000

* Se refiere principalmente a los gastos por muebles y enseres, equipo de oficina, comunicación, equipamiento.

m) SUB GRUPO 46000. Descripción de estudios y proyectos para inversión

Partida	Productos textiles y vestuarios	Cantidad	Costo/Unitario	Total
46100	Para Construcción de Bienes de Dominio Privado			
Total				

* Se refiere principalmente a los gastos por servicios de terceros para la realización de investigaciones y otras actividades técnico – Profesionales necesarias para la construcción y mejoramiento de bienes.

n) SUB GRUPO 49000. Descripción del gasto de Otros Activos

Partida	**Tipos de productos ***	**Cantidad**	**Costo/Unitario**	**Total**
49100	Activos Intangibles			
Total				

* Se refiere a los gastos en la compra de sotfwear, licencias.

II. COMPONENTES

II.1 MEJORAMIENTO DEL CONTROL DE LA INFORMACION DE LOS PACIENTES HOSPITALIZADOS EN LA CAJA NACIONAL DE SALUD REGIONAL TARIJA.

II.1.1 Plan de Desarrollo del Software

II.1.1.1 Introducción

Este Plan de Desarrollo de Software es una versión preparada para ser incluida en la propuesta elaborada como respuesta al proyecto de la asignatura de Taller III de la Carrera de Ingeniería Informática de la Facultad de Ciencias y Tecnología de la Universidad Autónoma Juan Misael Saracho. Este documento provee la visión global del plan de desarrollo del software propuesto para la CAJA NACIONAL DE SALUD-REGIONAL TARIJA.

El proyecto ha sido ofertado por Garcia Zambrana Rodrigo basado en una metodología de Rational Unified Process en la que únicamente se procederá a cumplir con las tres primeras fases que marca la metodología, constando únicamente en la tercera fase de dos iteraciones. Es importante destacar esto puesto que utilizaremos la terminología RUP en este documento. Se incluirá el detalle para las fases de Inicio y Elaboración y adicionalmente se esbozarán las fases posteriores de Construcción y Transición para dar una visión global de todo proceso.

El enfoque desarrollo propuesto constituye una configuración del proceso RUP de acuerdo a las características del proyecto, seleccionando los roles de los participantes, las actividades a realizar y los artefactos (entregables) que serán generados. Este documento es a su vez uno de los artefactos de RUP.

II.1.1.1.1 Propósito

El propósito del Plan de Desarrollo de Software es proporcionar la información necesaria para controlar el proyecto. En él se describe el enfoque de desarrollo del software.

Los usuarios del Plan de Desarrollo del Software son:

- El jefe del proyecto lo utiliza para organizar la agenda y necesidades de recursos, y para realizar su seguimiento.
- Los miembros del equipo de desarrollo lo usan para entender lo qué deben hacer, cuándo deben hacerlo y qué otras actividades dependen de ello.

II.1.1.1.2 Alcance

El Plan de Desarrollo de Software analiza y describe mejorar el control de la información de los pacientes hospitalizados de la caja nacional de salud- Regional Tarija, utilizando las tres primeras fases de la metodología RUP. Las fases se beneficiaran con un cronograma de cada una de las actividades a realizar. Especificando los detalles de construcción del proyecto. Este Plan contiene los objetivos, alcances y propósito del proyecto, además de las responsabilidades, suposiciones y restricciones, etc.

II.1.1.1.3 Resumen

Después de esta introducción, el resto del documento está organizado en las siguientes secciones:

Vista General del Proyecto — proporciona una descripción del propósito, alcance y objetivos del proyecto, estableciendo los artefactos que serán producidos y utilizados durante el proyecto.

Organización del Proyecto — describe la estructura organizacional del equipo de desarrollo.

Gestión del Proceso — explica los costos y planificación estimada, define las fases e hitos del proyecto y describe cómo se realizará su seguimiento.

Planes y Guías de aplicación — proporciona una vista global del proceso de desarrollo de software, incluyendo métodos, herramientas y técnicas que serán utilizadas.

II.1.1.2 Vista General del Proyecto

II.1.1.2.1 Propósito, Alcance y Objetivos

La información que a continuación se incluye ha sido extraída de las diferentes reuniones que se han realizado con los usuarios finales del producto desde el inicio del proyecto.

II.1.1.2.1.1 Propósito

La CNS-Regional Tarija tiene como objetivo principal brindar atenciones en salud con calidad a la población asegurada con la implementación de planes, programas y control de calidad.

Para coadyuvar a cumplir con los objetivos de la CNS- Regional Tarija y los problemas existentes se necesita administrar la información de los pacientes hospitalizados, asegurados y no asegurados de forma eficiente y eficaz.

El presente proyecto tiene como propósito llevar un Control de la información oportuna y precisa sobre los pacientes hospitalizados en la CNS-Regional Tarija mejorada.

II.1.1.2.1.2 Alcance

Entre los alcances del mismo tenemos los siguientes:

- Registro de pacientes hospitalizados
- Registro de médicos
- Registro de la historia clínica de los PI
- Registro de ingreso del interno
- Registro de egreso del interno
- Reportes de los pacientes hospitalizados
- Resguardo de backup
- Información relevante de la CNS-Regional Tarija

II.1.1.2.1.3 Objetivos

II.1.1.2.1.3.1 Objetivos Generales

El objetivo Principal del presente producto Mejorar el control de la información de manera oportuna y precisa sobre los pacientes hospitalizados de la CNS-Regional Tarija

II.1.1.2.1.3.2 Objetivos Específicos

1.- Desarrollar un sistema informático de control de la información sobre los pacientes hospitalizados en la CNS-Regional Tarija

2.- Capacitar al personal administrativo de la CNS- Regional Tarija en las TIC y el manejo del sistema informático

II.1.1.2.2 Suposiciones y Restricciones

II.1.1.2.2.1 Suposiciones

- El sistema no proporcionará nada útil a menos que haya alguien que introduzca los datos. Se asumirá, por tanto, que el administrador (encargado de sistema) será responsable de realizar una carga inicial de datos de los afiliados, usuarios e información adicional.
- La Sociedad tarijeña hace uso de los servicios que brinda el proyecto.
- Presupuesto suficiente para la implementación del sistema informático
- Se cuenta con recursos Humanos suficientes en el área de administración
- Los miembros del personal administrativo han reconocido la importancia de la capacitación.
- Existe disponibilidad por parte del personal de la institución, para brindar información sobre el manejo de los registros
- Se cuenta con la disponibilidad de los recursos económicos.

Los miembros del área administración han asignado su tiempo para capacitarse

Ambiente de Capacitación adecuado.

- Disponibilidad del personal para asistir a la capacitación..

II.1.1.2.2.2 Restricciones

Dado que el sistema implementará la política y los procesos actualmente vigentes de la CNS-Regional Tarija es de esperar que futuros cambios en los modos de trabajo o en las políticas, ejerzan cierto impacto sobre el sistema.

Por un lado, se espera para la versión alpha la BD se realice en SQL-SERVER, sin embargo la versión BETA se la realizara en postgresSQL.

Otra restricción importante es la naturaleza de la infraestructura software a utilizar, pues siempre será preferible utilizar software libre., por lo tanto el lenguaje de programación a usar será JAVA SERVLET y un servidor multiplataforma y libre.

Por último se implementara SSL para brindar mayor seguridad el sistema informático

II.1.1.2.3 Entregables del proyecto

A continuación se indican y describen cada uno de los artefactos que serán generados y utilizados por el proyecto y que constituyen los entregables. Esta lista constituye la configuración de RUP desde la perspectiva de artefactos, y que proponemos para este proyecto.

Es preciso destacar que de acuerdo a la filosofía de RUP (y de todo proceso iterativo e incremental), todos los artefactos son objeto de modificaciones a lo largo del proceso de desarrollo, con lo cual, sólo al término del proceso podríamos tener una versión definitiva y completa de cada uno de ellos. Sin embargo, el resultado de cada iteración y los hitos del proyecto están enfocados a conseguir un cierto grado de

completitud y estabilidad de los artefactos. Esto será indicado más adelante cuando se presenten los objetivos de cada iteración.

1) Plan de Desarrollo del Software

Es el presente documento.

2) Modelo de Casos de Uso del Negocio

Es un modelo de las funciones de negocio vistas desde la perspectiva de los actores externos (Agentes de registro, solicitantes finales, otros sistemas etc.). permite situar al sistema en el contexto organizacional haciendo énfasis en los objetivos en este ámbito. Este modelo se representa con un Diagrama de Casos de Uso usando estereotipos específicos para este modelo.

3) Modelo de Objetos del Negocio

Es un modelo que describe la realización de cada caso de uso del negocio, estableciendo los actores hospitalizados, la información que en términos generales manipulan y los flujos de trabajo (workflows) asociados al caso de uso del negocio. Para la representación de este modelo se utilizan Diagramas de Colaboración (para mostrar actores externos, hospitalizados y las entidades (información) que manipulan, un Diagrama de Clases para mostrar gráficamente las entidades del sistema y sus relaciones, y Diagramas de Actividad para mostrar los flujos de trabajo.

4) Glosario

Es un documento que define los principales términos usados en el proyecto. Permite establecer una terminología consensuada.

5) Modelo de Casos de Uso

El modelo de Casos de Uso presenta las funciones del sistema y los actores que hacen uso de ellas. Se representa mediante Diagramas de Casos de Uso.

6) **Visión**

Este documento define la visión del producto desde la perspectiva del cliente,

especificando las necesidades y características del producto. Constituye una base de acuerdo en cuanto a los requisitos del sistema.

7) Especificaciones de Casos de Uso

Para los casos de uso que lo requieran (cuya funcionalidad no sea evidente o que no baste con una simple descripción narrativa) se realiza una descripción detallada utilizando una plantilla de documento, donde se incluyen: precondiciones, post-condiciones, flujo de eventos, requisitos no-funcionales asociados. También, para casos de uso cuyo flujo de eventos sea complejo podrá adjuntarse una representación gráfica mediante un Diagrama de Actividad.

8) Prototipos de Interfaces de Usuario

Se trata de prototipos que permiten al usuario hacerse una idea más o menos precisa de las interfaces que proveerá el sistema y así, conseguir retroalimentación de su parte respecto a los requisitos del sistema. Estos prototipos se realizarán como: dibujos a mano en papel, dibujos con alguna herramienta gráfica o prototipos ejecutables interactivos, siguiendo ese orden de acuerdo al avance del proyecto. Sólo los de este último tipo serán entregados al final de la fase de Elaboración, los otros serán desechados. Asimismo, este artefacto, será desechado en la fase de Construcción en la medida que el resultado de las iteraciones vayan desarrollando el producto final.

9) Modelo de Análisis y Diseño

Este modelo establece la realización de los casos de uso en clases y pasando desde una representación en términos de análisis (sin incluir aspectos de implementación) hacia una de diseño (incluyendo una orientación hacia el entorno de implementación), de acuerdo al avance del proyecto.

10) Modelo de Datos

Previendo que la persistencia de la información del sistema será soportada por una base de datos relacional, este modelo describe la representación lógica de los datos persistentes, de acuerdo con el enfoque para modelado relacional de datos. Para expresar este modelo se utiliza un Diagrama de Clases (donde se utiliza un profile UML para Modelado de Datos, para conseguir la representación de tablas, claves, etc.).

11) Modelo de Implementación

Este modelo es una colección de componentes y los subsistemas que los contienen. Estos componentes incluyen: ficheros ejecutables, ficheros de código fuente, y todo otro tipo de ficheros necesarios para la implantación y despliegue del sistema. (Este modelo es sólo una versión preliminar al final de la fase de Elaboración, posteriormente tiene bastante refinamiento).

12) Modelo de Despliegue

Este modelo muestra el despliegue la configuración de tipos de nodos del sistema, en los cuales se hará el despliegue de los componentes.

13) Casos de Prueba

Cada prueba es especificada mediante un documento que establece las condiciones de ejecución, las entradas de la prueba, y los resultados esperados. Estos casos de prueba son aplicados como pruebas de regresión en cada iteración. Cada caso de prueba llevará asociado un procedimiento de prueba con las instrucciones para realizar la prueba, y dependiendo del tipo de prueba dicho procedimiento podrá ser automatizable mediante un script de prueba.

14) Plan de Iteracion

Es un conjunto de actividades y tareas ordenadas temporalmente, con recursos asignados. Se realiza para cada iteración, y para todas las fases.

15) Evaluación de Iteración

Este documento incluye le evaluación de los resultados de cada iteración, el grado en el cual se han conseguido los objetivos de la iteración, las lecciones aprendidas y los cambios a ser realizados.

16) Lista de Riesgos

Este documento incluye una lista de los riesgos conocidos y vigentes en el proyecto, ordenados en orden decreciente de importancia y con acciones específicas de contingencia o para su mitigación.

17) Manual de Instalación

Este documento incluye las instrucciones para realizar la instalación del producto.

18) Material de Apoyo al Usuario Final

Corresponde a un conjunto de documentos y facilidades de uso del sistema, incluyendo: Guías del Usuario, Guías de Operación, Guías de Mantenimiento y Sistema de Ayuda en Línea

19) Producto

Los ficheros del producto empaquetados y almacenadas en un CD con los mecanismos apropiados para facilitar su instalación. El producto, a partir de la primera iteración de la fase de Construcción es desarrollado incremental e iterativamente, obteniéndose una nueva release al final de cada iteración.

Los artefactos 19, 20 y 21 se generarán a partir de la fase de Construcción, con lo cual se han incluido aquí sólo para dar una visión global de todos los artefactos que se generarán en el proceso de desarrollo.

II.1.1.2.4 Evolución del Plan de Desarrollo del Software

El Plan de Desarrollo del Software se revisará semanalmente y se refinará antes del comienzo de cada iteración.

II.1.1.3 Organización del Proyecto

II.1.1.3.1 Participantes en el Proyecto

Director de Proyecto.

Garcia Zambrana Rodrigo

Por ser el único integrante del presente proyecto debo de estar al pendiente de todas las tareas, tener responsabilidad para mantener el orden y coordinación durante el desarrollo del proyecto.

Recabar la información necesaria, analizarla, identificar los problemas y proponer las soluciones.

Facilidad en el manejo de programas y dominio en el área de la programación.

Tener conocimiento metodológico en el desarrollo del software, herramientas de programación.

II.1.1.3.2 Interfaces Externas

no habrá ninguna interfaz software con sistemas externos. Interesa, no obstante, analizar la posibilidad de conectar el sistema con el sistema de afiliación (en caso de implementarse).En la interfaz se podrá obtener información es decir la generación de reportes.

II.1.1.3.3 Roles y Responsabilidades

A continuación se describen las principales responsabilidades de cada uno de los puestos en el equipo de desarrollo durante las fases de Inicio y Elaboración, de acuerdo con los roles que desempeñan en RUP.

Puesto	**Responsabilidad**
Director de Proyecto	El jefe de proyecto asigna los recursos, gestiona las prioridades, coordina las interacciones con los clientes y usuarios, y mantiene al equipo del proyecto enfocado en los objetivos. El jefe de proyecto

	también establece un conjunto de prácticas que aseguran la integridad y calidad de los artefactos del proyecto. Además, el jefe de proyecto se encargará de supervisar el establecimiento de la arquitectura del sistema. Gestión de riesgos. Planificación y control del proyecto. Captura, especificación y validación de requisitos, interactuando con el cliente y los usuarios mediante entrevistas. Elaboración del Modelo de Análisis y Diseño. Colaboración en la elaboración de las pruebas funcionales y el modelo de datos. Construcción de prototipos. Colaboración en la elaboración de las pruebas funcionales, modelo de datos y en las validaciones con el usuario Gestión de requisitos, gestión de configuración y cambios, elaboración del modelo de datos, preparación de las pruebas funcionales, elaboración de la documentación. Elaborar modelos de implementación y despliegue.

Tabla 5. Tabla de roles y responsabilidades

II.1.1.4 Gestión del Proceso

II.1.1.4.1 Estimaciones del Proyecto

El presupuesto del proyecto y los recursos involucrados se encuentran detallados en el perfil del proyecto.

II.1.1.4.2 Plan del Proyecto

En esta sección se presenta la organización en fases e iteraciones y el calendario del proyecto.

II.1.1.4.2.1 Plan de las Fases

El desarrollo se llevará a cabo en base a fases con una o más iteraciones en cada una de ellas. La siguiente tabla muestra una la distribución de tiempos y el número de iteraciones de cada fase (para las fases de Construcción y Transición es sólo una aproximación muy preliminar)

Fase	**Nro. Iteraciones**	**Duración**
Fase de Inicio	3	30
Fase de Elaboración	2	55
Fase de Construcción	1	80
Fase de Transición	1	20

Tabla 6. Tabla de fases

Los hitos que marcan el final de cada fase se describen en la siguiente tabla.

Descripción	**Hito**
Fase de Inicio	En esta fase desarrollarán los requisitos del producto desde la perspectiva del usuario, los cuales serán establecidos en el artefacto Visión. Los principales casos de uso serán identificados y se hará un refinamiento del Plan de Desarrollo del Proyecto. La aceptación del cliente /usuario

	del artefacto Visión y el Plan de Desarrollo marcan el final de esta fase.
Fase de Elaboración	En esta fase se analizan los requisitos y se desarrolla un prototipo de arquitectura (incluyendo las partes más relevantes y / o críticas del sistema). Al final de esta fase, todos los casos de uso correspondientes a requisitos que serán implementados en la primera release de la fase de Construcción deben estar analizados y diseñados (en el Modelo de Análisis / Diseño). La revisión y aceptación del prototipo de la arquitectura del sistema marca el final de esta fase. En nuestro caso particular, por no incluirse las fases siguientes, la revisión y entrega de todos los artefactos hasta este punto de desarrollo también se incluye como hito. La primera iteración tendrá como objetivo la identificación y especificación de los principales casos de uso, así como su realización preliminar en el Modelo de Análisis / Diseño, también permitirá hacer una revisión general del estado de los artefactos hasta este punto y ajustar si es necesario la planificación para asegurar el cumplimiento de los objetivos. Ambas iteraciones tendrán una duración de una semana.

Tabla 7. Tabla de fin de fase

Fase de Construcción	Durante la fase de construcción se terminan de analizar y diseñar todos los casos de uso, refinando el Modelo de Análisis / Diseño. El producto se construye en base a 1

	iteraciones, cada una produciendo una release a la cual se le aplican las pruebas y se valida con el cliente / usuario. Se comienza la elaboración de material de apoyo al usuario. Con la capacidad operacional parcial del producto que se haya considerado como crítica, lista para ser entregada a los usuarios para pruebas beta.
Fase de Transición	En esta fase se prepararán una releases para distribución, asegurando una implantación y cambio del sistema previo de manera adecuada, incluyendo el entrenamiento de los usuarios. El hito que marca el fin de esta fase incluye, la entrega de toda la documentación del proyecto con los manuales de instalación y todo el material de apoyo al usuario, la finalización del entrenamiento de los usuarios y el empaquetamiento del producto.

Tabla 8. Tabla de la construcción de fase

II.1.1.4.2.2 Calendario del Proyecto

A continuación se presenta un calendario de las principales tareas del proyecto incluyendo sólo las fases de Inicio y Elaboración. Como se ha comentado, el proceso iterativo e incremental de RUP está caracterizado por la realización en paralelo de todas las disciplinas de desarrollo a lo largo del proyecto, con lo cual la mayoría de los artefactos son generados muy tempranamente en el proyecto pero van desarrollándose en mayor o menor grado de acuerdo a la fase e iteración del proyecto. La siguiente figura ilustra este enfoque, en ella lo ensombrecido marca el énfasis de cada disciplina (workflow) en un momento determinado del desarrollo.

Para este proyecto se ha establecido el siguiente calendario. La fecha de aprobación indica cuando el artefacto en cuestión tiene un estado de completitud suficiente par

someterse a revisión y aprobación, con una probabilidad de modificación para posterior refinamiento y cambios.

II.1.1.4.3 Seguimiento y Control del Proyecto

II.1.1.4.3.1 Gestión de Requisitos

Este proceso se encarga de la identificación, asignación y seguimiento de los requisitos, conociendo las necesidades que tienen los interesados.

Los requisitos que se han empezado con el proyecto se encuentran en la etapa de análisis, posteriormente, dichos requisitos en el ciclo de vida del proyecto se irán modificando por lo que se establece el tratamiento y control de las actualizaciones y cambios a los mismos.

Debido a que todo proyecto está sujeto a de cambios, se irá actualizando incorporando funcionalidades y/o eliminar otras. Para logra esto se mantendrá un control y documentación del producto.

Reuniendo las necesidades del proyecto, e implementación, los requisitos son:

- Tamaño y complejidad del proyecto,
- Experiencia del personal del proyecto,
- Experiencia de los clientes del proyecto,
- Dominio de la aplicación,
- El propósito y uso de esta aplicación.

Tareas principales de la Gestión de Requisitos

ACTIVIDADES	**DESCRIPCIÓN**
Recolección	Se realizo la recolección y documentación de requisitos, para descubrir, y registrar una representación precisa de los requisitos del producto, identificando las aéreas de mayor importancia. Se procedió a realizar Entrevistas

	donde pretendemos obtener una información general de la Institución y así conocer las fronteras del sistema, conocer sus reglas, aumentar el entendimiento y el compromiso de los participantes.
Documentación	La información recopilada se analizó a detalle y se documentó.
Verificación	Una vez que la descripción de los requisitos ha sido desarrollada, se ha verificado.
Gestión de Cambios	Como el proyecto va evolucionando, Los requerimientos pueden cambiar o expandirse y deberán ajustarse al Proyecto. Para así lograr identificar, evaluar, trazar y reportar cambios propuestos y aprobados a la especificación del producto.

Tabla 9. Tabla de gestión de requisitos

Requisitos para el desarrollo del Sistema

1. Requisitos Organizacionales

El personal debe conocer la metodología RUP bajo un modelado con el lenguaje UML.

Se debe contar con un personal que conozca sobre la plataforma java. El modo de aplicación y trabajo. Contar con el software necesario para el desarrollo del sistema.

2. Requisitos de Personal y Usuarios

Se debe contar por lo menos con una persona informada y capacitada en el manejo del sistema.

Todo usuario debe contar con un usuario y clave.

3. Requisitos de Hardware y funcionamiento

Para el funcionamiento del sistema se debe contar por lo menos con diez ordenadores.

Una LAN (Red de Área Local) con un protocolo TCP/IP.

Los equipos deben contar con un procesador Dual Core y memoria RAM de 1Gb o superior.

Los monitores deben soportar la resolución de 1024 X 768 px.

4. Requisitos de Software y Construcción

El sistema debe tener la capacidad de administrar entre 2 a 3 equipos de computación con una concurrencia de 10 usuarios en línea en un intervalo x de tiempo.

El sistema debe utilizar tecnologías actuales.

Interactuar en una plataforma Windows.

El sistema se lo debe construir bajo una Java Virtual Machine v 6.

El sistema debe tener la capacidad de interactuar con algún motor de base de datos.

5. Requisitos de Software

Java Runtime versión 6 update 2.

Base de datos PostgreSQL 8.4 adelante.

II.1.1.4.3.1.2 Control de Plazos

ID	**FASES**	**Duración Días**	**FECHA DE ENTREGA**	**OBSERVACIONES**
ID	**Fase Inicio**			
0	**Plan de Desarrollo de Software**	**7**	**29/11/2011**	Se cumplió con la fecha establecida. Durante el desarrollo de esta

	(borrador)			fase se ha trabajado en los artefactos: Visión, Glosario, Caso de Uso del Negocio, Modelos de Objetos del Negocio, Casos del uso del sistema, Diagramas de Actividades, Diagramas de Secuencia, Diagramas de Colaboración y el diseño de Pantallas. Durante el desarrollo de esta fase se ha trabajado en los artefactos: Visión, Glosario, Caso de Uso del Negocio, Modelos de Objetos del Negocio, Casos del uso del sistema, Diagramas de Actividades, Diagramas de Secuencia, Diagramas **de** Colaboración, Diagramas de Clases, Diseño de la Base de Datos, Pantallas UML, Mapas Navegacionales y el prototipo en un 40% . Durante el desarrollo de esta fase se ha trabajado en los artefactos: Visión, Glosario, Caso de Uso del Negocio, Modelos de Objetos del

				Negocio, Casos del uso del sistema, Diagramas de Actividades, Diagramas de Secuencia, Diagramas de Colaboración, Diagramas de Clases, Diseño de la Base de Datos, Pantallas UML, Mapas Navegacionales, el prototipo en un 80%, Casos de Prueba (Prueba de Caja Blanca y Prueba de Caja Negra) y Especificación de Métodos. Se cumplió con la fecha establecida.

Tabla 10. Tabla Control de Plazos

II.1.1.4.3.3 Control de Calidad

Para garantizar la calidad en el análisis, diseño y desarrollo del sistema el grupo de desarrollo se baso en tres principios que nos parece importante:

- Control de Variación.
- Calidad en el Diseño.
- Calidad de Concordancia.
- Uso de Herramientas CASE para detección de Errores de Código

Control de Variación: Se fueron realizando controles a las variaciones que se hacía en cada versión del sistema.

Calidad en el Diseño: Se enfatizó el desarrollo de cada uno de los elementos del sistema y se tomó en cuenta los requerimientos planteados por el cliente para evitar problemas con el mismo al momento de su entrega.

Calidad de Concordancia: A medida que se desarrollaba el software se fue concordando el sistema con los requerimientos del cliente.

Se tomo en cuenta la apariencia del sistema verificando los siguientes aspectos.

- Autonomía.-Las interfaces de usuario presentan un ambiente flexible, y de fácil utilización.
- Percepción del color, se eligio el color e imágenes de acuerdo al trabajo del cliente, para dar una apariencia acorde al desarrollo y uso en la ejecución del sistema.
- Eficiencia del usuario.- se dispuso de iconos que vayan de acuerdo a su funcion, para evitar confusiones.
- Interfaces amigables.- Las interfaces son muy amigables, presentando una navegación sencilla, donde cada pantalla presenta enlaces faciles de usar.
- Consistencia, manteniendo una concordancia con el ambiente de trabajo.
- Legibilidad.- La Información presentada es legible por el usuario.

II.1.1.4.3.4 Control de Cambios

Es de vital importancia para el grupo el control de cambios ya que una diminuta perturbación en el código puede crear un gran fallo en el desarrollo del producto, para ello se debe llevar un control de cambios seguro, en el cual el grupo o cliente debe seguir el Siguiente Proceso:

Flujo Principal	**Flujo Alterno**
Se reconoce la necesidad del cambio	
El usuario suscribe la petición de cambio	

El desarrollador la evalúa	
Se genera un informe de cambio	
La autoridad de control de cambios decide	
La petición queda pendiente de actuación se genera la OCI	**Petición de cambios denegada**
Asignación personalizada a los objetos de configuración	**Información del usuario**
“Dar de Baja” objetos de configuración	
Realización del cambio	
Revisión de cambio (Auditoria)	
Los elementos de configuración han cambiado son “Dados de Alta”	
Establecimiento de una línea base para la prueba	
Realización de actividades de garantía de calidad y de prueba	
“Promoción ” de los cambios para ser incluidos en la siguiente versión	
Reconstrucción de la versión adecuada del software	
Revisión (auditoria) de los cambios en todos los elementos de configuración	

Cambios incluidos en la nueva versión	
Distribución de la nueva versión	

Tabla 11. Control de Cambios

II.1.1.4.3.5 Gestión de Configuración

Se realizará una gestión de configuración para llevar un registro de las versiones y de las modificaciones que éstas produzcan, informando y publicando dichos cambios para que sean accesibles a todo los participantes en el proyecto.

Para realizar la gestión de configuración realizaremos las siguientes actividades:

1. Identificación de Elementos de Gestión de Configuración.
2. Control de Versiones.
3. Control de Cambios.
4. Auditoria de Configuración.
5. Generación de Informes.

Se identifico cada uno de los elementos para la realización de la gestión de configuración para un mayor control de los mismos

ID	PDS
Nombre	Plan de Desarrollo de Software
Versión	0.0
Tipo	Documento
Proyecto	Mejoramiento del Control de la Informacion de los Pacientes Hospitalizados en la caja nacional

	de salud- Regional Tarija
Fecha	05/10/2012
Origen del Elemento	Primera fase del proyecto
Responsable	Jefe de Proyecto Rodrigo Garcia Zambrana
Información del Cambio	Se dio una vista preliminar del proyecto, objetivos y alcances.

Tabla 12. Gestión de configuración 1

ID	VS
Nombre	Visión
Versión	0.1
Tipo	Documento
Proyecto	Mejoramiento del Control de la Informacion de los Pacientes Hospitalizados en la caja nacional de salud- Regional Tarija
Fecha	05/10/2012
Origen del Elemento	Visión y requisitos del Sistema
Responsable	Rodrigo Garcia Zambrana
Información del Cambio	Documento de visión del Proyecto

Tabla 13. Gestión de configuración 2

II.1.1.4.3.6 Referencias

Para realizar el Plan de Desarrollo de Software se utilizaron las siguientes referencias:

Referencia
Análisis y diseño de Kendall & Kendall
Ingeniería de Software un enfoque Practico
Análisis y Diseño con Rational
Lenguaje de Modelado Unificado
http://www.dsic.upv.es/asignaturas/facultad/lsi/ejemplorup/Implementacion2.html
http://www.monografias.com/trabajos14/aplicacion-distrib/aplicacion-distrib.shtml
http://www.dsic.upv.es/asignaturas/facultad/lsi/ejemplorup/
http://www.inf.udec.cl/~aimcon/grupouml/apuntecolombia/implementacion01.html

Tabla 14. Referencias

II.1.2 Visión

Este documento define la visión del producto desde la perspectiva del cliente, especificando las necesidades y características del producto. Constituye una base de acuerdo en cuanto a los requisitos del sistema.

II.1.2.1 Introducción

II.1.2.1.1 Propósito

El propósito de éste documento es recoger, analizar y definir las necesidades de alto nivel y las características para mejorar el control de la información de los pacientes hospitalizados de la caja nacional de salud- Regional Tarija. El documento se centra en la funcionalidad requerida por los participantes en el proyecto y los usuarios finales, de modo que dicho departamento sea capaz de atender los distintos trámites, de una forma automatizada.

Los detalles de cómo el sistema cubre los requerimientos se pueden observar en la especificación de los casos de uso y otros documentos adicionales.

II.1.2.1.2 Alcance

Este documento Visión se ocupa, de mejorar el control de la información de los pacientes hospitalizados de la caja nacional de salud, tomando como prueba piloto a la caja nacional de salud- Regional Tarija. Dedicada a brindar atenciones en salud con calidad a la población asegurada con la implementación de planes, programas y control de calidad

El sistema permitirá a los encargados de la Institución Publica controlar todo lo relativo del seguimiento de documentación de trámites, proporcionando facilidad de información sobre el estado de los mismos.

II.1.2.1.3 Definiciones, Acrónimos, y Abreviaciones

C.N.S. Son las siglas de Caja Nacional de Salud

R.U.P. Son las siglas de Rational Unified Process. Se trata de una metodología para escribir el proceso de desarrollo de software.

SUMI.- seguro universal materno infantil

SSPAM.- seguro social de personas ancianas y mayores.

II.1.2.1.4 Referencias

- Glosario.
- Plan de desarrollo de software.
- RUP (Rational Unified Process).
- Diagrama de casos de uso.

I.1.2.2 Posicionamiento

I.1.2.2.1 Oportunidad de Negocio

Este sistema permitirá a la Institución Publica automatizar el control de la información, el cual permitirá a nuestro usuario acceder de manera factible y simple. Mediante interfaces gráficas manejables, los datos se almacenaran en una base de datos que se mantendrá siempre actualizada, permitiendo una administración eficiente y mejor. Coayudando a la toma de decisiones.

II.1.2.2.2. Sentencia que define el problema

El problema de	**Controlar** la administración de la información de los asegurados, beneficiarios, así también de los no asegurados como el SUMI, SPAM y otros que reciben atención en medicina interna. Gestionar la lista de pacientes hospitalizados Controlar el registro de médicos Gestionar la de la historia clínica de los PI Gestionar el registro de ingreso del interno

	Gestionar el registro de egreso del interno Gestionar los reportes de los pacientes hospitalizados Gestionar el resguardo de los backup. Gestionar la información relevante de la CNS-Regional Tarija
afecta a	Caja Nacional de Salud - Administrativos - Pacientes **-Poblacion asegurados**
El impacto asociado es	El proyecto está dirigido a mejorar la administración de la información de los asegurados, beneficiarios, así también de los no asegurados como el SUMI, SPAM y otros que reciben atención en medicina interna. El proyecto se centra en administrar la información de relevancia de los pacientes, historia clínica informatizada, para así de esta manera hacer un seguimiento del estado de salud de los mismos. Además de reducir las barreras de acceso a la información.
una adecuada solución sería	Automatizar el proceso, usando un sistema que contenga información al

	momento que se la requiera. Desarrollando una base de datos accesible desde los distintos puntos de la red local (si contase con ella) Generar una interfaz amigable y sencillas con las que las que se accede a la base de datos.

Tabla 15. Sentencia de Problema

II.1.2.2.3. Sentencia que define la posición del Producto

Para	Caja Nacionalde Salud -Personal de Estadisticas -Medicos -Pacientes hospitalizados -Poblacionen general ModuloSistema -Rol Acceso -backup Modulo Administracion -Camas -Salas -Especialidad -Personal Modulo Ingreso

	-Ingreso del Paciente Modulo Egreso -Egreso del Paciente Modulo Servicio Modulo Ayuda Modulo Reportes
Quienes	Controlan la información de los pacientes hospitalizados de la caja nacional de salud- Regional Tarija
El nombre del producto	Es una herramienta software. Mejoramiento del Control de la Informacion de los Pacientes Hospitalizados en la caja nacional de salud- Regional Tarija
Que	Permite mejorar la administración de la información de los asegurados, beneficiarios, así también de los no asegurados como el SUMI, SPAM y otros que reciben atención en medicina interna. El proyecto se centra en administrar la información de relevancia de los pacientes, historia clínica informatizada, para así de esta manera hacer un seguimiento del estado de salud de los mismos. Además de reducir

	las barreras de acceso a la información.
No como	Se maneja actualmente, de manera manual.
Nuestro producto	Con la elaboración del sistema de gestión, pretende manejar de manera eficaz y eficiente la información relacionada con los pacientes de la CNS-Regional Tarija, de esta manera se beneficiara a la parte administrativa, asegurados y/o familiares y población en general.

Tabla 16. Definición de posición del Proyecto

II.1.2.3 Descripción de Stakeholders (Participantes en el Proyecto) y Usuarios

Para proveer de una forma efectiva productos y servicios que se ajusten a las necesidades de los usuarios, es necesario identificar e involucrar a todos los participantes en el proyecto como parte del proceso de modelado de requerimientos. También es necesario identificar a los usuarios del sistema y asegurarse de que el conjunto de participantes en el proyecto los representa adecuadamente. Esta sección muestra un perfil de los participantes y de los usuarios involucrados en el proyecto, así como los problemas más importantes que éstos perciben para enfocar la solución propuesta hacia ellos. No describe sus requisitos específicos ya que éstos se capturan mediante otro artefacto. En lugar de esto proporciona la justificación de por qué estos requisitos son necesarios.

II.1.2.3.1 Resumen de Stakeholders

Nombre	descripción	responsabilidades
Director del hospital Obrero Nº7	Representante de la Caja Nacional de Salud frente a reuniones realizadas a nivel nacional. Además de ser la autoridad maxima dentro de dicha Institucion.	El stakeholder realiza: Representa a todos los usuarios posibles del sistema. Seguimiento del desarrollo del proyecto. Aprueba requisitos y funcionalidades
Administrador de Hospital Obrero	Encargado de coordinar todas las actividades del hospital de manera interna realizar control docente y otros.	El stakeholder realiza: Coordinar con los administrativo de la C.N.S., reuniones informar al director cualquier anomalía en la institucion, representante directa del director en causa de ausencia.

Tabla 17. Stakeholder

II.1.2.3.2 Resumen de Usuarios (Tomando en cuenta la Caja Nacional de Salud)

Nombre	Descripcion	Stakeholder
Director del Hospital	Responsable de controlar	Dr. Franco Bass Werner

Obrero	la administración y coordinación de la institución, administrar las camas, salas, personal, ingresos, egresos reportes y otros que impliquen latoma de desiciones. Manejo general del Hospital.	
Administrador del Hospital Obrero	Responsable de controlar al personal	Lic. Arturo Tavera Gonzales.
Encargada de Estadisticas	Responsable de controlar estadísticas de; camas, salas, pacientes, patologías, ingresos,egresos, diagnoticos, y otros.	Dra. Luisa Guerrero
Auxiliar de Estadisticas	Encargada de colaborar en la realización de las estadísticas a la dra.	Ines Cardozo Lopez

Tabla 18. Descripción de Usuarios

II.1.2.3.3 Entorno de usuario

Los usuarios entrarán al sistema identificándose sobre un ordenador con un sistema operativo Windows Vista y tras este paso entrarán introduciendo login y password a

la parte de aplicación diseñada para cada uno según su papel en la Intitucion (C.N.S.) Este sistema es similar a cualquier aplicación Windows y por tanto los usuarios estarán familiarizados con su entorno.

II.1.2.3.4 Perfil de los Stakeholders

1 Director del Hospital Obrero

Representante	Dr. Franco Bass Werner
Descripción	Director del Hospital Obrero
Tipo	Doctor
Responsabilidades	Encargado de controlar la administración y coordinación de la institución, las camas, salas, personal, ingreso, egresos reportes y otros que impliquen latoma de desiciones. Manejo general del Hospital.
Criterio Éxito	A definir por el cliente
Grado participación	Tienes privilegios para todos los módulos del sistema
Comentarios	Ninguno

Tabla 19. Stakeholder de Director

2 Administrador del Hospital Obrero

Representante	Lic. Arturo Tavera Gonzales.
Descripción	Administrador encargado de controlar al personal administrativo
Tipo	Licenciado

Responsabilidades	Encargado de coordinar las actividades del personal administrativo de la institución.
Criterio de Éxito	A definir por el cliente
Grado de participación	A definir por el cliente
Comentarios	Ninguno

Tabla 20. Stakeholder de administrador de H. Obrero

3 **Encargada de Estadisticas**

Representante	Dra. Luisa Guerrero
Descripción	Encargada de Estadisticas
Tipo	Doctora
Responsabilidades	Responsable de controlar estadísticas de; camas, salas, pacientes, patologías, ingreso,egresos, diagnoticos, y otros.
Criterio de Éxito	A definir por el cliente
Grado de participación	A definir por el cliente
Comentarios	Ninguno

Tabla 21. Stakeholder de Encargada de estadística

4 Auxiliar **de Estadisticas**

Representante	Ines Cardozo Lopez
Descripción	Encargada de Estadisticas
Tipo	auxiliar
Responsabilidades	Encargada de colaborar en la realización de las estadísticas a la dra.
Criterio de Éxito	A definir por el cliente
Grado de participación	A definir por el cliente
Comentarios	Ninguno

Tabla 21-1. Stakeholder de Auxiliar de estadistica

II.1.2.3.5 Perfiles de usuario

1 Director del Hospital Obrero Nº7

Representante	Dr. Franco Bass Werner
Descripción	Director del Hospital Obrero Nº7
Tipo	Doctor
Responsabilidades	Responsable de controlar la administración y coordinación de la institución, administrar las camas, salas, personal, ingreso, egresos reportes y otros que impliquen latoma de desiciones. Manejo general del Hospital.

Criterio de Éxito	A definir por el cliente
Grado de participación	Tienes privilegios para todos los módulos del sistema
Comentarios	Ninguno

Tabla 22. Usuario Director del H. Obrero

2 Administrador del Hospital Obrero

Representante	Lic. Arturo Tavera Gonzales.
Descripción	Administrador del Hospital Obrero
Tipo	Licenciado
Responsabilidades	Responsable de controlar al personal
Grado de participación	A definir por el cliente

Tabla 23. Usuario Administrador del hospital

3 Encargada de Estadisticas

Representante	Dra. Luisa Guerrero
Descripción	Encargada de Estadisticas
Tipo	Doctora
Responsabilidades	Responsable de controlar estadísticas de; camas, salas, pacientes, patologías, ingreso,egresos, diagnoticos, y otros.

Criterio de Éxito	A definir por el cliente
Grado de participación	A definir por el cliente
Comentarios	Ninguno

Tabla 24. Usuario Encargada de estadistica

II.1.2.4 Descripción Global del Producto

II.1.2.4.1 Perspectiva del producto

El producto a desarrollar es un sistema dirigido a la caja nacional de salud. Este sistema está siendo realizado tomando en cuenta la Regional Tarija, con la intención de agilizar su funcionamiento y tener un mejor control de las actividades que se realizan dentro. Las áreas a tratar por el sistema son: administración, ingreso, egresos, servicios, ayuda y reportes.

II.1.2.4.2 Resumen de características

A continuación se mostrará un listado con los beneficios que obtendrá el cliente a partir del producto:

Beneficio del cliente	**Características que lo apoyan**
Mayor agilidad al obtener las observaciones	Fácil acceso a la base de datos.
Mayor control en el estado de los tramites	Restringir los procesos no permitidos a algunos usuarios.
Mayor control en la administración de salas existentes dentro de la norma establecida.	El sistema no permitirá irregularidades en la administración de las salas.

Generar advertencias en el momento del ingreso del paciente.	Generar advertencias.
Mayor facilidad en la generación y recepción de reportes.	Obteniéndolas mediante herramientas especializadas.

Tabla 25. Resumen de características

ModuloSistema

Con interfaces sencillas y amigables la administración del sistema permite al usuario administrar los roles y generar los backup del sistema, donde de acuerdo al tipo de usuario que es accede a sus menús respectivos. Permitiendo

-Rol Acceso

-backup

Modulo Administracion

Con interfaces sencillas y amigables el modulo administración realiza la administración de las camas, salas, especialidad y personal, registrando sus datos e información necesaria para la C.N.S., permitiendo administrar:

-Camas

-Salas

-Especialidad

-Personal

Modulo Ingreso

Modulo en el que se realiza el registro del ingreso del paciente y todos los datos necesarios del mismo.

-Ingreso del Paciente

Modulo Egreso

Modulo en el que se realiza el registro del ingreso del paciente y todos los datos necesarios del mismo.

-Egreso del Paciente

Modulo Servicio

Modulo en que administra los servicios que brinda el hospital obrero, administración de los pacientes que los utilizan, los ingresos que manejan estos servicos.

Reportes

En este modulo el usuario genera reportes requeridos previa verificación de su clave.

Ayuda

Modulo en el se puede obtener ayuda o consultar sobre algúna aplicación del sistema, este modulo no tiene restricciones siendo permitido para cualquier usuario.

Restricciones

- *Poco uso del Teclado.*
- Poco uso de memoria.

II.1.3 GLOSARIO

II.1.3.1. Introducción

Este documento recoge todos y cada uno de los términos manejados a lo largo de todo el Proyecto Mejoramiento del Control de la Informacion de los Pacientes Hospitalizados en la caja nacional de salud- Regional Tarija. Se trata de un diccionario informal de datos y definiciones de la nomenclatura que se maneja, de tal modo que se crea un estándar para todo el proyecto.

II.1.3.1.1 Propósito

El propósito de este glosario es definir con exactitud y sin ambigüedad la terminología manejada en el proyecto de desarrollo, también sirve como guía de consulta para la clarificación de las palabras confusas.

II.1.3.1.2 Alcance

El alcance del presente documento se extiende a todos los componentes del proyecto. De tal manera que la terminología empleada, se refleje con claridad en este documento.

II.1.3.1.3 Referencias

El presente Glosario hace referencia a los siguientes documentos:

- Perfil del proyecto
- Documento Plan de Desarrollo de Software
- Documento de Caso de Uso de Negocio
- Documento Modelos de Objetos del Negocio
- Documento Visión
- Documento Modelo de Casos de Uso
- Documento Especificación de los Casos de Uso
- Documento Modelo de Análisis y Diseño

- Documento Modelo de Datos

- Documento Prototipos de Interfaces de Usuario

II.1.3.1.4 Organización del Glosario

El presente documento está organizado por definiciones de términos ordenados de forma ascendente según la ordenación alfabética tradicional de español.

II.1.3.2 Definiciones

A continuación se presentan todos los términos manejados a lo largo del desarrollo del proyecto Mejoramiento del Control de la Informacion de los Pacientes Hospitalizados en la caja nacional de salud- Regional Tarija.

- **ESalud** se defi ne como la aplicación de las Tecnologías deInformación y Comunicación (TIC) en el amplio rango de aspectosque afectan el cuidado de la salud.
- **Base de datos** Se encarga de almacenar toda la información requerida de la unidad Educativa.
- **Java (Java)** Lenguaje de programación desarrollado por la empresa Sun para la elaboración de pequeñas aplicaciones exportables a la red (*applets*) y capaces de operar sobre cualquier plataforma a través, normalmente, de navegadores WWW. Permite dar dinamismo a las páginas web y aplicaciones de Escritorio.
- **Núcleo** Conjunto de Redes en que están organizadas las unidades educativas**,** conformada por director de núcleo, quien tiene jurisdicción y competencia sobre el conjunto de unidades educativas que conforman un núcleo educativo**Red** Conformadas por unidades educativas, cada red está compuesta por el director, docentes, padres de familia, juntas escolares y alumnos
- **Prueba Piloto** Prueba del sistema por un periodo de tiempo, para monitorear el desenvolvimiento de los distintos actores, para detectar y corregir errores antes de implementarlo.

- **RUP** Son las siglas de Rational Unified Process. Se trata de una metodología para describir el proceso de desarrollo de software, basada en cuatro fases y cada una de ellas en iteraciones.
- **Sistema Informático** Conjunto de partes (hardware y software) que funcionan relacionándose entre sí con un objetivo preciso. Los usuarios son parte del sistema informático.
- **Sistema Operativo** Un sistema operativo (SO) es un conjunto de programas o software destinado a permitir la comunicación del usuario con un ordenador y gestionar sus recursos de manera cómoda y eficiente. Comienza a trabajar cuando se enciende el ordenador, y gestiona el hardware de la máquina desde los niveles más básicos .Ejemplos Windows, Linux, MacOS, Solaris.
- **TIC** Siglas que significan Tecnologías de información y Comunicación. Las TIC son medios que aportan un flujo ininterrumpido de información.
- **UML** Son las siglas de Lenguaje Unificado de Modelación. Es una lenguaje grafico para visualizar, especificar construir y documentar un sistema de software

II.1.4 MODELO DE CASO DE USO DEL NEGOCIO

II.1.4.1 Introducción

El modelo de Casos de Uso del Negocio. Es un artefacto de la disciplina Requisitos en la metodología RUP la cual se está implementando.

II.1.4.1.1 Propósito

- Comprender la estructura y la dinámica del establecimiento educativo
- Comprender los problemas actuales e identificar posibles mejoras.
- Comprender los procesos del negocio del establecimiento educativo

II.1.4.1.2 Alcance

- Describir los procesos de negocio y los usuarios
- Identificar y definir los procesos del negocio según los objetivos de la institucion.
- Definir un caso de uso del negocio para cada proceso del negocio (diagrama de casos de uso del negocio puede mostrar el contexto y los límites del establecimiento.)

II.1.4.2 Diagrama de Casos de Uso del Negocio

II.1.4.2 .1 director del Hospital Obrero

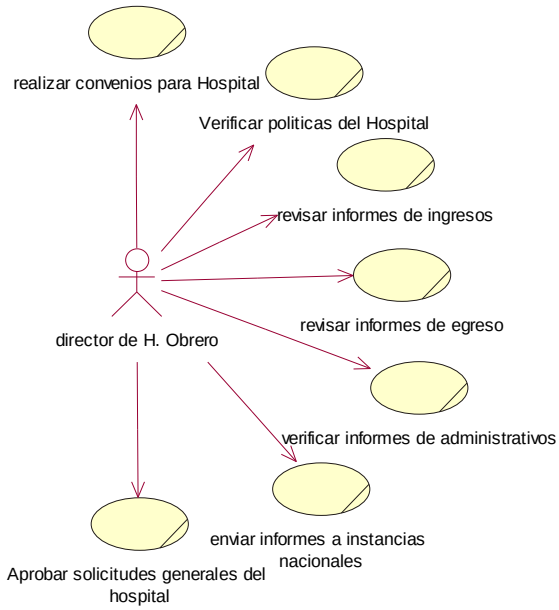


Figura 3. D.C.U. director del Hospital Obrero

II.1.4.2 .2 Administrador del Hospital Obrero

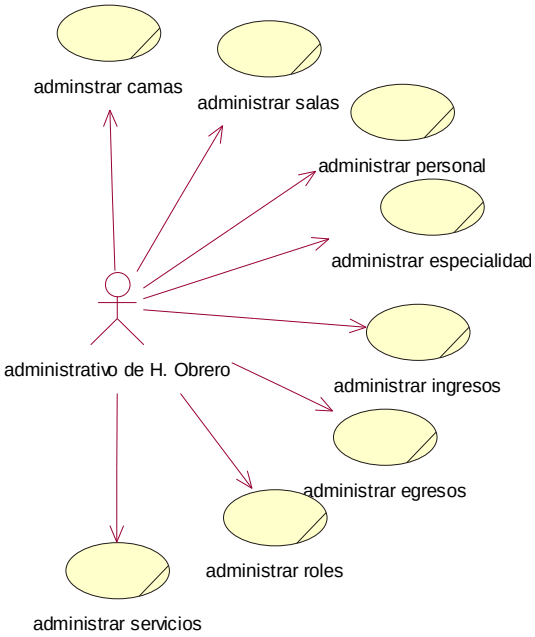


Figura 4. D.C.U. Administrador del Hospital Obrero

II.1.4.2 .3 Encargada de Estadisticas

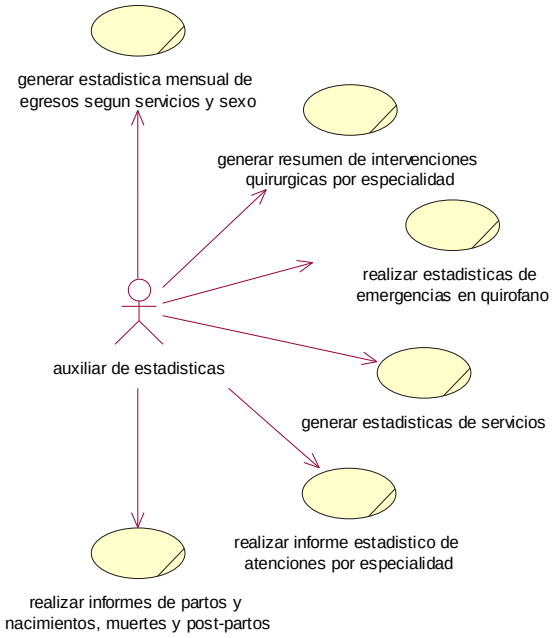


Figura 5. D.C.U. Encargada de Estadisticas

II.1.4.2 .4 Auxiliar de estadisticas

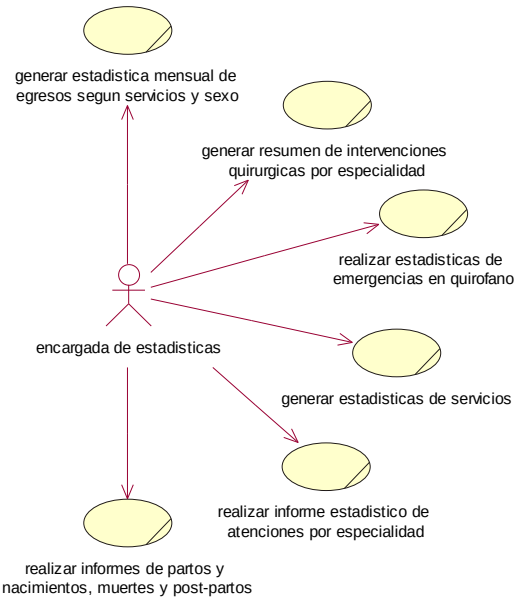


Figura 6. D.C.U. Auxiliar de estadisticas

II.1.5 MODELO DE OBJETO DEL NEGOCIO

II.1.5.1 Introducción

El Modelo de Objeto del Negocio es un artefacto de la disciplina Requisitos en la metodología RUP que se está implementando.

II.1.5.1.1 Propósito

- Comprender la estructura dinámica de los casos de uso del negocio del proyecto Mejoramiento del Control de la Información de los Pacientes Hospitalizados en la Caja Nacional de Salud Regional Tarija
- Comprender los procesos de negocio de la organización

II.1.5.1.2 Alcance

- Describe el comportamiento de los procesos de negocio
- Identificar y definir los objetos del negocio

II.1.5.2 Diagrama de Objetos del Negocio

II.1.5.2.1 director del Hospital Obrero

II.1.5.2.1.1 modelo de objeto realizar convenios para el hospital

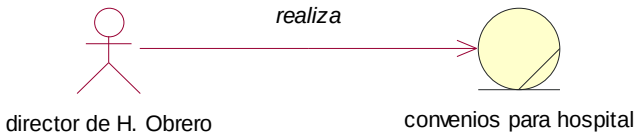


Figura 7. M.O.N. realizar convenios para el hospital

II.1.5.2.1.2 modelo de objeto verificar políticas del hospital

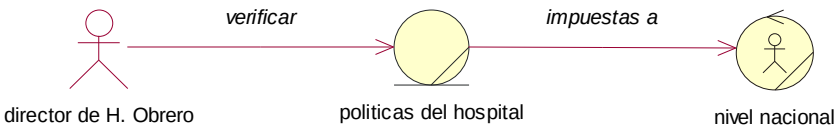


Figura 8. M.O.N. verificar políticas del hospital

II.1.5.2.1.3 modelo de objeto revisar informes de ingreso

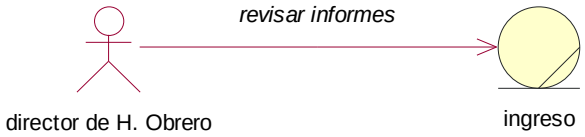


Figura 9. M.O.N. revisar informes de ingreso

II.1.5.2.1.4 modelo de objeto revisar informes de egreso

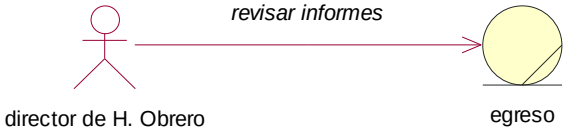


Figura 10. M.O.N. revisar informes de egreso

II.1.5.2.1.5 modelo de objeto revisar informes de administrativos

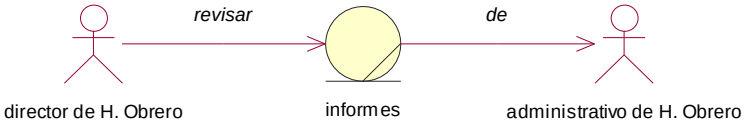


Figura 11. M.O.N. verificar informes de administrativos

II.1.5.2.1.6 modelo de objeto enviar informes a instancias nacionales

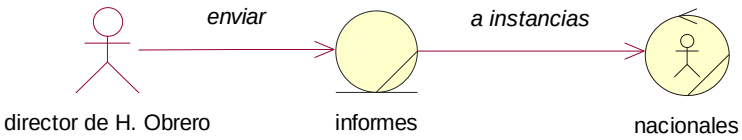


Figura 12. M.O.N. enviar informes a instancias nacionales

II.1.5.2.1.7 modelo de objeto aprobar solicitudes generales al hospital

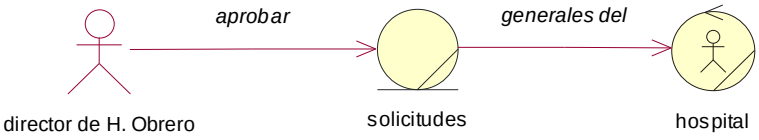


Figura 13. M.O.N. aprobar solicitudes generales al hospital

II.1.5.2.2 Administrativo del Hospital Obrero

II.1.5.2.2.1 modelo de objeto administrar camas

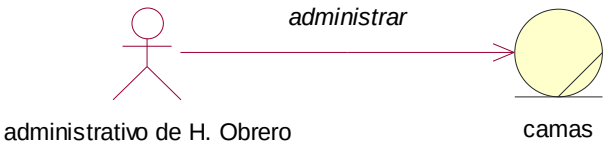


Figura 14. M.O.N. administrar camas

II.1.5.2.2.2 modelo de objeto administrar salas

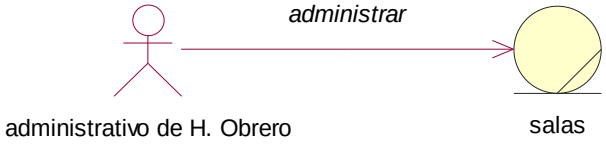


Figura 15. M.O.N. administrar camas

II.1.5.2.2.3 modelo de objeto administrar personal

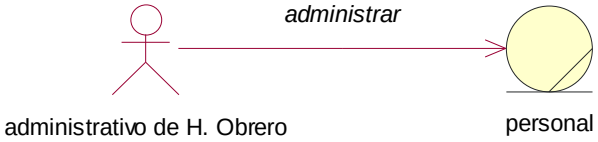


Figura 16. M.O.N. administrar personal

II.1.5.2.2.4 modelo de objeto administrar especialidad

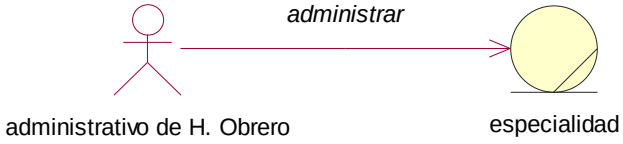


Figura 17. M.O.N. administrar especialidad

II.1.5.2.2.5 modelo de objeto administrar ingresos

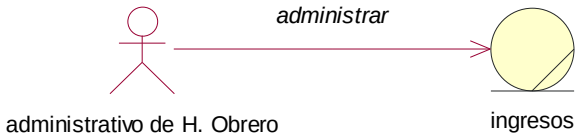


Figura 18. M.O.N. administrar ingresos

II.1.5.2.2.6 modelo de objeto administrar egresos

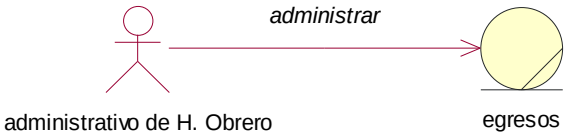


Figura 19. M.O.N. administrar egresos

II.1.5.2.2.7 modelo de objeto administrar servicios

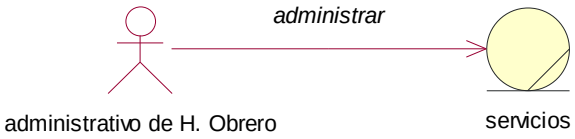


Figura 20. M.O.N. administrar servicios

II.1.5.2.3 Encargada de estadisticas

II.1.5.2.3 .1 modelo de objeto generar estadistica mensual de egresos segun servicios y sexo

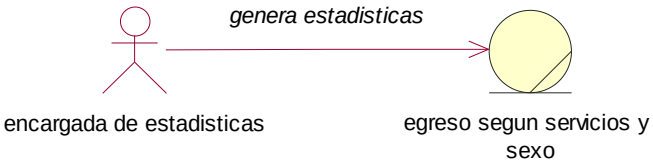


Figura 21. M.O.N. generar estadistica mensual de egresos segun servicios y sexo

II.1.5.2.3.2 modelo de objeto generar resumen de intervenciones quirurgicas por especialidad

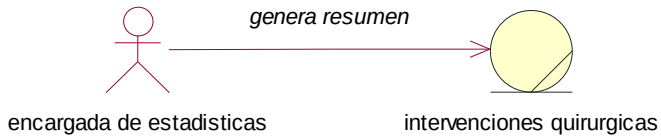


Figura 22. M.O.N. generar resumen de intervenciones quirurgicas por especialidad

II.1.5.2.3 .3 modelo de objeto realizar estadisticas de emergencias en quirofano

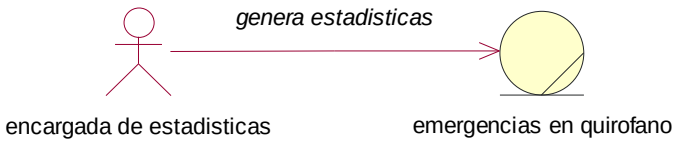


Figura 23. M.O.N. realizar estadisticas de emergencias en quirófano

II.1.5.2.3 .4 modelo de objeto generar estadisticas de servicios

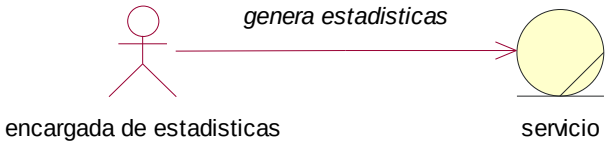


Figura 24. M.O.N. generar estadisticas de servicios

II.1.5.2.3.5 modelo de objeto realizar informe estadistico de atenciones por especialidad

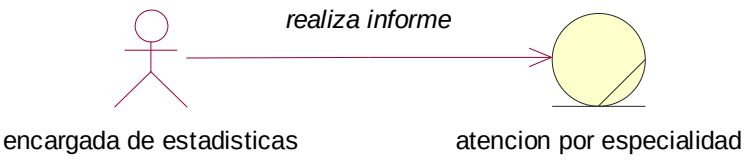


Figura 25. M.O.N. realizar informe estadistico de atenciones por especialidad

II.1.5.2.3.6 modelo de objeto realizar informes de partos y nacimientos, muertes y post-partos

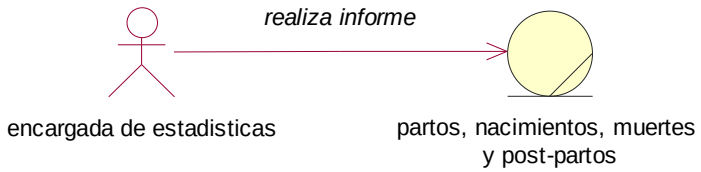


Figura 26. M.O.N. realizar informes de partos y nacimientos, muertes y post-partos

II.1.5.2.4 Auxiliar de estadistica

II.1.5.2.4.1 modelo de objeto generar estadistica mensual de egresos segun servicios y sexo

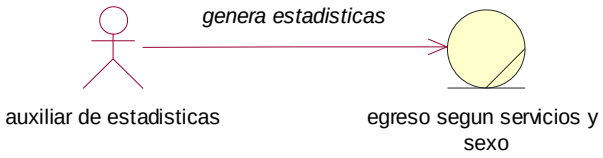


Figura 27. M.O.N. generar estadistica mensual de egresos segun servicios y sexo

II.1.5.2.4.2 modelo de objeto generar resumen de intervenciones quirurgicas por especialidad

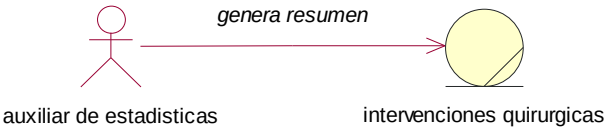


Figura 28. M.O.N. generar resumen de intervenciones quirurgicas por especialidad

II.1.5.2.4.3 modelo de objeto realizar estadisticas de emergencias en quirófano

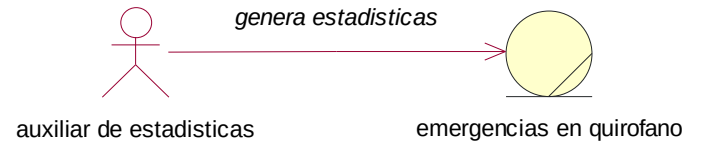


Figura 29. M.O.N. realizar estadisticas de emergencias en quirófano

II.1.5.2.4.4 modelo de objeto generar estadisticas de servicios

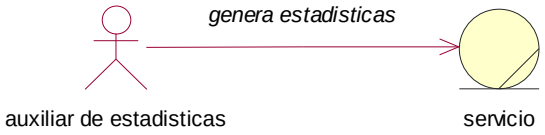


Figura 30. M.O.N. generar estadisticas de servicios

II.1.5.2.4.5 modelo de objeto realizar informe estadistico de atenciones por especialidad

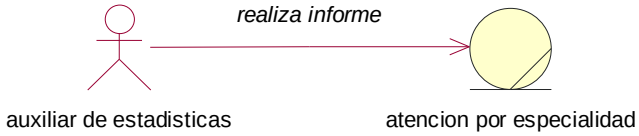


Figura 31. M.O.N. realizar informe estadistico de atenciones por especialidad

II.1.5.2.4.5 modelo de objeto realizar informes de partos y nacimientos, muertes y post-partos

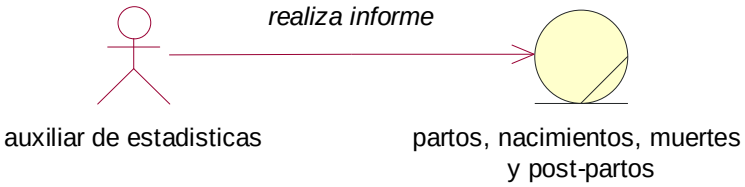


Figura 32. M.O.N. realizar informes de partos y nacimientos, muertes y post-partos

II.1.6 MODELOS DE CASO DE USO

II.1.6.1 Introducción

Los casos de uso nos sirven para presentar la manera de cómo el cliente interactúa con el sistema que se desarrolla

II.1.6.1.1 Propósito

- Modelar el contexto del sistema
- Modelar los requerimientos del sistema
- Identificar los procesos del sistema
- Estimular a que los usuarios potenciales hablen del sistema desde su propio punto de vista
- Involucrar a los usuarios en las etapas iniciales del análisis y diseño del sistema
- Ayudar en la obtención de los requerimientos desde el punto de vista del usuario

II.1.6.1.2 Alcance

- Describir lo que el sistema informático realizara dentro del negocio
- Describir los alcances del sistema
- Describir los procesos del sistema y del cliente

II.1.6.2 Diagrama de Casos De Uso

II.1.6.2.1 Caso de Uso General

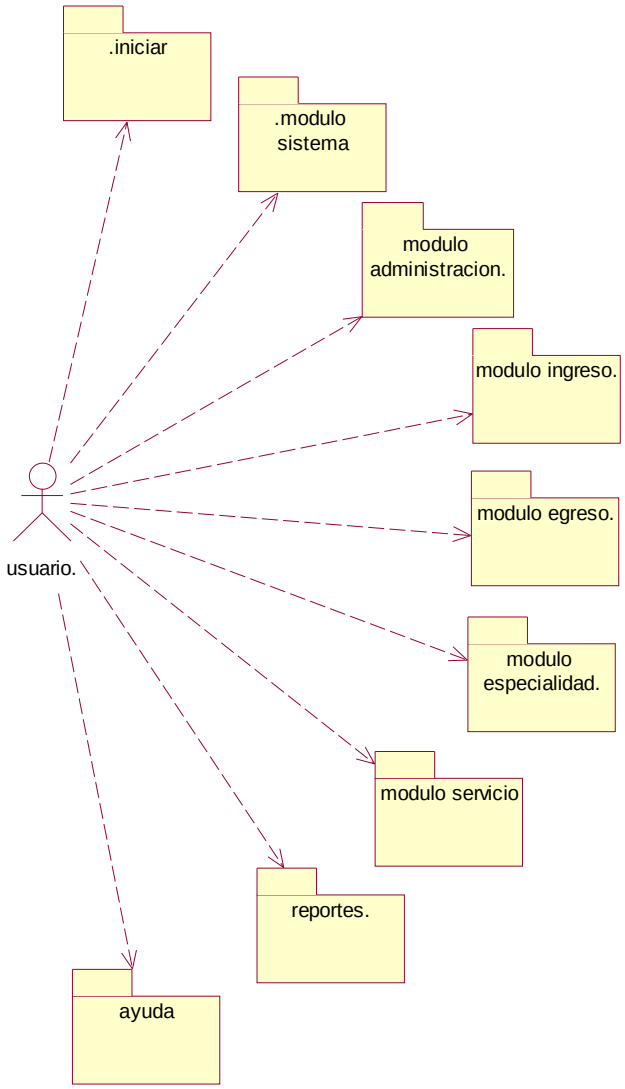


Figura 34. D.C.U. Caso de Uso General

II.1.6.2.2 Caso de Uso Específico

II.1.6.2.2.1. iniciar

Figura 36. D.C.U. iniciar

II.1.6.2.2.2. Modulo sistema

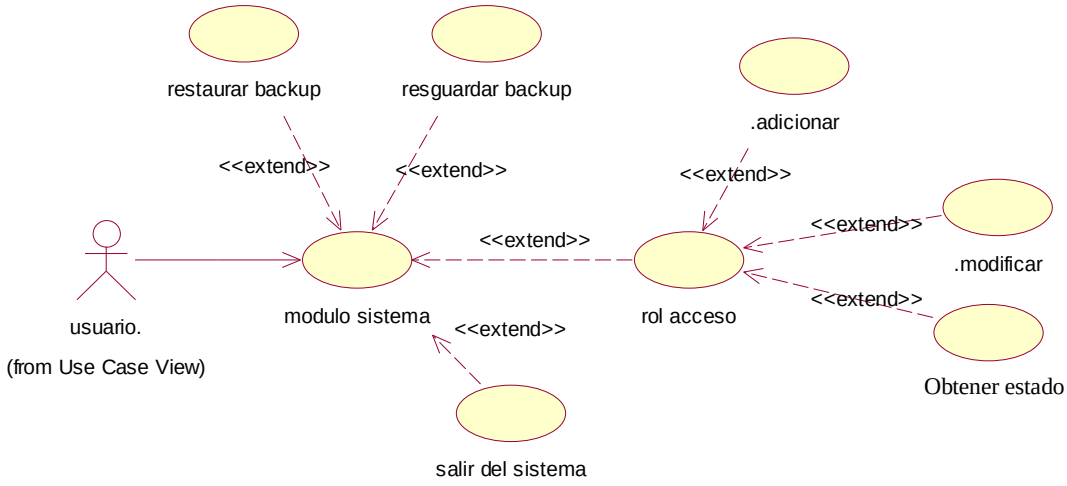


Figura 37. D.C.U. Modulo Sistema

II.1.6.2.2.3. Modulo administración

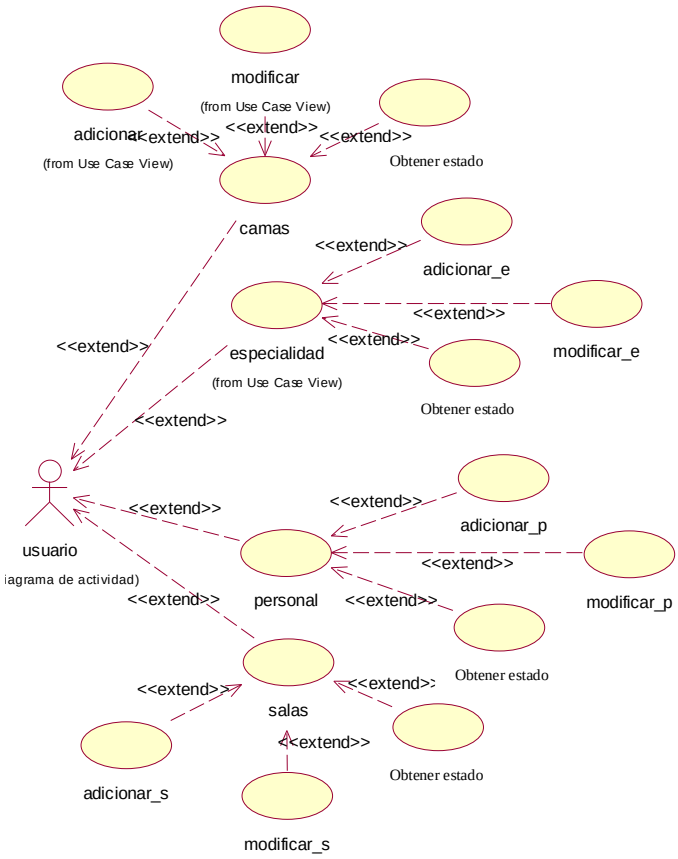


Figura 38. D.C.U. Modulo Administracion

II.1.6.2.2.4. Modulo Ingreso

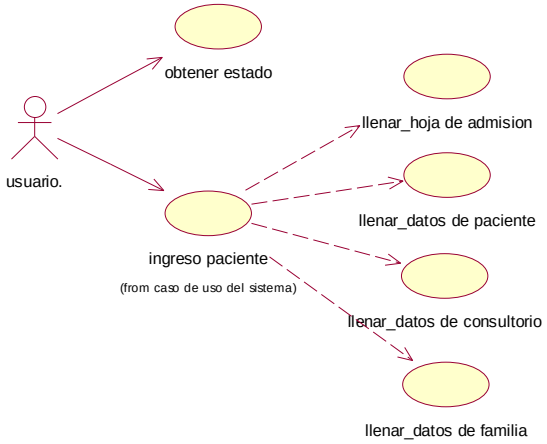


Figura 39. D.C.U. Modulo Ingreso

II.1.6.2.2.5. Modulo Egreso

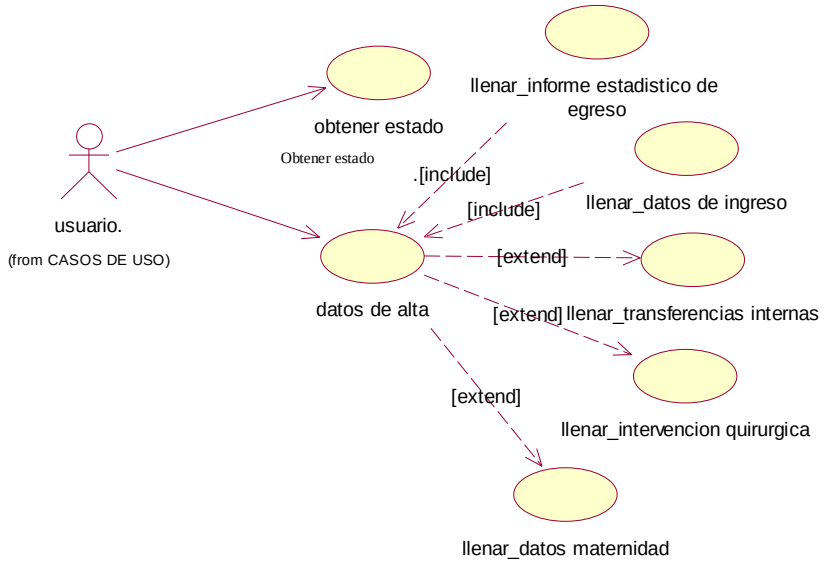


Figura 40. D.C.U. Modulo Egreso

II.1.6.2.2.6. Modulo servicios

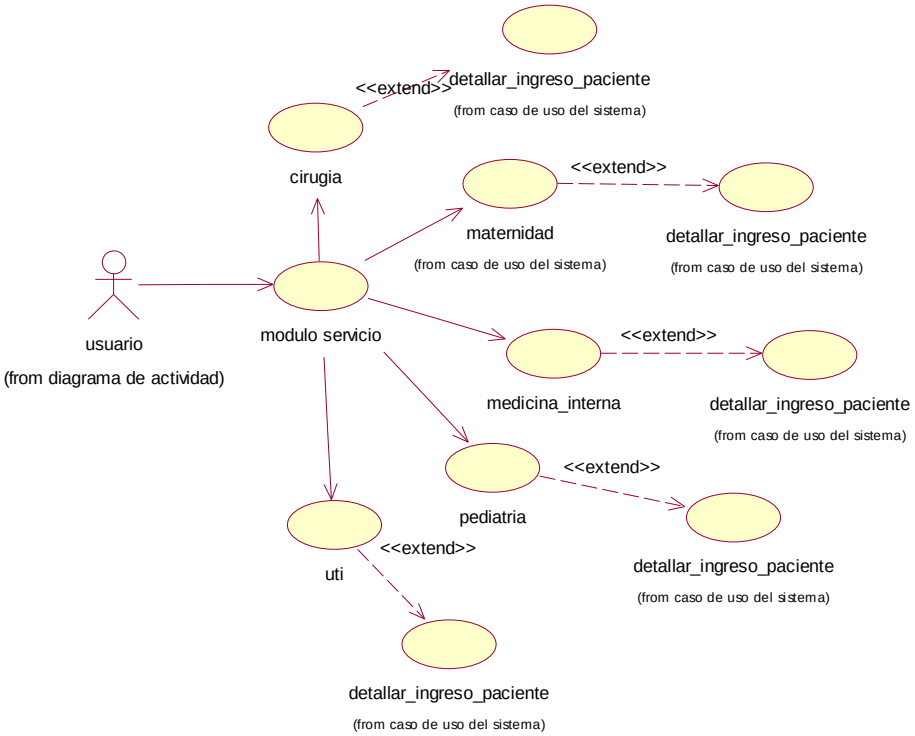


Figura 41. D.C.U. Modulo Servicios

II.1.6.2.2.7. Reportes

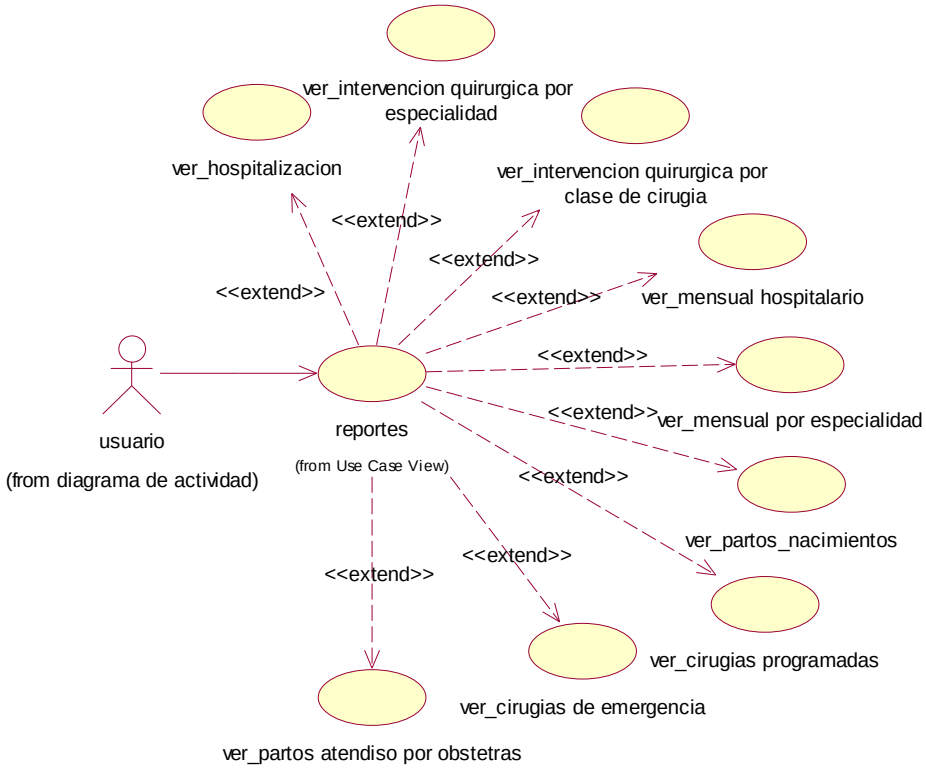


Figura 42. D.C.U. Reportes

II.1.6.2.2.8. Modulo Ayuda

Figura 43. D.C.U. Modulo Ayuda

II.1.7 Especificaciones de Casos de Uso

Para los casos de uso que lo requieran (cuya funcionalidad no sea evidente o que no baste con una simple descripción narrativa) se realiza una descripción detallada utilizando una plantilla de documento, donde se incluyen: precondiciones, post-condiciones, flujo de eventos, requisitos no-funcionales asociados. También, para casos de uso cuyo flujo de eventos sea complejo podrá adjuntarse una representación gráfica mediante un Diagrama de Actividad.

II.1.7.1. Introducción

Las especificaciones de los Casos de Uso es una descripción detallada de los casos de uso identificados en el análisis.

II.1.7.1.1. Propósito

Describir específicamente cada caso de uso. Comprender el funcionamiento de los casos de Uso del Sistema.

II.1.7.1.2. Alcance

- Describir los procesos hospitalizados del sistema en cada Caso de Uso
- Detallar los flujos de cada caso según lo establecido.

II.1.7.2. Especificación de los actores

II.1.7.2.1 Director del Hospital Obrero

Responsable de controlar la administración y coordinación de la institución, realizar convenios para Hospital, Verificar politicas del Hospital , revisar informes de ingresos, revisar informes de egreso, verificar informes de administrativos, enviar informes a instancias nacionales, Aprobar solicitudes generales del hospital, y actividades que impliquen la toma de desiciones. Manejo general del Hospital.

II.1.7.2.2. Administrativo del Hospital Obrero

Encargado de administrar las especialidades, servicios, camas, salas, personal, ingresos, egresos reportes y otros en general dentro del hospital

II.1.7.2.3. Encargada de estadisticas

Es la encargada de generar estadistica mensual de egresos segun servicios y sexo, generar resumen de intervenciones quirurgicas por especialidad, realizar estadisticas de emergencias en quirófano, generar estadisticas de servicios, realizar informe estadistico de atenciones por especialidad, realizar informes de partos y nacimientos, muertes y post-partos, y otras estadísticas en caso de requerirlas.

II.1.7.2.4. Auxiliar de estadísticas

Es la auxiliar y encargada de apoyar en la generación de reportes y estadísticas a la encargada de estadísticas y otras estadísticas en caso de requerirlas.

II.1.7.3. Especificación de casos de Uso

II.1.7.3.1 Iniciar

Caso de Uso	**Iniciar**
Usuario:	**Usuario**
Descripción: En este caso de uso el usuario debe introducir su usuario y clave	
Propósito: Tener acceso al sistema.	
Flujo Principal: 1.- El usuario introduce su usuario. 2.- El usuario introduce su clave.	
Subflujo: Acceso al sistema Mensaje: Clave i/o Contraseña incorrectas.	
Flujo de Excepción: Mensaje error: Introducir Clave i/o Contraseña	

Tabla 26. E.C.U. Iniciar

II.1.7.3.2 Modulo Sistema

Caso de Uso	**Modulo Sistema**
Usuario:	**Usuario**
Descripción: En este caso de uso se administra el sistema y este modulo nos permite; restauración de backup, resguardo de backup, rol-acceso, desconeccion del sistema y salir del mismo.	
Propósito: Tener acceso al sistema.	
Flujo Principal: 1.- Restaurar backup 2.-Resguardar backup. 3.-rol-acceso. 3.1.-adicionar rol 3.2.-modificar rol 3.3.- obtener estado (rol) 4.-Desconectarse del sistema 5.Salir	
Subflujo: Mensaje: Clave i/o Contraseña incorrectas.	
Flujo de Excepción: Mensaje error: Introducir Clave i/o Contraseña	

Tabla 27. E.C.U. Modulo Sistema

II.1.7.3.3 Resguadar backup

Caso de Uso	**Resguadar backup**
Usuario:	**Usuario**

Descripción: En este caso de uso nos permite crear un resguardo de la base de datos.
Propósito: Tener seguridad de los datos.
Precondición: 1.- Estar dentro del sistema 2.- Estar logeado como director o administrativo
Flujo Principal:Resguardar datos Selecciona ubicación para crear el backup Selecciona botón generar
Flujo de Excepción: **Mensaje:** La ubicación no es valida **Mensaje:** ya existe un archivo con el mismo nombre, desea reemplazarlo? **Mensaje:** El proceso tuvo errores durante su generación Mensaje: Se realizo el backup correctamente

Tabla 28. E.C.U. Resguadar backup

II.1.7.3.4 Restaurar backup

Caso de Uso	**Restaurar backup**
Usuario:	**Usuario**
Descripción: En este caso de uso nos permite extraer un backup de alguna dirección especifica donde fue creado el backup con anterioridad.	
Propósito: Extraer datos generados de una anterior base de datos.	
Precondición: 1.- Estar dentro del sistema	

2.- Estar logeado como director o administrativo
Flujo Principal:Resguardar datos Busca la direccioonm donde esta guardado el backup Selecciona botón restaurar
Flujo de Excepción: **Mensaje:** La ubicación no es valida **Mensaje:** El archivo noescompatible. **Mensaje:** El proceso tuvo errores durante la restauracion Mensaje: Se restauro correctamente el backup

Tabla 29. E.C.U. Restaurar backup

II.1.7.3.5 Rol_acceso

Caso de Uso: Rol_acceso
Usuario: Director o administrativo
Descripción: En este caso de uso el director se encarga de la administración de los roles de personal, en este modulo nos permite adicionar nuevo rol, modificar rol, y cambiar el estado de rol de activo a pasivo o viceversa.
Propósito: Realizar un registro previo de los roles para su posterior asignación al personal administrativo. Este modulo nos permite: 1.- Ver los roles existentes 2.- Agregar un nuevo rol 3.- Modificar un rol 4.- Dar de baja un rol 5.- Dar de Alta un rol

Flujo Principal: 1.- El usuario Selecciona modulo sistema 2.-El usuario selecciona opción rol_acceso 3.- Se listan los datos de los rol ya registrados. 3.- El usuario selecciona alguna opción 3.1.-El usuario puede seleccionar adicionar rol. 3.2.-El usuario puede Seleccionar modificarrol donde se modificará algunos datos. 3.3.-El usuario puede seleccionar dar de baja a un rol. 3.4.- El usuario puede seleccionar dar de alta, en caso de que el rol este dado de baja.
Subflujo: Mensaje: Se adiciono satisfactoriamente. Mensaje: Se modifico satisfactoriamente. Mensaje:¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente)
Flujo de Excepción: Mensaje error: el rol introducido ya se encuentra registrado. (Ocurre al momento de agregar un nuevo rol, si el nombre de rol ya estaba registrado. Mensaje error: los campos marcados están vacíos o son erróneos. (se da cuando al realizar el registro existen algunos campos que están vacíos o están con datos que no corresponden a ese campo para su registro).

Tabla 30. E.C.U. Rol_acceso

II.1.7.3.6 Adicionar rol

Caso de Uso: Adicionar rol
Usuario: Director o administrativo

Descripción: Es el caso de uso donde el director puede adicionar un nuevo rol.
Propósito: Adicionar un nuevo rol.
Precondición : Estar logeada como director o administartivo, para gozar de estos privilegios.
Flujo Principal: Selecciona opción modulo sistema Selecciona opción rol_acceso Selecciona Opción Adicionar (rol) Llena los datos del nuevo rol Selecciona botón Adicionar
Flujo de Excepción: **Mensaje:** Los campos no son validos **Mensaje:** campo rol ya existe

Tabla 31. E.C.U. Adicionar rol

II.1.7.3.7 Modificar rol

Caso de Uso	**Modificar rol**
Usuario:	**Director o administrativo**
Descripción: Es el caso de uso donde el director o administrativo puede modificar algunos datos del rol dentro de la base de datos.	
Propósito: Modificar los datos del rol	
Precondición : Estar dentro del Sistema.	

Estar Logueada como director o adminitrativo
Flujo Principal: Selecciona opción modulo sistema Selecciona opción rol_acceso Selecciona Opción Modificar (rol) Llena los nuevos datos de rol Selecciona botón Modificar
Flujo de Excepción: Mensaje: Los campos no son validos. (Si alguno de los datos no es válido) **Flujo de Excepción:** Mensaje los datos ya existen

Tabla 32. E.C.U. Modificar rol

II.1.7.3.8 Obtener Estado (rol)

Caso de Uso: Obtener Estado (rol)
Usuario: director o administrativo
Descripción: dar de baja o alta a algún rol de administrativo, por algún motivo de cambio o reestructuraccion de la caja.
Propósito: Cambiar de estado al rol seleccionado.
Precondición : Estar Logueado como director o administrativo.
Flujo Principal: Selecciona opción modulo sistema Selecciona opción rol_acceso Selecciona Opción estado (rol) Seguro de cambiar de estado rol?

Selecciona botón aceptar
Subflujo: **Mensaje:** ¿Seguro de cambiar de estado? **Mensaje:** se cambio de estado satisfactoriamente

Tabla 33. E.C.U. obtener estado (rol)

II.1.7.3.9 Modulo de administración

Caso de Uso Modulo de administración
Usuario: Director o administrativo
Descripción: En este caso de uso el administrativo registra las salas, camas, especialidades, servcios y personal, permitiendo adicionar, hacer modificaciones, dar de baja o dar de alta en todas las opciones, para ello se lista todos los datos registrados en cada una de las opciones que se hayan registrado en la base de datos.
Propósito: Ingresar al modulo administración, mediante el cual se puede, la cual también podemos: 1.- administrar camas 2.- Administrar Salas 3.- Administrar especialidad 4.- Administrar Personal
Flujo Principal: 1.- El usuario Selecciona modulo administrativo 2.- Se listan las opciones administrar camas, Administrar Salas, Administrar especialidad, Administrar Personal. 3.- El usuario selecciona alguna opción 3.1.-El usuario puede seleccionar administrar camas. Donde tendrá las subopciones

Adicionar, modificar y obtener estado. 3.2.- El usuario puede seleccionar Administrar Salas. Donde tendrá las subopciones Adicionar, modificar y obtener estado. 3.3.- El usuario puede seleccionar Administrar especialidad Donde tendrá las subopciones Adicionar, modificar y obtener estado. 3.4.- El usuario puede seleccionar Administrar Personal Donde tendrá las subopciones Adicionar, modificar y obtener estado.
Subflujo: Mensaje: Se adiciono satisfactoriamente. Mensaje: Se modifico satisfactoriamente. Mensaje: (dar de baja) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente) Mensaje: (dar de alta) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente)
Flujo de Excepción: Mensaje error: Los datos introducidos ya se encuentra registrados. (Ocurre al momento de agregar nuevos datos, y estos ya se encuentran registrados en la base de datos). Mensaje error: los campos marcados están vacíos o son erróneos. (Este mensaje se da cuando al realizar el registro existen algunos campos que están vacíos o están con datos que no corresponden a ese campo para su registro).

Tabla 34. E.C.U. Modulo de administración

II.1.7.3.10 Camas

Caso de Uso: Camas
Usuario: Director o administrativo
Descripción: En este caso de uso el administrativo se encarga de administrar, este modulo permite adicionar, hacer modificaciones, dar de baja o dar de alta, para ello se

lista todos los datos registrados en la tabla camas.
Propósito: Ingresar al modulo administración seleccionar opción administrar camas, mediante el cual se podemos: 1.- adicionar camas 2.- modificar camas 3.- Obtener Estado (camas)
Flujo Principal: 1.- El usuario Selecciona modulo adminitrativo 2.- Selecciona opción camas 3.- El usuario selecciona alguna opción 3.1.-El usuario puede seleccionar adicionar camas. 3.2.- El usuario puede seleccionar modificar camas. 3.3.- El usuario puede seleccionar estado. 3.4.- El usuario puede utilizar el filtro para buscar alguna cama según el numero
Subflujo: Mensaje: Se adiciono satisfactoriamente. Mensaje: Se modifico satisfactoriamente. Mensaje: (dar de baja) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente) Mensaje: (dar de alta) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente)
Flujo de Excepción: Mensaje error: Los datos introducidos ya se encuentra registrados. (Ocurre al momento de agregar nuevos datos, y estos ya se encuentran registrados en la base de datos). Mensaje error: los campos marcados están vacíos o son erróneos. (Este mensaje se da

cuando al realizar el registro existen algunos campos que están vacíos o están con datos que no corresponden a ese campo para su registro).

Tabla 35. E.C.U. Camas

II.1.7.3.11 Adicionar camas

Caso de Uso: Adicionar camas
Usuario: Director o administrativo
Descripción: Caso de uso donde el director puede adicionar una nueva cama.
Propósito: Adicionar un nuevo cama.
Precondición : Estar logeada como director o administrativo.
Flujo Principal: 1.- El usuario Selecciona modulo administrativo 2.- Selecciona opción camas 3.-seleccionar adicionar camas. 4.-Llena los datos de la nueva cama 5.-Selecciona Adicionar
Flujo de Excepción: **Mensaje:** Los campos no son validos **Mensaje:** El numero de cama ya existe

Tabla 36. E.C.U. Adicionar camas

II.1.7.3.12 Modificar camas

Caso de Uso	**Modificar camas**
Usuario:	**Director**

Descripción: Es el caso de uso donde el director puede modificar algunos datos de la cama dentro de la base de datos.
Propósito: Modificar datos de camas
Precondición : Estar dentro del Sistema. Estar Logueada como director o administrativo.
Flujo Principal: 1.- El usuario Selecciona modulo administrativo 2.- Selecciona opción camas 3.-seleccionar modificar camas. 4.-Llena los nuevos datos de la cama 5.-Selecciona modificar
Flujo de Excepción: Los campos no son validos.

Tabla 37. E.C.U. Modificar camas

II.1.7.3.13 Obtener Estado (cama)

Caso de Uso	**Obtener Estado (cama)**
Usuario:	**Director o administrativo**
Descripción: Obtiene el estado cama, y permite dar de baja por motivo de caduco o dar de alta encaso de reemplazo de cama u otros.	
Propósito: cambiar de estado cama seleccionada.	
Precondición : Estar Logueado como director o administrativo.	
Flujo Principal:	

1.- El usuario Selecciona modulo administrativo 2.- Selecciona opción camas 3.-seleccionar estado (cama). 4.-¿Seguro de cambiar estado?
Subflujo: **Mensaje:** ¿Seguro de cambiar de estado? **Mensaje:** se cambio de estado satisfactoriamente

Tabla 38 E.C.U. Obtener Estado (cama)

II.1.7.3.14 Especialidad

Caso de Uso: Especialidad
Usuario: Director o administrativo
Descripción: En este caso de uso el administrativo se encarga de administrar la especialidad, este modulo permite adicionar, hacer modificaciones, dar de baja o dar de alta, para ello se lista todos los datos registrados en la tabla Especialidad.
Propósito: Ingresar al modulo administración seleccionar opción administrar especialidad, mediante el cual se podemos: 1.- adicionar especialidad 2.- modificar especialidad 3.- Obtener Estado (especialidad) 4.- filtrar datos según nombre de la especialidad
Flujo Principal: 1.- El usuario Selecciona modulo administrativo 2.- Selecciona opción especialidad

3.- El usuario selecciona alguna opción 3.1.-El usuario puede seleccionar adicionar especialidad. 3.2.- El usuario puede seleccionar modificar especialidad. 3.3.- El usuario puede seleccionar estado. Donde le permitira al usuario cambiar el estado de activo a pasivo o viceversa según sea el caso 3.4.- El usuario puede utilizar el filtro para buscar alguna especialidad según el numero
Subflujo: Mensaje: Se adiciono satisfactoriamente. Mensaje: Se modifico satisfactoriamente. Mensaje: (dar de baja) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente) Mensaje: (dar de alta) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente)
Flujo de Excepción: Mensaje error: Los datos introducidos ya se encuentra registrados. (Ocurre al momento de agregar nuevos datos, y estos ya se encuentran registrados en la base de datos). Mensaje error: los campos marcados están vacíos o son erróneos. (Este mensaje se da cuando al realizar el registro existen algunos campos que están vacíos o están con datos que no corresponden a ese campo para su registro).

Tabla 39 E.C.U. Especialidad

II.1.7.3.15 Adicionar especialidad

Caso de Uso: Adicionar especialidad
Usuario: Director o administrativo
Descripción: Es el caso de uso donde el director puede adicionar una nueva especialidad.

Propósito: Adicionar un nuevo especialidad.
Precondición : Estar logeada como director o administrativo, para gozar de estos privilegios.
Flujo Principal: 1.- El usuario Selecciona modulo administrativo 2.- Selecciona opción especialidad 3.-seleccionar adicionar especialidad. 4.-Llena los datos de la nueva especialidad 5.-Selecciona Adicionar especialidad
Flujo de Excepción: **Mensaje:** Los campos no son validos **Mensaje:** La especialidad ya existe

Tabla 40. E.C.U. Adicionar especialidad

II.1.7.3.16 Modificar especialidad

Caso de Uso	**Modificar especialidad**
Usuario:	**Director**
Descripción: Es el caso de uso donde el director puede modificar algunos datos de la especialidad dentro de la base de datos.	
Propósito: Modificar datos de especialidad	
Precondición : Estar Logueada como director o administrativo.	
Flujo Principal: 1.- El usuario Selecciona modulo administrativo	

2.- Selecciona opción especialidad 3.-seleccionar modificar especialidad 4.-Llena los nuevos datos de la especialidad 5.-Selecciona modificar
Flujo de Excepción: Los campos no son validos.

Tabla 41. E.C.U. Modificar especialidad

II.1.7.3.17 Obtener Estado (especialidad)

Caso de Uso	**Obtener Estado (especialidad)**
Usuario:	**Director o administrativo**
Descripción: Obtiene el estado de especialidad, y permite dar de baja por motivo de cambios, reestructuracion u otros.	
Propósito: cambiar de estado especialidad seleccionada.	
Precondición : Estar Logueado como director o administrativo.	
Flujo Principal: 1.- El usuario Selecciona modulo administrativo 2.- Selecciona opción especialidad 3.-seleccionar estado (especialidad) 4.-¿Seguro de cambiar estado?	
Subflujo: **Mensaje:** ¿Seguro de cambiar de estado? **Mensaje:** se cambio de estado satisfactoriamente	

Tabla 42. E.C.U. Obtener estado (especialidad)

II.1.7.3.18 Personal

Caso de Uso: Personal
Usuario: Director o administrativo
Descripción: En este caso de uso el administrativo se encarga de administrar Personal, este modulo permite adicionar, hacer modificaciones, dar de baja o dar de alta, para ello se lista todos los datos registrados en la tabla Personal.
Propósito: Ingresar al modulo administración seleccionar opción administrar Personal, mediante el cual se podemos: 1.- adicionar Personal 2.- modificar Personal 3.- Obtener Estado (personal) 4.- filtrar datos según nombre de la Personal
Flujo Principal: 1.- El usuario Selecciona modulo administrativo 2.- Selecciona opción Personal 3.- El usuario selecciona alguna opción 3.1.-El usuario puede seleccionar adicionar Personal 3.2.- El usuario puede seleccionar modificar Personal 3.3.- El usuario puede seleccionar estado. Donde lepermitira al usuario cambiar el estado de activo a pasivo o viceversa según sea el caso 3.4.- El usuario puede usar el filtro para buscar alguna Persona según el apellido paterno
Subflujo: Mensaje: Se adiciono satisfactoriamente. Mensaje: Se modifico satisfactoriamente.

Mensaje: (dar de baja) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente) Mensaje: (dar de alta) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente)
Flujo de Excepción: Mensaje error: Los datos introducidos ya se encuentra registrados. (Ocurre al momento de agregar nuevos datos, y estos ya se encuentran registrados en la base de datos). Mensaje error: los campos marcados están vacíos o son erróneos. (Este mensaje se da cuando al realizar el registro existen algunos campos que están vacíos o están con datos que no corresponden a ese campo para su registro).

Tabla 43. E.C.U. Personal

II.1.7.3.19 Adicionar personal

Caso de Uso: Adicionar personal
Usuario: Director o administrativo
Descripción: Caso de uso donde el director puede adicionar datos de persona
Propósito: Adicionar un nuevo personal
Precondición : Estar logeada como director o administrativo, para gozar de estos privilegios.
Flujo Principal: 1.- El usuario Selecciona modulo administrativo 2.- Selecciona opción personal 3.-seleccionar adicionar personal 4.-Llena los datos de la nueva personal 5.-Selecciona Adicionar

Flujo de Excepción: **Mensaje:** Los campos no son validos **Mensaje:** campo ci solo aceptan números **enteros** **Mensaje:** El ci de persona ya existe

Tabla 44. E.C.U. Adicionar personal

II.1.7.3.20 Modificar personal

Caso de Uso	**Modificar personal**
Usuario:	**Director o administrativos**
Descripción: Es el caso de uso donde el director puede modificar algunos datos de personal dentro de la base de datos.	
Propósito: Modificar datos de personal	
Precondición : Estar dentro del Sistema. Estar Logueada como director o administrativo.	
Flujo Principal: 1.- El usuario Selecciona modulo administrativo 2.- Selecciona opción personal 3.-seleccionar modificar personal 4.-Llena los nuevos datos de la personal 5.-Selecciona modificar	
Flujo de Excepción: Los campos no son validos.	

Tabla 45. E.C.U. Modificar personal

II.1.7.3.21 Obtener Estado (personal)

Caso de Uso	**Obtener Estado (personal)**
Usuario:	**Director o administrativo**
Descripción: Obtiene el estado de personal, permitiendo dar de baja por motivo de despido , renuncia, reincorporación u otros	
Propósito: cambiar de estado personal seleccionada.	
Precondición : Estar Logueado como director o administrativo.	
Flujo Principal: 1.- El usuario Selecciona modulo administrativo 2.- Selecciona opción personal 3.-seleccionar estado (personal) 4.-¿Seguro de cambiar estado?	
Subflujo: **Mensaje:** ¿Seguro de cambiar de estado? **Mensaje:** se cambio de estado satisfactoriamente	

Tabla 46. E.C.U. Obtener Estado (personal)

II.1.7.3.22 Salas

Caso de Uso: Salas
Usuario: Director o administrativo
Descripción: En este caso de uso el administrativo se encarga de administrar Salas, este modulo permite adicionar, hacer modificaciones, dar de baja o dar de alta, para ello se lista todos los datos registrados en la tabla Salas

Propósito: Ingresar al modulo administración seleccionar opción administrar Salas, mediante el cual se podemos: 1.- adicionar Salas 2.- modificar Salas 3.- Obtener Estado (salas) 4.- filtrar datos según numero de la Salas
Flujo Principal: 1.- El usuario Selecciona modulo adminitrativo 2.- Selecciona opción Salas 3.- El usuario selecciona alguna opción 3.1.-El usuario puede seleccionar adicionar Salas 3.2.- El usuario puede seleccionar modificar Salas 3.3.- El usuario puede seleccionar Obtener estado. Donde le permitira al usuario cambiar el estado de activo a pasivo o viceversa según sea el caso 3.4.- El usuario puede utilizar el filtro para buscar alguna Salas según el numero
Subflujo: Mensaje: Se adiciono satisfactoriamente. Mensaje: Se modifico satisfactoriamente. Mensaje: (dar de baja) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente) Mensaje: (dar de alta) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente)
Flujo de Excepción: Mensaje error: Los datos introducidos ya se encuentra registrados. (Ocurre al momento de agregar nuevos datos, y estos ya se encuentran registrados en la base de datos).

Mensaje error: los campos marcados están vacíos o son erróneos. (Este mensaje se da cuando al realizar el registro existen algunos campos que están vacíos o están con datos que no corresponden a ese campo para su registro).

Tabla 47. E.C.U. Salas

II.1.7.3.23 Adicionar salas

Caso de Uso: Adicionar salas
Usuario: Director o administrativo
Descripción: Es el caso de uso donde el director puede adicionar una nueva sala
Propósito: Adicionar un nuevo sala
Precondición : Estar logeada como director o administrativo.
Flujo Principal: 1.- El usuario Selecciona modulo administrativo 2.- Selecciona opción salas 3.-seleccionar adicionar salas 4.-Llena los datos de la nueva sala 5.-Selecciona Adicionar
Flujo de Excepción: Mensaje: Los campos no son validos **Mensaje:** La sala ya existe

Tabla 48. E.C.U. Adicionar salas

II.1.7.3.24 Modificar salas

Caso de Uso	**Modificar salas**
Usuario:	**Director o administrativo**

Descripción: Es el caso de uso donde el director puede modificar algunos datos de la salas dentro de la base de datos.
Propósito: Modificar datos de salas
Precondición : Estar dentro del Sistema. Estar Logueada como director o administrativo.
Flujo Principal: 1.- El usuario Selecciona modulo administrativo 2.- Selecciona opción salas 3.-seleccionar modificar salas 4.-Llena los nuevos datos de la sala 5.-Selecciona modificar
Flujo de Excepción: Los campos no son validos.

Tabla 49. E.C.U. Modificar salas

II.1.7.3.25 Obtener Estado (sala)

Caso de Uso	**Obtener Estado (sala)**
Usuario:	**Director o administrativo**
Descripción: Obtiene el estado de sala, permite dar de baja por motivo de caduco o dar de alta encaso de reemplazo de salas u otros.	
Propósito: cambiar de estado de sala seleccionada.	
Precondición : Estar Logueado como director o administrativo.	
Flujo Principal:	

1.- El usuario Selecciona modulo administrativo 2.- Selecciona opción salas 3.-seleccionar estado (sala) 4.-¿Seguro de cambiar estado?
Subflujo: **Mensaje:** ¿Seguro de cambiar de estado? **Mensaje:** se cambio de estado satisfactoriamente

Tabla 50. E.C.U. Obtener estado (sala)

II.1.7.3.26 Modulo Ingreso

Caso de Uso: Modulo Ingreso
Usuario: administrativo
Descripción: Caso de uso donde se registran los ingresos de los pacientes al hospital
Propósito: tener un registro de los paciente atendidos en el H. obrero
Precondición : Estar logeada como administrativo para gozar de estos privilegios.
Flujo Principal: 1.- El usuario Selecciona opción modulo ingreso 2.- introduce datos en la opción llenar_hoja de admisión (guardar) 3.- introduce datos en opción llenar_datos paciente(guardar) 4.- introduce datos en opcion llenar_consultorio(guardar) 5.- introduce datos en datos de familiar(guardar)
Subflujo:Mensaje: ingreso de paciente correctamente registrado

Tabla 51. E.C.U. Modulo Ingreso

II.1.7.3.27 Modulo Egreso

Caso de Uso: Modulo Egreso
Usuario: administrativo
Descripción: Caso de uso donde se registra el Egreso de los pacientes al hospital
Propósito: tener un registro de los paciente atendidos en el H. obrero
Precondición : Estar logeada como administrativo para gozar de estos privilegios.
Flujo Principal: 1.- El usuario Selecciona opción modulo egreso 2.- introduce datos en opción informe estadístico de egreso(guardar) 3.- introduce datos de ingreso (guardar) 4.- introduce datos de transferencias internas(guardar) 5.- introduce datos de atención quirurgica(guardar)
Subflujo:Mensaje: egreso de paciente correctamente registrado

Tabla 52. E.C.U. Modulo Egreso

II.1.7.3.28 Modulo Servicio

Caso de Uso: Modulo Servicio
Usuario: administrativo
Descripción: Caso de uso donde se registra los servicio ofrecido a un paciente del hospital
Propósito: tener un registro de los servicios brindados a un paciente atendidos en el H. obrero
Precondición : Estar logeada como administrativo para gozar de estos privilegios.

Flujo Principal: 1.- El usuario Selecciona opción modulo servicio 2.- El usuario puede seleccionar opcion cirugía 3.- El usuario puede seleccionar opcion maternidad 4.- El usuario puede seleccionar opcion medicina interna 5.- El usuario puede seleccionar opcion pediatria 6.- El usuario puede seleccionar opcion uti

Tabla 53. E.C.U. Modulo Servicio

II.1.7.3.29 Opcion cirugia

Caso de Uso: Opcion cirugia
Usuario: administrativo
Descripción: Caso de uso donde se visualiza las especialidades que tiene esta opción, con la lista de pacientes que llenaron su hoja de admisión y a quienes se la asigno una cama
Propósito: brindar datos sobre la evolución del paciente en los días de hospitalizacion en el H. obrero
Precondición : Estar logeada como administrativo para gozar de estos privilegios.
Flujo Principal: 1.- El usuario Selecciona opción modulo servicio 2.- El usuario seleccionar opcion cirugía 3.- El usuario puede seleccionar opcion detallar_ingreso_paciente 4.-El usuario puede seleccionar opción cancelar

Tabla 54. E.C.U. Opcion Cirugia

II.1.7.3.30 Opcion maternidad

Caso de Uso: Opcion maternidad
Usuario: administrativo
Descripción: Caso de uso donde se visualiza las especialidades que tiene esta opción, con la lista de pacientes que llenaron su hoja de admisión y a quienes se la asigno una cama
Propósito: brindar datos sobre la evolución del paciente en los días de hospitalizacion en el H. obrero
Precondición : Estar logeada como administrativo para gozar de estos privilegios.
Flujo Principal: 1.- El usuario Selecciona opción modulo servicio 2.- El usuario seleccionar opcion maternidad 3.- El usuario puede seleccionar opcion detallar_ingreso_paciente 4.-El usuario puede seleccionar opción cancelar

Tabla 55. E.C.U. opción maternidad

II.1.7.3.31 Opcion medicina_interna

Caso de Uso: Opcion medicina_interna
Usuario: administrativo
Descripción: Caso de uso donde se visualiza las especialidades que tiene esta opción, con la lista de pacientes que llenaron su hoja de admisión y a quienes se la asigno una cama
Propósito: brindar datos sobre la evolución del paciente en los días de hospitalizacion en el H. obrero

Precondición : Estar logeada como administrativo para gozar de estos privilegios.
Flujo Principal: 1.- El usuario Selecciona opción modulo servicio 2.- El usuario seleccionar opcion medicina_interna 3.- El usuario puede seleccionar opcion detallar_ingreso_paciente 4.-El usuario puede seleccionar opción cancelar

Tabla 56. E.C.U. Opcion medicina_interna

II.1.7.3.32 Opcion pediatria

Caso de Uso: Opcion pediatria
Usuario: administrativo
Descripción: Caso de uso donde se visualiza las especialidades que tiene esta opción, con la lista de pacientes que llenaron su hoja de admisión y a quienes se la asigno una cama
Propósito: brindar datos sobre la evolución del paciente en los días de hospitalizacion en el H. obrero
Precondición : Estar logeada como administrativo para gozar de estos privilegios.
Flujo Principal: 1.- El usuario Selecciona opción modulo servicio 2.- El usuario seleccionar opcion pediatria 3.- El usuario puede seleccionar opcion detallar_ingreso_paciente 4.-El usuario puede seleccionar opción cancelar

Tabla 57. E.C.U. Opcion pediatría

II.1.7.3.33 Opcion UTI

Caso de Uso: Opcion UTI
Usuario: administrativo
Descripción: Caso de uso donde se visualiza las especialidades que tiene esta opción, con la lista de pacientes que llenaron su hoja de admisión y a quienes se la asigno una cama
Propósito: brindar datos sobre la evolución del paciente en los días de hospitalizacion en el H. obrero
Precondición : Estar logeada como administrativo para gozar de estos privilegios.
Flujo Principal: 1.- El usuario Selecciona opción modulo servicio 2.- El usuario seleccionar opcion uti 3.- El usuario puede seleccionar opcion detallar_ingreso_paciente 4.-El usuario puede seleccionar opción cancelar

Tabla 57. E.C.U. Opcion UTI

II.1.7.3.34 Reportes

Caso de Uso	**reportes**
Usuario:	**Director**
Descripción: En este caso de uso el director se encarga de ver los reportes requeridos y definidos en las entrevistas realizadas en el inicio del desarrollo del proyecto. En este caso el usuario selecciona la opción ver el reporte de la lista ya definida.	

Propósito: Ver lo reportes requeridos en el momento que sea necesario para una buena toma de decisiones, tomando en cuenta los datos actualizados del Hospital Obrero 1.- ver reportes.
Flujo Principal: 1.- El usuario Selecciona opción reportes 2.- El usuario puede seleccionar la opción ver_hospitalización 3.- El usuario puede seleccionar la opción ver_intervencion quirurgica por especialidad 4.- El usuario puede seleccionar la opción ver_intervencion quirurgica por clase de cirugia 5.- El usuario puede seleccionar la opción ver_mensual hospitalario 6.- El usuario puede seleccionar la veropción mensual por especialidad 7.- El usuario puede seleccionar la opción partos_nacimientos 8.- El usuario puede seleccionar la opción cirugias programadas 9.- El usuario puede seleccionar la opción cirugias de emergencia 10.- El usuario puede seleccionar la opción partos atendiso por obstetras

Tabla 58. E.C.U. Reportes

II.1.7.3.35 Hospitalización

Caso de Uso	**Hospitalizacion**
Usuario:	**Director**
Descripción: En este caso de uso el director selecciona ver los reportes de la hopsitalizacion de pacientes. Donde visualiza las especialidades y servicios brindados y la cantidad de pacientes atendidos	

Propósito: Ver lo reportes requeridos en el momento que sea necesario para una buena toma de decisiones, tomando en cuenta los datos actualizados del Hospital Obrero 1.- ver reporte Hospitalizacion
Flujo Principal: 1.- El usuario Selecciona opción modulo reportes 2.- El usuario selección Hospitalizacion 3.- el usuario puede seleccionar la opción imprimir 4.- El usuario puede seleccionar la opción cancelar

Tabla 59. E.C.U. Hospitalizacion

II.1.7.3.36 Intervencion quirurgica por especialidad

Caso de Uso	**Intervencion quirurgica por especialidad**
Usuario:	**Director**
Descripción: En este caso de uso el director selecciona ver los reportes de Intervencion quirurgica por especialidad. Donde visualiza las especialidades y servicios brindados y la cantidad de pacientes atendidos	
Propósito: Ver lo reportes requeridos en el momento que sea necesario para una buena toma de decisiones, tomando en cuenta los datos actualizados del Hospital Obrero 1.- ver reporte Intervencion quirurgica por especialidad	

Flujo Principal: 1.- El usuario Selecciona opción modulo reportes 2.- El usuario selección Intervencion quirurgica por especialidad 3.- el usuario puede seleccionar la opción imprimir 4.- El usuario puede seleccionar la opción cancelar

Tabla 60. E.C.U. Intervencion quirurgica por especialidad

II.1.7.3.37 Intervencion quirurgica por clase de cirugia

Caso de Uso	**Intervencion quirurgica por clase de cirugia**
Usuario:	**Director**
Descripción: En este caso de uso el director selecciona ver los reportes de Intervencion quirurgica por clase de cirugia. Donde visualiza las especialidades y servicios brindados y la cantidad de pacientes atendidos	
Propósito: Ver lo reportes requeridos en el momento que sea necesario para una buena toma de decisiones, tomando en cuenta los datos actualizados del Hospital Obrero 1.- ver reporte Intervencion quirurgica por clase de cirugia	
Flujo Principal: 1.- El usuario Selecciona opción modulo reportes 2.- El usuario selección Intervencion quirurgica por clase de cirugia 3.- el usuario puede seleccionar la opción imprimir 4.- El usuario puede seleccionar la opción cancelar	

Tabla 61. E.C.U. Intervencion quirurgica por clase de cirugía

II.1.7.3.38 Mensual hospitalario

Caso de Uso	**Mensual hospitalario**
Usuario:	**Director**
Descripción: En este caso de uso el director selecciona ver los reportes Mensual hospitalario. Donde visualiza las especialidades y servicios brindados y la cantidad de pacientes atendidos	
Propósito: Ver lo reportes requeridos en el momento que sea necesario para una buena toma de decisiones, tomando en cuenta los datos actualizados del Hospital Obrero 1.- ver reporte Mensual hospitalario	
Flujo Principal: 1.- El usuario Selecciona opción modulo reportes 2.- El usuario selecciona reporte Mensual hospitalario 3.- el usuario puede seleccionar la opción imprimir 4.- El usuario puede seleccionar la opción cancelar	

Tabla 62. E.C.U. Mensual hospitalario

II.1.7.3.39 Mensual por especialidad

Caso de Uso	**Mensual por especialidad**
Usuario:	**Director**
Descripción: En este caso de uso el director selecciona ver los reportes Mensual por especialidad.	

Donde visualiza las especialidades y servicios brindados y la cantidad de pacientes atendidos
Propósito: Ver lo reportes requeridos en el momento que sea necesario para una buena toma de decisiones, tomando en cuenta los datos actualizados del Hospital Obrero 1.- ver reporte Mensual por especialidad
Flujo Principal: 1.- El usuario Selecciona opción modulo reportes 2.- El usuario selecciona reporte Mensual por especialidad 3.- el usuario puede seleccionar la opción imprimir 4.- El usuario puede seleccionar la opción cancelar

Tabla 63. E.C.U. Mensual por especialidad

II.1.7.3.40 Partos_nacimientos

Caso de Uso	**Partos_nacimientos**
Usuario:	**Director**
Descripción: En este caso de uso el director selecciona ver los reportes de los Partos_nacimientos. Donde visualiza las especialidades y servicios brindados y la cantidad de pacientes atendidos	
Propósito: Ver lo reportes requeridos en el momento que sea necesario para una buena toma de decisiones, tomando en cuenta los datos actualizados del Hospital Obrero 1.- ver reporte Partos_nacimientos	

Flujo Principal: 1.- El usuario Selecciona opción modulo reportes 2.- El usuario selecciona reporte Partos_nacimientos 3.- el usuario puede seleccionar la opción imprimir 4.- El usuario puede seleccionar la opción cancelar

Tabla 64. E.C.U. Partos_nacimientos

II.1.7.3.41 Cirugias programadas

Caso de Uso	**Cirugias programadas**
Usuario:	**Director**
Descripción: En este caso de uso el director selecciona ver los reportes de los Cirugias programadas. Donde visualiza las especialidades y servicios brindados y la cantidad de pacientes atendidos	
Propósito: Ver lo reportes requeridos en el momento que sea necesario para una buena toma de decisiones, tomando en cuenta los datos actualizados del Hospital Obrero 1.- ver reporte Cirugias programadas	
Flujo Principal: 1.- El usuario Selecciona opción modulo reportes 2.- El usuario selecciona reporte Cirugias programadas 3.- el usuario puede seleccionar la opción imprimir 4.- El usuario puede seleccionar la opción cancelar	

Tabla 65. E.C.U. Cirugias programadas

II.1.7.3.42 Cirugias de emergencia

Caso de Uso	**Cirugias de emergencia**
Usuario:	**Director**
Descripción: En este caso de uso el director selecciona ver los reportes de los Cirugias de emergencia. Donde visualiza las especialidades y servicios brindados y la cantidad de pacientes atendidos	
Propósito: Ver lo reportes requeridos en el momento que sea necesario para una buena toma de decisiones, tomando en cuenta los datos actualizados del Hospital Obrero 1.- ver reporte Cirugias de emergencia	
Flujo Principal: 1.- El usuario Selecciona opción modulo reportes 2.- El usuario selecciona reporte Cirugias de emergencia 3.- el usuario puede seleccionar la opción imprimir 4.- El usuario puede seleccionar la opción cancelar	

Tabla 66. E.C.U. Cirugias de emergencia

II.1.7.3.43 Partos atendidos por obstetras

Caso de Uso	**Partos atendidos por obstetras**
Usuario:	**Director**
Descripción: En este caso de uso el director selecciona ver los reportes de los Partos atendidos por	

obstetras Donde visualiza las especialidades y servicios brindados y la cantidad de pacientes atendidos
Propósito: Ver lo reportes requeridos en el momento que sea necesario para una buena toma de decisiones, tomando en cuenta los datos actualizados del Hospital Obrero 1.- ver reporte Partos atendidos por obstetras
Flujo Principal: 1.- El usuario Selecciona opción modulo reportes 2.- El usuario selecciona reporte Partos atendidos por obstetras 3.- el usuario puede seleccionar la opción imprimir 4.- El usuario puede seleccionar la opción cancelar

Tabla 67. E.C.U. Partos atendidos por obstetras

II.1.7.3.44 Modulo ayuda

Caso de Uso	**ayuda**
Usuario:	**Todos los usuarios registrados**
Descripción: En este caso de uso todos los usuarios tienen acceso ya que la ayuda será el manual de usuario que brindara ayuda en cualquier momento que sea requerido.	
Propósito: ayudar al usuario en el momento requerido, brindando el manual.	
Flujo Principal: 1.- El usuario Selecciona opción modulo ayuda 2.- selecciona opción ayuda.	

Tabla 68. E.C.U. Modulo ayuda

II.1.8 MODELO DE ANALISIS Y DISEÑO

II.1.8.1 Modelado de Diagramas de actividades

II.1.8.1.1 Introducción

El diagrama de actividad es un artefacto de la disciplina Requisitos en la metodología RUP la cual estamos implementando.

Los diagramas de actividad se utilizan para modelar los aspectos dinámicos de un sistema, esto implica modelar los pasos secuenciales de un proceso.

II.1.8.1.1.1 Propósito

- Comprender la estructura y la dinámica del sistema deseado para la organización
- Identificar posibles mejoras

II.1.8.1.1.2 Alcance

- Describir los procesos de sistema y los usuarios
- Identificar y definir los procesos de los casos de uso según los objetivos de la organización
- Definir un diagrama de actividad para cada caso de uso.

II.1.8.1.2 Diagramas de Actividad

II.1.8.1.2.1 iniciar

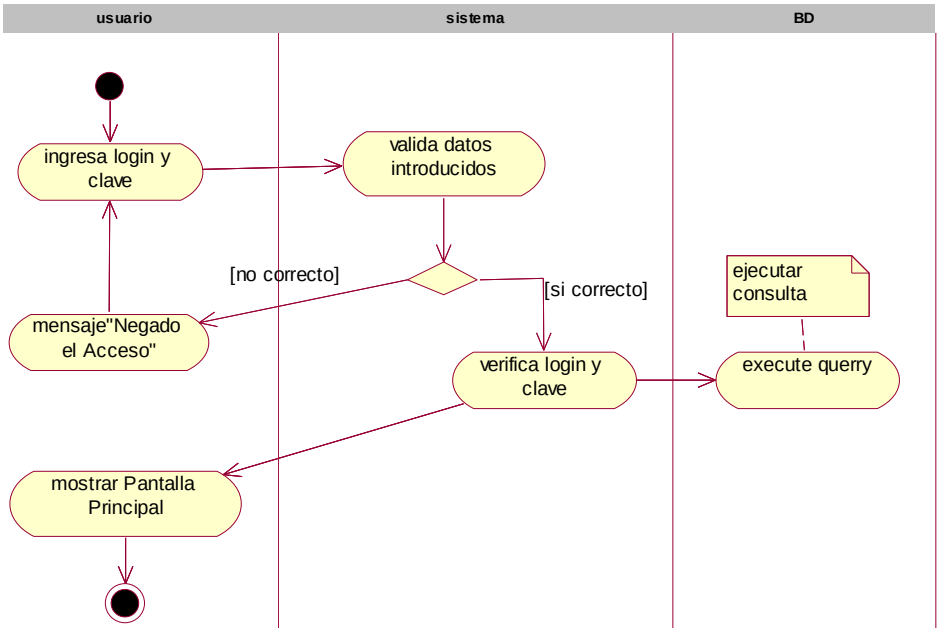


Figura 45. D.A. iniciar

II.1.8.1.2.2 Modulo Sistema

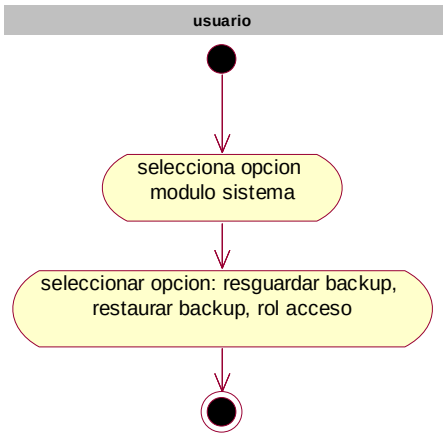


Figura 46. D.A. Modulo Sistema

II.1.8.1.2.3 Resguadar backup

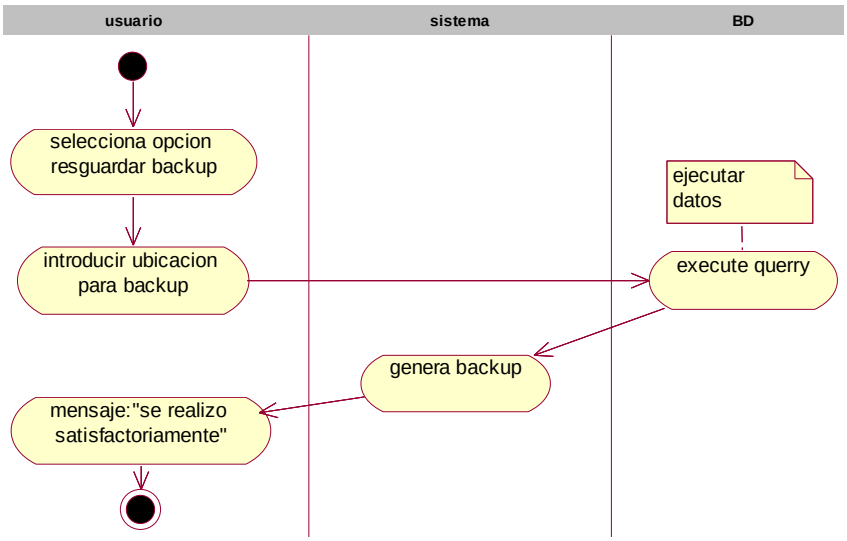


Figura 47. D.A. Resguardar backup

II.1.8.1.2.4 Restaurar backup

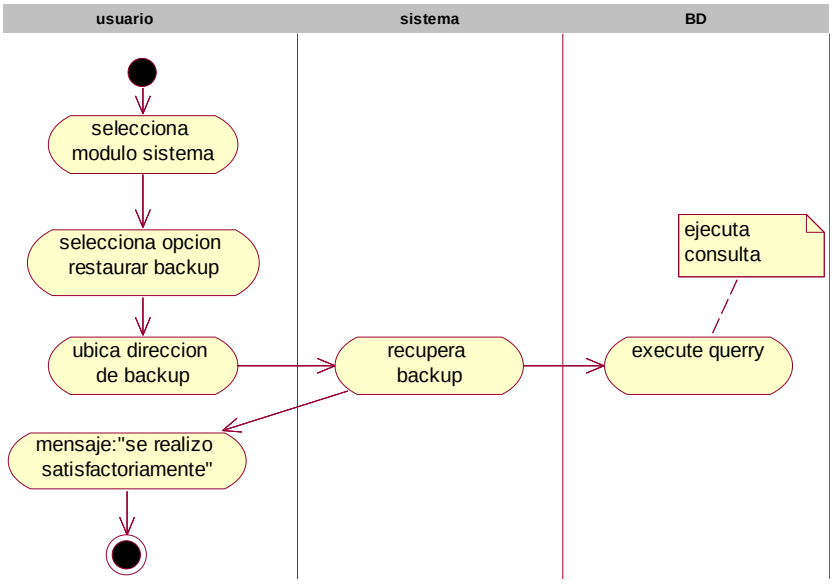


Figura 48. D.A. Restaurar Backup

II.1.8.1.2.5 Rol_acceso

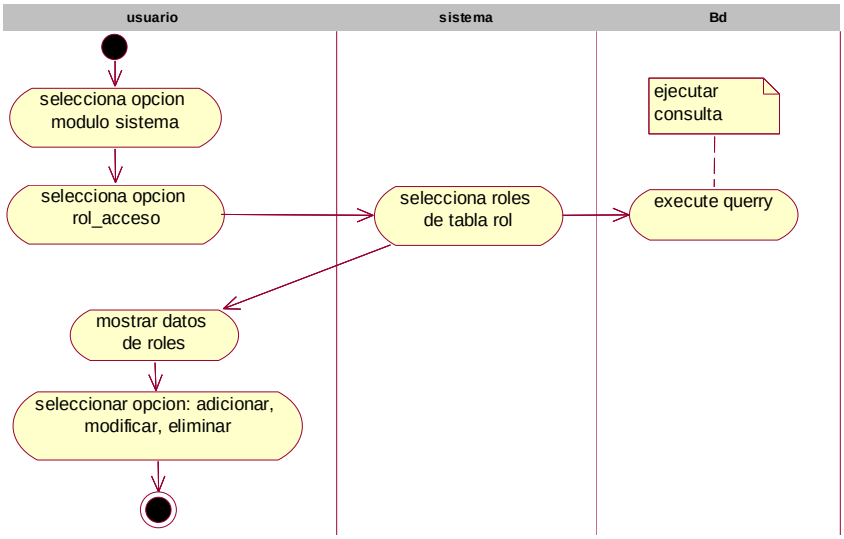


Figura 49. D.A. Rol_acceso

II.1.8.1.2.6 Adicionar rol

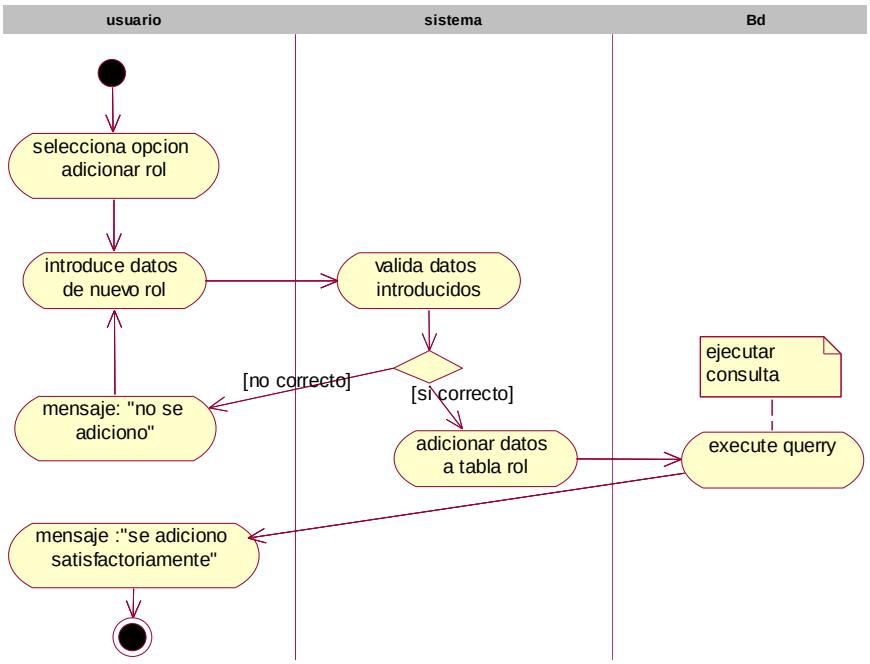


Figura 50. D.A. adicionar rol

II.1.8.1.2.7 Modificar rol

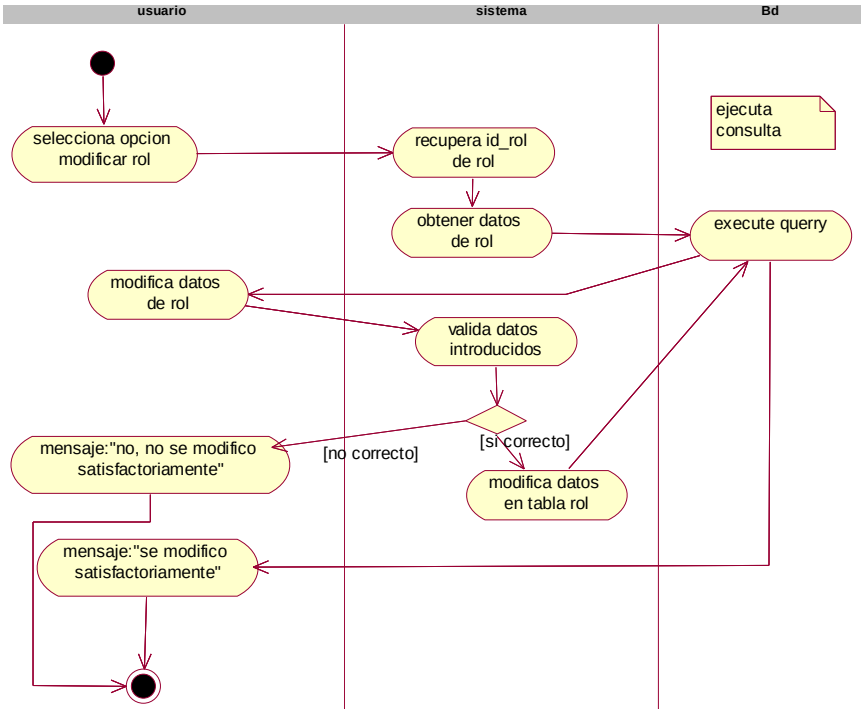


Figura 51. D.A. modificar rol

II.1.8.1.28. Obtener Estado (rol)

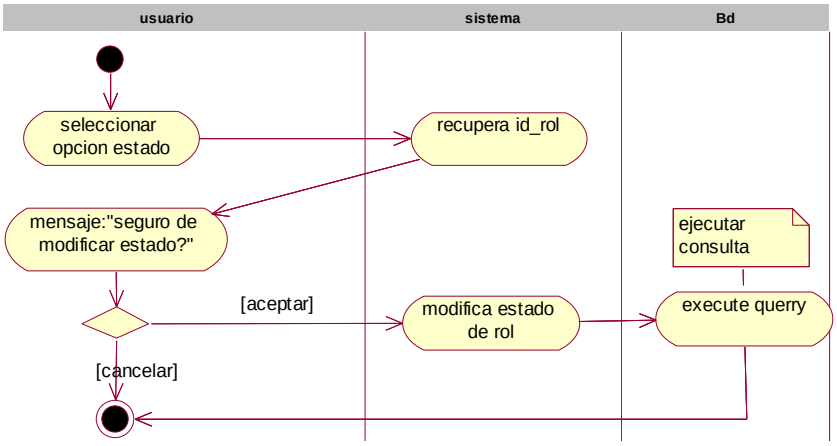


Figura 52. D.A. Obtener estado (rol)

II.1.8.1.2.9 Modulo de administración

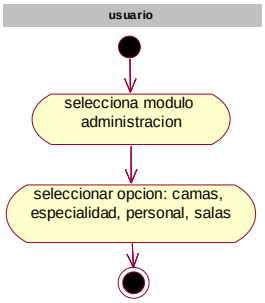


Figura 53. D.A. modulo de administracion

II.1.8.1.2.10 Camas

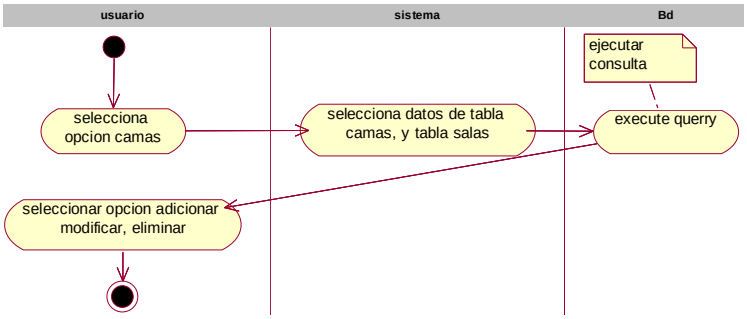


Figura 54. D.A. camas

II.1.8.1.2.11 Adicionar camas

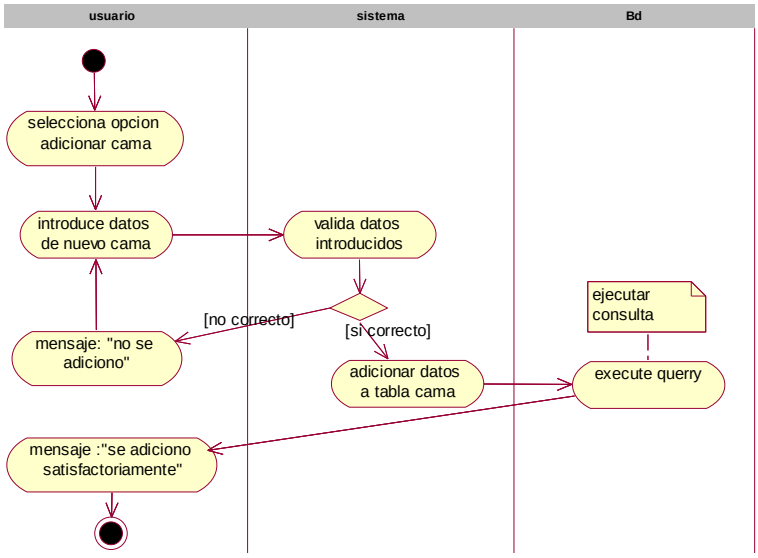


Figura 55. D.A. adicionar camas

II.1.8.1.2.12 Modificar camas

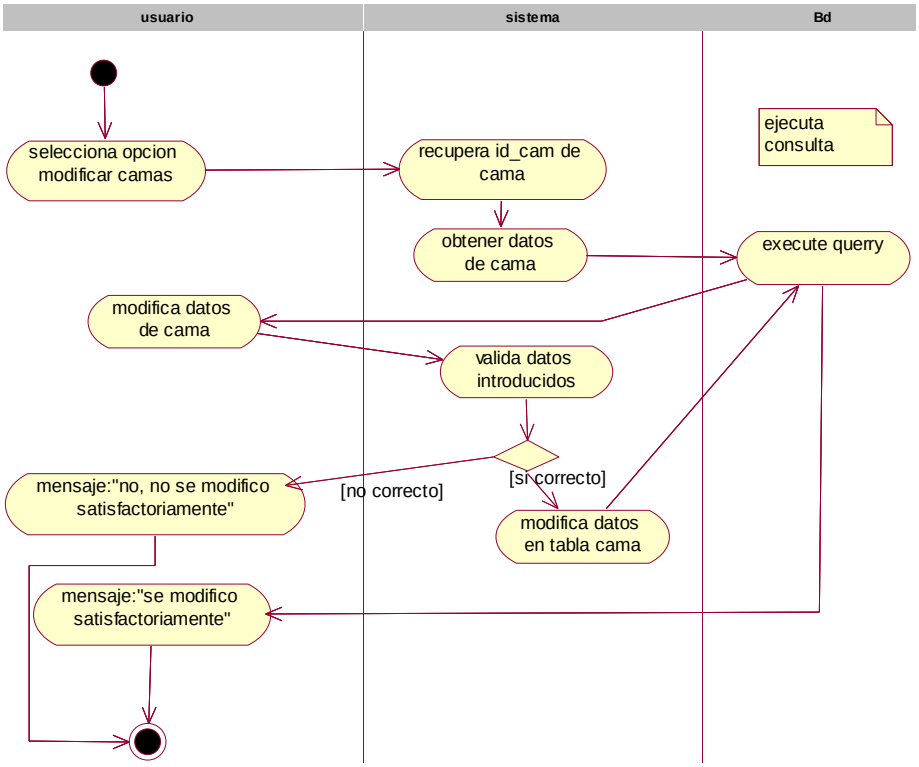


Figura 56. D.A. modificar camas

II.1.8.1.2.13 Obtener Estado (cama)

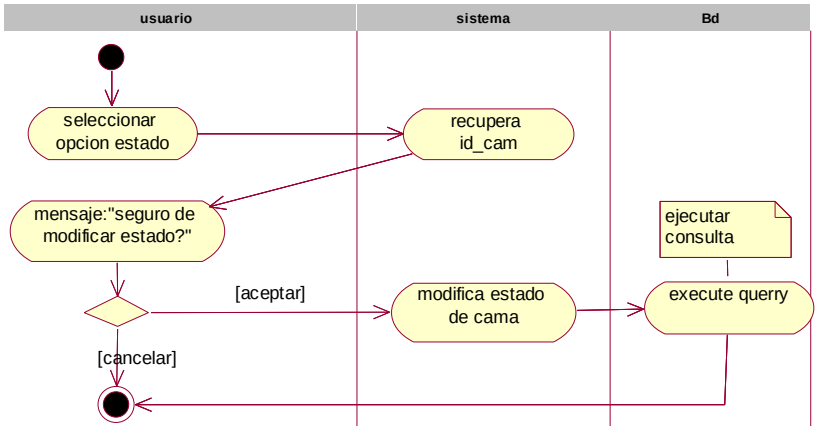


Figura 57. D.A. Obtener estado (cama)

II.1.8.1.2.14 Especialidad

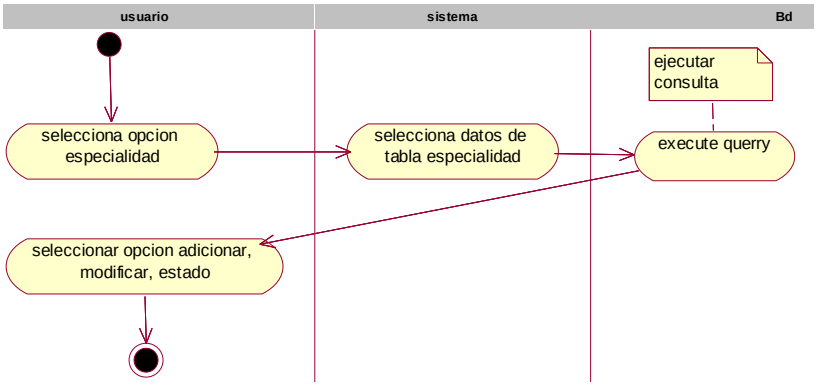


Figura 58. D.A. especialidad

II.1.8.1.2.15 Adicionar especialidad

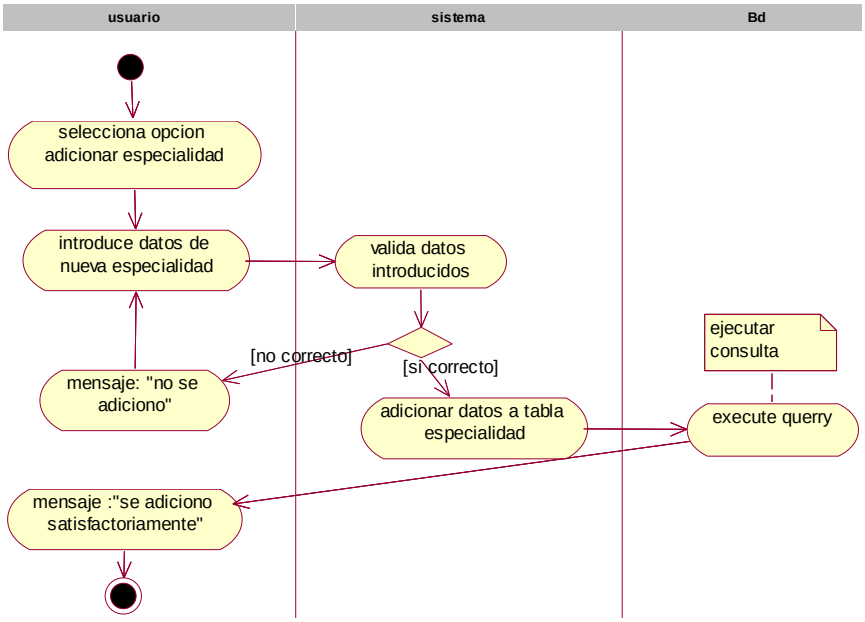


Figura 59. D.A. adicionar especialidad

II.1.8.1.2.16 Modificar especialidad

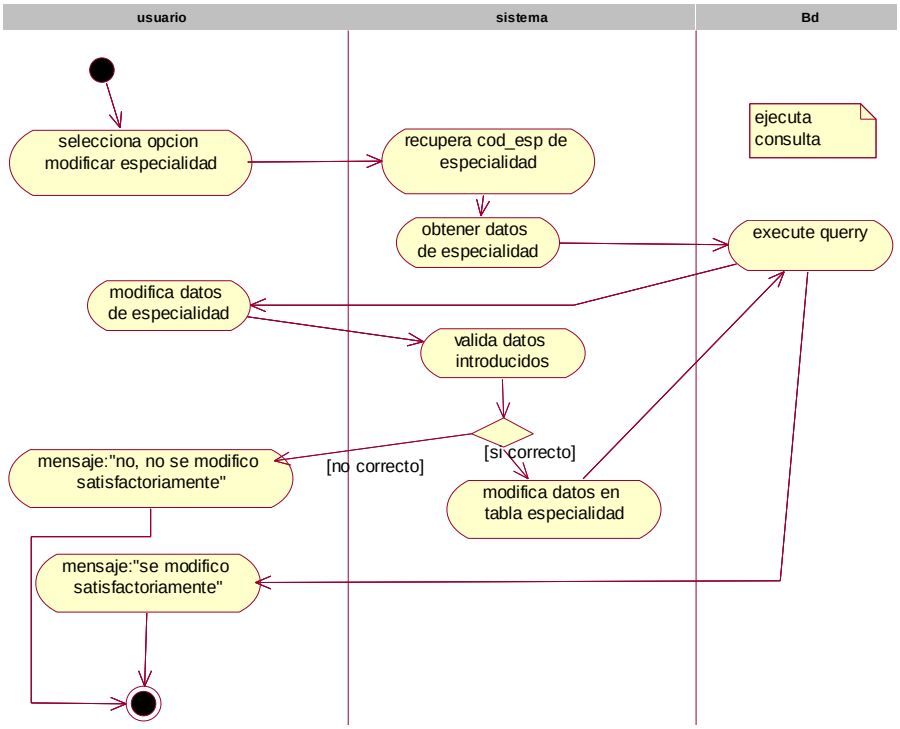


Figura 60. D.A. modificar especialidad

II.1.8.1.2.17 Obtener Estado (especialidad)

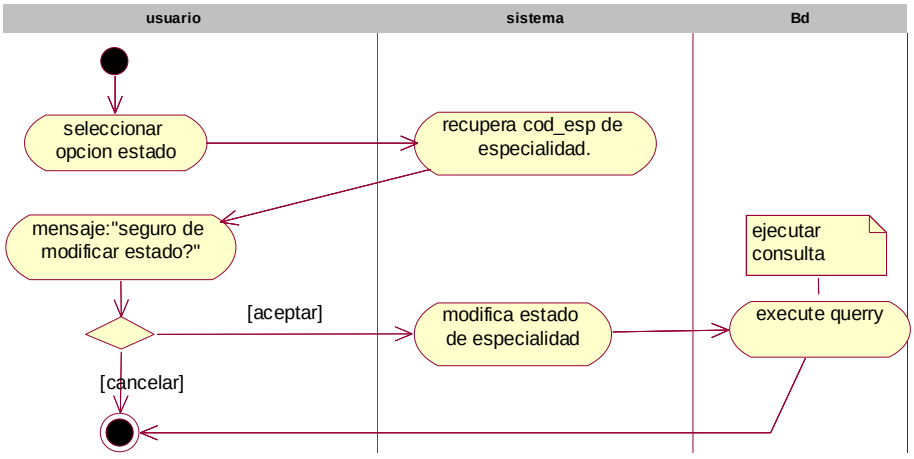


Figura 61. D.A. Obtener estado (especialidad)

II.1.8.1.2.18 Personal

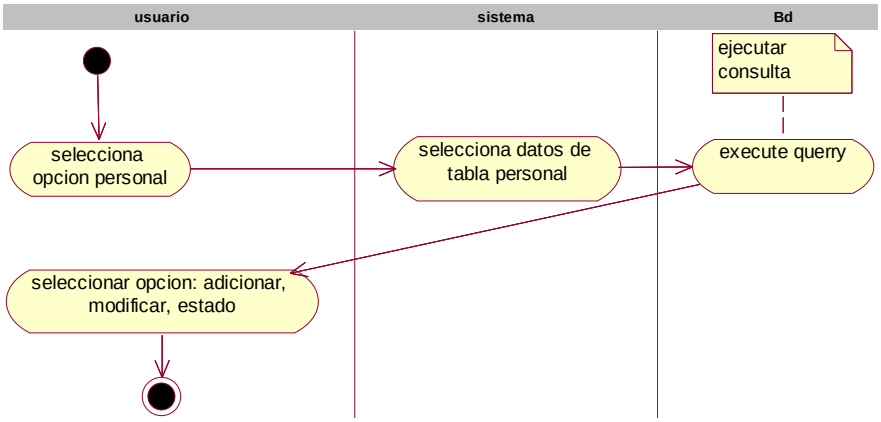


Figura 62. D.A. personal

II.1.8.1.2.19 Adicionar personal

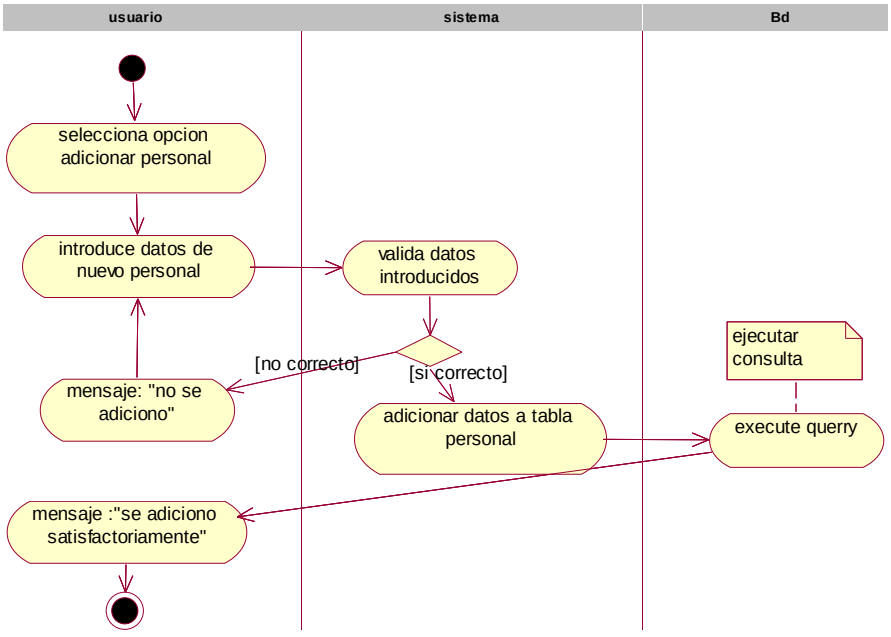


Figura 63. D.A. adicionar personal

II.1.8.1.2.20 Modificar personal

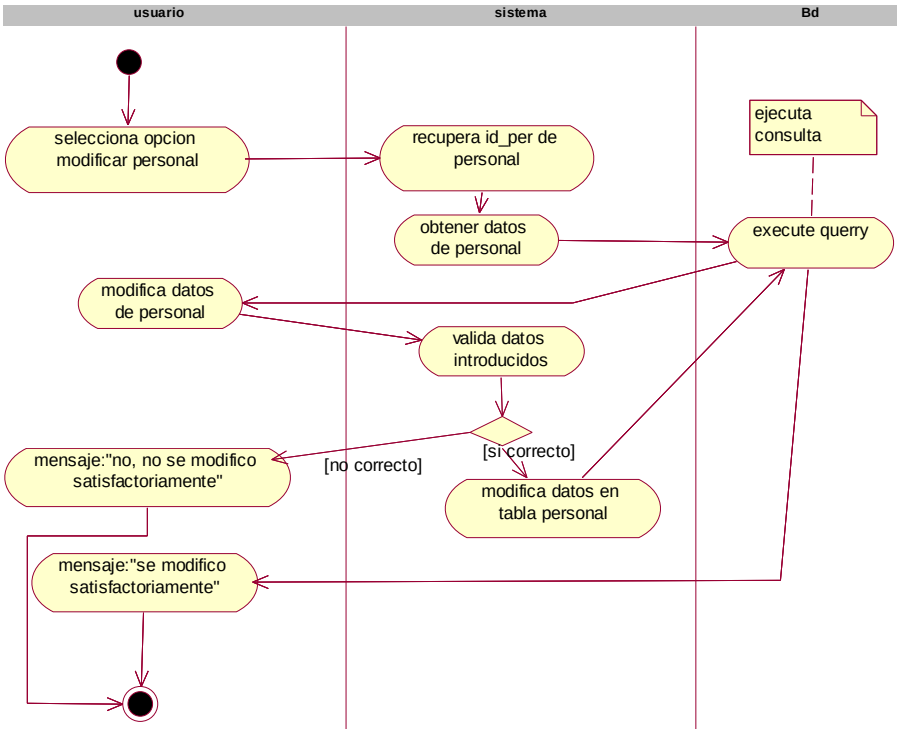


Figura 64. D.A. modificar personal

II.1.8.1.2.21 Obtener Estado (personal)

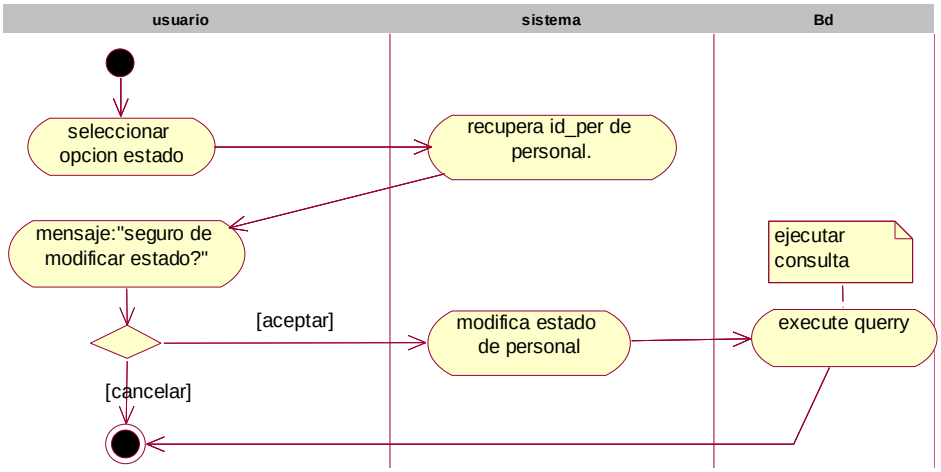


Figura 65. D.A. Obtener estado (personal)

II.1.8.1.2.22 Salas

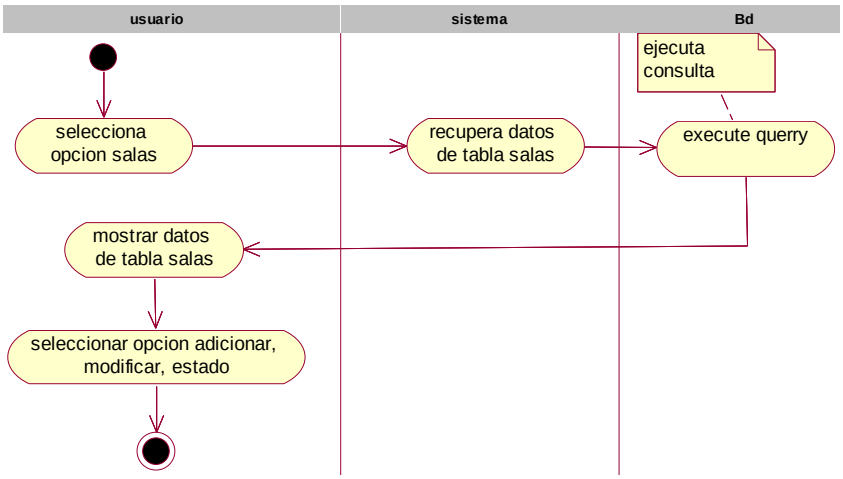


Figura 66. D.A. salas

II.1.8.1.2.23 Adicionar salas

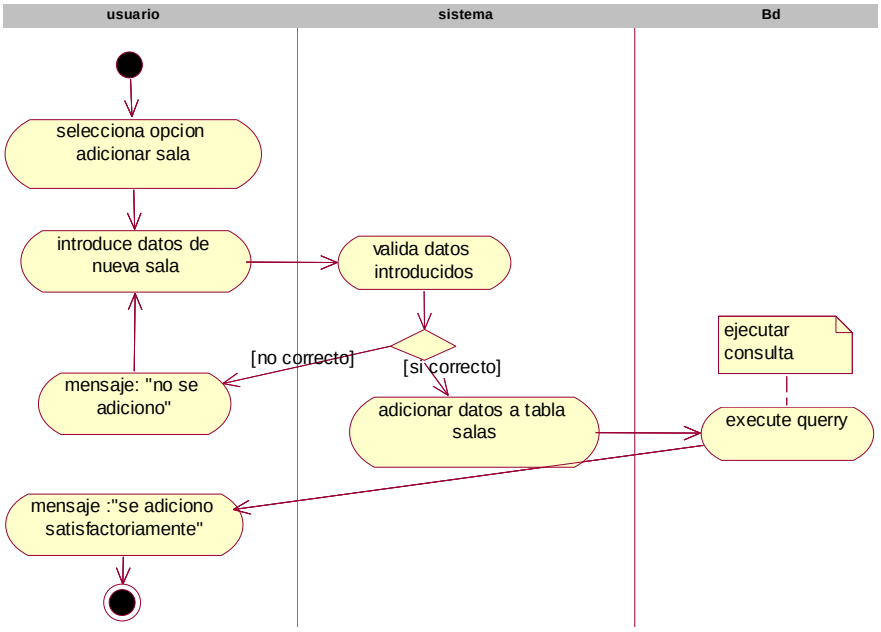


Figura 67. D.A. adicionar salas

II.1.8.1.2.24 Modificar salas

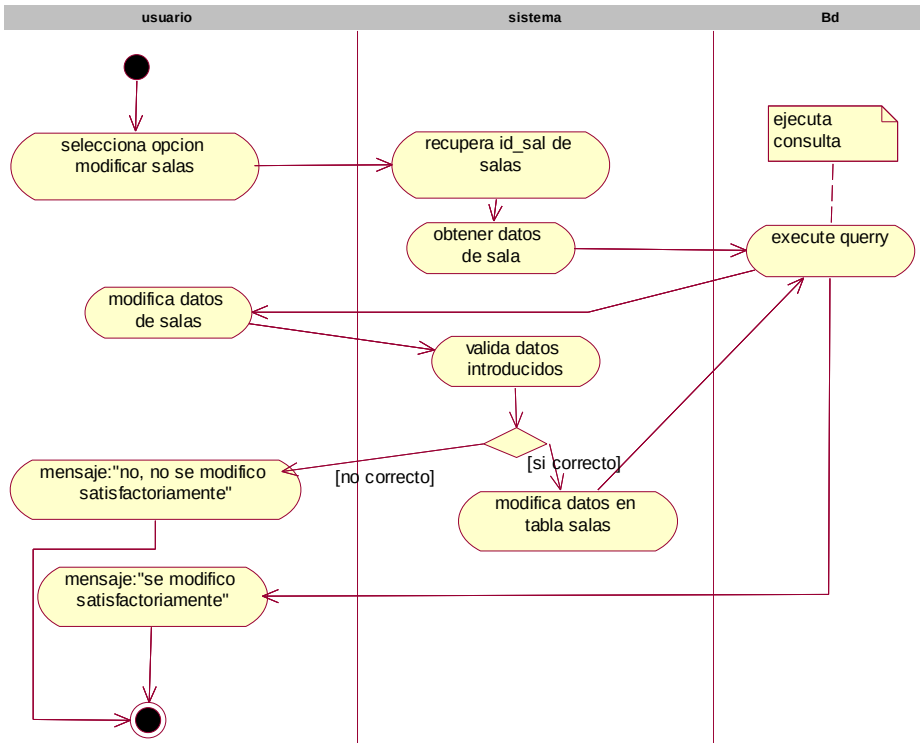


Figura 68. D.A. modificar salas

II.1.8.1.2.25 Obtener Estado (sala)

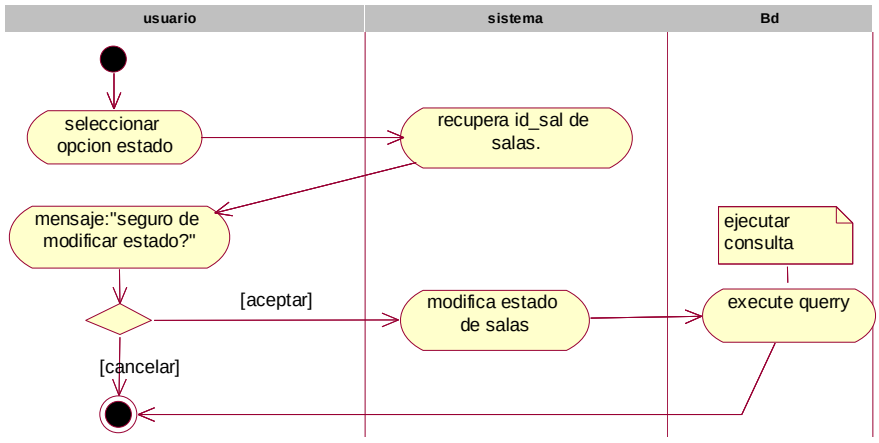


Figura 69. D.A. Obtener estado (sala)

II.1.8.1.2.26 Modulo Ingreso

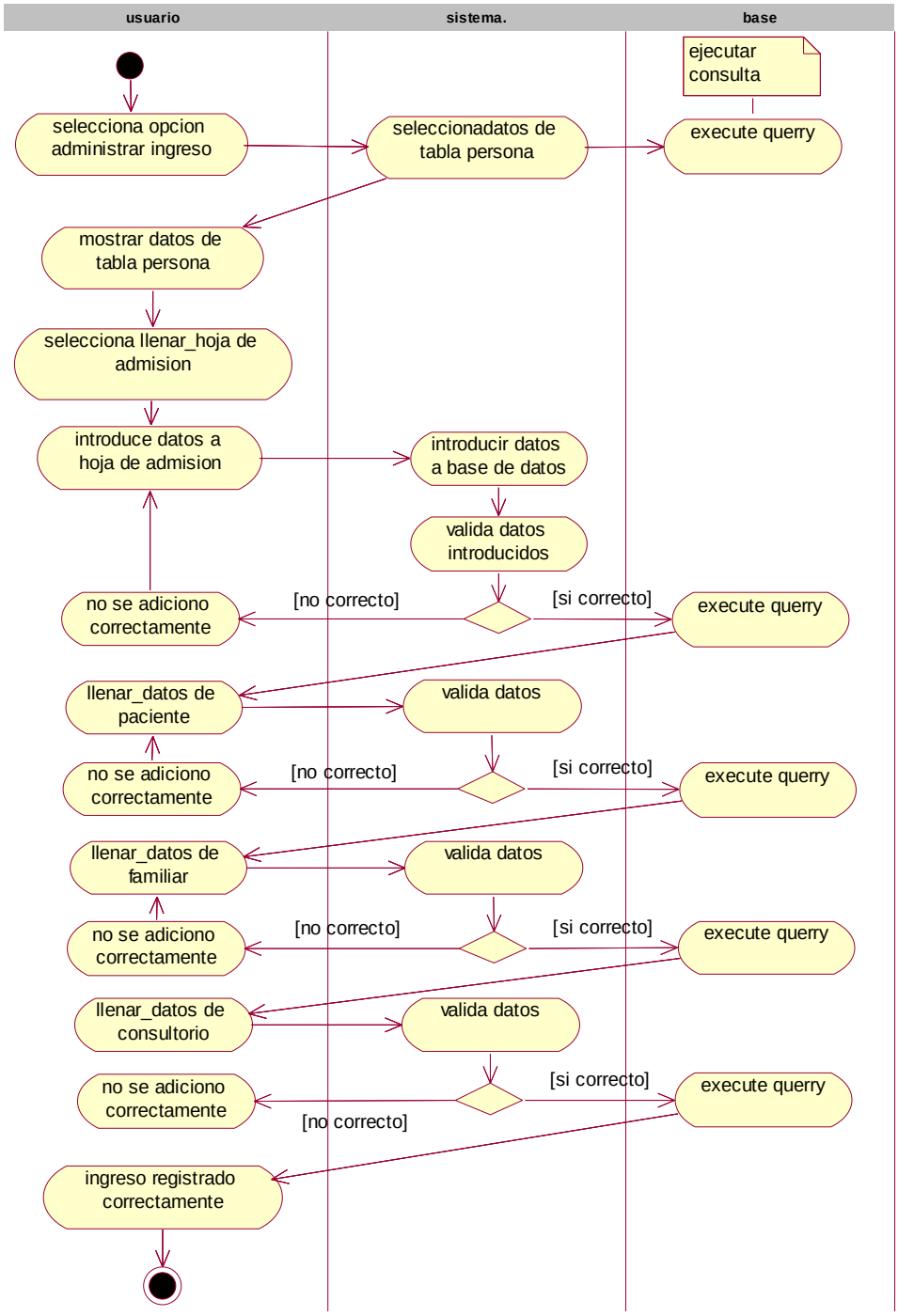


Figura 70. D.A. modulo ingreso

II.1.8.1.2.27 Modulo Egreso

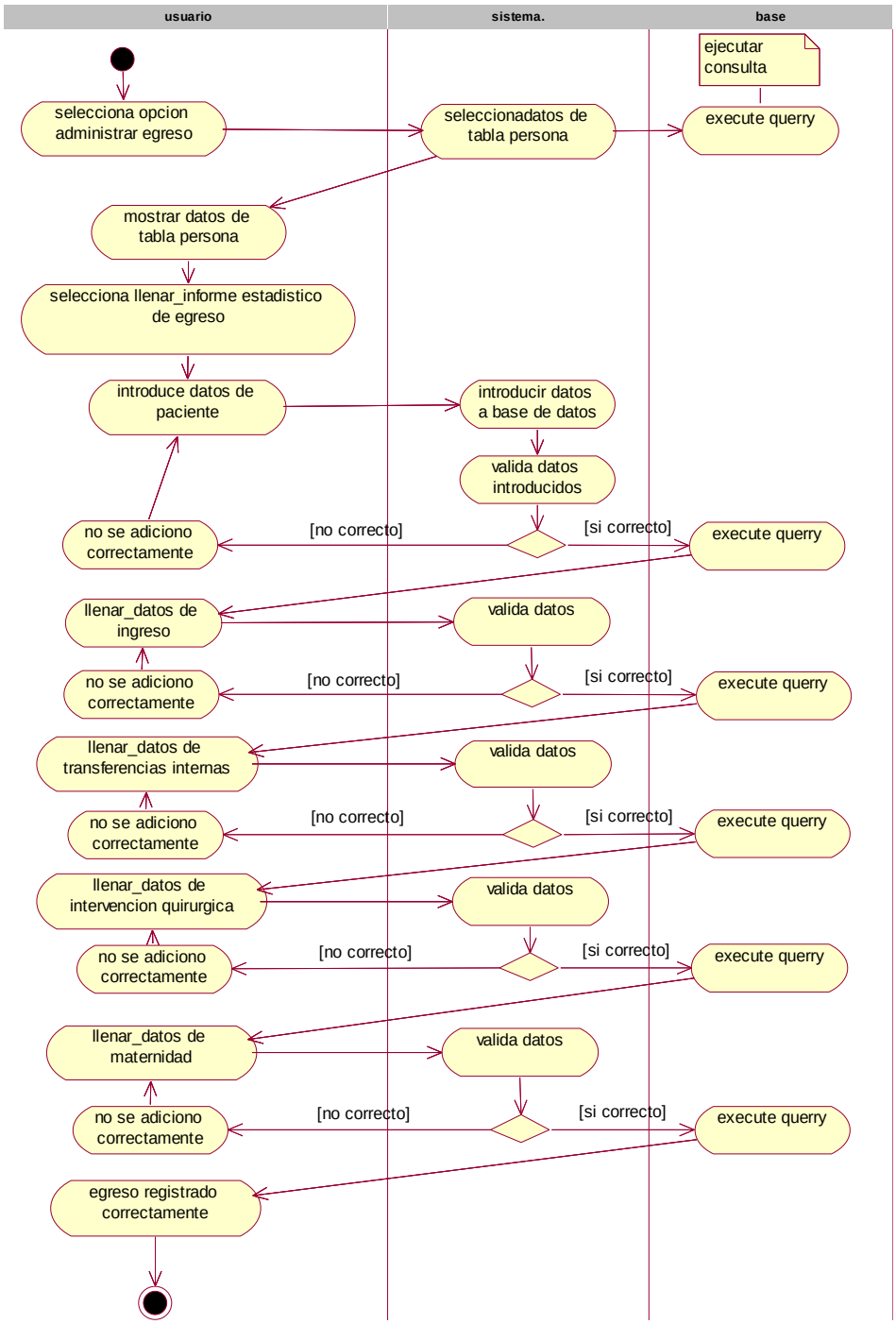


Figura 71. D.A. modulo egreso

II.1.8.1.2.28 Modulo Servicio

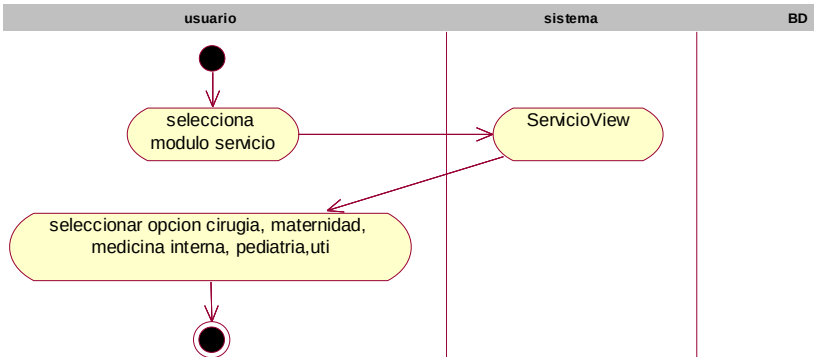


Figura 72. D.A. modulo servicio

II.1.8.1.2.30 Cirugía

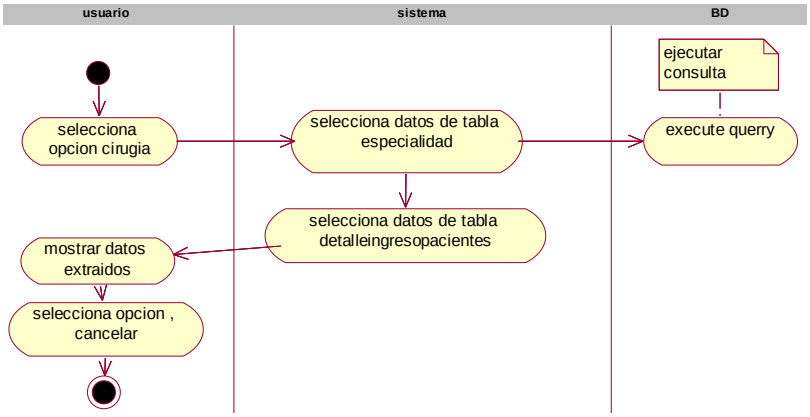


Figura 73. D.A. Cirugía

II.1.8.1.2.31 Maternidad

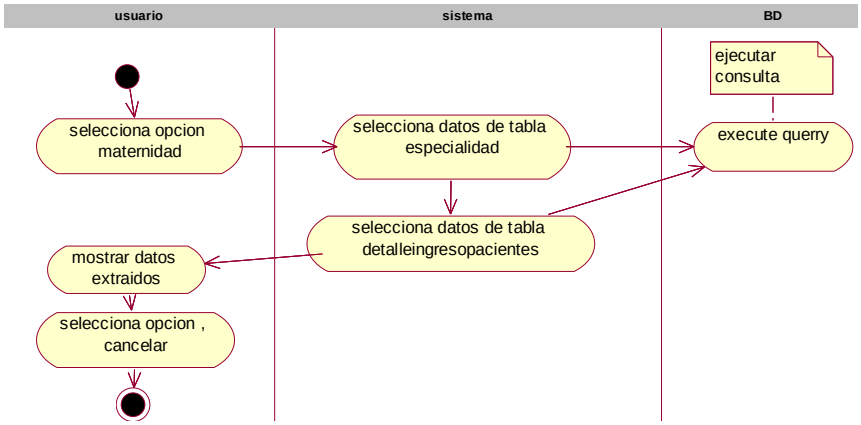


Figura 74. D.A. Maternidad

II.1.8.1.2.32 Medicina interna

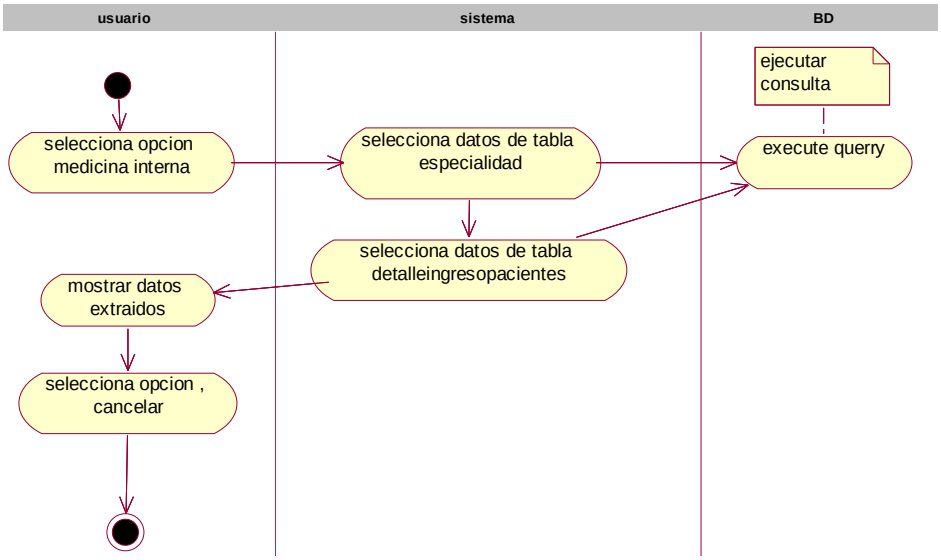


Figura 75. D.A. Medicina interna

II.1.8.1.2.33 Pediatria

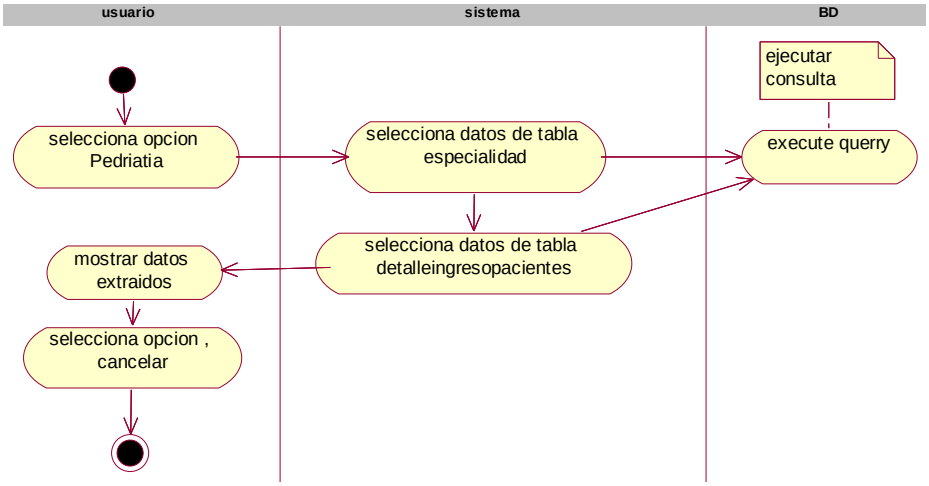


Figura 76. D.A. Pediatria

II.1.8.1.2.34 UTI

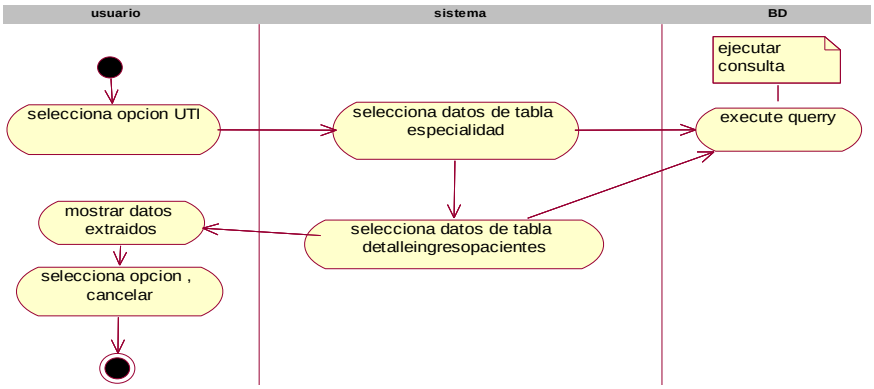


Figura 77. D.A. UTI

II.1.8.1.2.35 Reportes

II.1.8.1.2.36 Ver Hospitalizacion

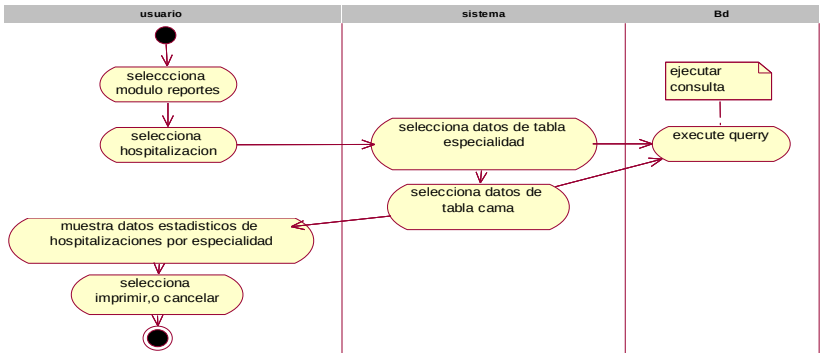


Figura 79. D.A. Ver Hospitalizacion

II.1.8.1.2.37 VerIntervencion quirúrgica por especialidad

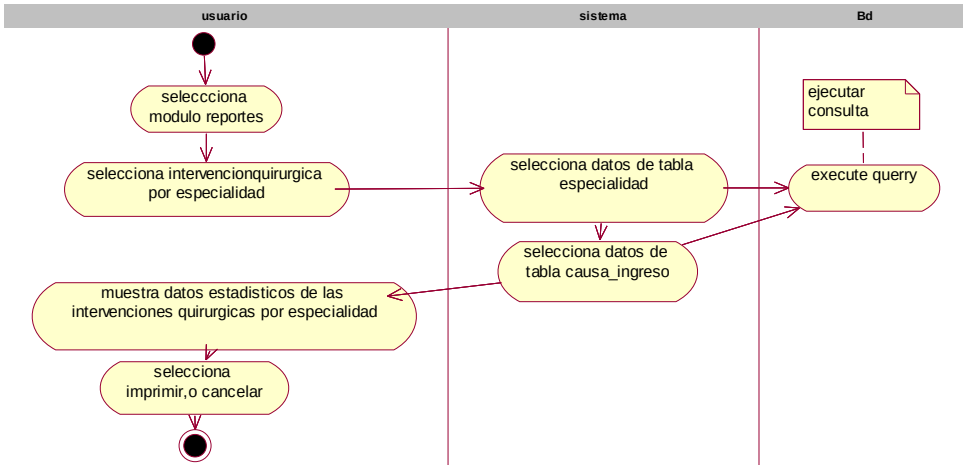


Figura 80. D.A. Ver Intervencion quirúrgica por especialidad

II.1.8.1.2.38 Ver Intervencion quirúrgica por clase de cirugía

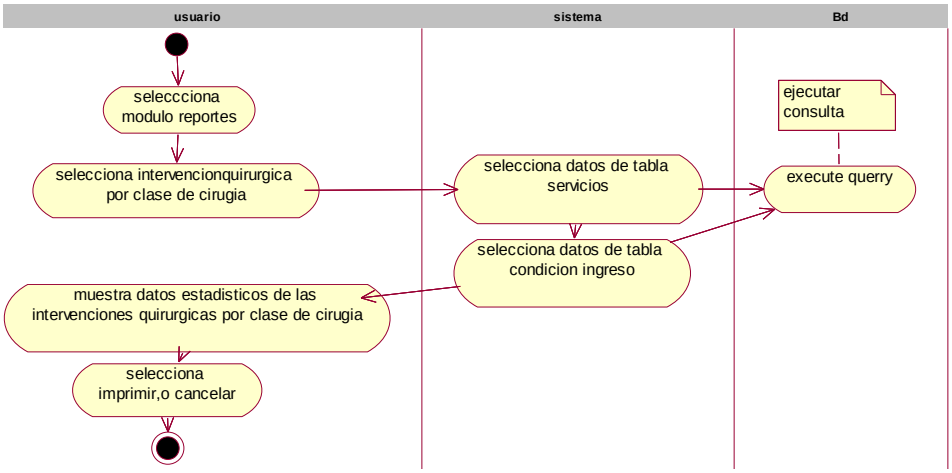


Figura 81. D.A. Ver Intervencion quirúrgica por clase de cirugia

II.1.8.1.2.39 Ver Mensual hospitalario

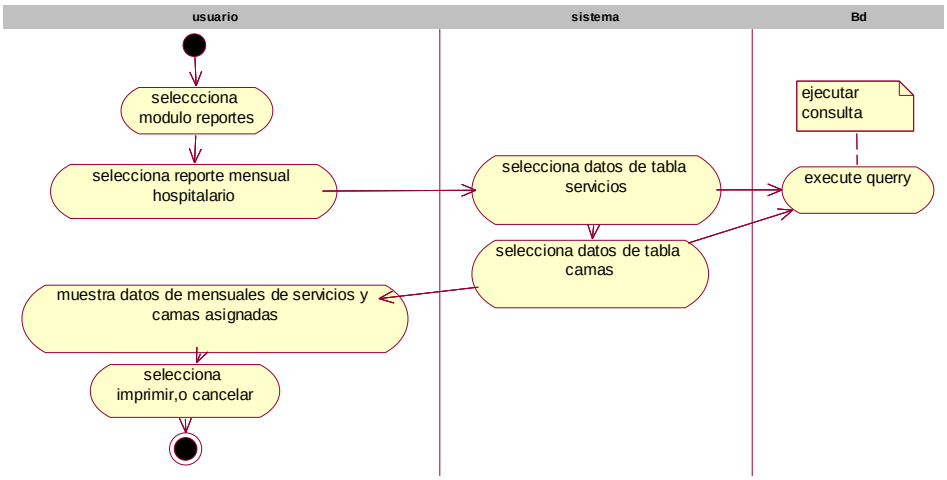


Figura 82. D.A. Ver Mensual hospitalario

II.1.8.1.2.40 Ver Mensual hospitalario por especialidad

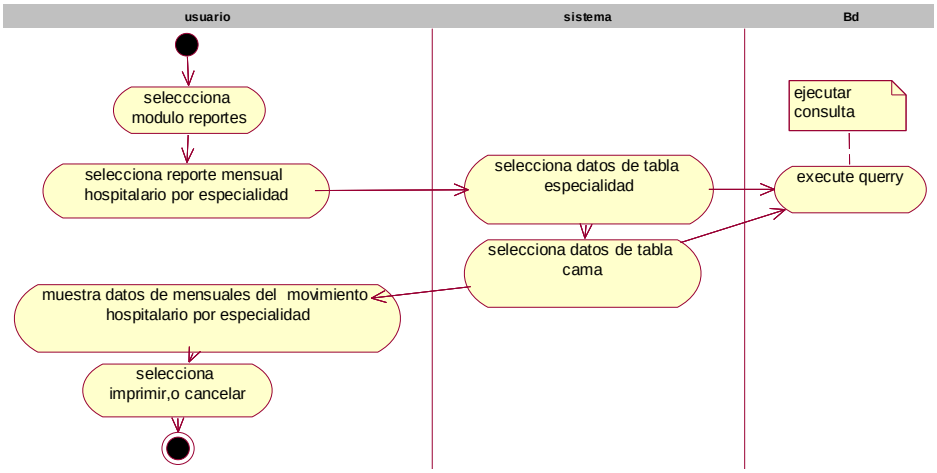


Figura 83. D.A. Ver Hospitalizacion por especialidad

II.1.8.1.2.41 Ver Partos y nacimientos

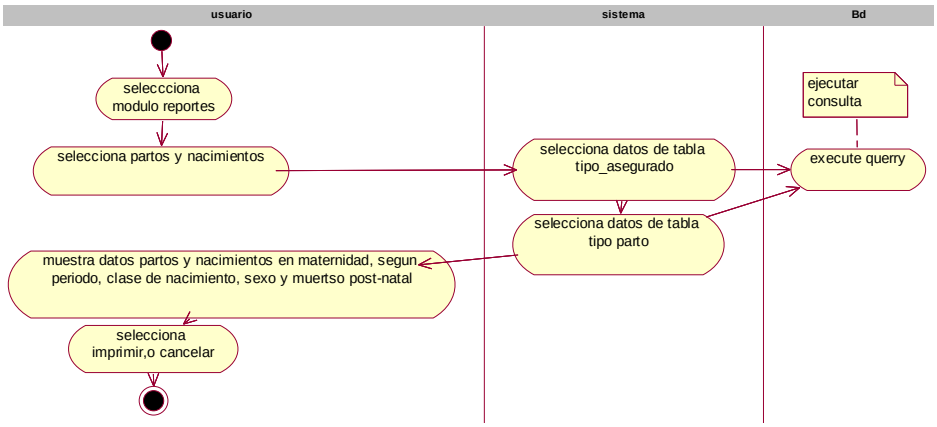


Figura 84. D.A. Ver Partos y nacimientos

II.1.8.1.2.42 Ver Cirugias programadas

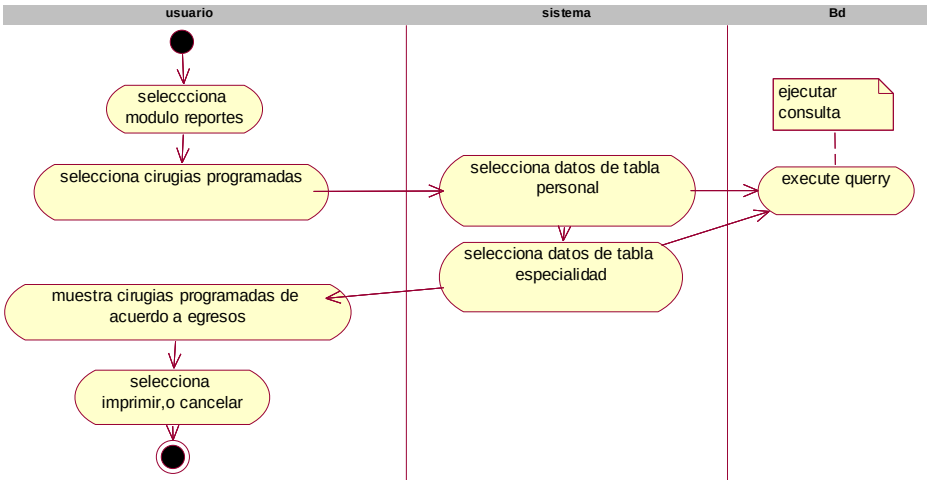


Figura 85. D.A. Ver Cirugias programadas

II.1.8.1.2.43 Ver Cirugias de emergencia

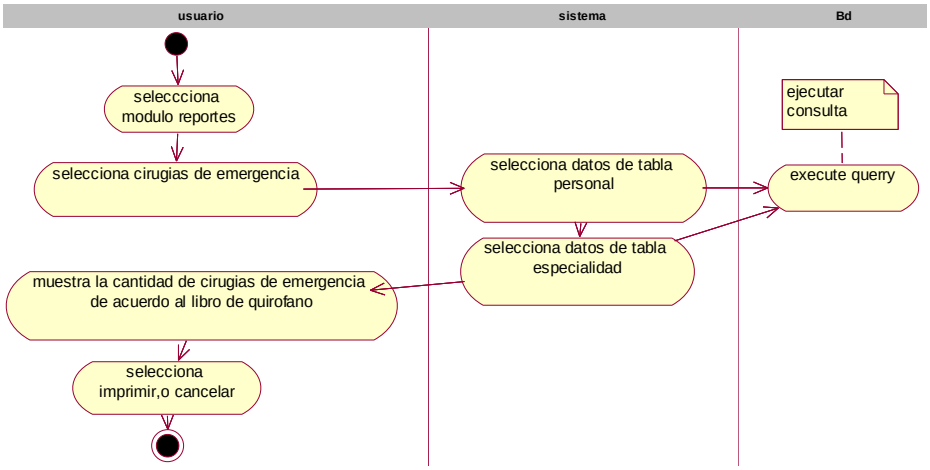


Figura 86. D.A. Ver Cirugias de emergencia

II.1.8.1.2.44 Ver Partos atendidos por obstetras

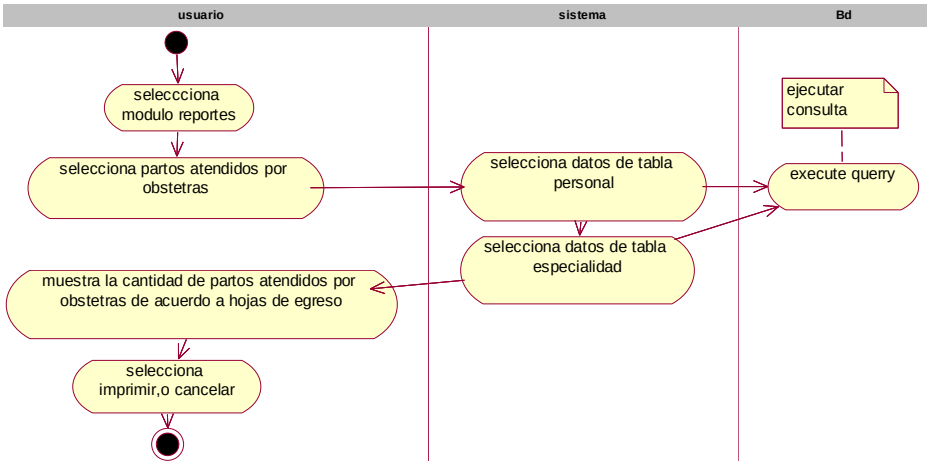


Figura 87. D.A. Ver Partos atendidos por obstetras

II.1.8.1.2.45 ayuda

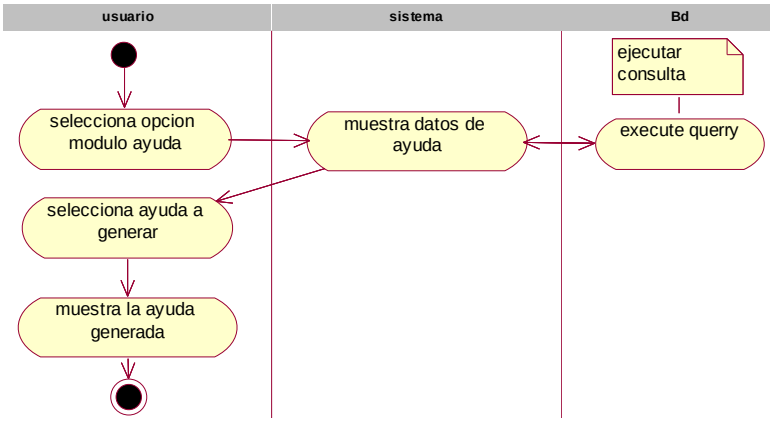


Figura 88. D.A. ayuda

II.1.8.2 Modelado de Diagramas de Secuencia

II.1.8.2.1 Introducción

El diagrama de secuencia consiste en un conjunto de objetos y sus relaciones.

II.1.8.2.1.1 Propósito

- Comprender la dinámica, sistema deseado para la organización
- Identificar las clases de análisis y diseño

II.1.8.2.1.2 Alcance

- Describir la dinámica del sistema en el tiempo de vida de las clases u objetos
- Definir un diagrama de secuencia para cada caso de uso.

II.1.8.2.2 Diagramas de Secuencia

II.1.8.2.2.1 iniciar

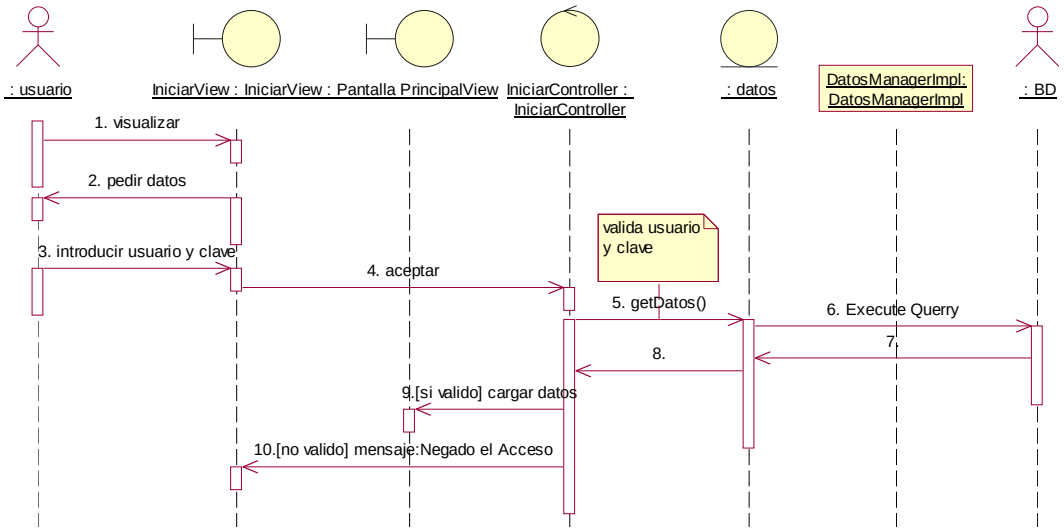


Figura 89. D.S. iniciar

II.1.8.2.2.2 Modulo Sistema-Resguadar backup

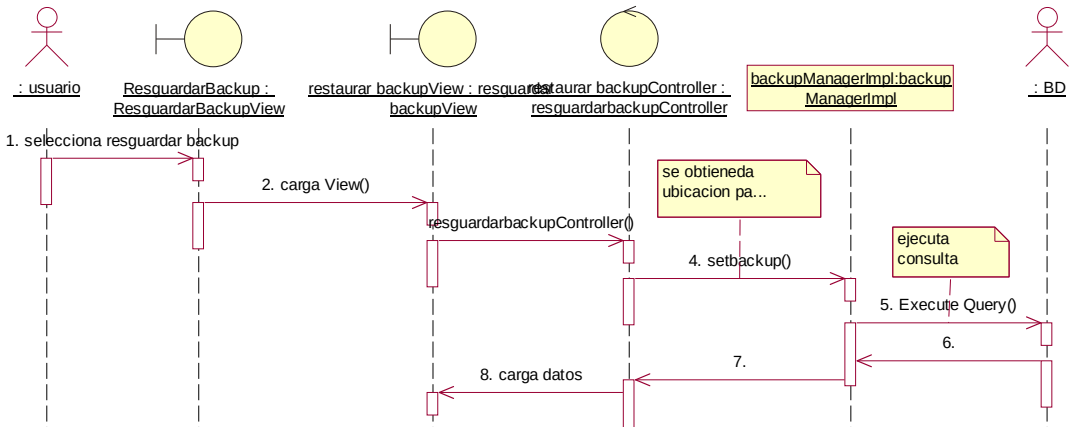


Figura 90. D.S. resguardar backup

II.1.8.2.2.2.3 Restaurar backup

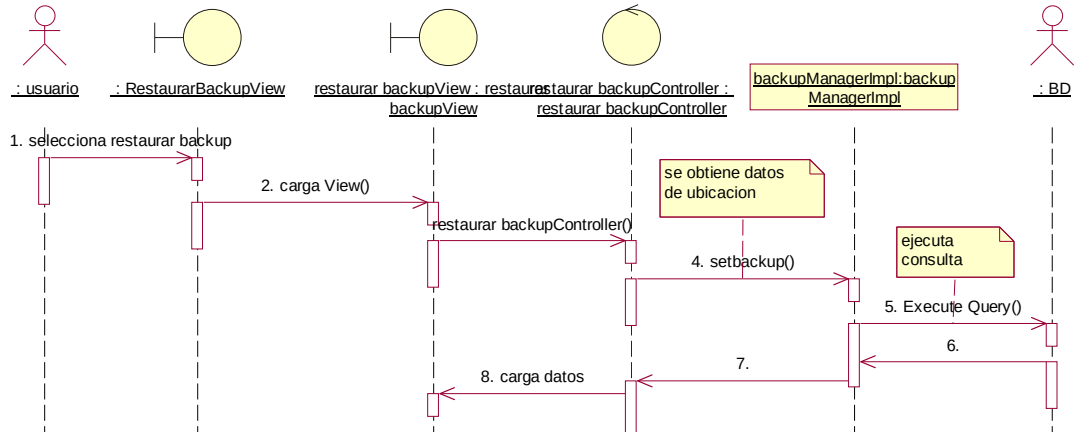


Figura 91. D.S. restaurar backup

II.1.8.2.2.4 Rol_acceso

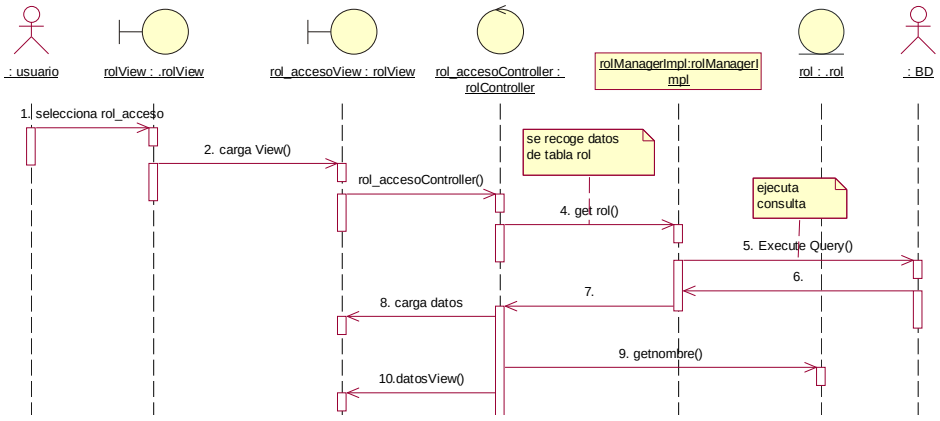


Figura 92. D.S. rol_acceso

II.1.8.2.2.5 Adicionar rol

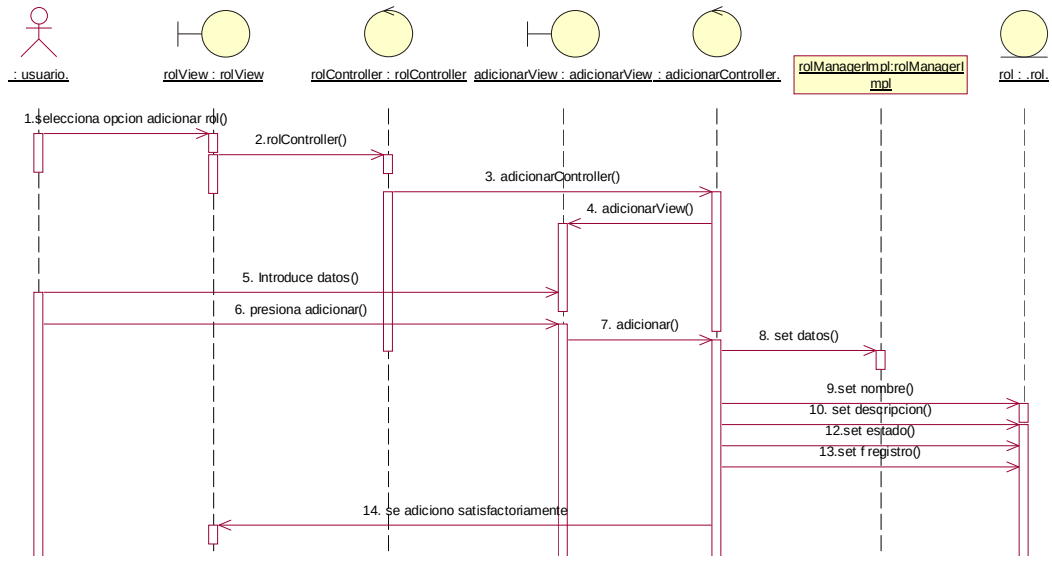


Figura 93. D.S. adicionar rol

II.1.8.2.2.6 Modificar rol

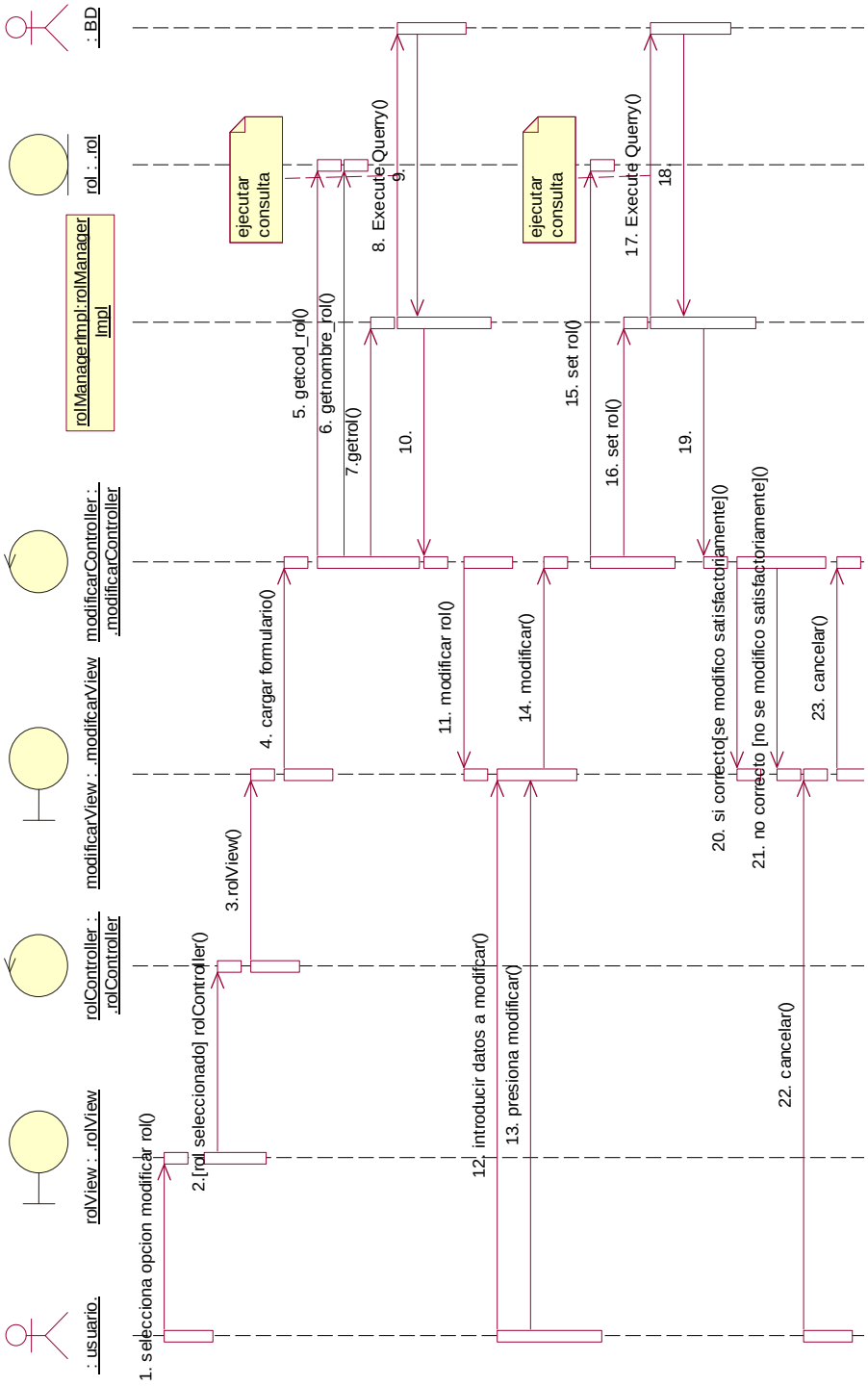


Figura 94. D.S. modificar rol

II.1.8.2.2.7 Obtener Estado (rol)

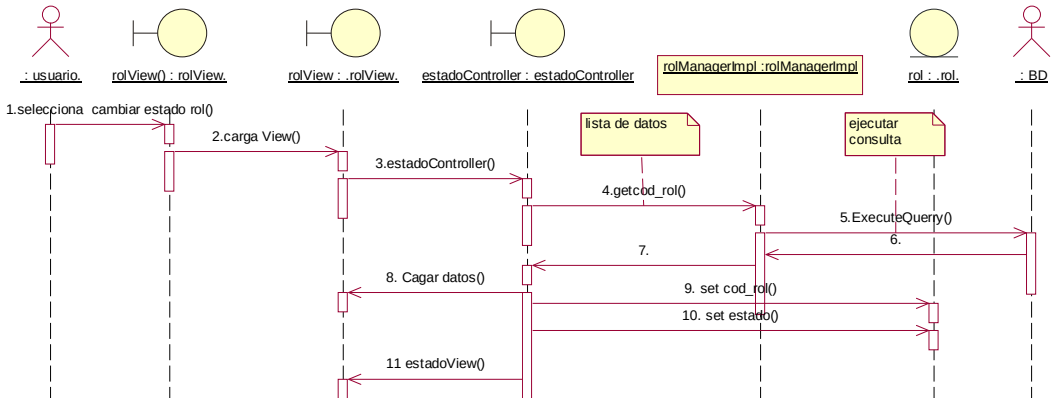


Figura 95. D.S. Obtener Estado (rol)

II.1.8.2.2.8 Modulo de administración- Camas

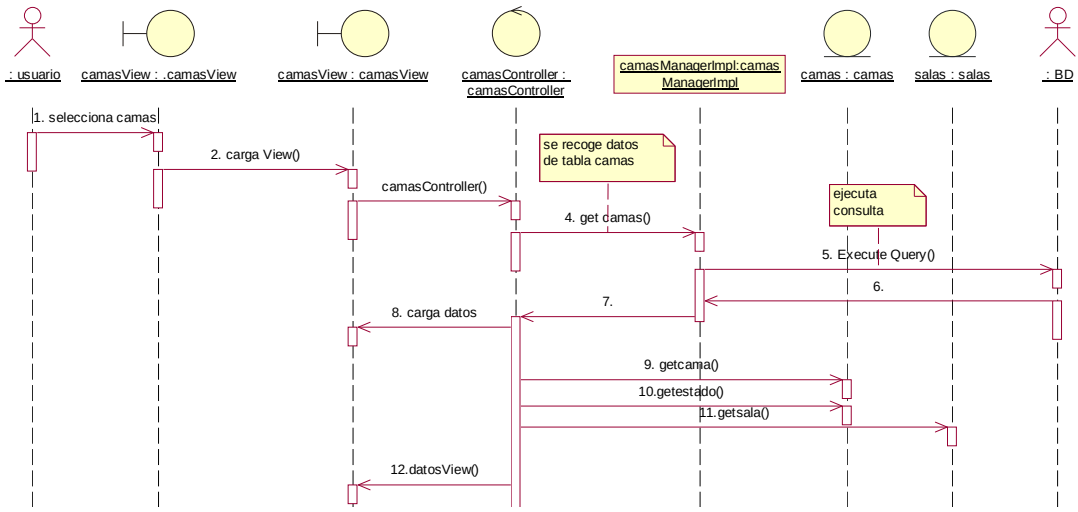


Figura 96. D.S. Modulo de administración Camas

II.1.8.2.2.9 Adicionar camas

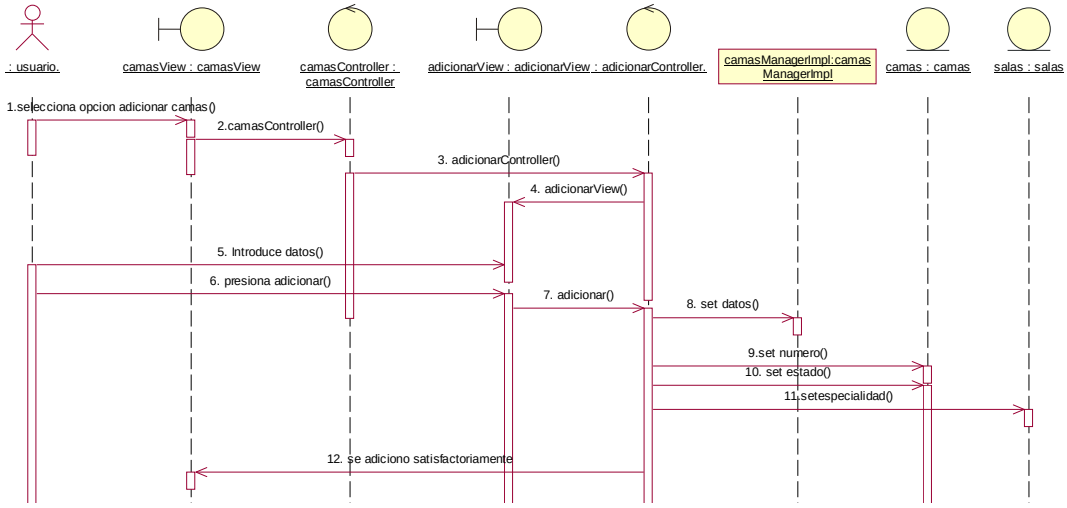


Figura 97. D.S. Adicionar camas

II.1.8.2.2.10 Modificar camas

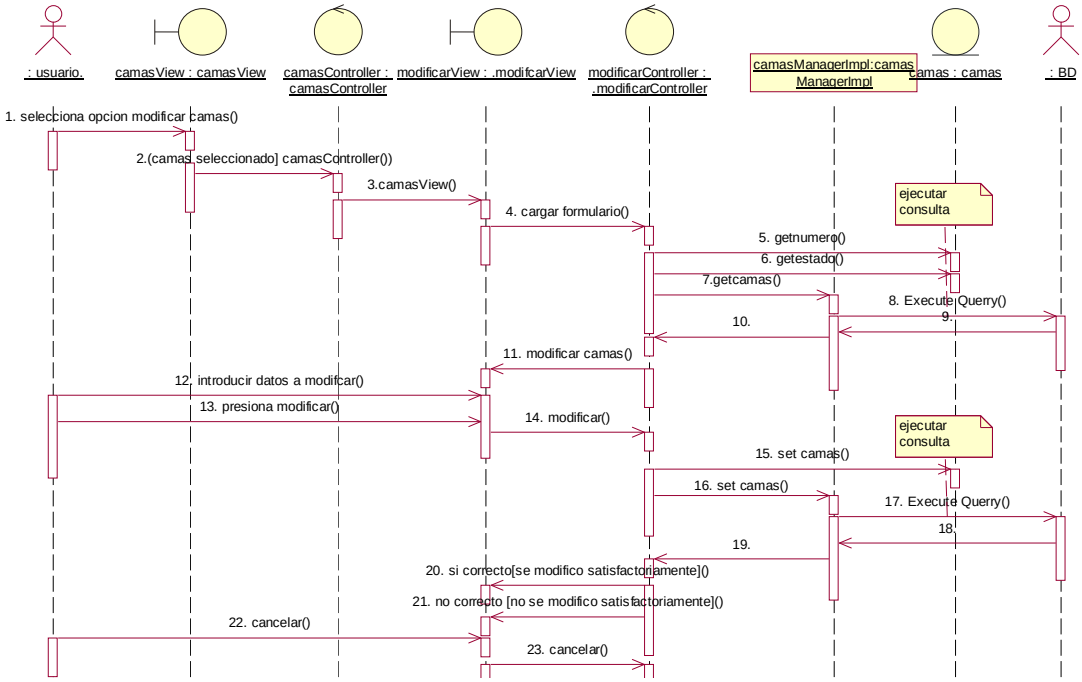


Figura 98. D.S. Modificar camas

II.1.8.2.2.11 Obtener Estado (cama)

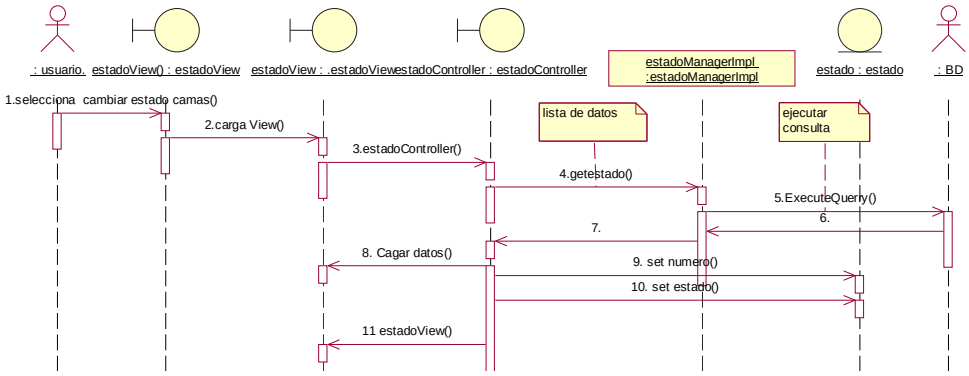


Figura 99. D.S. Obtener Estado (cama)

II.1.8.2.2.12 Especialidad

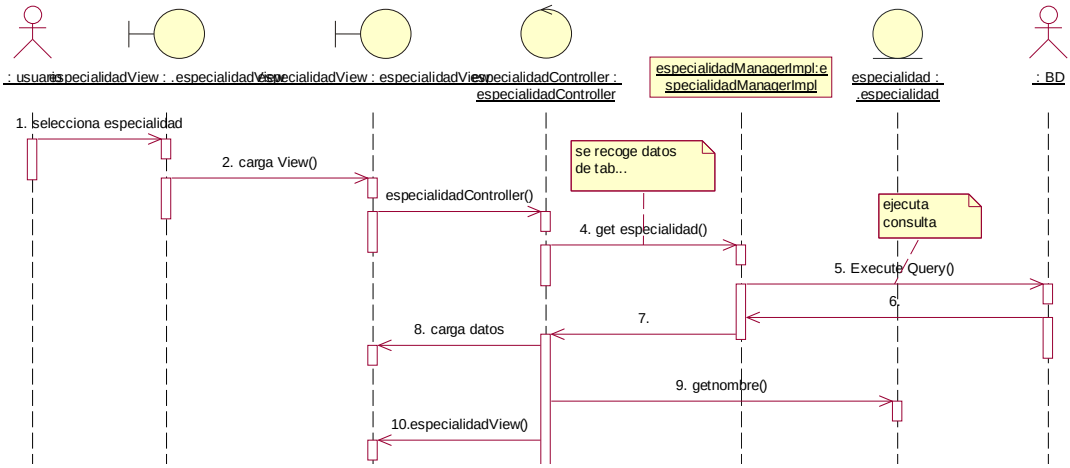


Figura 100. D.S. Especialidad

II.1.8.2.2.13 Adicionar especialidad

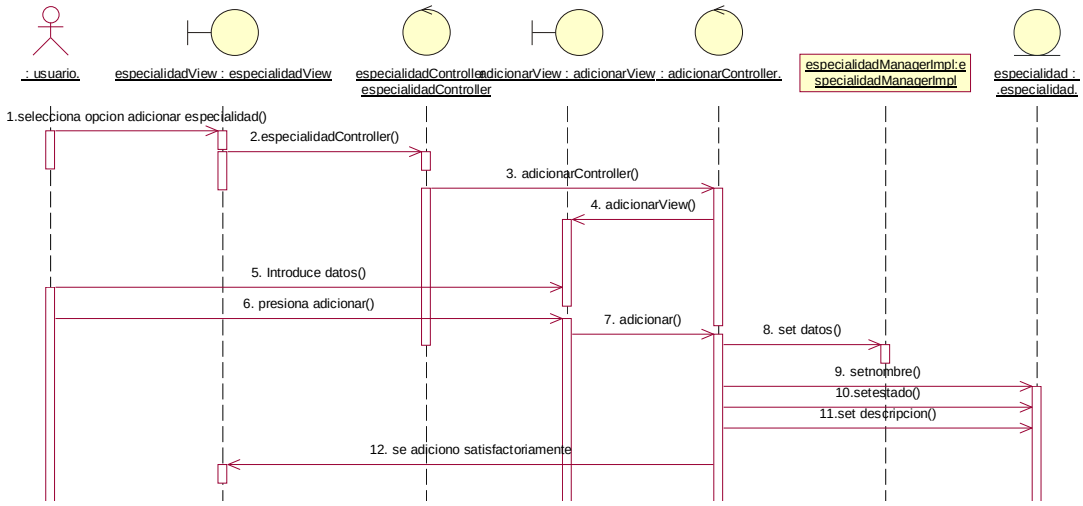


Figura 101. D.S. Adicionar especialidad

II.1.8.2.2.14 Modificar especialidad

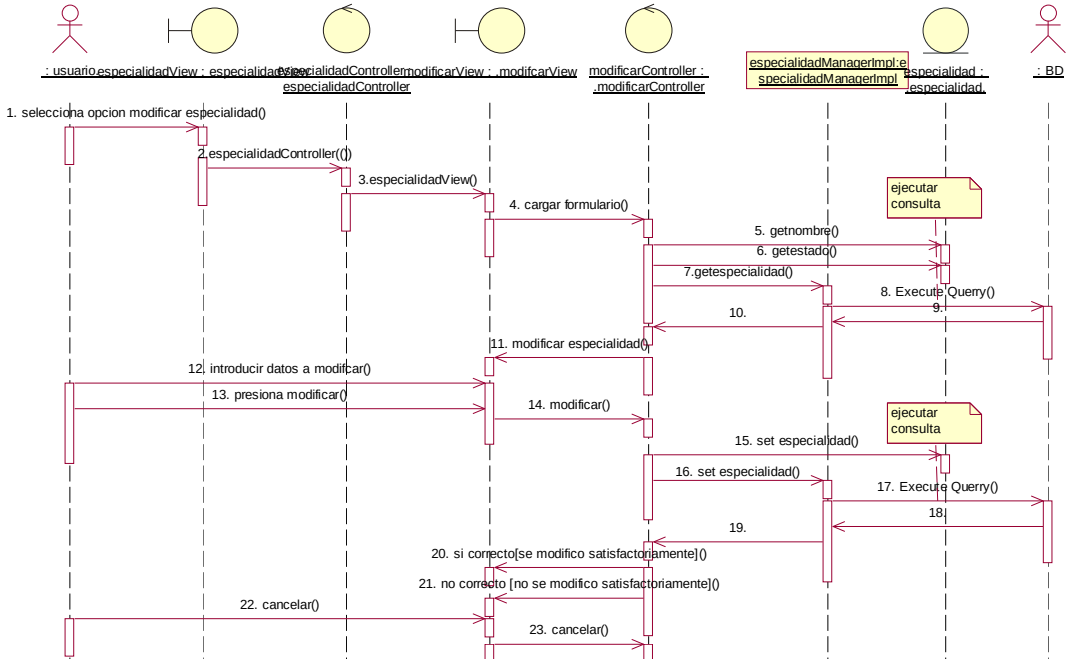


Figura 102. D.S. Modificar especialidad

II.1.8.2.2.15 Obtener Estado (especialidad)

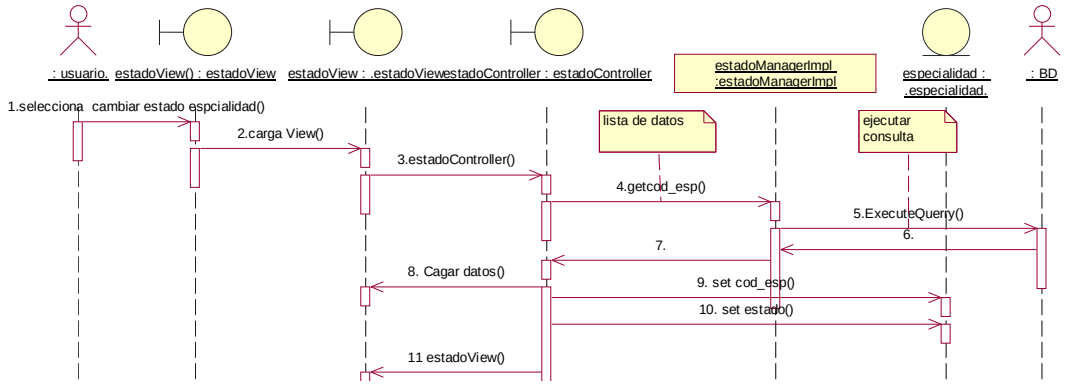


Figura 103. D.S. Obtener Estado (especialidad)

II.1.8.2.2.16 Personal

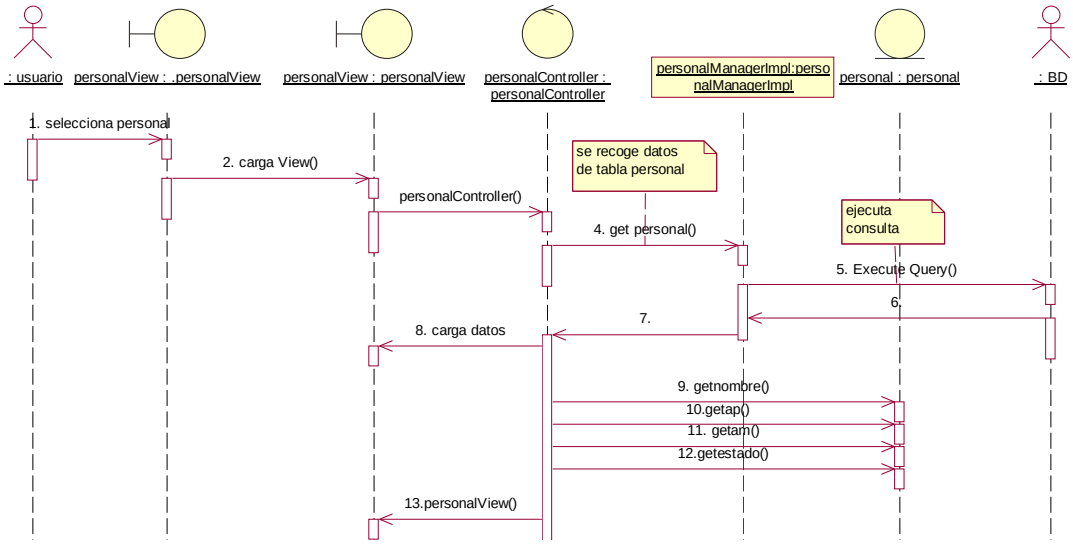


Figura 104. D.S. Personal

II.1.8.2.2.17 Adicionar personal

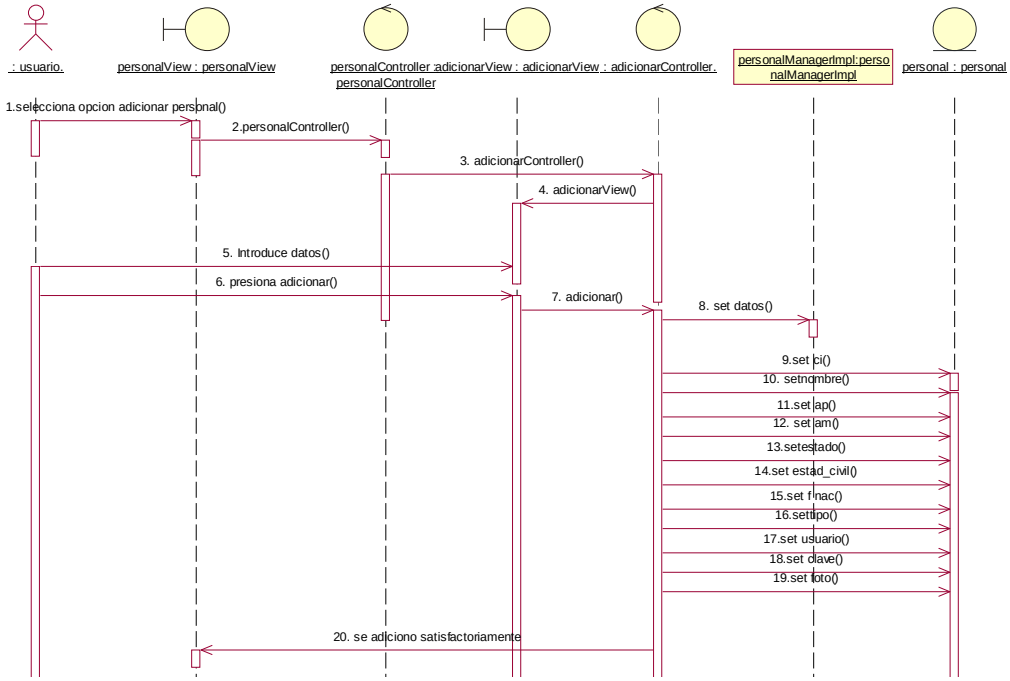


Figura 105. D.S. Adicionar personal

II.1.8.2.2.18 Modificar personal

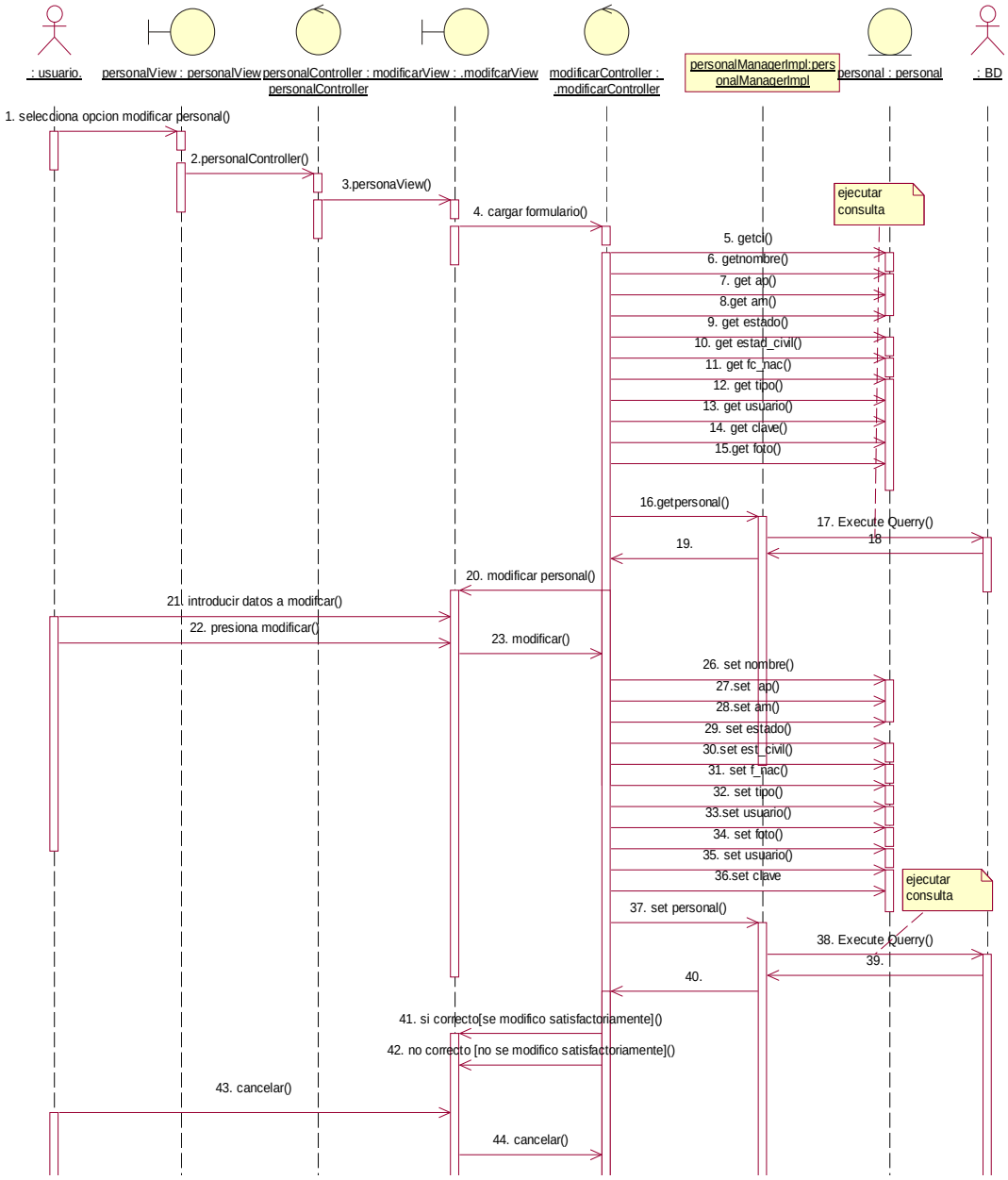


Figura 106. D.S. Modificar personal

II.1.8.2.2.19 Obtener Estado (personal)

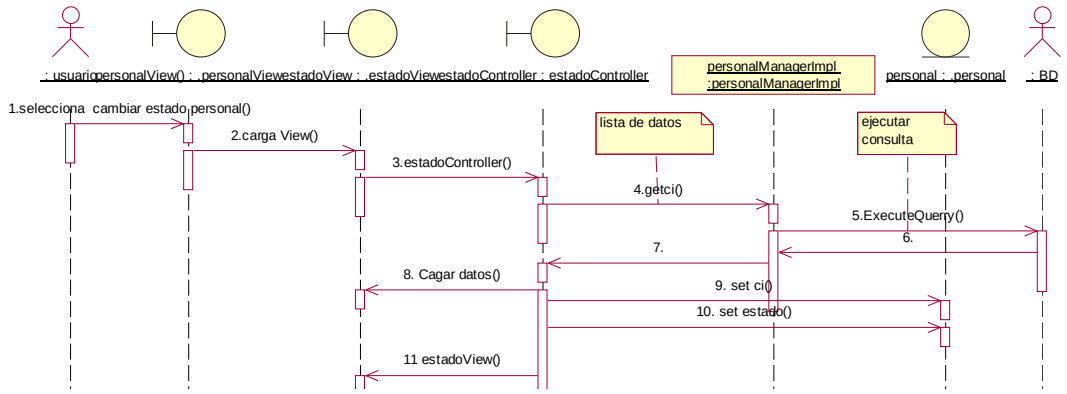


Figura 107. D.S. Obtener Estado (personal)

II.1.8.2.2.20 Salas

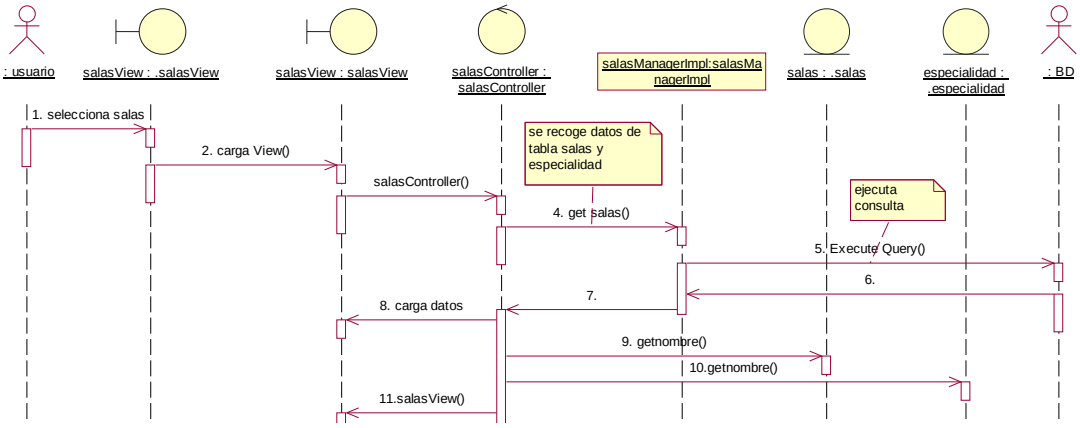


Figura 108. D.S. Salas

II.1.8.2.2.21 Adicionar salas

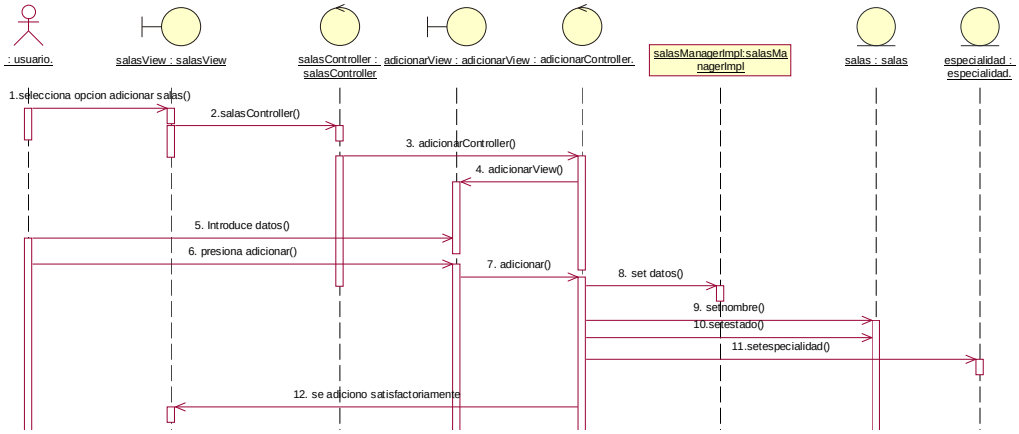


Figura 109. D.S. Adicionar salas

II.1.8.2.2.22 Modificar salas

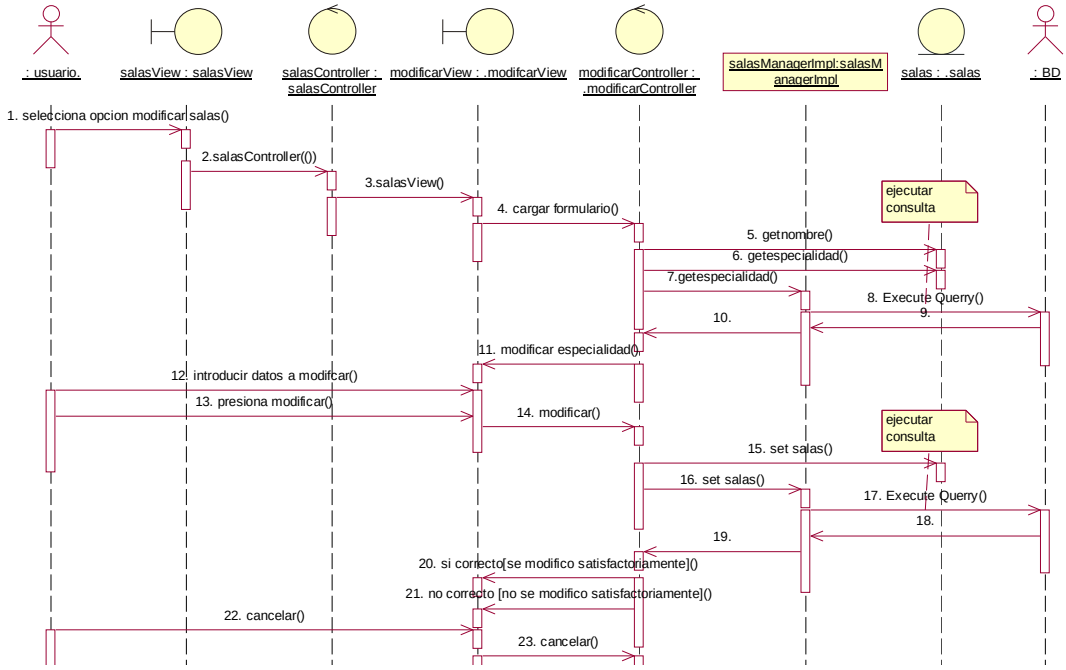


Figura 110. D.S. Modificar salas

II.1.8.2.2.23 Obtener Estado (sala)

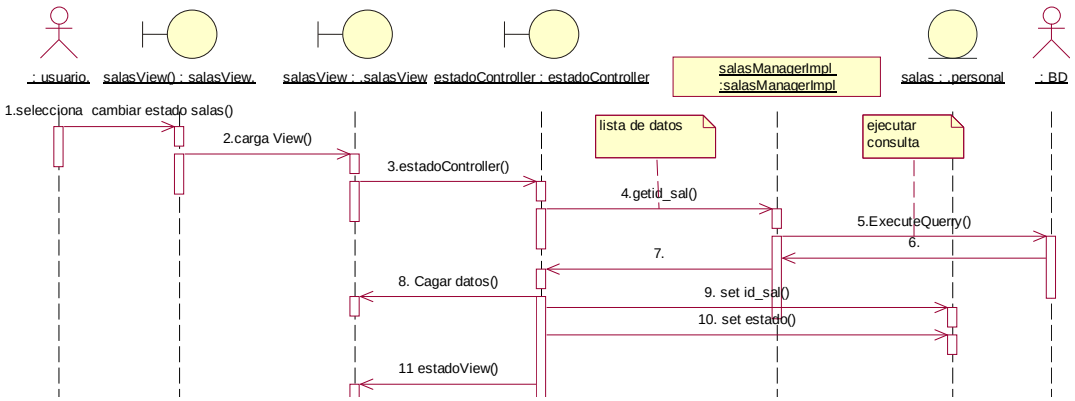


Figura 111. D.S. Obtener Estado (sala)

II.1.8.2.2.24 Modulo Ingreso

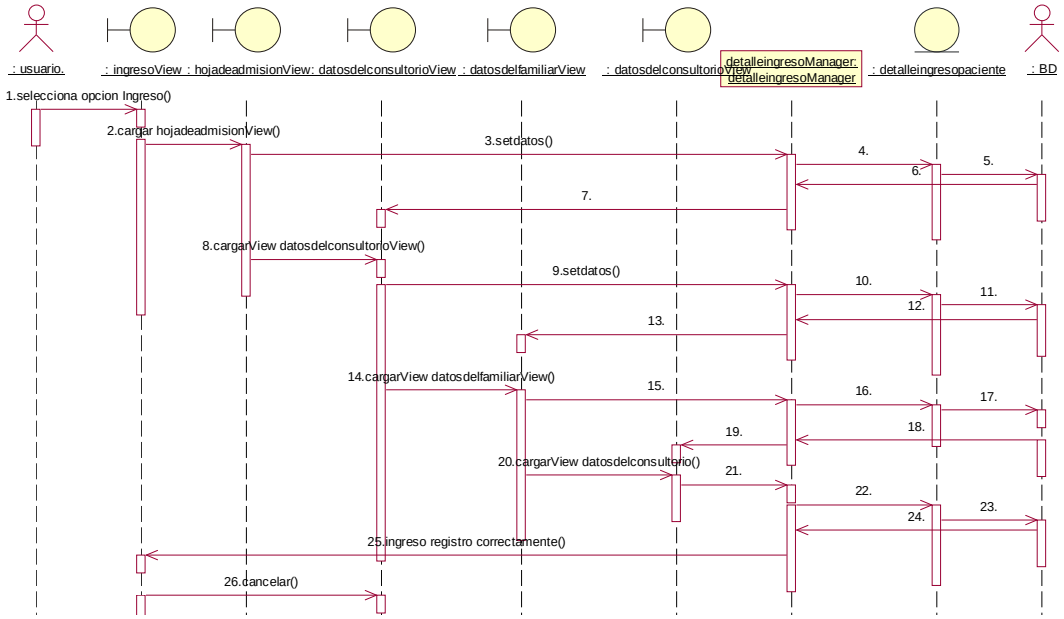


Figura 112. D.S. Modulo Ingreso

II.1.8.2.2.25 Modulo Egreso

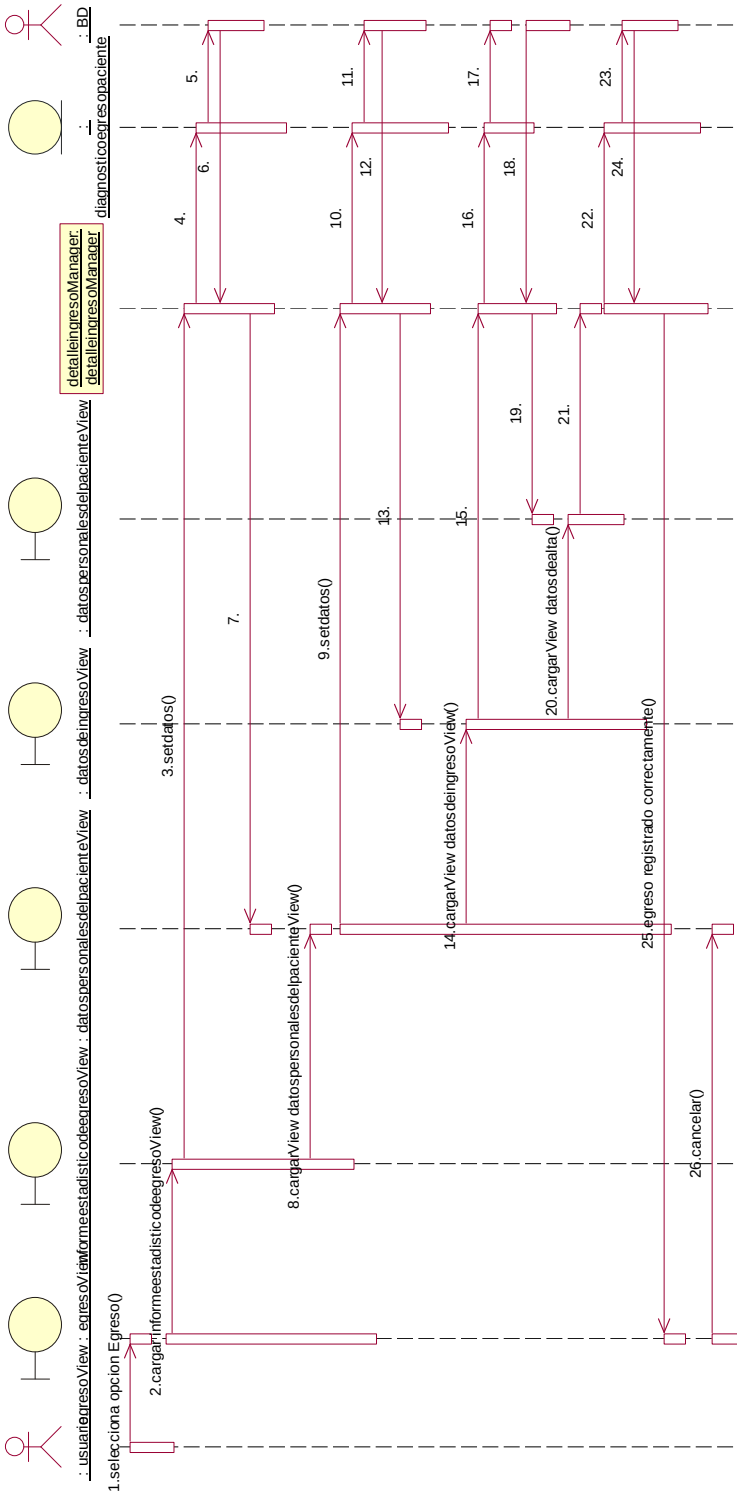


Figura 113. D.S. Modulo Egreso

II.1.8.2.2.26 Modulo Servicio

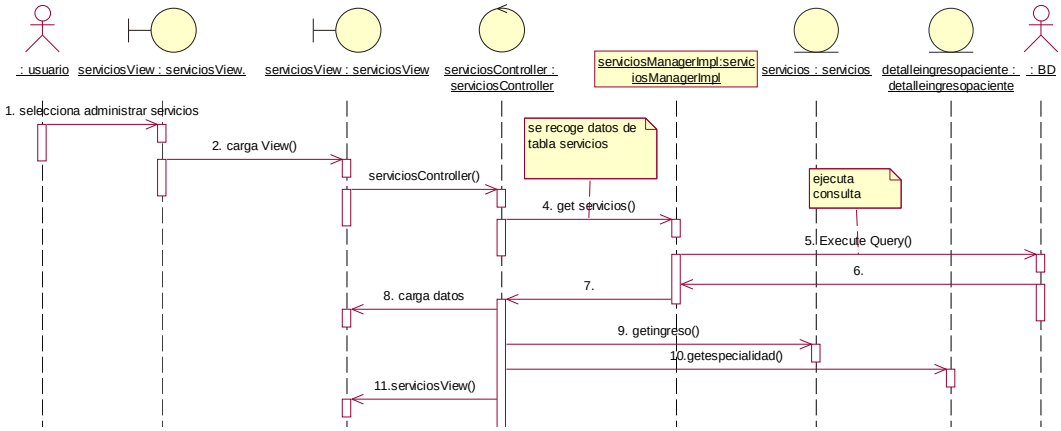


Figura 104. D.S. Modulo Servicio

II.1.8.2.2.27 Cirugia

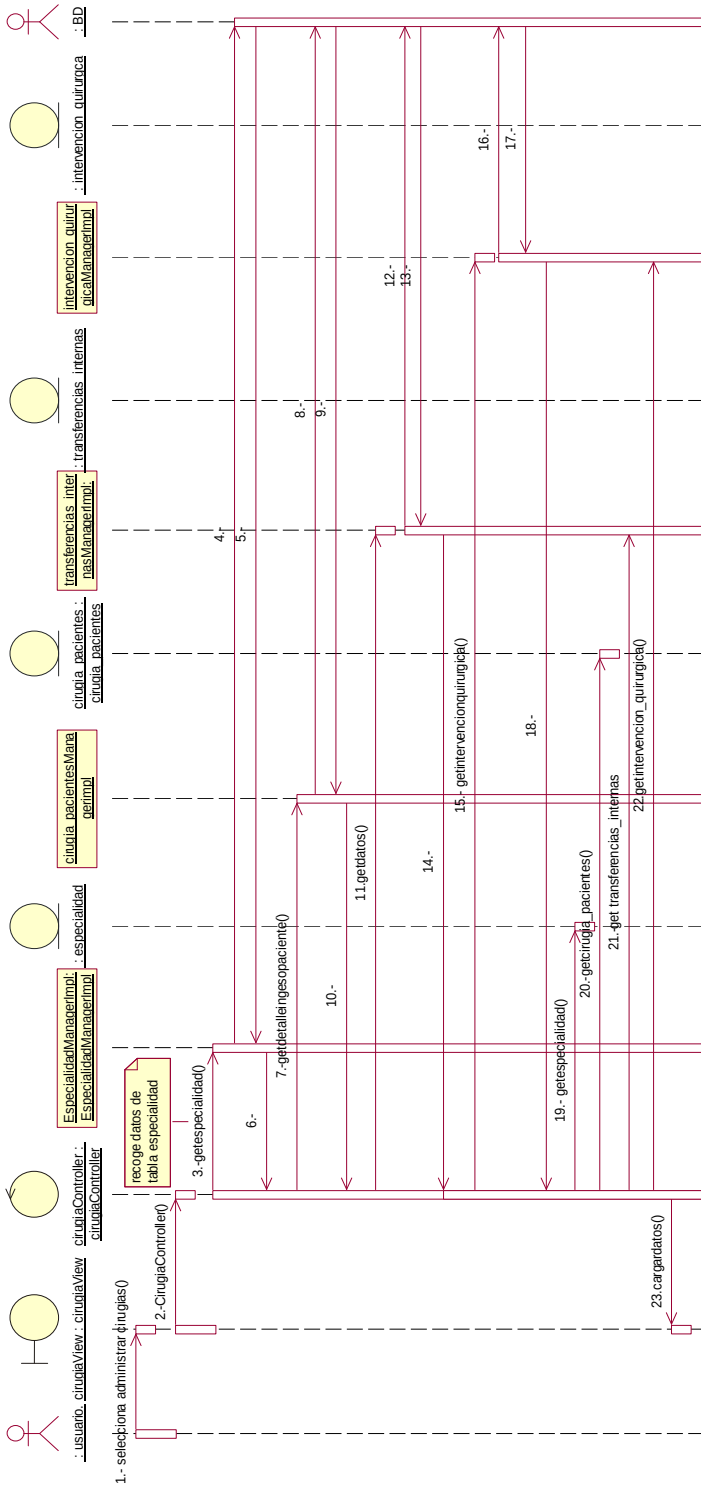


Figura 105. D.S. Cirugia

II.1.8.2.2.28 Maternidad

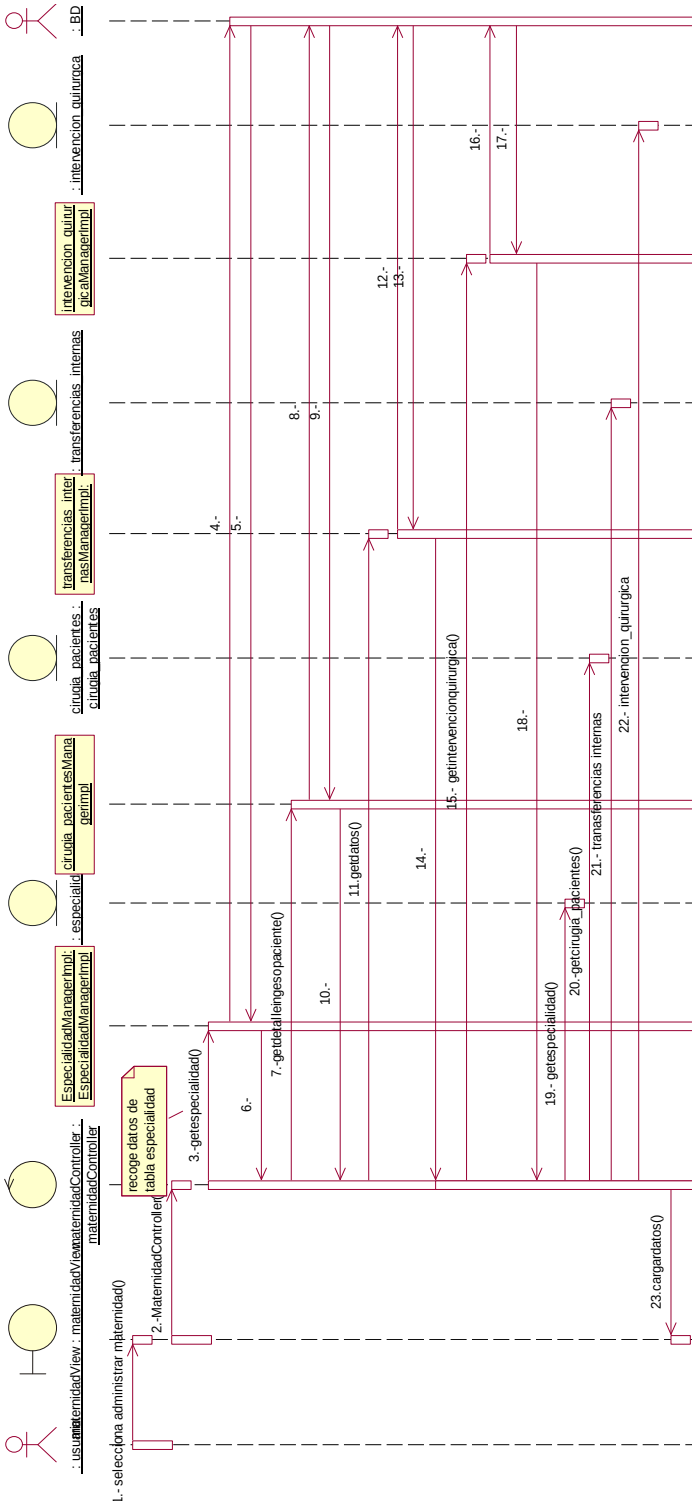


Figura 106. D.S. Maternidad

II.1.8.2.2.29 Medicina Interna

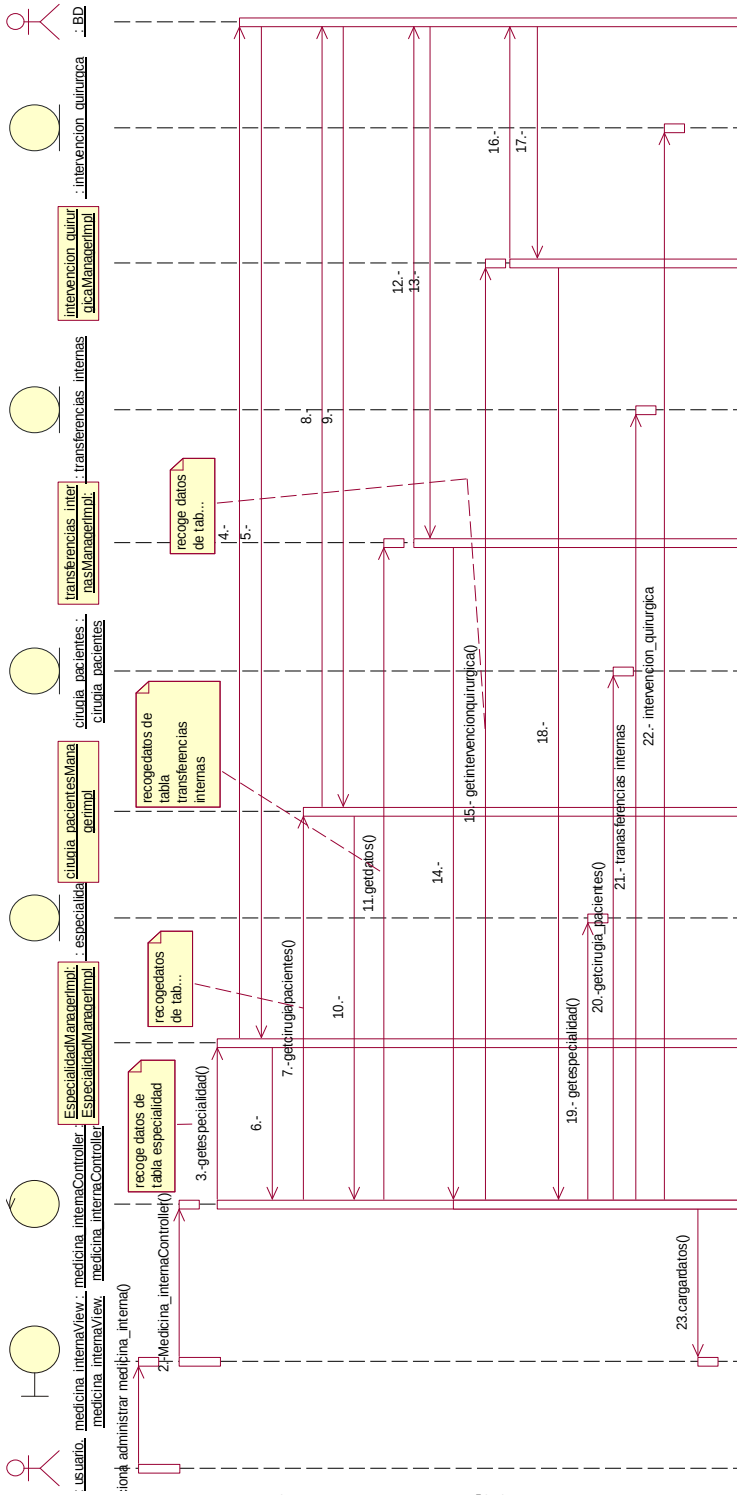


Figura 107. D.S. Medicina Interna

II.1.8.2.2.30 Pediatria

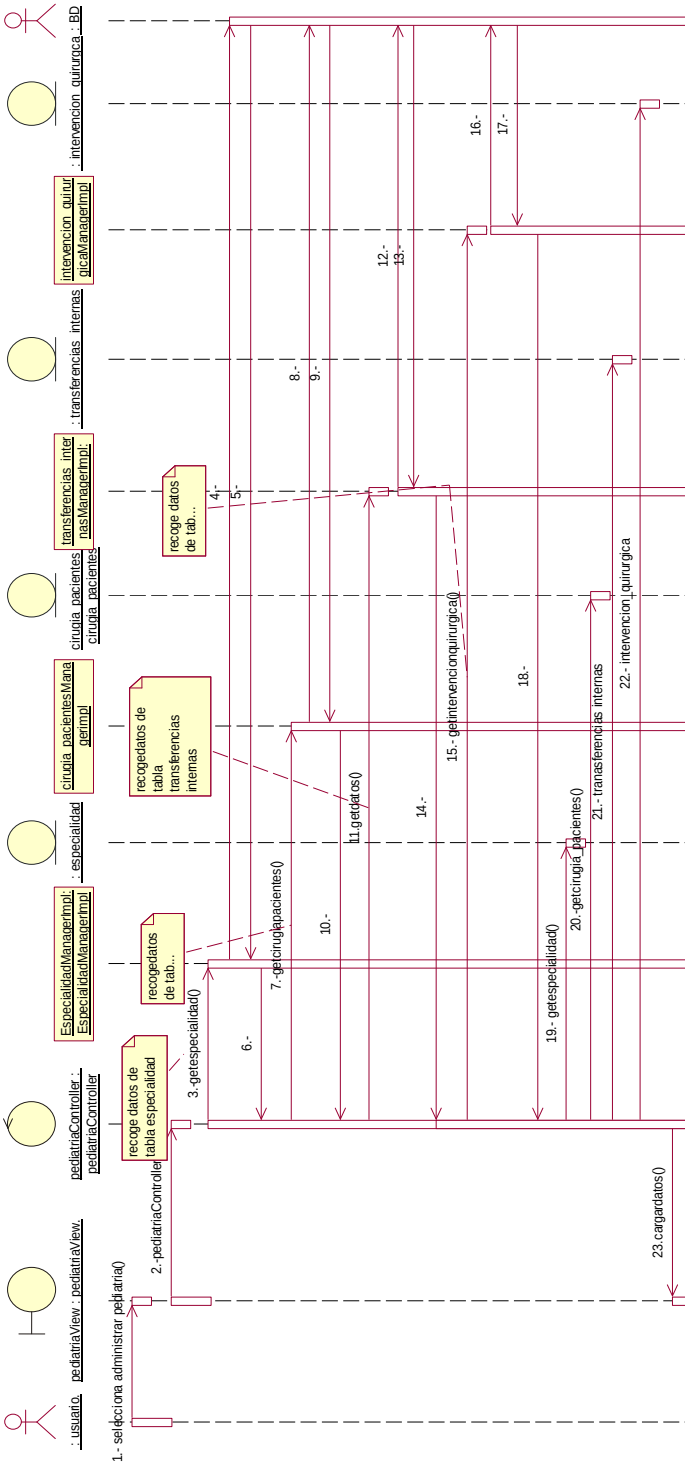


Figura 108. D.S. Pediatria

II.1.8.2.2.31 UTI

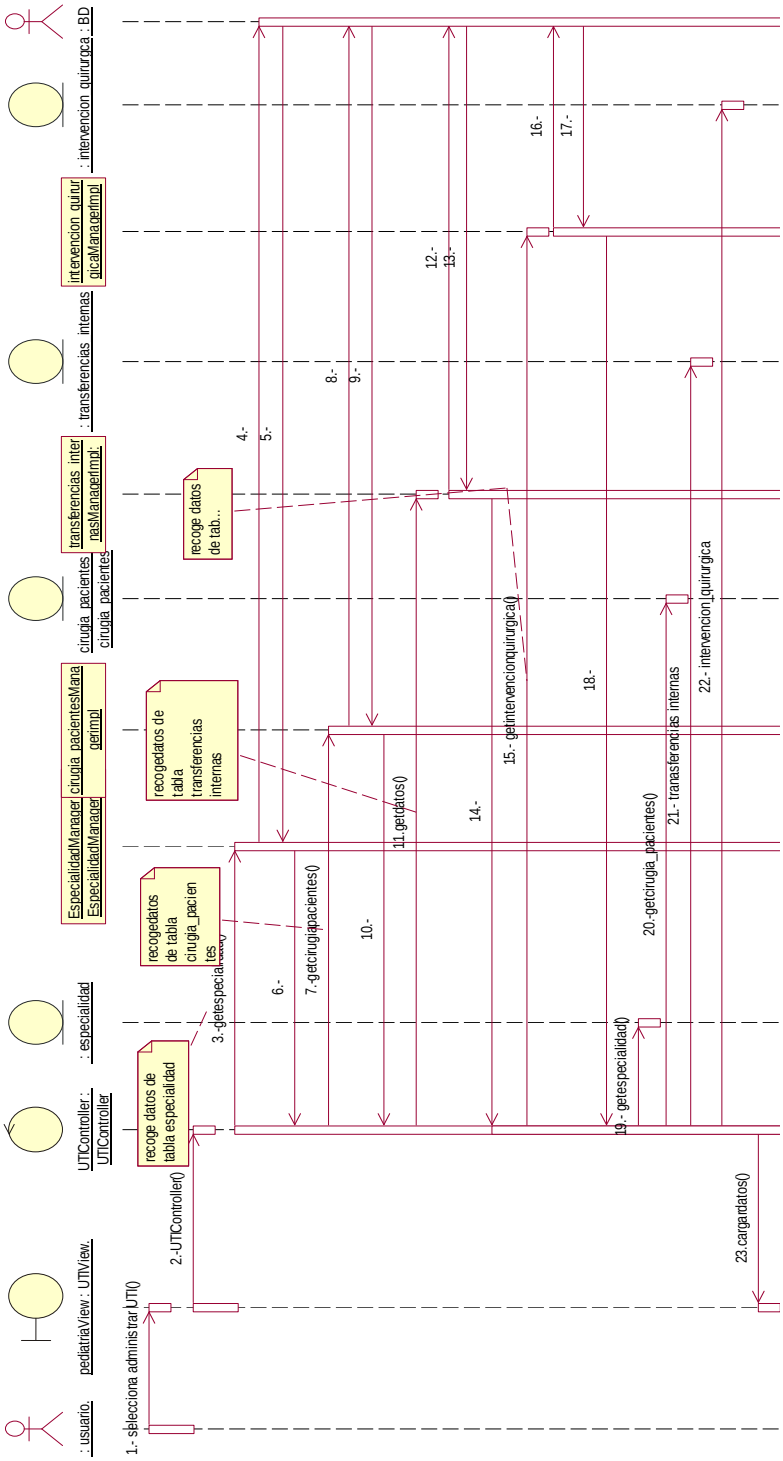


Figura 109. D.S. UTI

II.1.8.2.2.32 Reportes

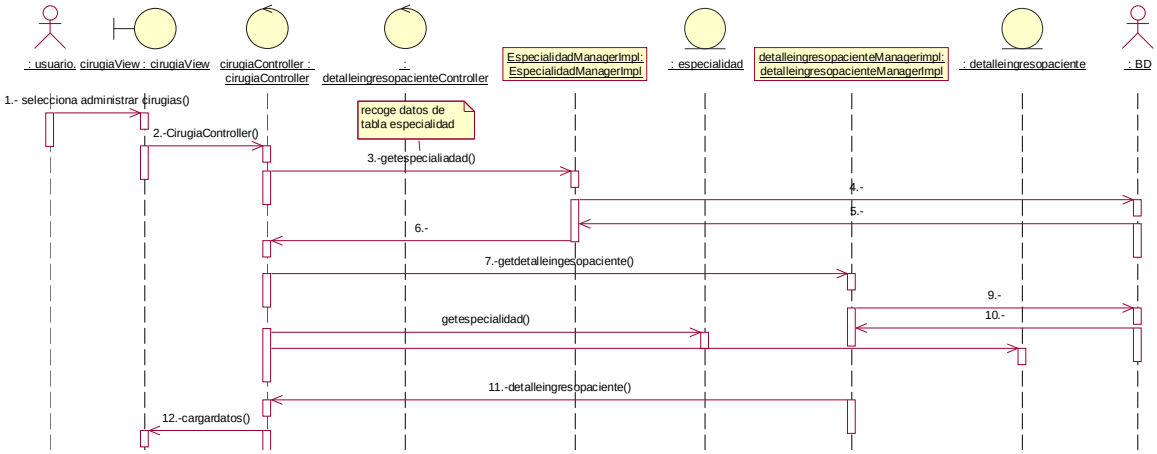


Figura 110. D.S. Modulo Reporte

II.1.8.2.2.33 Ver Hospitalizacion

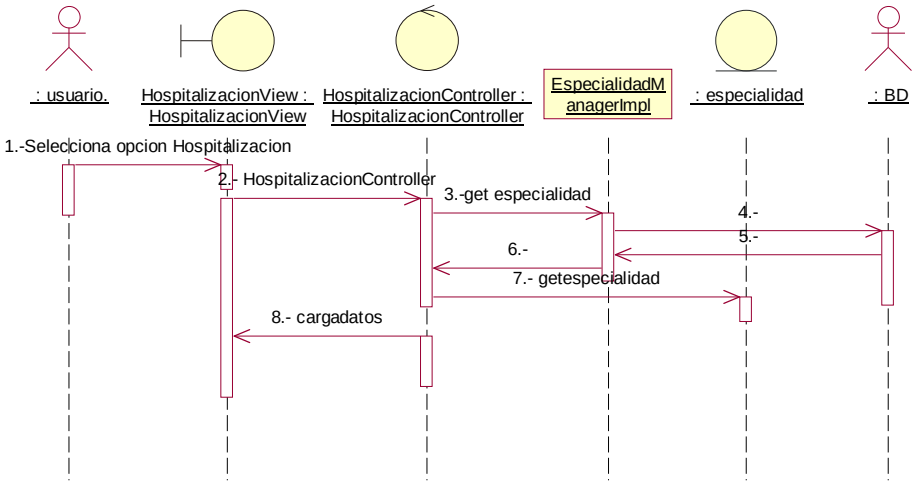


Figura 111. D.S. Ver Hospitalizacion

II.1.8.2.2.34 Ver Intervencion quirurgica por especialidad

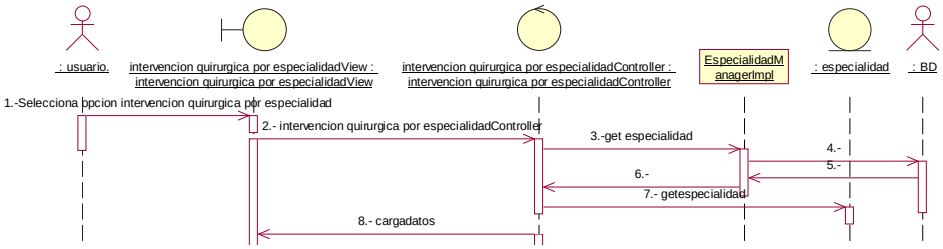


Figura 112. D.S. Ver Intervencion quirurgica por especialidad

II.1.8.2.2.35 Ver Intervencion quirurgica por clase de cirugía

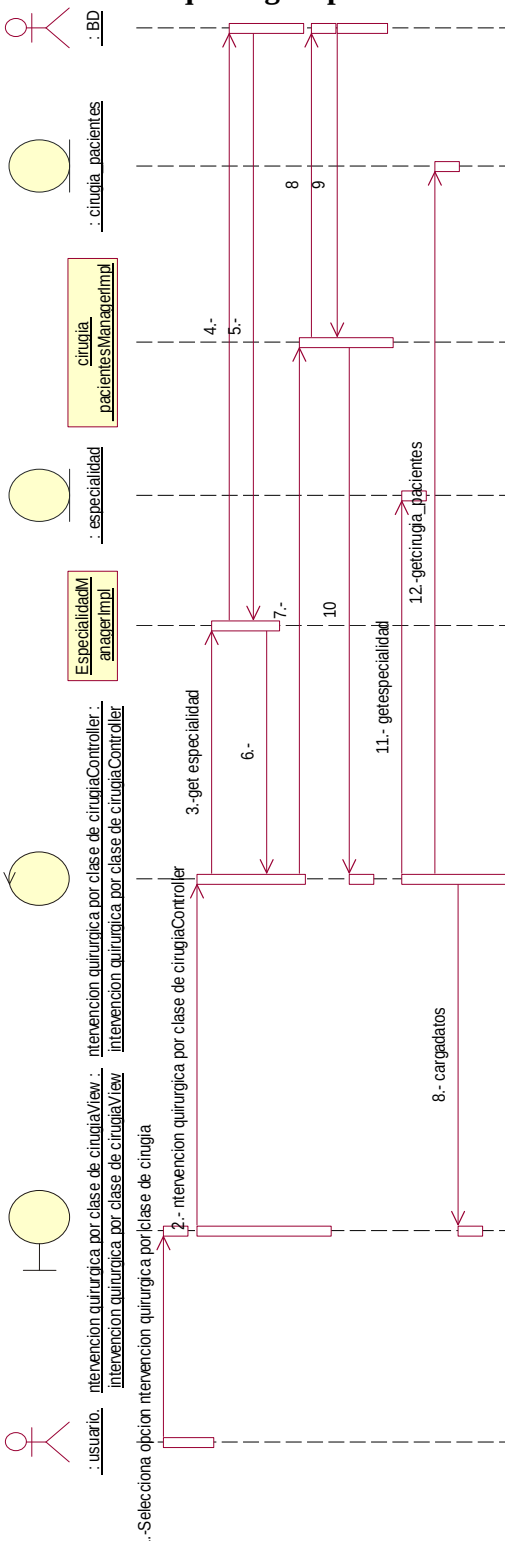


Figura 113. D.S. Ver Intervencion quirurgica por clase de cirugia

II.1.8.2.2.36 Ver Mensual hospitalario

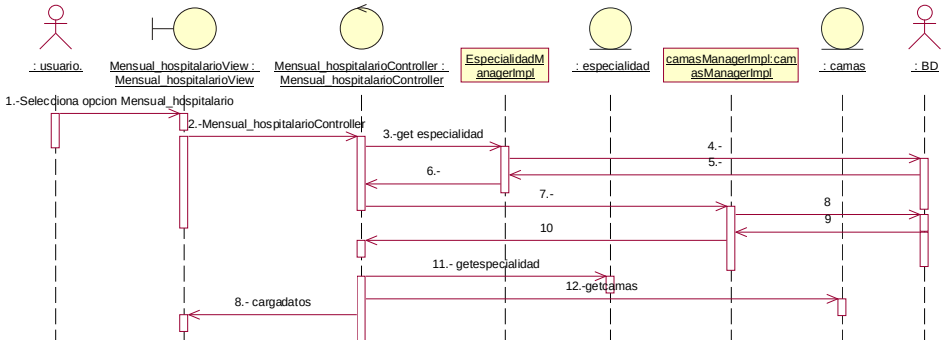


Figura 114. D.S. Ver Mensual hospitalario

II.1.8.2.2.37 Ver Mensual por especialidad

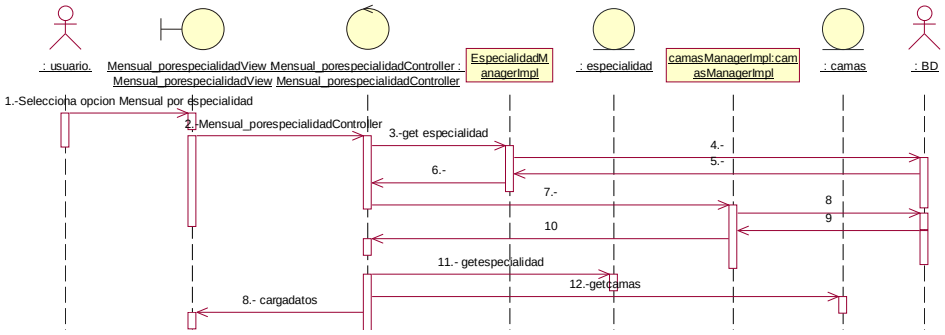


Figura 115. D.S. Mensual por especialidad

II.1.8.2.2.38 Ver Partos_nacimientos

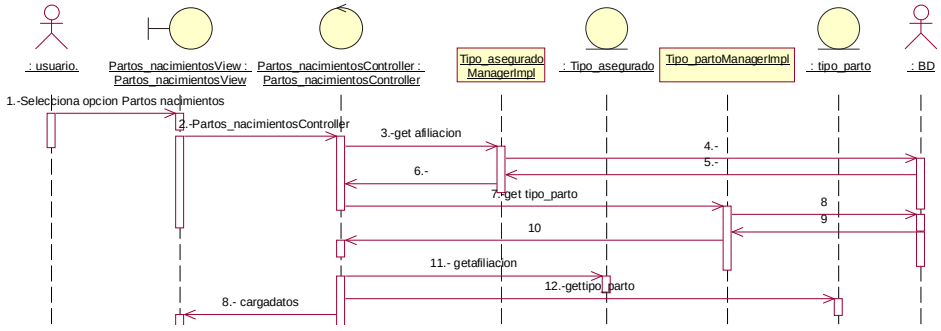


Figura 116. D.S. Ver Partos_nacimientos

II.1.8.2.2.39 Ver cirugias programadas

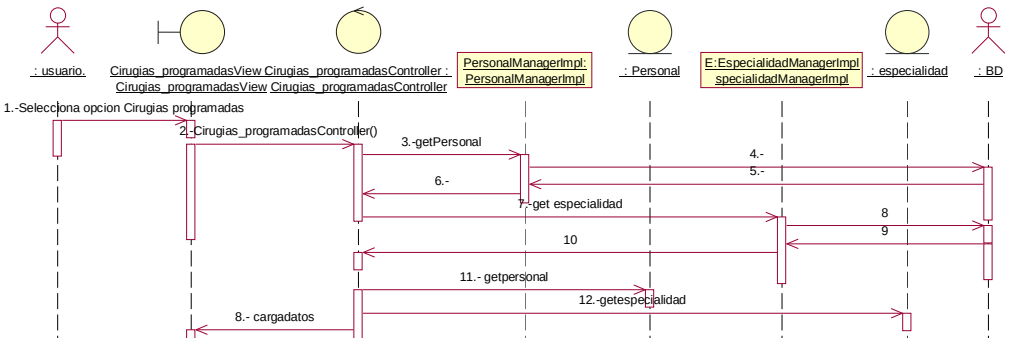


Figura 117. D.S. Ver cirugias programadas

II.1.8.2.2.40 Ver cirugias de emergencia

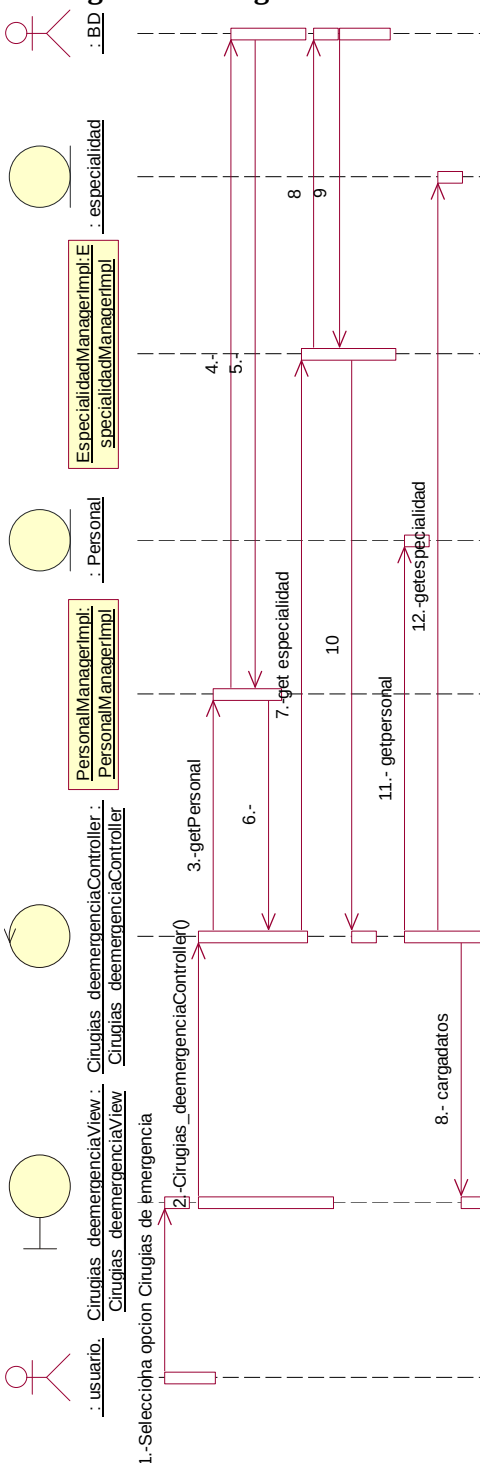


Figura 118. D.S. Ver cirugias de emergencia

II.1.8.2.2.41 Ver partos atendiso por obstetras

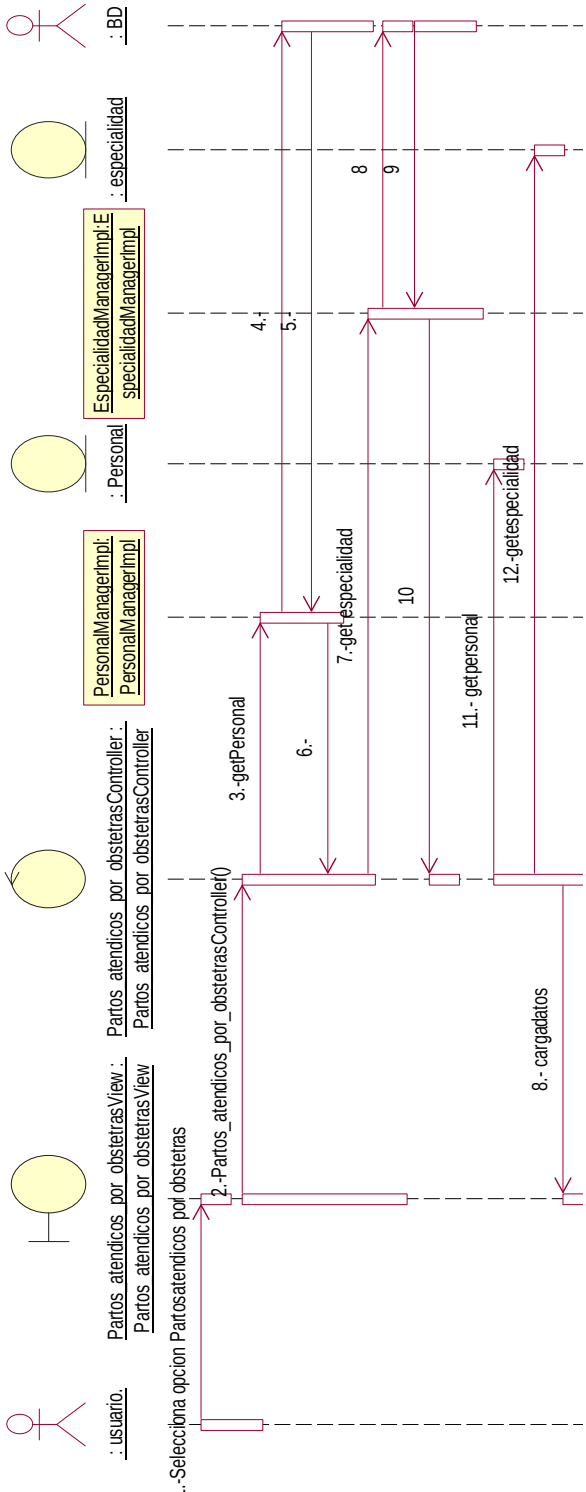


Figura 119. D.S. Ver partos atendidos por obstetras

II.1.8.2.2.42 ayuda

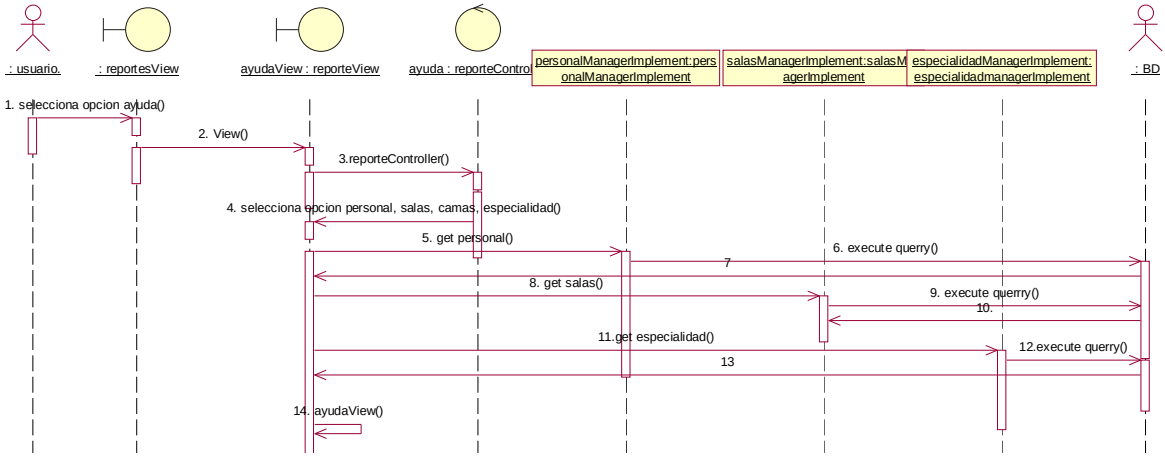


Figura 120. D.S. Ayuda

II.1.8.3 Modelado de Diagramas de Colaboracion

II.1.8.3.1 iniciar

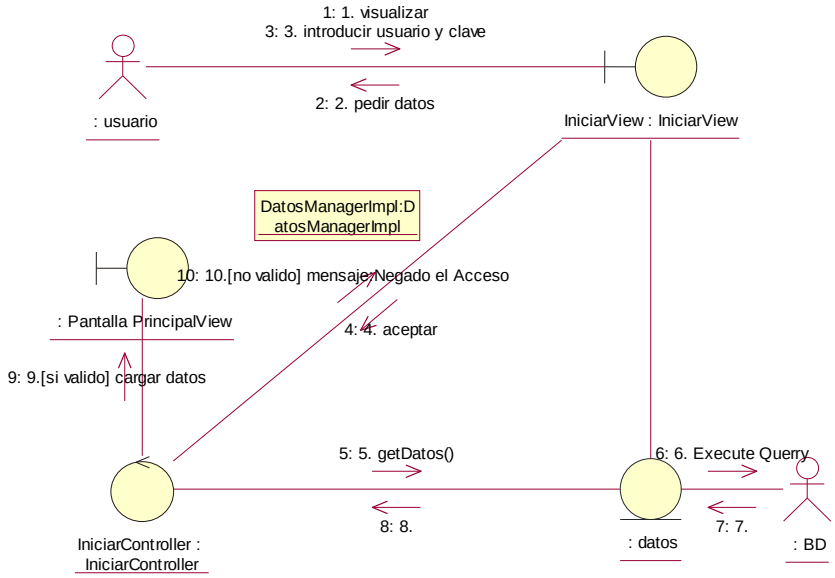


Figura 121. D.C. iniciar

II.1.8.3.2 Modulo Sistema-Resguadar backup

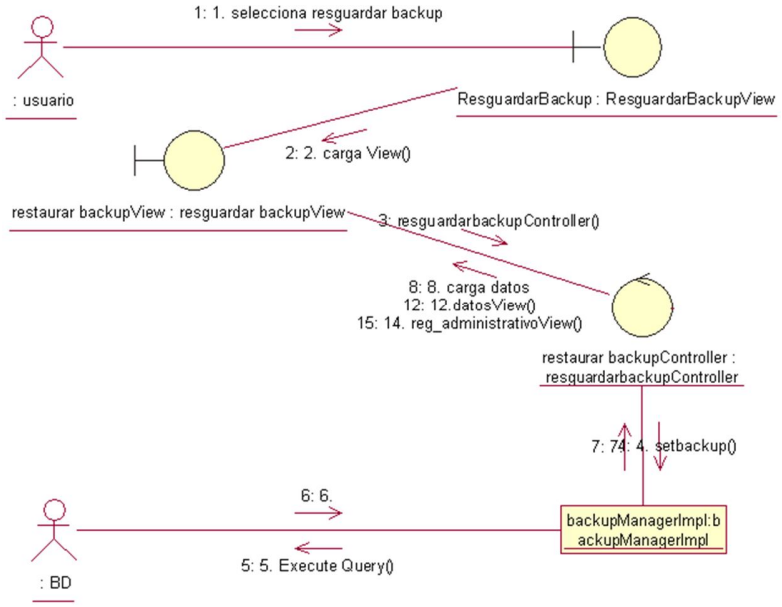


Figura 122. D.C. Modulo Sistema-Resguadar backup

II.1.8.3.3 Restaurar backup

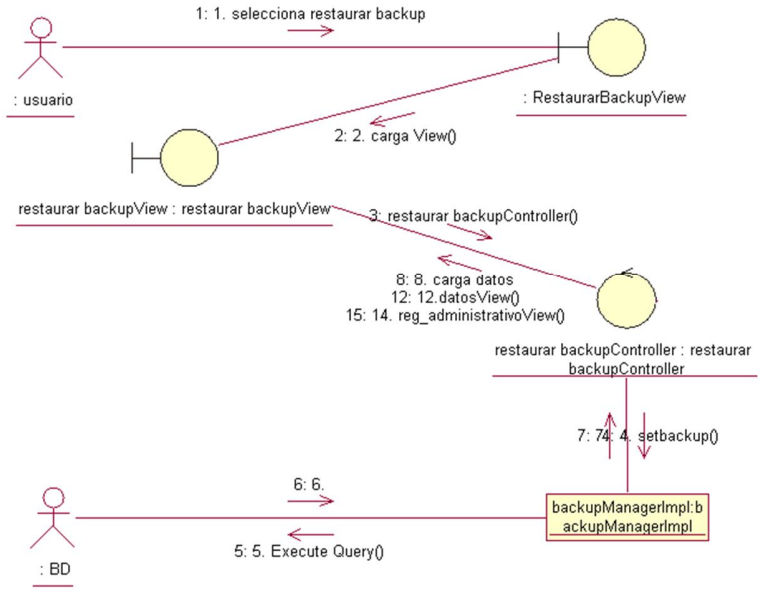


Figura 123. D.C. Restaurar backup

II.1.8.3.4 Rol_acceso

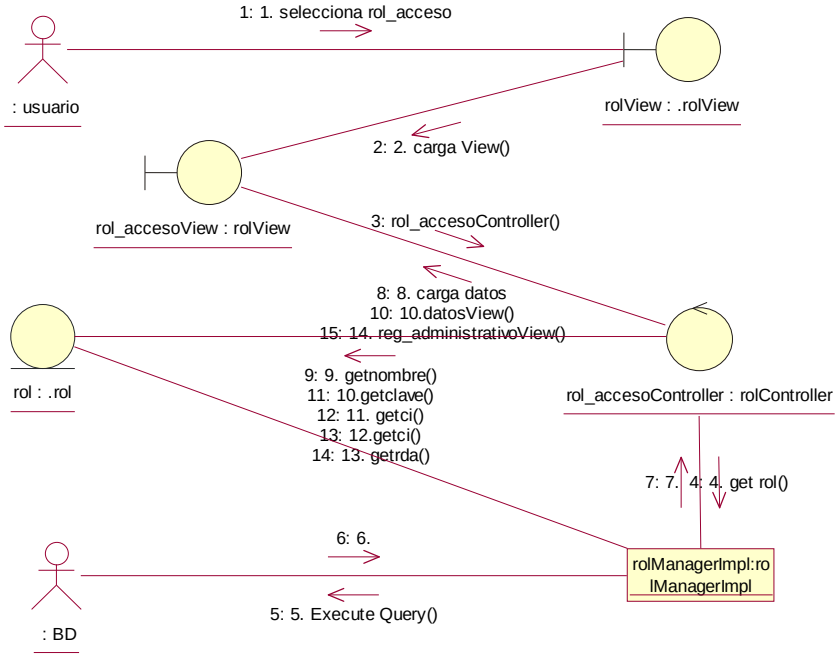


Figura 124. D.C. Rol_acceso

II.1.8.3.5 Adicionar rol

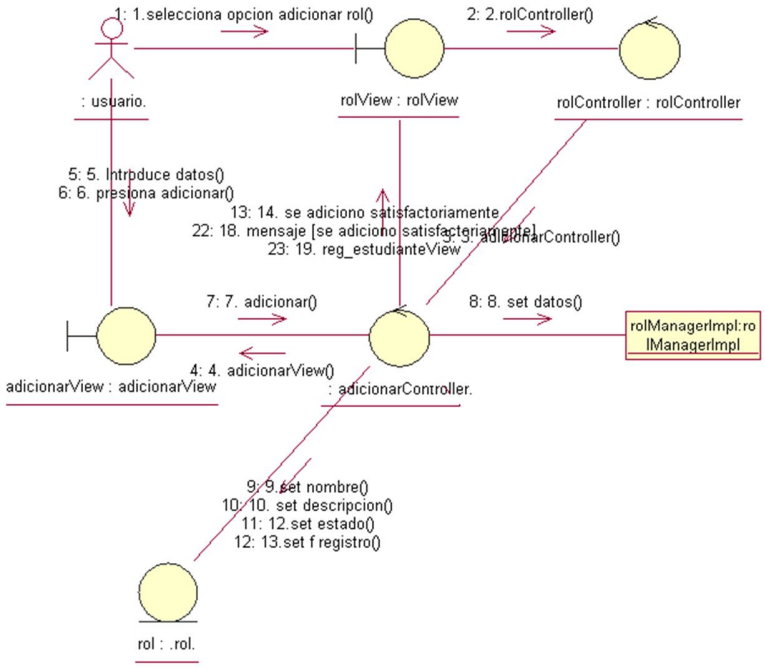


Figura 125. D.C. Adicionar rol

II.1.8.3.6 Modificar rol

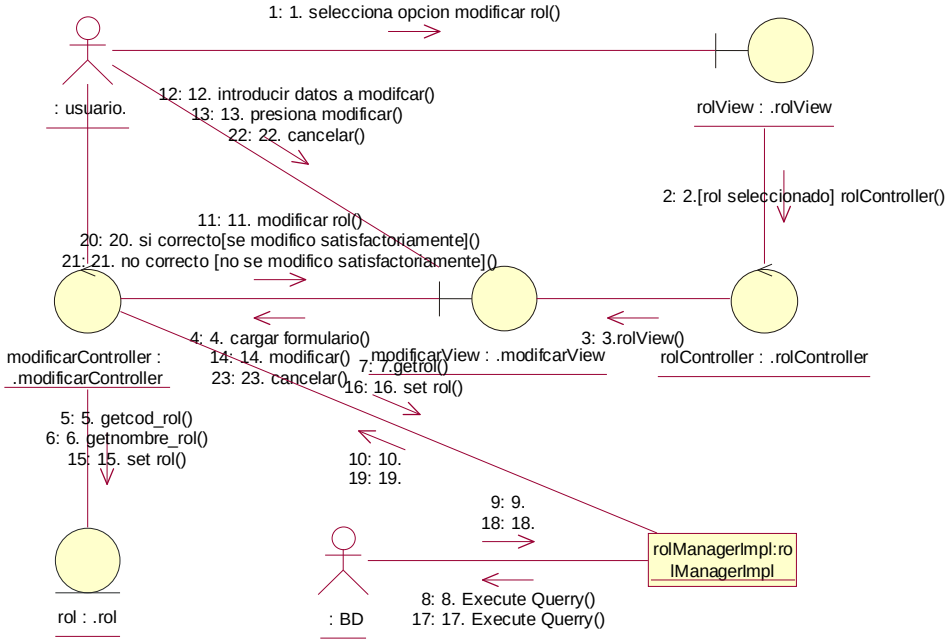


Figura 126. D.C. Modificar rol

II.1.8.3.7 Obtener Estado (rol)

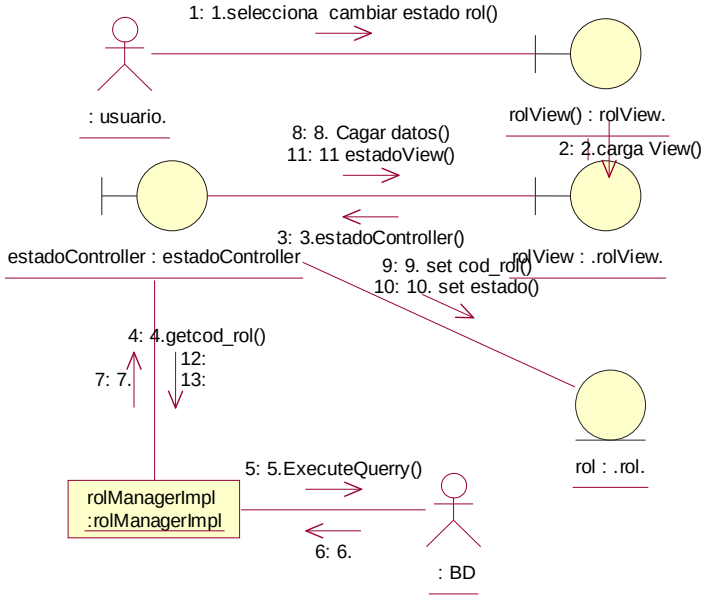


Figura 127. D.C. Obtener Estado (rol)

II.1.8.3.8 Modulo de administración-Camas

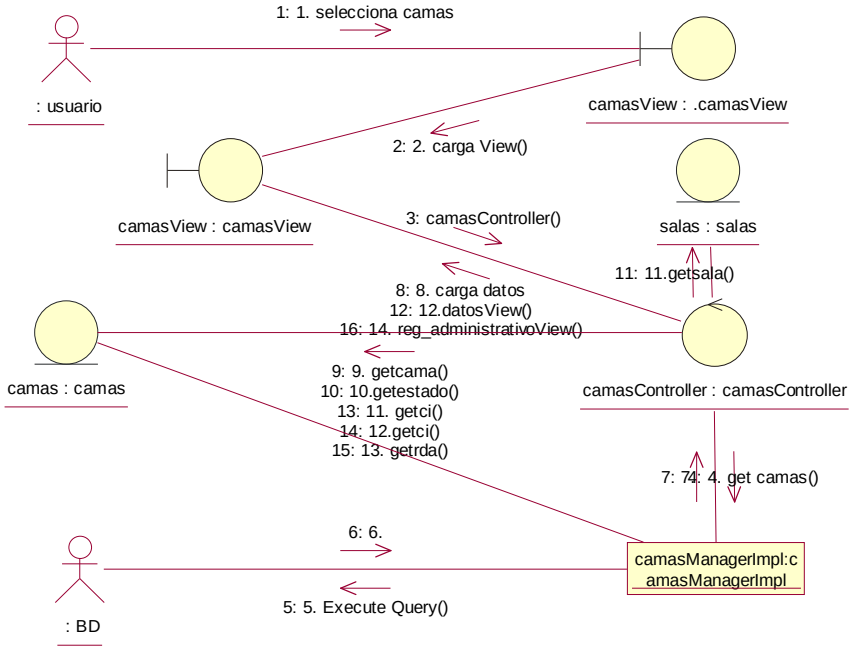


Figura 128. D.C. Camas

II.1.8.3.9 Adicionar camas

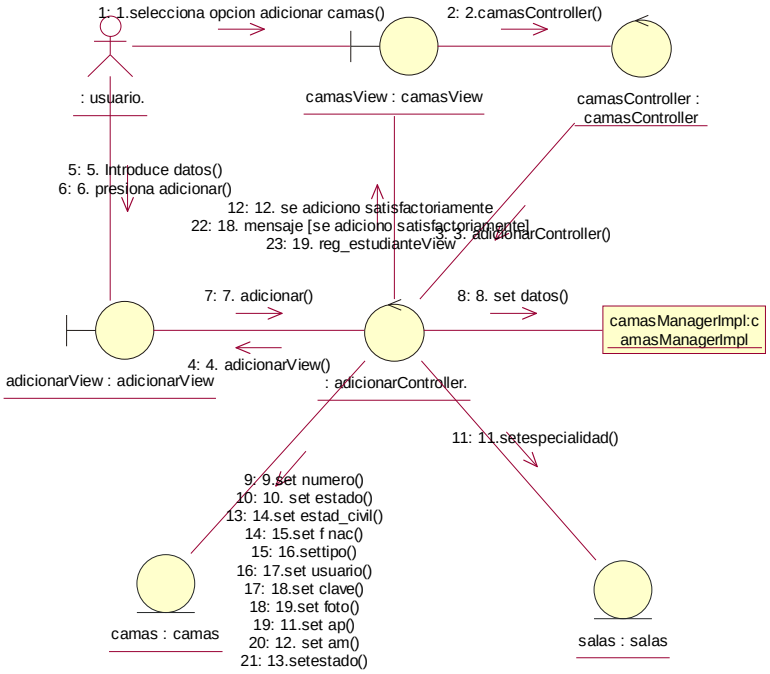


Figura 129. D.C. Adicionar camas

II.1.8.3.10 Modificar camas

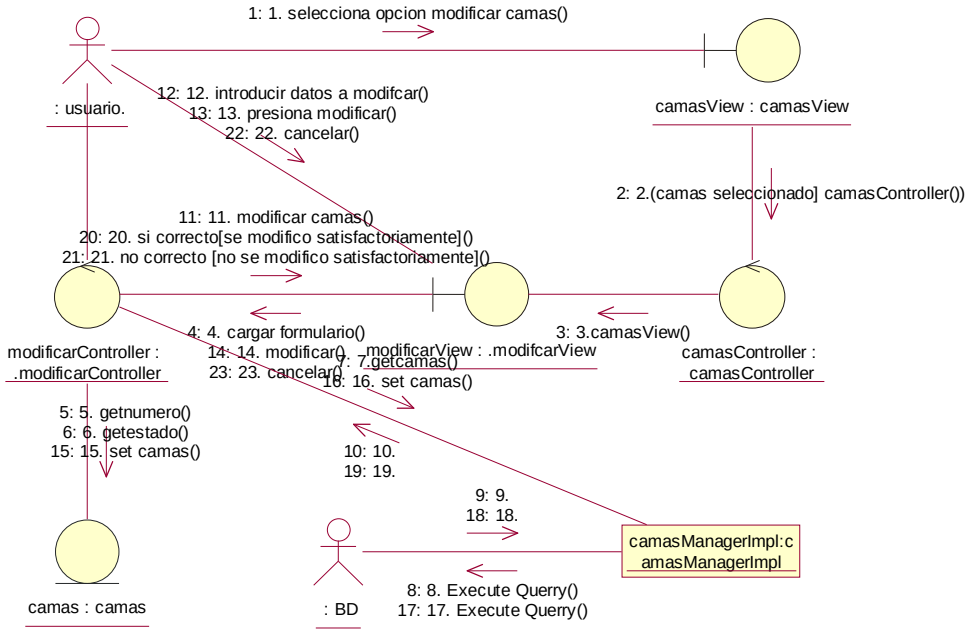


Figura 130. D.C. Modificar camas

II.1.8.3.11 Obtener Estado (cama)

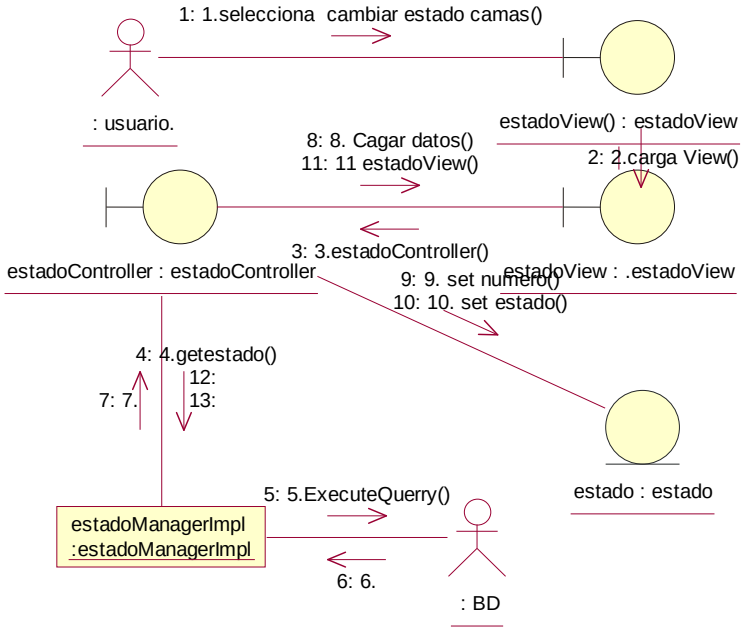


Figura 131. D.C. Obtener Estado (cama)

II.1.8.3.12 Especialidad

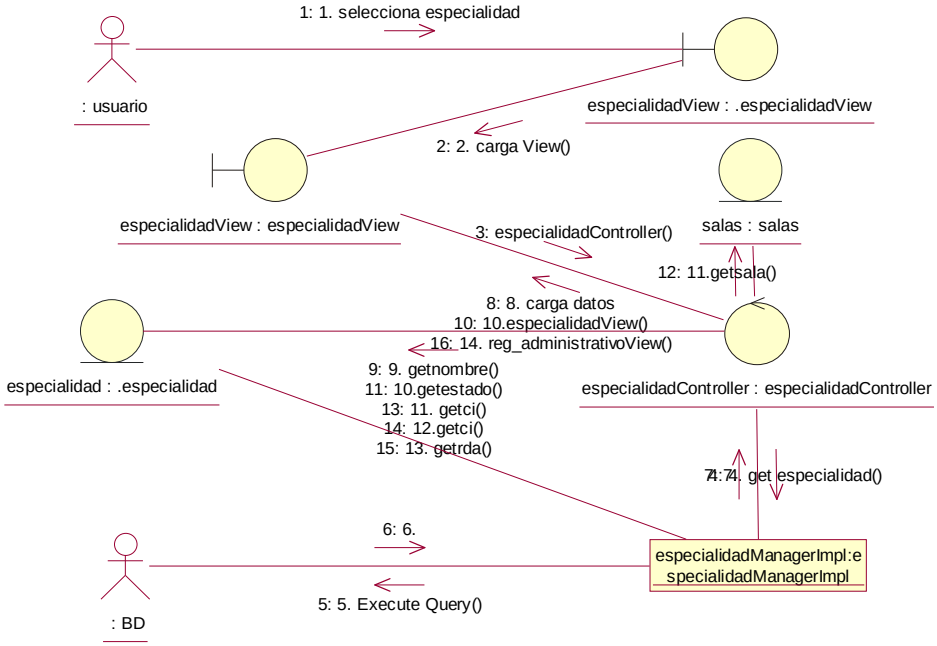


Figura 132. D.C. Especialidad

II.1.8.3.13 Adicionar especialidad

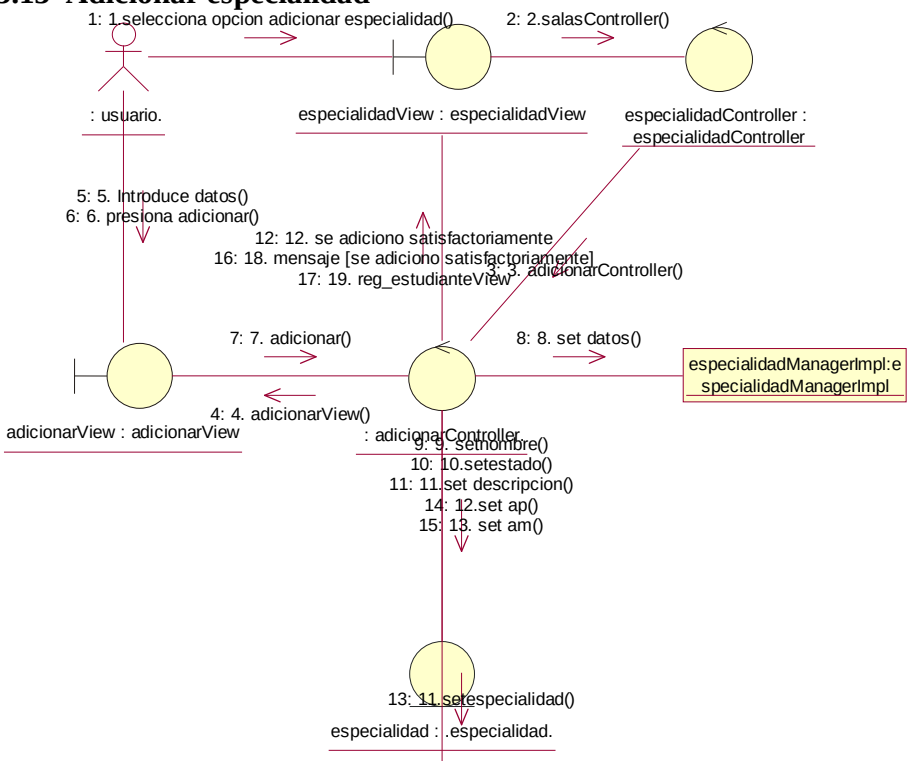


Figura 133. D.C. Adicionar especialidad

II.1.8.3.14 Modificar especialidad

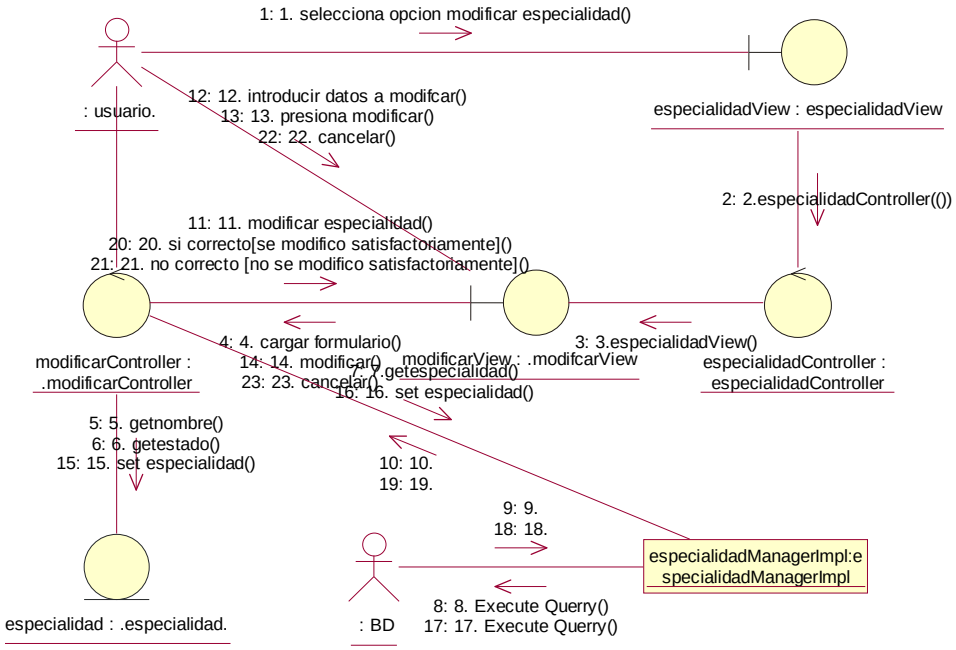


Figura 134. D.C. Modificar especialidad

II.1.8.3.15 Obtener Estado (especialidad)

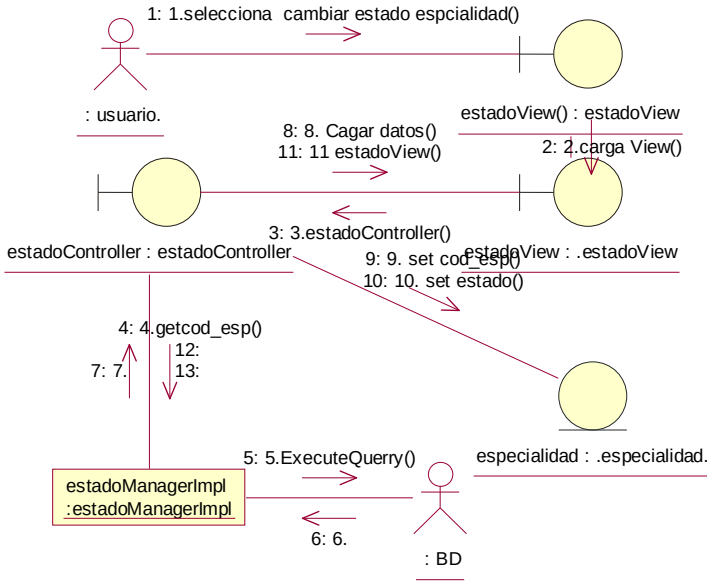


Figura 135. D.C. Obtener Estado (especialidad)

II.1.8.3.16 Personal

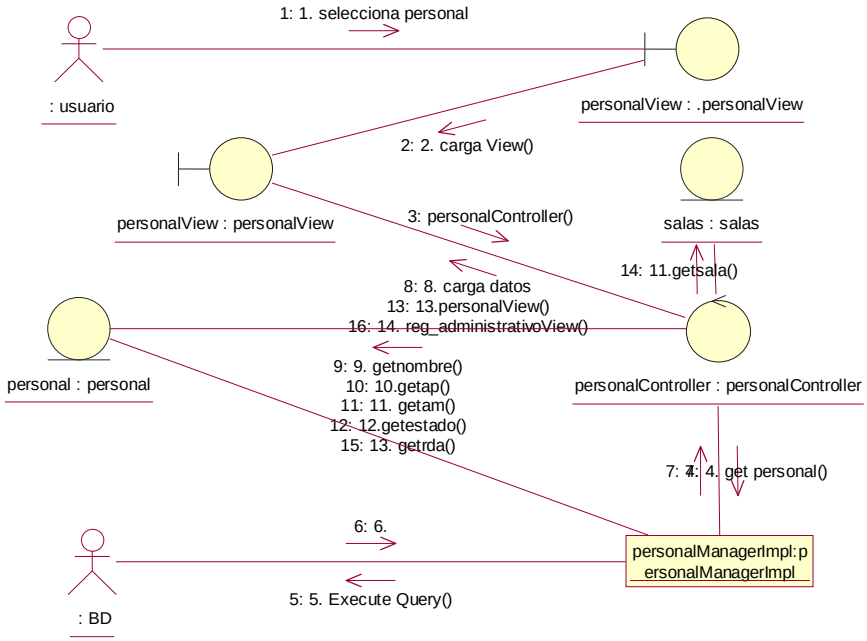


Figura 136. D.C. Personal

II.1.8.3.17 Adicionar personal

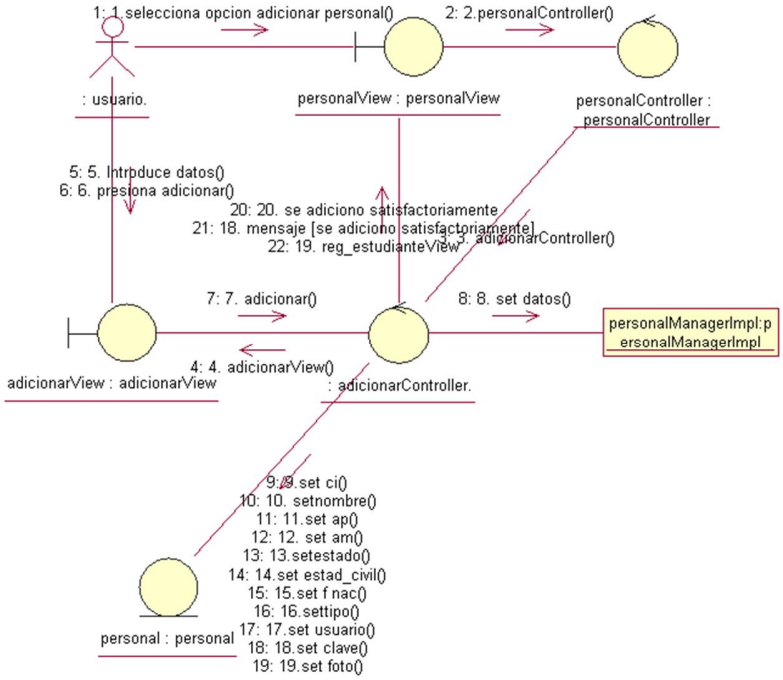


Figura 137. D.C. Adicionar personal

II.1.8.3.18 Modificar personal

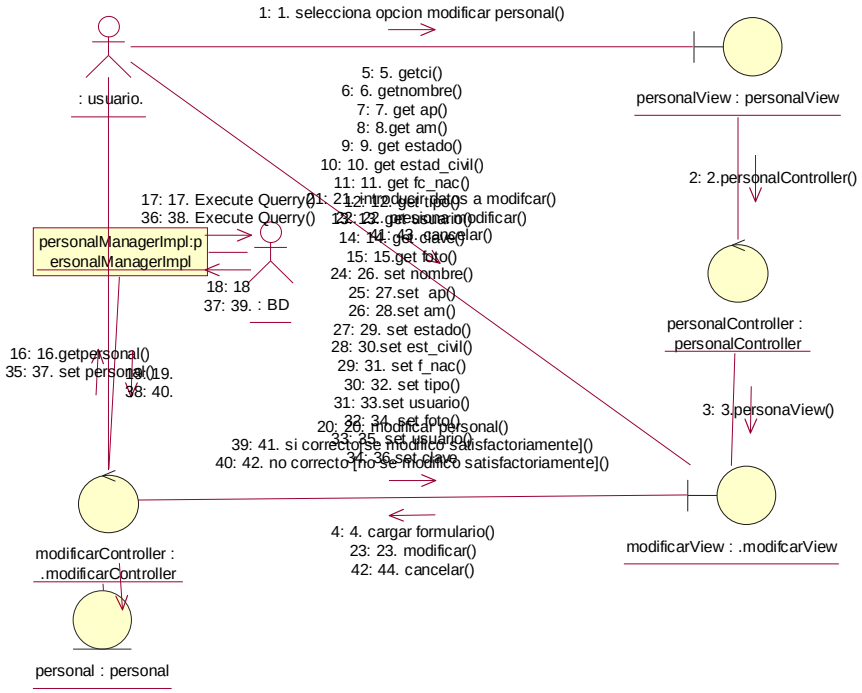


Figura 138. D.C. Modificar personal

II.1.8.3.19 Obtener Estado (personal)

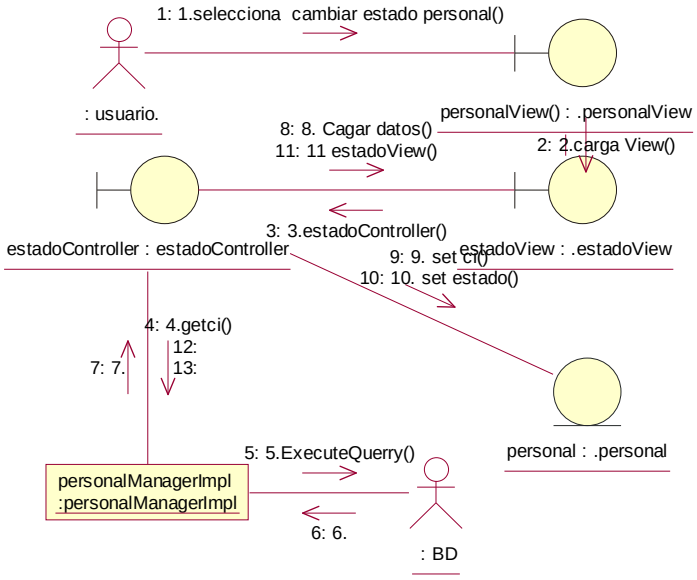


Figura 139. D.C. Obtener Estado (personal)

II.1.8.3.20 Salas

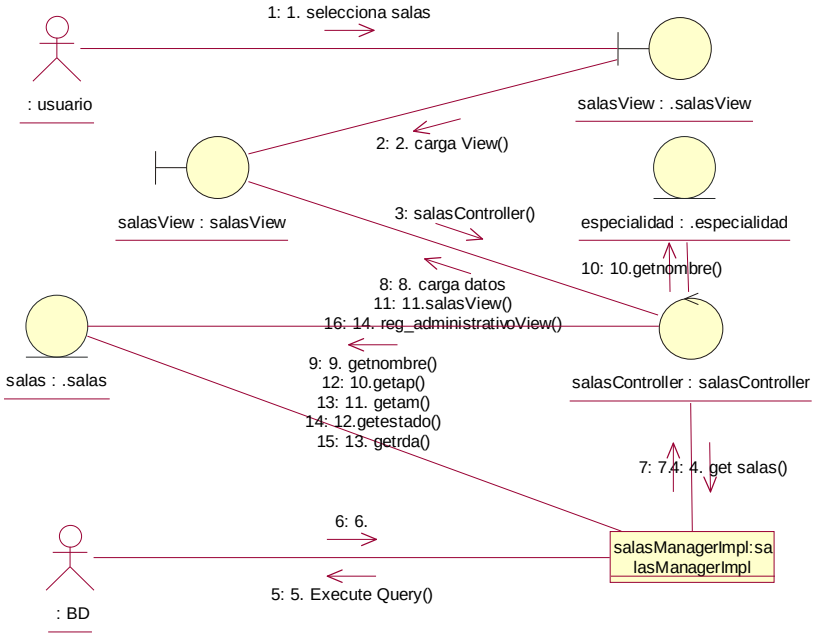


Figura 140. D.C. Salas

II.1.8.3.21 Adicionar salas

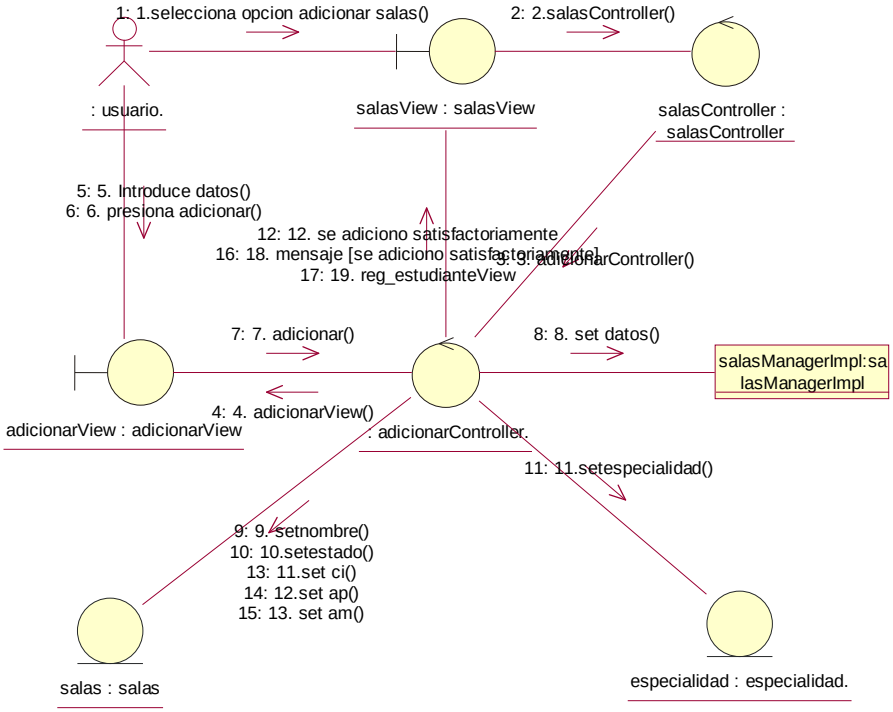


Figura 141. D.C. Adicionar salas

II.1.8.3.22 Modificar salas

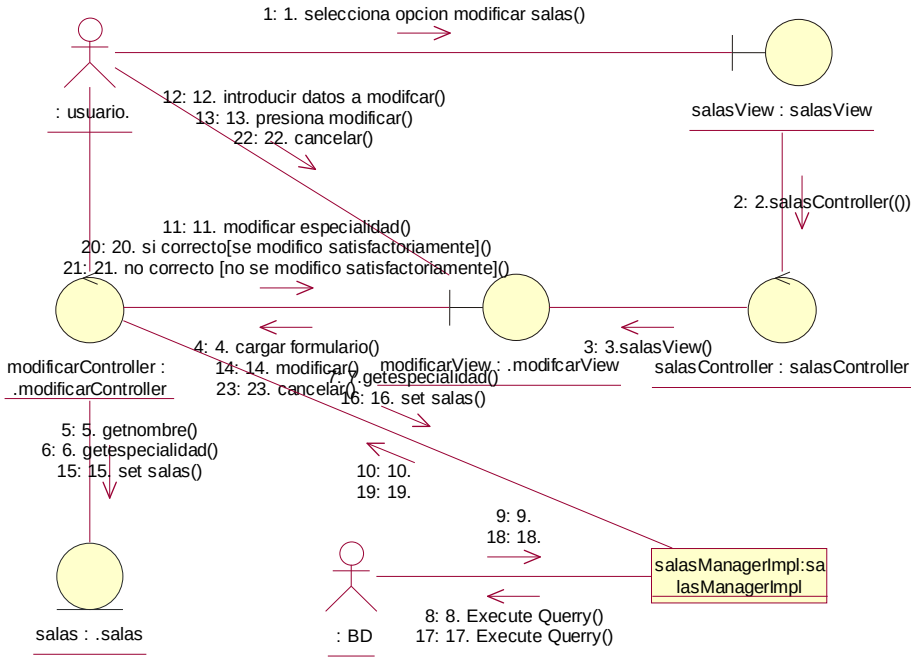


Figura 142. D.C. Modificar salas

II.1.8.3.23 Obtener Estado (sala)

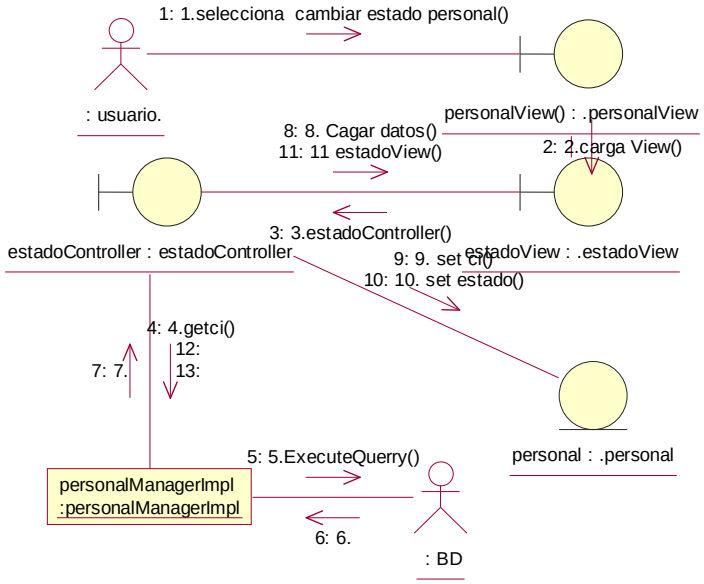


Figura 143. D.C. Obtener Estado (sala)

II.1.8.3.24 Modulo Ingreso

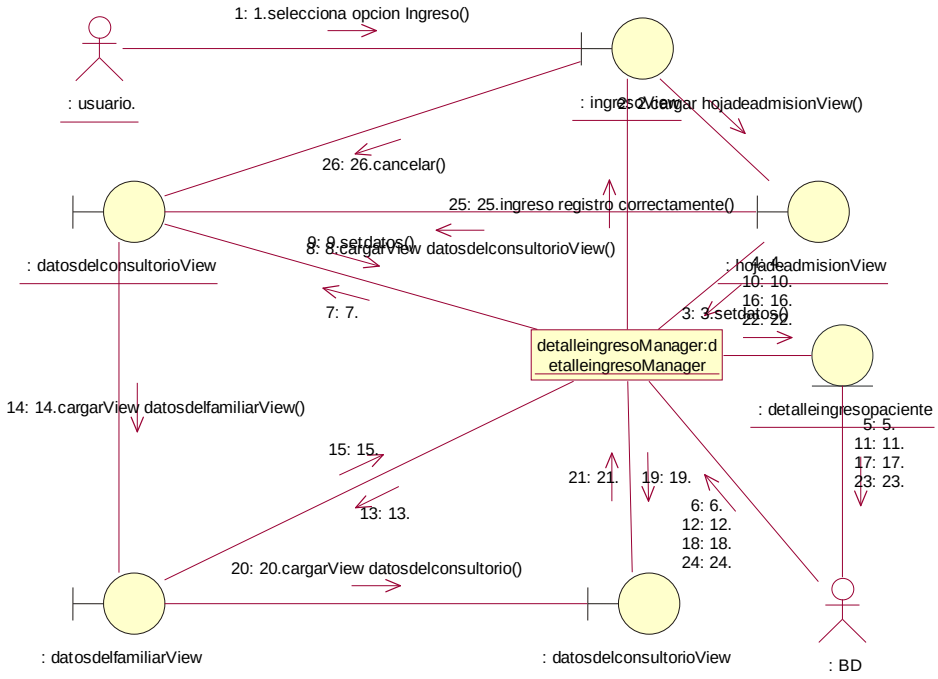


Figura 144. D.C. Modulo Ingreso

II.1.8.3.25 Modulo Egreso

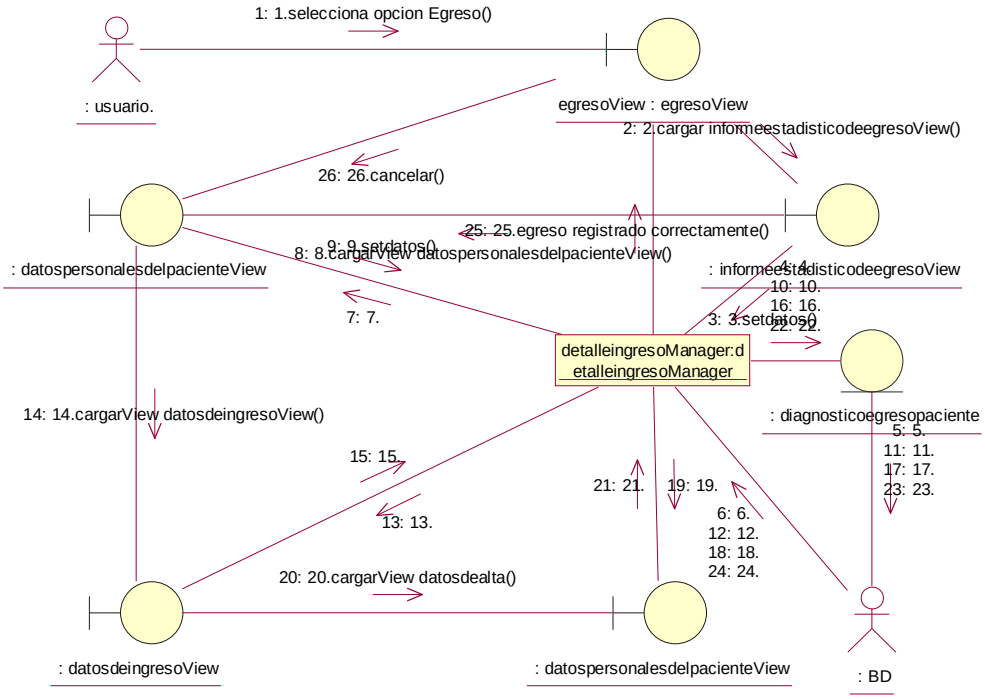


Figura 145. D.C. Modulo Egreso

II.1.8.3.26 Modulo Servicio

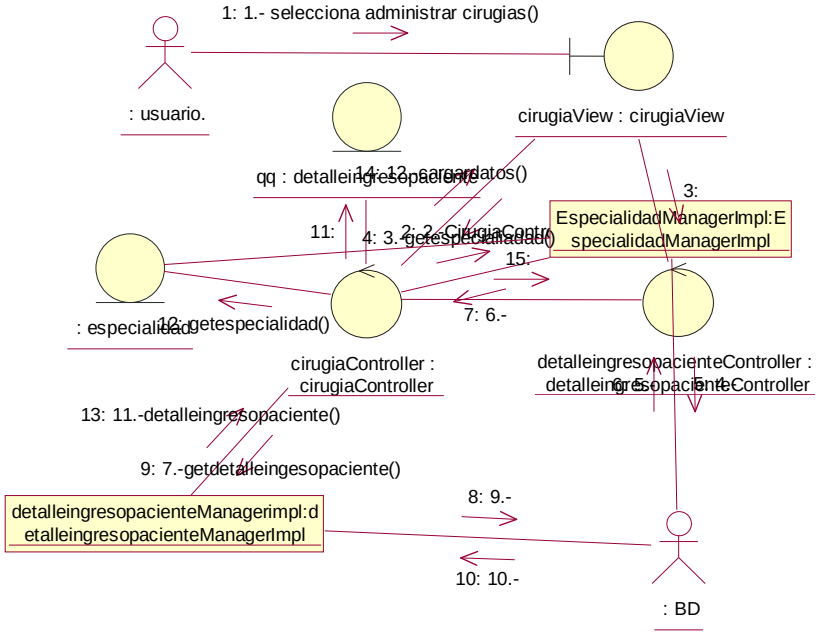


Figura 146. D.C. Modulo Servicio

II.1.8.3.27 Cirugía

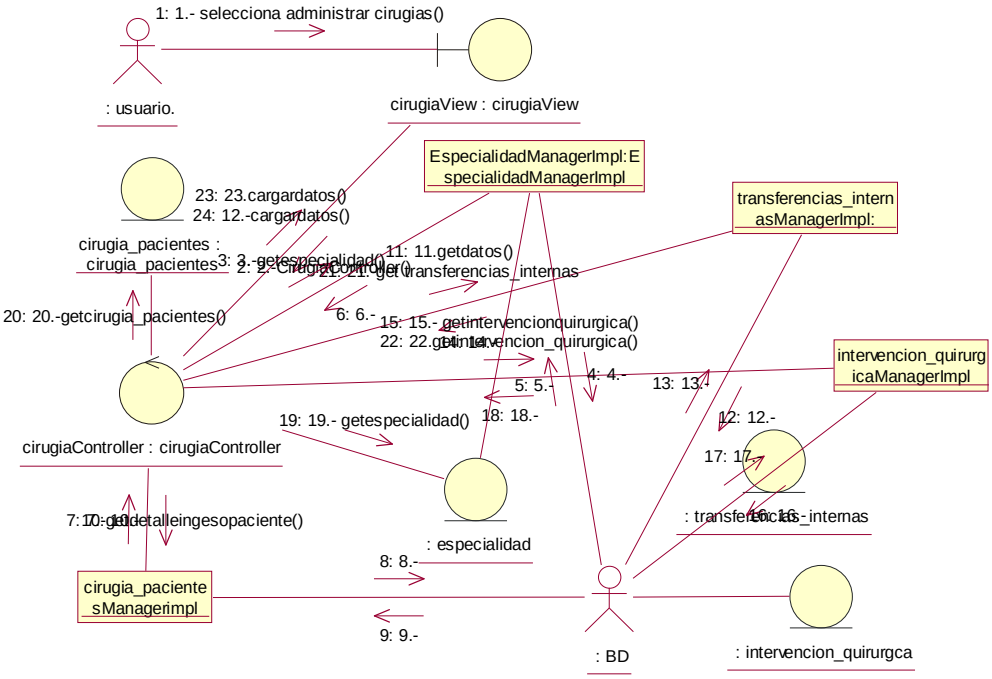


Figura 147. D.C. Cirugía

II.1.8.3.28 Maternidad

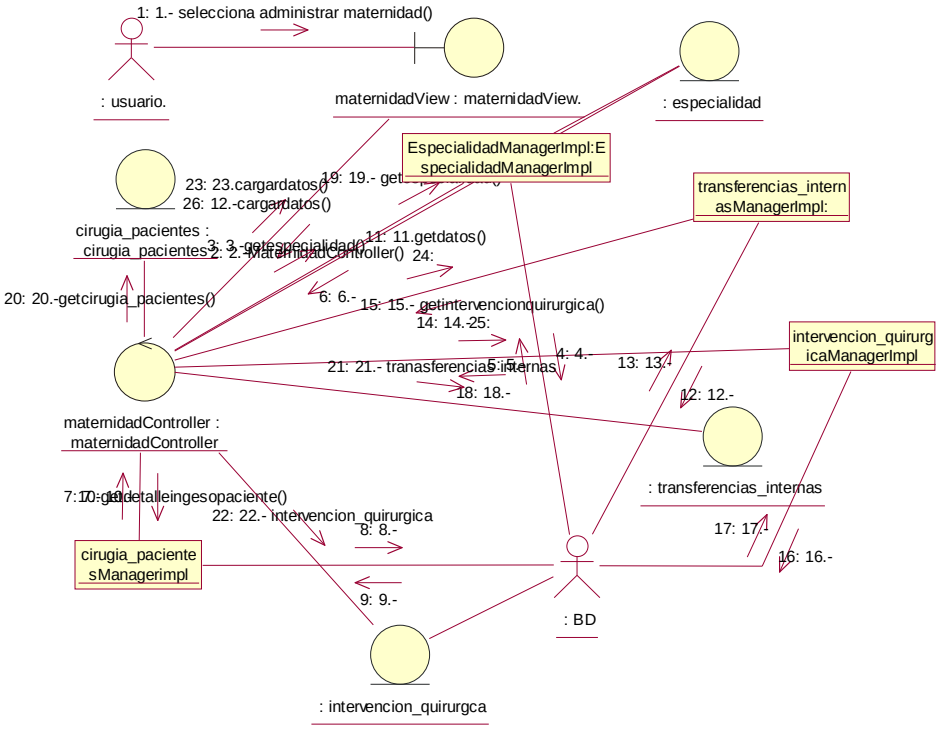


Figura 148. D.C. Maternidad

II.1.8.3.29 Medicina interna

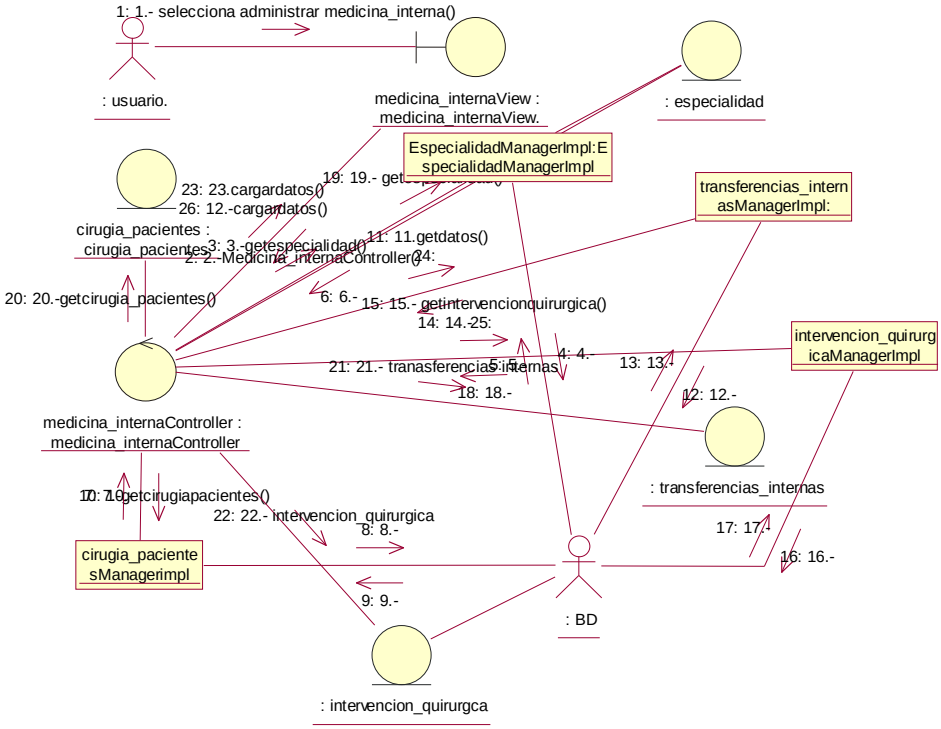


Figura 149. D.C. Medicina interna

II.1.8.3.30 Pediatria

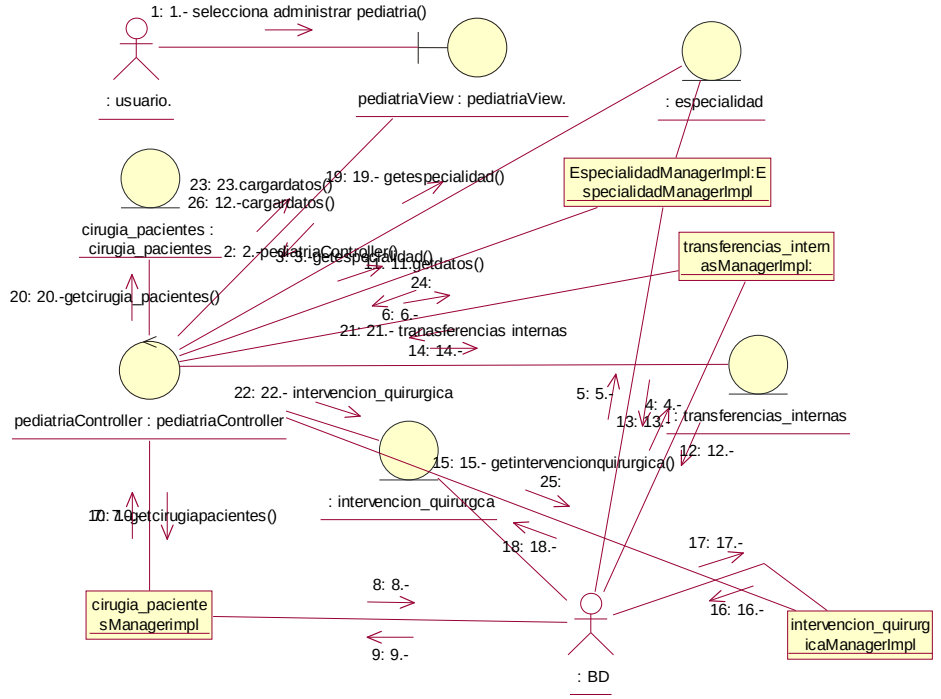


Figura 150. D.C. Pediatria

II.1.8.3.31 UTI

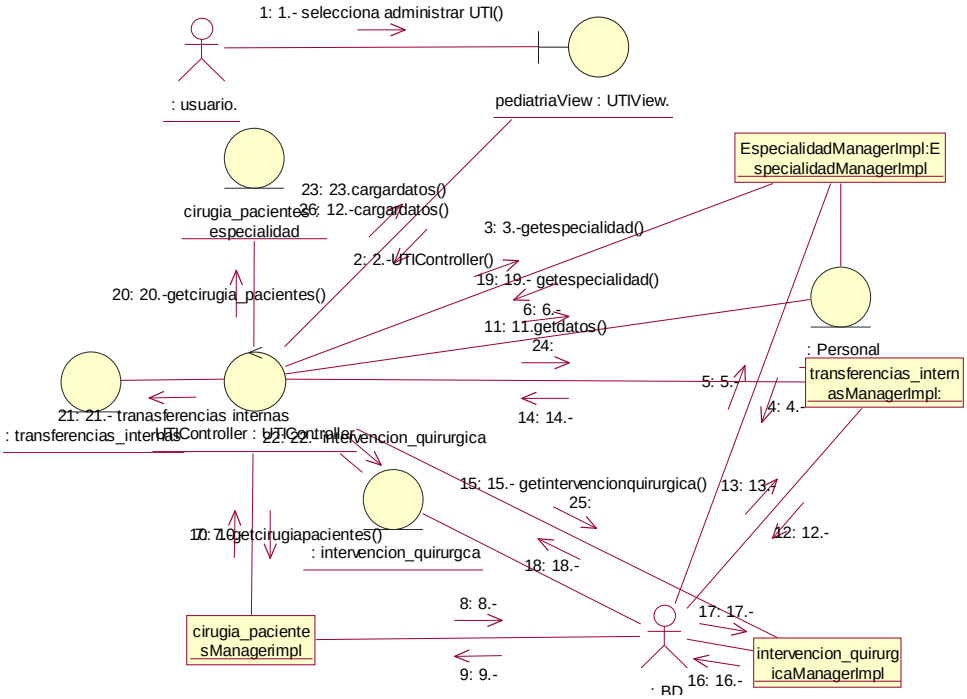


Figura 151 D.C. UTI

II.1.8.3.32 Reportes

II.1.8.3.33 Ver Hospitalización

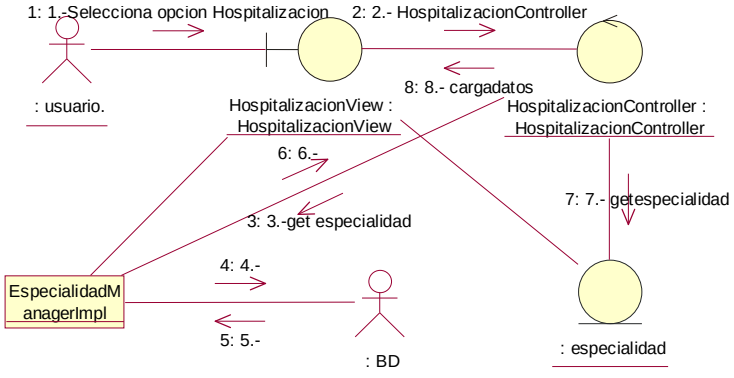


Figura 153. D.C. Ver Hospitalización

II.1.8.3.34 Ver Intervencion quirurgica por especialidad

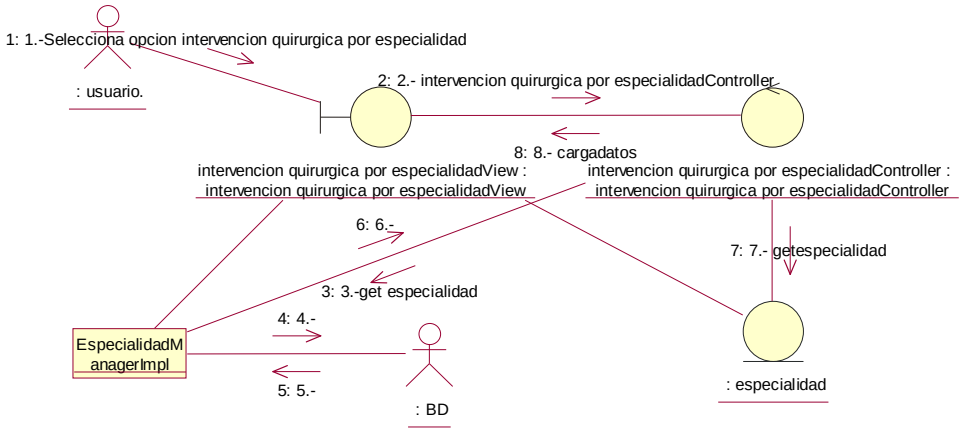


Figura 154. D.C. Ver Intervencion quirurgica por especialidad

II.1.8.3.35 Ver Intervencion quirurgica por clase de cirugía

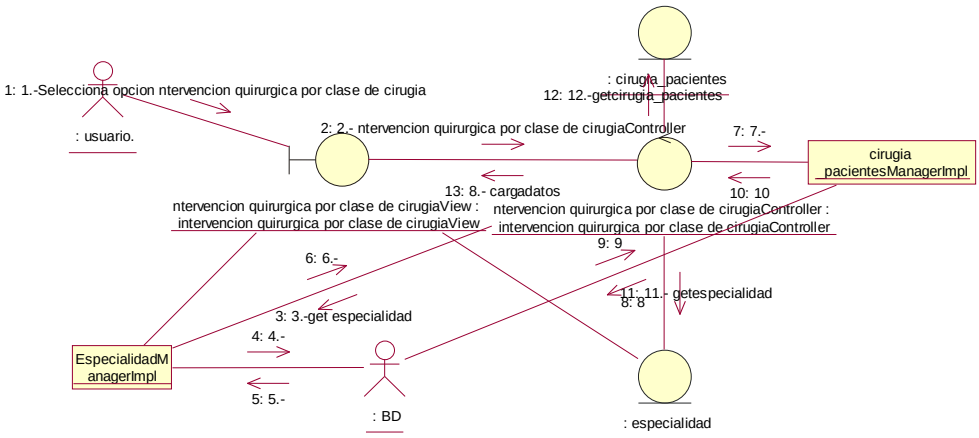


Figura 155. D.C. Ver Intervencion quirurgica por clase de cirugia

II.1.8.3.36 Ver Mensual hospitalario

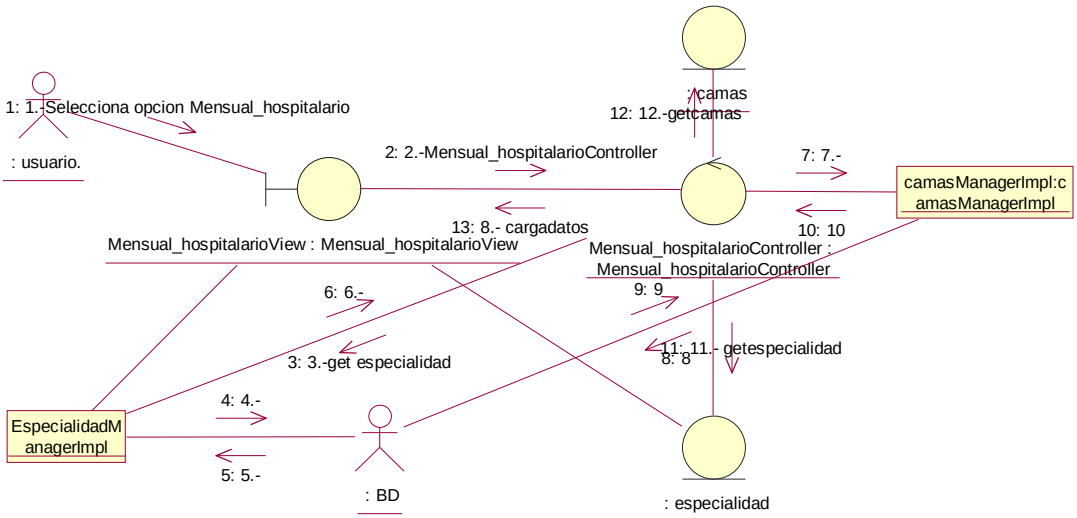


Figura 156. D.C. Ver Mensual hospitalario

II.1.8.3.37 Ver Mensual por especialidad

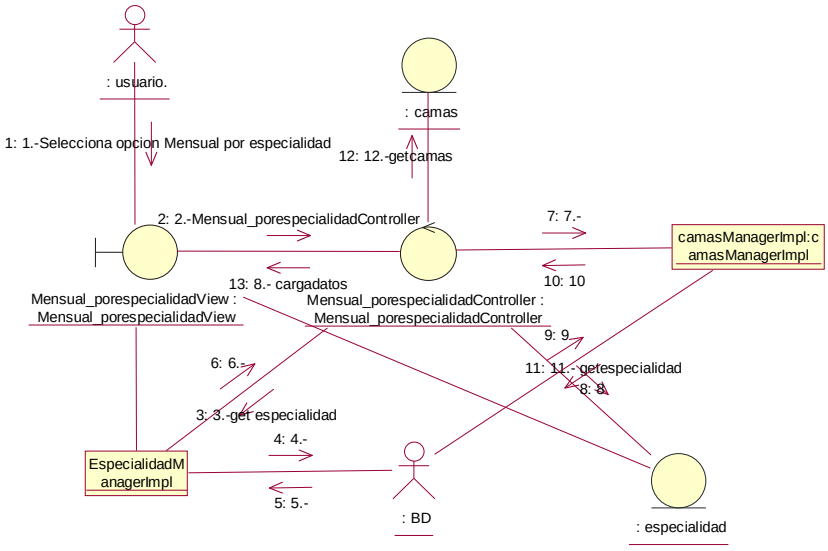


Figura 157. D.C. Ver Mensual por especialidad

II.1.8.3.38 Ver Partos_nacimientos

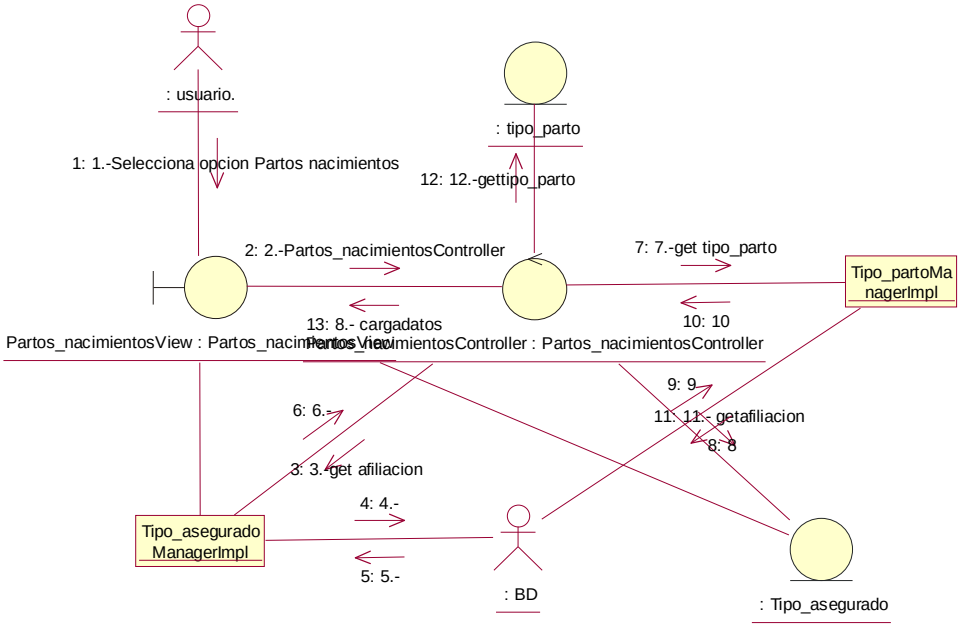


Figura 158 D.C. Ver Partos_nacimientos

II.1.8.3.39 Ver Cirugias programadas

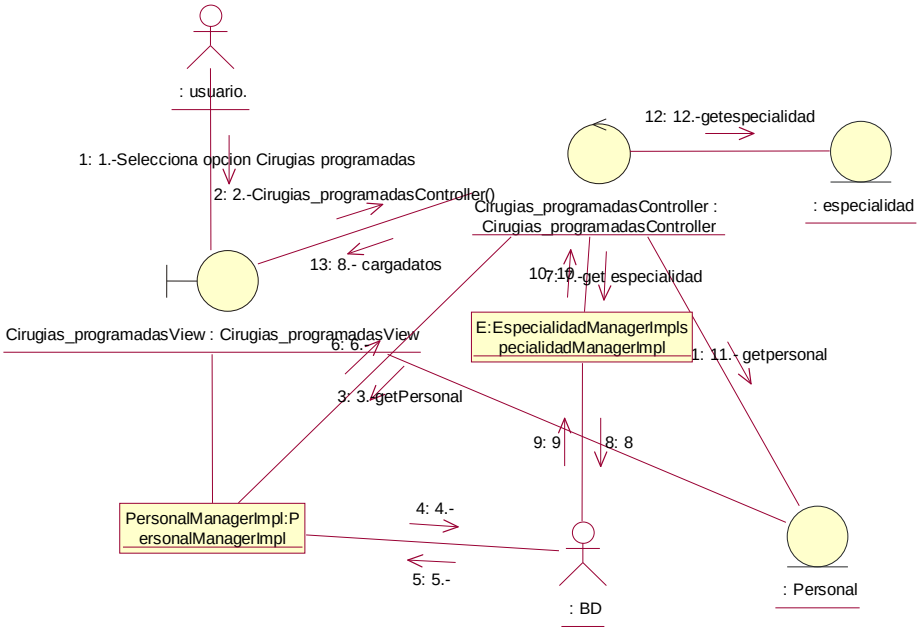


Figura 159. D.C. Ver Cirugias programadas

II.1.8.3.40 Ver Cirugias de emergencia

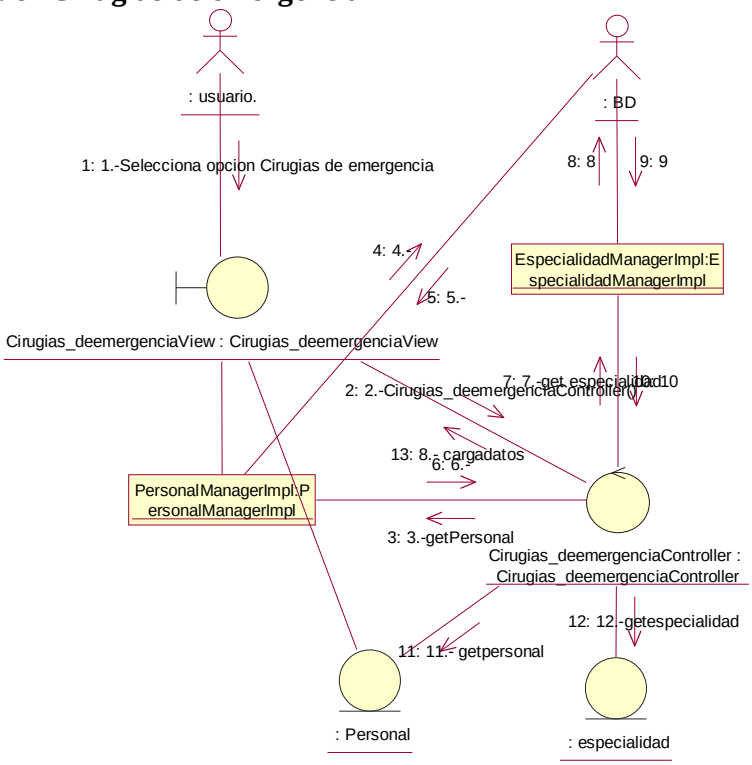


Figura 160. D.C. Ver Cirugias de emergencia

II.1.8.3.41 Ver Partos atendidos por obstetras

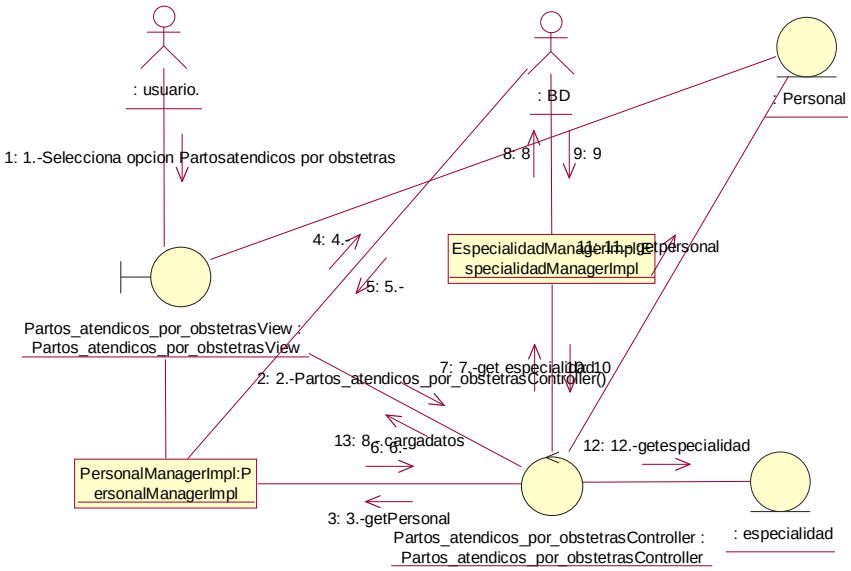


Figura 161. D.C. Ver Partos atendidos por obstetras

II.1.8.3.42 Ayuda

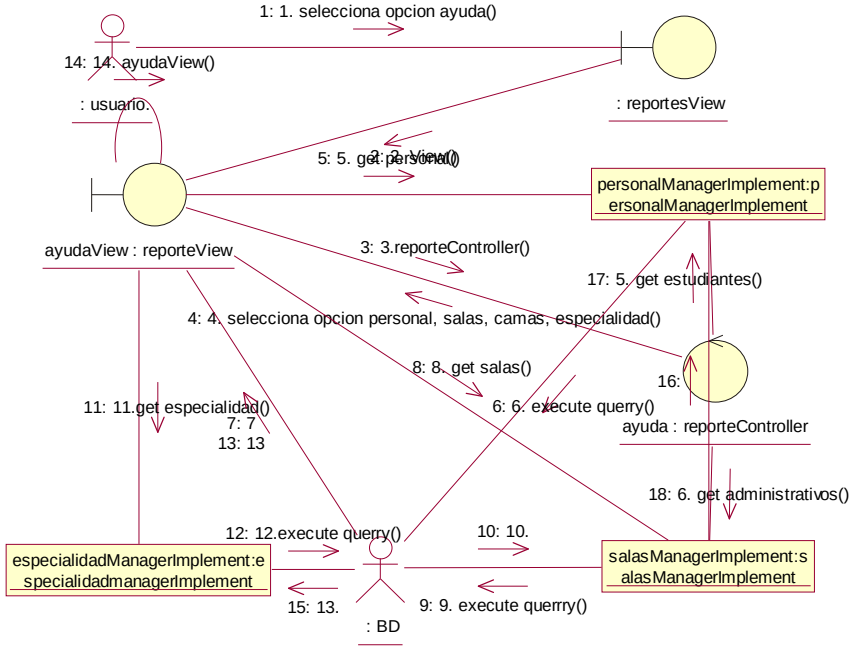


Figura 162. D.C. Ayuda

II.1.9 MODELO DE DATOS

II.1.9.1 Modelado de Diagramas de clases

II.1.9.1.1 Introducción

El modelado de Diagrama de Clases en un artefacto de la disciplina de Análisis y Diseño en la Metodología RUP que se está implementando

II.1.9.1.1.1 Propósito

- Comprender la estructura del sistema deseado para el Hospital Obrero
- Identificar clases de análisis y diseño.

II.1.9.1.1.2 Alcance

- Describir las clases y objetos de diseño del sistema en su segunda iteración
- Identificar y definir los objetos del sistema según los objetos del sistema deseado, aprobado administrativos del Hospital Obrero

II.1.9.1.2 DIAGRAMAS DE CLASES

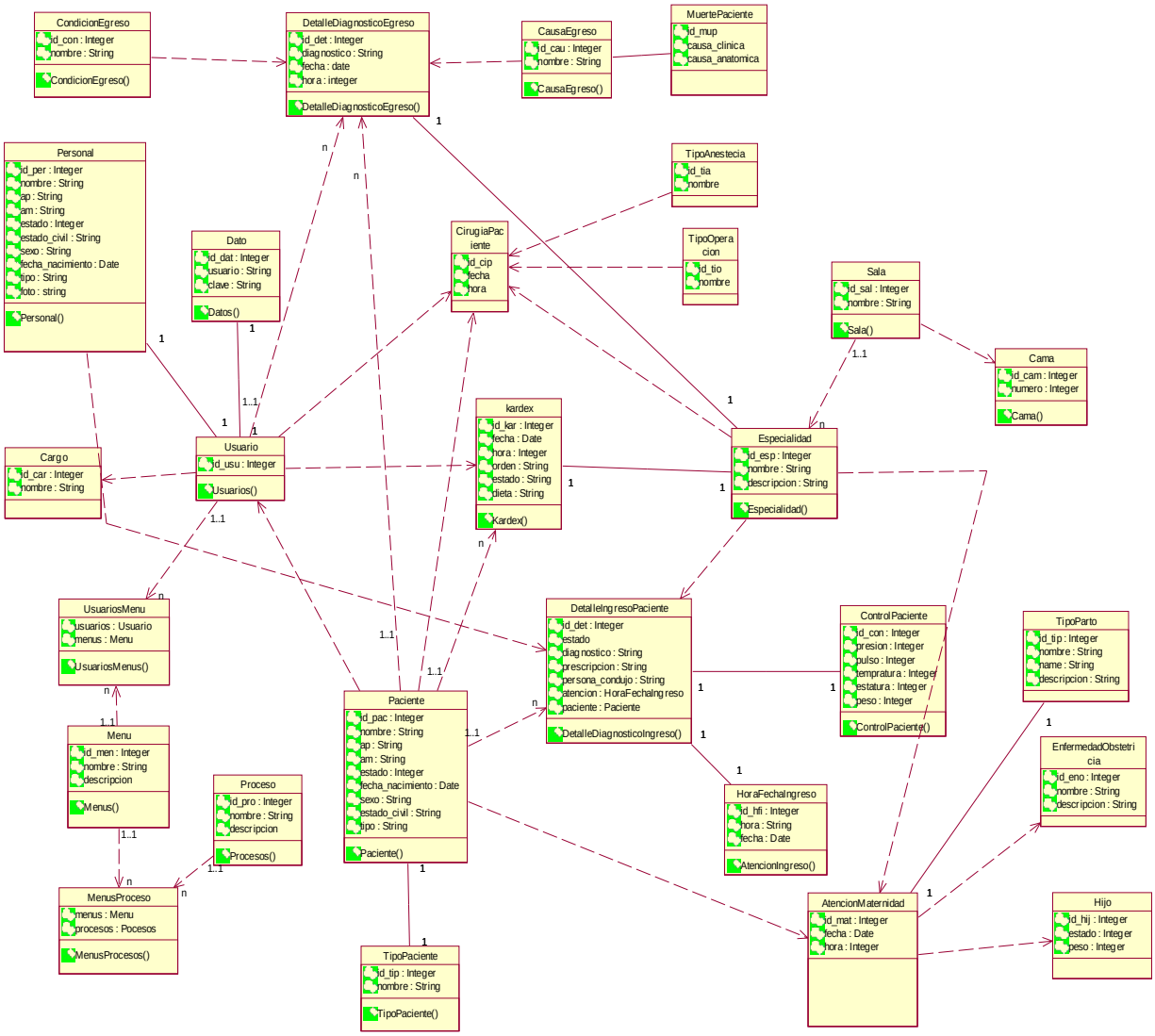
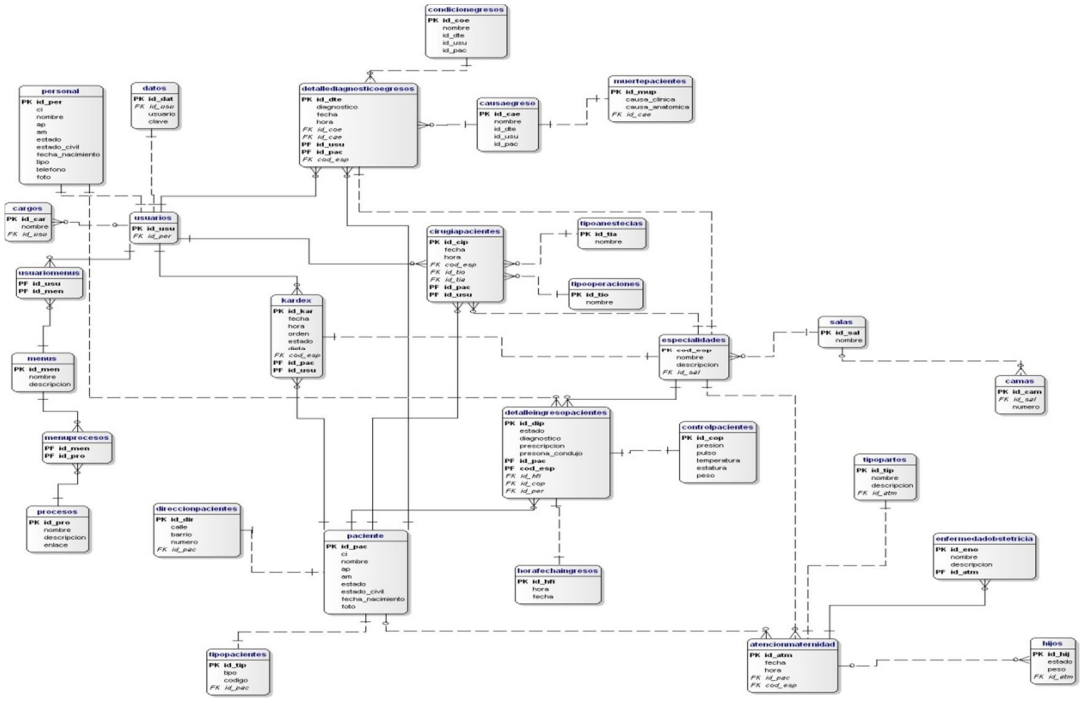


Figura 163. DIAGRAMAS DE CLASES


condicionegresos
PK id_coe
nombre
id_dte
id_usu
id_pac
personal
PK id_per
ci
nombre
ap
am
estado
estado_civil
fecha_nacimiento
tipo
telefono
foto
datos
PK id_dat
FK id_usu
usuario
clave
detallediagnosticoegresos
PK id_dte
diagnostico
fecha
hora
FK id_coe
FK id_cae
PF id_usu
PF id_pac
FK cod_esp
causaegreso
PK id_cae
nombre
id_dte
id_usu
id_pac
muertepacientes
PK id_mep
causa_clinica
causa_anatomica
FK id_cae
cargos
PK id_car
nombre
FK id_usu
usuarios
PK id_usu
FK id_per
cirugiapacientes
PK id_cip
fecha
hora
FK cod_esp
FK id_tio
FK id_tia
PF id_pac
PF id_usu
tipoanestecias
PK id_tia
nombre
tipooperaciones
PK id_tio
nombre
usuariomenus
PF id_usu
PF id_men
kardex
PK id_kar
fecha
hora
orden
estado
dieta
FK cod_esp
PF id_pac
PF id_usu
especialidades
PK cod_esp
nombre
descripcion
FK id_sal
salas
PK id_sal
nombre
camas
PK id_cam
FK id_sal
numero
menus
PK id_men
nombre
descripcion
detalleingresopacientes
PK id_dip
estado
diagnostico
prescripcion
persona_condujo
PF id_pac
PF cod_esp
FK id_hfi
FK id_cop
FK id_per
controlpacientes
PK id_cop
presion
pulso
temperatura
estatura
peso
tipopartos
PK id_tip
nombre
descripcion
FK id_atm
menuprocesos
PF id_men
PF id_pro
procesos
PK id_pro
nombre
descripcion
enlace
direccionpacientes
PK id_dir
calle
barrio
numero
FK id_pac
paciente
PK id_pac
ci
nombre
ap
am
estado
estado_civil
fecha_nacimiento
foto
horafechaingresos
PK id_hfi
hora
fecha
enfermedadobstetricia
PK id_eno
nombre
descripcion
PF id_atm
tipopacientes
PK id_tip
tipo
codigo
FK id_pac
atencionmaternidad
PK id_atm
fecha
hora
FK id_pac
FK cod_esp
hijos
PK id_hij
estado
peso
FK id_atm

II.1.9.2 MODELO DE DATOS

II.1.9.2.1 Introducción

El modelado de Datos nos sirve para tener un detalle de las tablas con sus respectivos campos de la base de datos

II.1.9.2.1.1 Propósito

- Comprender las estructuras de las tablas y sus campos, en la base de datos de nuestro sistema deseado para los establecimientos educativos
- Identificar los tipos de campos de cada tabla de la base de datos

II.1.9.2.1.2 Alcance

- Describir los campos de cada tabla de la base de datos especificando el tipo, longitud y descripción de cada campo.
- Identificar y definir relaciones entre las diferentes tablas de la base de datos de nuestro sistema deseado para el hospital Obrero

II.1.9.2.2 Scrip de la Base de Datos

II.1.9.2.2.1 BASE DE DATOS CNS

II.1.9.2.2.1.1 TABLE atención maternidad

```
CREATE TABLE atencionmaternidad
(  id_atm integer NOT NULL,
  fecha date,
  hora time without time zone,
  cod_persona integer,
  id_eno integer,
  id_tip integer,
  CONSTRAINT pk_atencionmaternidad PRIMARY KEY (id_atm),
```

CONSTRAINT enfermedadobstetricia_atencionmaternidad FOREIGN KEY (id_eno, id_atm)

REFERENCES enfermedadobstetricia (id_eno, id_atm) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT tipopartos_atencionmaternidad FOREIGN KEY (id_tip)

REFERENCES tipopartos (id_tip) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION);

II.1.9.2.2.1.2 TABLE camas

CREATE TABLE camas

(id_cam serial NOT NULL,

id_sal integer,

numero integer,

estado integer,

CONSTRAINT pk_camas PRIMARY KEY (id_cam),

CONSTRAINT camas_id_sal_fkey FOREIGN KEY (id_sal)

REFERENCES salas (id_sal) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE)

II.1.9.2.2.1.3 TABLE cargos

CREATE TABLE cargos

(id_car serial NOT NULL,

nombre character varying(40),

id_usu integer,

CONSTRAINT pk_cargos PRIMARY KEY (id_car),

CONSTRAINT usuarios_cargos FOREIGN KEY (id_usu)

REFERENCES usuarios (id_usu) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE)

II.1.9.2.2.1.4 TABLE causa_anatomicas

```
CREATE TABLE causa_anatomicas
(  id_caa serial NOT NULL,
  nombre character varying(100),
  id_cae integer,
  CONSTRAINT causa_anatomicas_pkey PRIMARY KEY (id_caa),
  CONSTRAINT causa_anatomicas_id_cae_fkey FOREIGN KEY (id_cae)
      REFERENCES causa_egresos (id_cae) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE )
```

II.1.9.2.2.1.5 TABLE causa_clinicas

```
CREATE TABLE causa_clinicas
(  id_cac serial NOT NULL,
  nombre character varying(100),
  id_cae integer,
  CONSTRAINT causa_clinicas_pkey PRIMARY KEY (id_cac),
  CONSTRAINT causa_clinicas_id_cae_fkey FOREIGN KEY (id_cae)
      REFERENCES causa_egresos (id_cae) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE )
```

II.1.9.2.2.1.6 TABLE causa_egresos

```
CREATE TABLE causa_egresos
( id_cae integer NOT NULL DEFAULT nextval('causaegreso_id_cae_seq'::regclass),
```

```
  nombre character varying(40),
  CONSTRAINT pk_causaegreso PRIMARY KEY (id_cae) )
```

II.1.9.2.2.1.7 TABLE cirugiapacientes

```
CREATE TABLE cirugiapacientes
(  id_cip serial NOT NULL,
  fecha date,
  hora time without time zone,
  id_esp integer NOT NULL,
  id_tio integer NOT NULL,
  id_tia integer NOT NULL,
  id_usu integer NOT NULL,
  cod_persona integer NOT NULL,
  CONSTRAINT pk_cirugiapacientes PRIMARY KEY (id_cip, id_usu,
cod_persona),
  CONSTRAINT especialidades_cirugiapacientes FOREIGN KEY (id_esp)
     REFERENCES especialidades (id_esp) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT tipoanestecias_cirugiapacientes FOREIGN KEY (id_tia)
     REFERENCES tipoanestecias (id_tia) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT tipooperaciones_cirugiapacientes FOREIGN KEY (id_tio)
     REFERENCES tipooperaciones (id_tio) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT usuarios_cirugiapacientes FOREIGN KEY (id_usu)
```

REFERENCES usuarios (id_usu) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.1.8 TABLE condicion_egresos

CREATE TABLE condicion_egresos

(id_coe integer NOT NULL DEFAULT nextval('condicionegresos_id_coe_seq'::regclass),

nombre character varying(40),

CONSTRAINT pk_condicionegresos PRIMARY KEY (id_coe))

II.1.9.2.2.1.9 TABLE controlpacientes

CREATE TABLE controlpacientes

(id_cop integer NOT NULL DEFAULT nextval('controlpacientes_id_cop_seq1'::regclass),

peso integer,

estatura integer,

temperatura integer,

pulso integer,

presion integer,

CONSTRAINT pk_controlpacientes PRIMARY KEY (id_cop))

II.1.9.2.2.1.10 TABLE datos

CREATE TABLE datos

(id_dat serial NOT NULL,

id_usu integer NOT NULL,

usuario character varying(40),

clave character varying(40),

```
  CONSTRAINT pk_datos PRIMARY KEY (id_dat),
  CONSTRAINT usuarios_datos FOREIGN KEY (id_usu)
      REFERENCES usuarios (id_usu) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE )
```

II.1.9.2.2.1.11 TABLE detalle_egreso_pacientes

```
CREATE TABLE detalle_egreso_pacientes
(  id_dep integer NOT NULL,
  id_coe integer NOT NULL,
  id_cae integer NOT NULL,
  id_usu integer NOT NULL,
  cod_persona integer NOT NULL,
  id_tid integer NOT NULL,
  id_hfi integer NOT NULL,
  id_cam integer NOT NULL,
  cod_beneficiario integer,
  CONSTRAINT detalle_egreso_pacientes_pkey PRIMARY KEY (id_dep, id_usu),
  CONSTRAINT detalle_egreso_pacientes_id_cae_fkey FOREIGN KEY (id_cae)
      REFERENCES causa_egresos (id_cae) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT detalle_egreso_pacientes_id_cam_fkey FOREIGN KEY (id_cam)
      REFERENCES camas (id_cam) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT detalle_egreso_pacientes_id_coe_fkey FOREIGN KEY (id_coe)
```

REFERENCES condicion_egresos (id_coe) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT detalle_egreso_pacientes_id_hfi_fkey FOREIGN KEY (id_hfi)

REFERENCES horafechaingresos (id_hfi) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT detalle_egreso_pacientes_id_tid_fkey FOREIGN KEY (id_tid)

REFERENCES tipo_diagnosticos (id_tid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT detalle_egreso_pacientes_id_usu_fkey FOREIGN KEY (id_usu)

REFERENCES usuarios (id_usu) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.1.12 TABLE detallar ingresopacientes

CREATE TABLE detalleingresopacientes

(id_dip integer NOT NULL DEFAULT nextval('detalleingresopacientes_id_dip_seq1'::regclass),

id_hfi integer NOT NULL,

id_per integer NOT NULL,

id_tie integer NOT NULL,

id_tid integer NOT NULL,

id_tip integer NOT NULL,

id_cam integer NOT NULL,

id_cop integer NOT NULL,

cod_persona integer NOT NULL,

persona_condujo character varying(100),

```
cod_beneficiario integer,
tipo character varying,
id_esp integer,
CONSTRAINT detalleingresopacientes_pkey PRIMARY KEY (id_dip),
CONSTRAINT detalleingresopacientes_id_cam_fkey FOREIGN KEY (id_cam)
   REFERENCES camas (id_cam) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT detalleingresopacientes_id_cop_fkey FOREIGN KEY (id_cop)
   REFERENCES controlpacientes (id_cop) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT detalleingresopacientes_id_esp_fkey FOREIGN KEY (id_esp)
   REFERENCES especialidades (id_esp) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT detalleingresopacientes_id_hfi_fkey FOREIGN KEY (id_hfi)
   REFERENCES horafechaingresos (id_hfi) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT detalleingresopacientes_id_per_fkey FOREIGN KEY (id_per)
   REFERENCES personal (id_per) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT detalleingresopacientes_id_tid_fkey FOREIGN KEY (id_tid)
   REFERENCES tipo_diagnosticos (id_tid) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT detalleingresopacientes_id_tie_fkey FOREIGN KEY (id_tie)
```

```
    REFERENCES tipo_estados (id_tie) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT detalleingresopacientes_id_tip_fkey FOREIGN KEY (id_tip)
    REFERENCES tipo_prescripciones (id_tip) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION )
```

II.1.9.2.2.1.13 TABLE dietas

```
CREATE TABLE dietas
(  id_die serial NOT NULL,  nombre character varying(100),
  CONSTRAINT pk_dietas PRIMARY KEY (id_die) )
```

II.1.9.2.2.1.14 TABLE direccion_personal

```
CREATE TABLE direccion_personal
(  id_dip serial NOT NULL,
  zona character varying(40),
  calle character varying(40),
  numero integer,  id_per integer,
  CONSTRAINT direccion_personal_pkey PRIMARY KEY (id_dip),
  CONSTRAINT direccion_personal_id_per_fkey FOREIGN KEY (id_per)
    REFERENCES personal (id_per) MATCH SIMPLE
    ON UPDATE CASCADE ON DELETE CASCADE )
```

II.1.9.2.2.1.15 TABLE enfermedadobstetricia

```
CREATE TABLE enfermedadobstetricia
(  id_eno integer NOT NULL,
  nombre character varying(40),
```

```
  descripcion character varying(40),
  id_atm integer NOT NULL,
  CONSTRAINT pk_enfermedadobstetricia PRIMARY KEY (id_eno, id_atm))
```

II.1.9.2.2.1.16 TABLE especialidades

```
CREATE TABLE especialidades
(  id_esp integer NOT NULL,
  nombre character varying(40),
  descripcion character varying(40),
  estado integer DEFAULT 1,
  id_ser integer,
  CONSTRAINT pk_especialidades PRIMARY KEY (id_esp),
  CONSTRAINT especialidades_id_ser_fkey FOREIGN KEY (id_ser)
     REFERENCES servicios (id_ser) MATCH SIMPLE
     ON UPDATE CASCADE ON DELETE CASCADE )
```

II.1.9.2.2.1.17 TABLE hijos

```
CREATE TABLE hijos
(  id_hij serial NOT NULL,
  estado integer,
  peso integer,
  id_atm integer,
  CONSTRAINT hijos_pkey PRIMARY KEY (id_hij),
  CONSTRAINT atencionmaternidad_hijos FOREIGN KEY (id_atm)
     REFERENCES atencionmaternidad (id_atm) MATCH SIMPLE
```

```
ON UPDATE CASCADE ON DELETE CASCADE )
```

II.1.9.2.2.1.18 TABLE horafechaingresos

```
CREATE TABLE horafechaingresos
(  id_hfi serial NOT NULL,
  fecha date,
  hora character varying(10),
  CONSTRAINT pk_horafechaingresos PRIMARY KEY (id_hfi) )
```

II.1.9.2.2.1.19 TABLE kardex

```
CREATE TABLE kardex
(  id_kar serial NOT NULL,
  fecha date,
  hora time without time zone,
  id_ord integer,
  id_sus integer,
  id_die integer,
  id_cam integer,
  id_usu integer NOT NULL,
  cod_persona integer NOT NULL,
  CONSTRAINT pk_kardex PRIMARY KEY (id_kar, id_usu, cod_persona),
  CONSTRAINT camas_kardex FOREIGN KEY (id_cam)
     REFERENCES camas (id_cam) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT dietas_kardex FOREIGN KEY (id_die)
```

REFERENCES dietas (id_die) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT ordenes_kardex FOREIGN KEY (id_ord)

REFERENCES ordenes (id_ord) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT suspendidos_kardex FOREIGN KEY (id_sus)

REFERENCES suspendidos (id_sus) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT usuarios_kardex FOREIGN KEY (id_usu)

REFERENCES usuarios (id_usu) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.1.20 TABLE menus

CREATE TABLE menus

(id_men serial NOT NULL,

nombre character varying(40),

descripcion character varying(40),

CONSTRAINT pk_menus PRIMARY KEY (id_men))

II.1.9.2.2.1.21 TABLE ordenes

CREATE TABLE ordenes

(id_ord serial NOT NULL,

nombre character varying(100),

CONSTRAINT pk_ordenes PRIMARY KEY (id_ord))

II.1.9.2.2.1.22 TABLE personal

```
CREATE TABLE personal
(  id_per serial NOT NULL,
  ci integer,
  nombre character varying(40),
  ap character varying(40),
  am character varying(40),
  estado integer,
  estado_civil character varying(40),
  fecha_nacimiento date,
  foto character varying(100),
  clave character varying(40),
  CONSTRAINT pk_personal PRIMARY KEY (id_per))
```

II.1.9.2.2.1.23 TABLE procesos

```
CREATE TABLE procesos
(  id_pro serial NOT NULL,
nombre character varying(60),
"60" character varying(40),
enlace character varying(60),
id_men integer,
CONSTRAINT pk_procesos PRIMARY KEY (id_pro),
CONSTRAINT procesos_id_men_fkey FOREIGN KEY (id_men)
REFERENCES menus (id_men) MATCH SIMPLE
```

ON UPDATE CASCADE ON DELETE CASCADE)

II.1.9.2.2.1.24 TABLE roles

CREATE TABLE roles

(id_rol serial NOT NULL,

nombre character varying(40),

descripcion character varying(100),

estado integer DEFAULT 1,

CONSTRAINT roles_pkey PRIMARY KEY (id_rol))

II.1.9.2.2.1.25 TABLE roles_procesos

CREATE TABLE roles_procesos

(id_rol integer NOT NULL,

id_pro integer NOT NULL,

CONSTRAINT pk_roles_procesos PRIMARY KEY (id_rol, id_pro),

CONSTRAINT procesos_roles_procesos FOREIGN KEY (id_pro)

REFERENCES procesos (id_pro) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT roles_roles_procesos FOREIGN KEY (id_rol)

REFERENCES roles (id_rol) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE)

II.1.9.2.2.1.26 TABLE salas

CREATE TABLE salas(

id_sal serial NOT NULL,

nombre character varying(40),

id_esp integer,

estado integer,

CONSTRAINT pk_salas PRIMARY KEY (id_sal),

CONSTRAINT salas_cod_esp_fkey FOREIGN KEY (id_esp)

REFERENCES especialidades (id_esp) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE)

II.1.9.2.2.1.27 TABLE servicios

CREATE TABLE servicios(

id_ser integer NOT NULL,

nombre character varying(40),

descripcion character varying(40),

estado integer,

CONSTRAINT servicios_pkey PRIMARY KEY (id_ser))

II.1.9.2.2.1.28 TABLE suspendidos

CREATE TABLE suspendidos(

id_sus serial NOT NULL,

nombre character varying(40),

CONSTRAINT pk_suspendidos PRIMARY KEY (id_sus))

II.1.9.2.2.1.29 TABLE tipo_diagnosticos

CREATE TABLE tipo_diagnosticos

(id_tid serial NOT NULL,

nombre character varying(100),

descripcion character varying(100),

CONSTRAINT tipo_diagnosticos_pkey PRIMARY KEY (id_tid))

II.1.9.2.2.1.30 TABLE tipo_estados

CREATE TABLE tipo_estados(

id_tie serial NOT NULL,

nombre character varying(100),

descripcion character varying(100),

CONSTRAINT tipo_estados_pkey PRIMARY KEY (id_tie))

II.1.9.2.2.1.31 TABLE tipo_prescripciones

CREATE TABLE tipo_prescripciones(

id_tip serial NOT NULL,

nombre character varying(100),

descripcion character varying(100),

CONSTRAINT tipo_prescripciones_pkey PRIMARY KEY (id_tip))

II.1.9.2.2.1.32 TABLE tipoanestecias

CREATE TABLE tipoanestecias(

id_tia serial NOT NULL,

nombre character varying(40),

CONSTRAINT pk_tipoanestecias PRIMARY KEY (id_tia))

II.1.9.2.2.1.33 TABLE tipooperaciones

CREATE TABLE tipooperaciones(

id_tio serial NOT NULL,

nombre character varying(40),

CONSTRAINT pk_tipooperaciones PRIMARY KEY (id_tio))

II.1.9.2.2.1.34 TABLE tipopartos

CREATE TABLE tipopartos(

id_tip serial NOT NULL,

nombre character varying(40),

descripcion character varying(40),

CONSTRAINT pk_tipopartos PRIMARY KEY (id_tip))

II.1.9.2.2.1.35 TABLE usuarios

CREATE TABLE usuarios(

id_usu serial NOT NULL,

id_per integer NOT NULL,

id_rol integer,

CONSTRAINT pk_usuarios PRIMARY KEY (id_usu),

CONSTRAINT personal_usuarios FOREIGN KEY (id_per)

REFERENCES personal (id_per) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT usuarios_id_rol_fkey FOREIGN KEY (id_rol)

REFERENCES roles (id_rol) MATCH SIMPLE

ON UPDATE CASCADE ON DELETE CASCADE)

II.1.9.2.2.2 BASE DE DATOS AFILIACION

II.1.9.2.2.2.1 TABLE actividad_economica

CREATE TABLE actividad_economica(

cod_actividad_economica serial NOT NULL,

nombre character varying(100) NOT NULL,

```
descripcion character varying(300),
estado integer NOT NULL DEFAULT 1,
fecha_registro date NOT NULL,
CONSTRAINT pk2 PRIMARY KEY (cod_actividad_economica))
```

II.1.9.2.2.2.2 TABLE asegurado

```
CREATE TABLE asegurado(
cod_asegurado serial NOT NULL,
nro_asegurado character varying(25) NOT NULL,
fecha_registro date NOT NULL,
estado integer NOT NULL DEFAULT 1,
cod_persona integer NOT NULL,
norma integer NOT NULL DEFAULT 1,
CONSTRAINT pk21 PRIMARY KEY (cod_asegurado),
CONSTRAINT fk6d53c91baa678dac FOREIGN KEY (cod_persona)
REFERENCES persona (cod_persona) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refpersona31 FOREIGN KEY (cod_persona)
REFERENCES persona (cod_persona) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.3 TABLE avc_01

```
CREATE TABLE avc_01(
cod_avc_01 serial NOT NULL,
fecha_registro date NOT NULL,
```

fecha_recepcion date NOT NULL,

lugar_presentacion character varying(100) NOT NULL,

fecha_presentacion date NOT NULL,

estado integer NOT NULL,

observaciones character varying(150),

revisado integer NOT NULL DEFAULT 0,

cod_empresa integer NOT NULL,

nfor character varying(50),

CONSTRAINT pk1 PRIMARY KEY (cod_avc_01),

CONSTRAINT fkac33551239b26f66 FOREIGN KEY (cod_empresa)

REFERENCES empresa (cod_empresa) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refempresa83 FOREIGN KEY (cod_empresa)

REFERENCES empresa (cod_empresa) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.4 TABLE avc_02

CREATE TABLE avc_02(

cod_avc_02 serial NOT NULL,

lugar character varying(100),

fecha date,

copia integer NOT NULL,

cod_avc_01 integer NOT NULL,

nfor character varying(50),

```
fecha_registro date NOT NULL,
CONSTRAINT pk9 PRIMARY KEY (cod_avc_02),
CONSTRAINT fkac3355136f6c47e3 FOREIGN KEY (cod_avc_01)
REFERENCES avc_01 (cod_avc_01) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refavc_0120 FOREIGN KEY (cod_avc_01)
REFERENCES avc_01 (cod_avc_01) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.5 TABLE avc_03

```
CREATE TABLE avc_03(
 cod_avc_03 serial NOT NULL,
fecha_recepcion date NOT NULL,
fecha date,
  fecha_registro date NOT NULL,
  observaciones character varying(150),
  tipo integer NOT NULL,
  fecha_presentacion date,
  lugar_presentacion character varying(100),
  estado integer NOT NULL,
  cod_avc_01 integer NOT NULL,
  nfor character varying(50),
  CONSTRAINT pk12 PRIMARY KEY (cod_avc_03),
CONSTRAINT fkac3355146f6c47e3 FOREIGN KEY (cod_avc_01)
```

REFERENCES avc_01 (cod_avc_01) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refavc_0125 FOREIGN KEY (cod_avc_01)

REFERENCES avc_01 (cod_avc_01) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.6 TABLE avc_04

CREATE TABLE avc_04(

cod_avc_04 integer NOT NULL,

cod_ocupacion integer NOT NULL,

observacion character varying(150),

fecha_registro date NOT NULL,

estado integer NOT NULL DEFAULT 1,

tipo integer NOT NULL DEFAULT 1,

salario_mensual real NOT NULL,

fecha_ingreso_trabajo date NOT NULL,

lugar_presentacion character varying(50) NOT NULL,

fecha_presentacion date NOT NULL,

fecha_recepcion date NOT NULL,

revisado integer NOT NULL DEFAULT 0,

num_municipio_localidad integer NOT NULL,

cod_asegurado integer NOT NULL,

cod_avc_01 integer NOT NULL,

nfor character varying(50)

CONSTRAINT pk22 PRIMARY KEY (cod_avc_04),

CONSTRAINT fkac33551518415682 FOREIGN KEY (cod_ocupacion)

REFERENCES ocupacion (cod_ocupacion) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT fkac33551523ad3b8a FOREIGN KEY (cod_asegurado)

REFERENCES asegurado (cod_asegurado) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT fkac3355156f6c47e3 FOREIGN KEY (cod_avc_01)

REFERENCES avc_01 (cod_avc_01) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT fkac3355158e1d7e5f FOREIGN KEY (num_municipio_localidad)

REFERENCES municipio_localidad (num_municipio_localidad) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refasegurado35 FOREIGN KEY (cod_asegurado)

REFERENCES asegurado (cod_asegurado) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refavc_0185 FOREIGN KEY (cod_avc_01)

REFERENCES avc_01 (cod_avc_01) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refmunicipio_localidad57 FOREIGN KEY (num_municipio_localidad)

REFERENCES municipio_localidad (num_municipio_localidad) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refocupacion36 FOREIGN KEY (cod_ocupacion)

```
REFERENCES ocupacion (cod_ocupacion) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.7 TABLE avc_04_a

```
CREATE TABLE avc_04_a(
  cod_avc_04_a serial NOT NULL,
  cod_asegurado integer NOT NULL,
  cod_tipo_seguro integer NOT NULL,
  num_municipio_localidad integer NOT NULL,
  observacion character varying(150),
  fecha_registro date NOT NULL,
  estado integer NOT NULL DEFAULT 1,
  documento_identidad character varying(25) NOT NULL,
  numero integer NOT NULL DEFAULT 1,
  localidad character varying(50) NOT NULL,
  lugar character varying(50) NOT NULL,
  fecha date NOT NULL,
  fecha_recepcion date NOT NULL,
  revisado integer NOT NULL DEFAULT 0,
  nfor character varying(50),
CONSTRAINT pk22_2 PRIMARY KEY (cod_avc_04_a),
CONSTRAINT fk6cb26fb723ad3b8a FOREIGN KEY (cod_asegurado)
REFERENCES asegurado (cod_asegurado) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
```

CONSTRAINT fk6cb26fb78e1d7e5f FOREIGN KEY (num_municipio_localidad)

REFERENCES municipio_localidad (num_municipio_localidad) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT fk6cb26fb7fbc9fd77 FOREIGN KEY (cod_tipo_seguro)

REFERENCES tipo_seguro (cod_tipo_seguro) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refasegurado54 FOREIGN KEY (cod_asegurado)

REFERENCES asegurado (cod_asegurado) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refmunicipio_localidad79 FOREIGN KEY (num_municipio_localidad)

REFERENCES municipio_localidad (num_municipio_localidad) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT reftipo_seguro76 FOREIGN KEY (cod_tipo_seguro)

REFERENCES tipo_seguro (cod_tipo_seguro) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.8 TABLE avc_05

CREATE TABLE avc_05(

cod_avc_05 integer NOT NULL,

cod_avc_04 integer NOT NULL,

lugar character varying(100),

fecha date,

copia integer NOT NULL,

nfor character varying(50),

```
fecha_registro date,
CONSTRAINT pk9_1 PRIMARY KEY (cod_avc_05),
CONSTRAINT fkac3355166f6c47e9 FOREIGN KEY (cod_avc_04)
REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refavc_0433 FOREIGN KEY (cod_avc_04)
REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.9 TABLE avc_06

```
CREATE TABLE avc_06(
  cod_avc_06 integer NOT NULL,
  observacion character varying(150),
  fecha_registro date NOT NULL,
  estado integer NOT NULL DEFAULT 1,
  lugar_aviso character varying(50) NOT NULL,
fecha_aviso date NOT NULL,
cod_avc_04 integer,
nfor character varying(50),
CONSTRAINT pk22_1_1 PRIMARY KEY (cod_avc_06),
CONSTRAINT fkac3355176f6c47e9 FOREIGN KEY (cod_avc_04)
REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refavc_0488 FOREIGN KEY (cod_avc_04)
```

```
REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.10 TABLE avc_07

```
CREATE TABLE avc_07(
  cod_avc_07 integer NOT NULL,
  cod_avc_04 integer NOT NULL,
  observacion character varying(150),
  fecha_registro date NOT NULL,
  estado integer NOT NULL DEFAULT 1,
  salario_retiro real NOT NULL DEFAULT 1,
  fecha_baja date NOT NULL,
  motivo character varying(150) NOT NULL,
  lugar_presentacion character varying(50) NOT NULL,
  fecha_presentacion date NOT NULL,
  fecha_recepcion date NOT NULL,
  nfor character varying(50),
  CONSTRAINT pk22_1 PRIMARY KEY (cod_avc_07),
  CONSTRAINT fkac3355186f6c47e9 FOREIGN KEY (cod_avc_04)
REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refavc_0438 FOREIGN KEY (cod_avc_04)
REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.11 TABLE avc_07_a

```
CREATE TABLE avc_07_a(
cod_avc_07_a serial NOT NULL,
observacion character varying(150),
fecha_registro date NOT NULL,
estado integer NOT NULL DEFAULT 1,
fecha_baja date NOT NULL,
motivo character varying(150) NOT NULL,
lugar character varying(50) NOT NULL,
fecha date NOT NULL,
fecha_recepcion date NOT NULL,
cod_avc_04_a integer NOT NULL,
nfor character varying(50),
CONSTRAINT pk22_1_2 PRIMARY KEY (cod_avc_07_a),
CONSTRAINT fk6cb27afa13f57e52 FOREIGN KEY (cod_avc_04_a)
REFERENCES avc_04_a (cod_avc_04_a) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refavc_04_a55 FOREIGN KEY (cod_avc_04_a)
REFERENCES avc_04_a (cod_avc_04_a) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.12 TABLE avc_08

```
CREATE TABLE avc_08(
  cod_avc_08 integer NOT NULL,
```

```
  cod_centro_salud integer NOT NULL,
  cod_avc_04 integer NOT NULL,
  salario real NOT NULL DEFAULT 0,
  regional_adscripcion character varying(50),
  observacion character varying(150),
  fecha_registro date NOT NULL,
  estado integer NOT NULL DEFAULT 1,
  lugar character varying(50) NOT NULL,
  atension_en character varying(150) NOT NULL,
  fecha date NOT NULL,
  nfor character varying(50),
CONSTRAINT pk22_2_1 PRIMARY KEY (cod_avc_08),
CONSTRAINT fkac3355191eb0741d FOREIGN KEY (cod_centro_salud)
REFERENCES centro_salud (cod_centro_salud) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT fkac3355196f6c47e9 FOREIGN KEY (cod_avc_04)
REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refavc_0449 FOREIGN KEY (cod_avc_04)
REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refcentro_salud51 FOREIGN KEY (cod_centro_salud)
REFERENCES centro_salud (cod_centro_salud) MATCH SIMPLE
```

```
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.13 TABLE avc_09

```
CREATE TABLE avc_09(
  cod_avc_09 integer NOT NULL,
  cod_avc_04 integer NOT NULL,
  incapacidad_hasta date NOT NULL,
  incapacidad_desde date NOT NULL,
  clave character varying(50),
  salario real NOT NULL DEFAULT 0,
  importe_subsidio real NOT NULL DEFAULT 0,
  observacion character varying(150),
  fecha_registro date NOT NULL,
  estado integer NOT NULL DEFAULT 1,
  lugar character varying(50) NOT NULL,
  riesgo character varying(50) NOT NULL,
  fecha date NOT NULL,
  dias_incapacidad character varying(150) NOT NULL,
  nfor character varying(50),
  CONSTRAINT pk22_2_1_1 PRIMARY KEY (cod_avc_09),
  CONSTRAINT fkac33551a6f6c47e9 FOREIGN KEY (cod_avc_04)
REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refavc_0450 FOREIGN KEY (cod_avc_04)
```

REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.14 TABLE backup

CREATE TABLE backup(

cod_backup serial NOT NULL,

path character varying(250) NOT NULL,

tipo character varying(250) NOT NULL,

file character varying(250),

fecha date NOT NULL,

cod_usuario integer NOT NULL,

CONSTRAINT pk58 PRIMARY KEY (cod_backup),

CONSTRAINT fkacc075c2eb71d4b0 FOREIGN KEY (cod_usuario)

REFERENCES usuario (cod_usuario) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refusuario103 FOREIGN KEY (cod_usuario)

REFERENCES usuario (cod_usuario) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.15 TABLE beneficiario

CREATE TABLE beneficiario(

cod_beneficiario integer NOT NULL,

fnac date NOT NULL,

apemat character varying(50) NOT NULL,

nombre character varying(70) NOT NULL,

```
apepat character varying(50),
estado integer NOT NULL DEFAULT 1,
fecha_registro date NOT NULL,
cod_asegurado integer NOT NULL,
codigo integer NOT NULL,
cod_parentesco integer NOT NULL,
norma integer NOT NULL DEFAULT 1,
CONSTRAINT pk17_2 PRIMARY KEY (cod_beneficiario),
CONSTRAINT fkebd5a43a23ad3b8a FOREIGN KEY (cod_asegurado)
REFERENCES asegurado (cod_asegurado) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT fkebd5a43aba9413b4 FOREIGN KEY (cod_parentesco)
REFERENCES parentesco (cod_parentesco) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refasegurado97 FOREIGN KEY (cod_asegurado)
REFERENCES asegurado (cod_asegurado) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refparentesco98 FOREIGN KEY (cod_parentesco)
REFERENCES parentesco (cod_parentesco) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.16 TABLE cambio_persona

```
CREATE TABLE cambio_persona(
  cod_cambio_persona serial NOT NULL,
```

```
  ci integer NOT NULL,
  sexo character varying(10) NOT NULL,
  estadocivil character varying(50) NOT NULL,
  fnac date NOT NULL,
  edad integer DEFAULT 0,
  apemat character varying(50) NOT NULL,
  nombre character varying(70) NOT NULL,
  apepat character varying(50),
  estado integer NOT NULL DEFAULT 1,
  fecha_registro date NOT NULL,
  cod_persona integer NOT NULL,
  CONSTRAINT pk17_1 PRIMARY KEY (cod_cambio_persona),
  CONSTRAINT fkde854446aa678dac FOREIGN KEY (cod_persona)
REFERENCES persona (cod_persona) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.17 TABLE centro_salud

```
CREATE TABLE centro_salud(
  cod_centro_salud serial NOT NULL,
  numero character varying(20),
  estado integer NOT NULL DEFAULT 1,
  nombre character varying(50) NOT NULL,
  descripcion character varying(300),
  direccion character varying(150),
```

telefono character varying(15),

fecha_registro date NOT NULL,

CONSTRAINT pk31_2_1_1 PRIMARY KEY (cod_centro_salud))

II.1.9.2.2.2.18 TABLE cumplido

CREATE TABLE cumplido(

cod_avc_04_a integer NOT NULL,

cod_requisitos integer NOT NULL,

CONSTRAINT pk41 PRIMARY KEY (cod_avc_04_a, cod_requisitos),

CONSTRAINT fk19aaa1dd13f57e52 FOREIGN KEY (cod_avc_04_a)

REFERENCES avc_04_a (cod_avc_04_a) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT fk19aaa1dd1e2e1d55 FOREIGN KEY (cod_requisitos)

REFERENCES requisitos (cod_requisitos) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT fk19aaa1dde9b966cc FOREIGN KEY (cod_requisitos)

REFERENCES requisitos (cod_requisitos) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refrequisitos42 FOREIGN KEY (cod_requisitos)

REFERENCES requisitos (cod_requisitos) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.19 TABLE departamento

CREATE TABLE departamento(

cod_departamento serial NOT NULL,

```
nombre character varying(100) NOT NULL,
descripcion character varying(300),
CONSTRAINT pk3 PRIMARY KEY (cod_departamento))
```

II.1.9.2.2.2.20 TABLE direccion

```
CREATE TABLE direccion(
  cod_direccion serial NOT NULL,
  calle character varying(300) NOT NULL,
  numero character varying(10) NOT NULL,
  estado integer DEFAULT 1,
  fecha_registro date NOT NULL,
  cod_zona integer NOT NULL,
  cod_avc_04_a integer NOT NULL,
  CONSTRAINT pk7_1_1 PRIMARY KEY (cod_direccion),
  CONSTRAINT fkc6984d3013f57e52 FOREIGN KEY (cod_avc_04_a)
REFERENCES avc_04_a (cod_avc_04_a) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT fkc6984d302d75197c FOREIGN KEY (cod_zona)
REFERENCES zona (cod_zona) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refavc_04_a81 FOREIGN KEY (cod_avc_04_a)
REFERENCES avc_04_a (cod_avc_04_a) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refzona80 FOREIGN KEY (cod_zona)
```

```
REFERENCES zona (cod_zona) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.21 TABLE direccion_asegurado

```
CREATE TABLE direccion_asegurado(
  cod_direccion_asegurado integer NOT NULL,
  cod_zona integer NOT NULL,
  cod_avc_04 integer NOT NULL,
  calle character varying(300) NOT NULL,
  numero character varying(10),
  estado integer DEFAULT 1,
  fecha_registro date NOT NULL,
  CONSTRAINT pk7_1 PRIMARY KEY (cod_direccion_asegurado),
  CONSTRAINT fk31d254cc2d75197c FOREIGN KEY (cod_zona)
REFERENCES zona (cod_zona) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT fk31d254cc6f6c47e9 FOREIGN KEY (cod_avc_04)
REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refavc_0439 FOREIGN KEY (cod_avc_04)
REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refzona40 FOREIGN KEY (cod_zona)
REFERENCES zona (cod_zona) MATCH SIMPLE
```

```
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.22 TABLE direccion_empresa

```
CREATE TABLE direccion_empresa(
  cod_direccion_empresa integer NOT NULL,
  cod_empresa integer NOT NULL,
  calle character varying(300) NOT NULL,
  numero character varying(10),
  estado integer DEFAULT 1,
  fecha_registro date NOT NULL,
  cod_zona integer NOT NULL,
CONSTRAINT pk7 PRIMARY KEY (cod_direccion_empresa),
CONSTRAINT fk3c95047a2d75197c FOREIGN KEY (cod_zona)
REFERENCES zona (cod_zona) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT fk3c95047a39b26f66 FOREIGN KEY (cod_empresa)
REFERENCES empresa (cod_empresa) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
 CONSTRAINT refempresa29 FOREIGN KEY (cod_empresa)
REFERENCES empresa (cod_empresa) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refzona14 FOREIGN KEY (cod_zona)
REFERENCES zona (cod_zona) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.23 TABLE empresa

```
CREATE TABLE empresa(
  cod_empresa serial NOT NULL,
  codemp character varying(30) NOT NULL,
  nombre_razonsocial character varying(100) NOT NULL,
  descripcion character varying(300),
  numero_trabajadores integer NOT NULL,
  n_padron_renta character varying(15),
  fecha_iniciacion_actividad date NOT NULL,
  estado integer NOT NULL,
  fecha_registro date NOT NULL,
  cod_actividad_economica integer NOT NULL,
  cod_tipo_empresa integer NOT NULL,
  num_municipio_localidad integer NOT NULL,
  norma integer NOT NULL DEFAULT 1,
CONSTRAINT pk11 PRIMARY KEY (cod_empresa),
CONSTRAINT fk9f3540898e1d7e5f FOREIGN KEY (num_municipio_localidad)
REFERENCES municipio_localidad (num_municipio_localidad) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT fk9f354089d08f4d35 FOREIGN KEY (cod_tipo_empresa)
REFERENCES tipo_empresa (cod_tipo_empresa) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT fk9f354089fde2ba97 FOREIGN KEY (cod_actividad_economica)
```

REFERENCES actividad_economica (cod_actividad_economica) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refactividad_economica3 FOREIGN KEY (cod_actividad_economica)

REFERENCES actividad_economica (cod_actividad_economica) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refmunicipio_localidad59 FOREIGN KEY (num_municipio_localidad)

REFERENCES municipio_localidad (num_municipio_localidad) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT reftipo_empresa4 FOREIGN KEY (cod_tipo_empresa)

REFERENCES tipo_empresa (cod_tipo_empresa) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.24 TABLE funcionalidad

CREATE TABLE funcionalidad(

cod_funcionalidad character varying(10) NOT NULL,

link character varying(100) NOT NULL,

nombre character(50) NOT NULL,

descripcion character varying(250),

estado integer NOT NULL,

cod_modulo integer NOT NULL,

CONSTRAINT pk48 PRIMARY KEY (cod_funcionalidad),

CONSTRAINT fk405b68ed782e4418 FOREIGN KEY (cod_modulo)

REFERENCES modulo (cod_modulo) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refmodulo60 FOREIGN KEY (cod_modulo)

REFERENCES modulo (cod_modulo) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.25 TABLE grupo_familiar

CREATE TABLE grupo_familiar(

cod_grupo_familiar integer NOT NULL,

cod_beneficiario integer NOT NULL,

estado integer NOT NULL DEFAULT 1,

fecha_exticion date NOT NULL,

causas character varying(150),

fecha_registro date NOT NULL,

cod_avc_04 integer NOT NULL,

tipo integer NOT NULL,

cod_avc_06 integer NOT NULL,

CONSTRAINT pk28 PRIMARY KEY (cod_grupo_familiar),

CONSTRAINT fkdeffed9b6f6c47e9 FOREIGN KEY (cod_avc_04)

REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT fkdeffed9b6f6c47ed FOREIGN KEY (cod_avc_06)

REFERENCES avc_06 (cod_avc_06) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT fkdeffed9b817f2ea0 FOREIGN KEY (cod_beneficiario)

REFERENCES beneficiario (cod_beneficiario) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refavc_0487 FOREIGN KEY (cod_avc_04)

REFERENCES avc_04 (cod_avc_04) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refavc_0699 FOREIGN KEY (cod_avc_06)

REFERENCES avc_06 (cod_avc_06) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refbeneficiario94 FOREIGN KEY (cod_beneficiario)

REFERENCES beneficiario (cod_beneficiario) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.26 TABLE grupo_familiar_avc_08

CREATE TABLE grupo_familiar_avc_08(

cod_avc_08 integer NOT NULL,

cod_beneficiario integer NOT NULL,

CONSTRAINT pk56 PRIMARY KEY (cod_avc_08, cod_beneficiario),

CONSTRAINT fkbda4f03d6f6c47f1 FOREIGN KEY (cod_avc_08)

REFERENCES avc_08 (cod_avc_08) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT fkbda4f03d817f2ea0 FOREIGN KEY (cod_beneficiario)

REFERENCES beneficiario (cod_beneficiario) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refavc_0892 FOREIGN KEY (cod_avc_08)

```
REFERENCES avc_08 (cod_avc_08) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refbeneficiario101 FOREIGN KEY (cod_beneficiario)
REFERENCES beneficiario (cod_beneficiario) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.27 TABLE modulo

```
CREATE TABLE modulo(
cod_modulo serial NOT NULL,
nombre character varying(50) NOT NULL,
descripcion character varying(250),
estado integer NOT NULL DEFAULT 1,
CONSTRAINT pk47 PRIMARY KEY (cod_modulo))
```

II.1.9.2.2.2.28 TABLE municipio_localidad

```
CREATE TABLE municipio_localidad(
  num_municipio_localidad serial NOT NULL,
  cod_provincia integer NOT NULL,
  numero character varying(25) NOT NULL,
  estado integer NOT NULL DEFAULT 1,
  nombre character varying(50) NOT NULL,
  descripcion character varying(300),
  CONSTRAINT pk31_2 PRIMARY KEY (num_municipio_localidad),
  CONSTRAINT fk69a8be977c820ae FOREIGN KEY (cod_provincia)
REFERENCES provincia (cod_provincia) MATCH SIMPLE
```

```
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refprovincia52 FOREIGN KEY (cod_provincia)
REFERENCES provincia (cod_provincia) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.29 TABLE nombre_empresa

```
CREATE TABLE nombre_empresa(
  cod_nombre_empresa integer NOT NULL,
  nombre_razonsocial character varying(100) NOT NULL,
  fecha_registro date NOT NULL,
  estado integer NOT NULL,
  cod_empresa integer NOT NULL,
CONSTRAINT pk8_1 PRIMARY KEY (cod_nombre_empresa),
CONSTRAINT fk31c55e1339b26f66 FOREIGN KEY (cod_empresa)
REFERENCES empresa (cod_empresa) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refempresa86 FOREIGN KEY (cod_empresa)
REFERENCES empresa (cod_empresa) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.30 TABLE ocupacion

```
CREATE TABLE ocupacion(
cod_ocupacion serial NOT NULL,
nombre character varying(50) NOT NULL,
descripcion character varying(150),
```

```
estado integer NOT NULL DEFAULT 1,
fecha_registro date NOT NULL,
CONSTRAINT pk24 PRIMARY KEY (cod_ocupacion))
```

II.1.9.2.2.2.31 TABLE parentesco

```
CREATE TABLE parentesco(
  cod_parentesco serial NOT NULL,
  nombre character varying(50) NOT NULL,
  descripcion character varying(150),
  estado integer NOT NULL DEFAULT 1,
  fecha_registro date NOT NULL,
  CONSTRAINT pk24_1 PRIMARY KEY (cod_parentesco))
```

II.1.9.2.2.2.32 TABLE persona

```
CREATE TABLE persona(
  cod_persona serial NOT NULL,
  ci integer NOT NULL,
  sexo character varying(10) NOT NULL,
  estadocivil character varying(50) NOT NULL,
  fnac date NOT NULL,
  edad integer DEFAULT 0,
  apemat character varying(50) NOT NULL,
  nombre character varying(70) NOT NULL,
  apepat character varying(50),
  estado integer NOT NULL DEFAULT 1,
```

```
  fecha_registro date NOT NULL,
  CONSTRAINT pk17 PRIMARY KEY (cod_persona))
```

II.1.9.2.2.2.33 TABLE propietario

```
CREATE TABLE propietario(
cod_propietario serial NOT NULL,
fecha_registro date NOT NULL,
estado integer NOT NULL DEFAULT 1,
cod_persona integer NOT NULL,
CONSTRAINT pk10 PRIMARY KEY (cod_propietario),
CONSTRAINT fke4a932caa678dac FOREIGN KEY (cod_persona)
REFERENCES persona (cod_persona) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT refpersona15 FOREIGN KEY (cod_persona)
REFERENCES persona (cod_persona) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.34 TABLE propietario_empleador

```
CREATE TABLE propietario_empleador(
  cod_propietario_empleador serial NOT NULL,
  cod_propietario integer NOT NULL,
  cod_empresa integer NOT NULL,
  fecha_registro date NOT NULL,
  estado integer DEFAULT 1,
CONSTRAINT pk18 PRIMARY KEY (cod_propietario_empleador),
```

CONSTRAINT fk67a09fb439b26f66 FOREIGN KEY (cod_empresa)

REFERENCES empresa (cod_empresa) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT fk67a09fb43ad5caac FOREIGN KEY (cod_propietario)

REFERENCES propietario (cod_propietario) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refempresa23 FOREIGN KEY (cod_empresa)

REFERENCES empresa (cod_empresa) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refpropietario22 FOREIGN KEY (cod_propietario)

REFERENCES propietario (cod_propietario) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.35 TABLE provincia

CREATE TABLE provincia(

cod_provincia serial NOT NULL,

cod_departamento integer NOT NULL,

estado integer NOT NULL DEFAULT 1,

nombre character varying(50) NOT NULL,

descripcion character varying(300),

CONSTRAINT pk31_2_1 PRIMARY KEY (cod_provincia),

CONSTRAINT fkdf613bad11ce3e34 FOREIGN KEY (cod_departamento)

REFERENCES departamento (cod_departamento) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refdepartamento53 FOREIGN KEY (cod_departamento)

REFERENCES departamento (cod_departamento) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.36 TABLE requisitos

CREATE TABLE requisitos(

cod_requisitos serial NOT NULL,

cod_tipo_seguro integer NOT NULL,

estado integer NOT NULL DEFAULT 1,

requisito character varying(500) NOT NULL,

descripcion character varying(300),

fecha_registro date NOT NULL,

CONSTRAINT pk31_1 PRIMARY KEY (cod_requisitos),

CONSTRAINT fk904f2d0fbc9fd77 FOREIGN KEY (cod_tipo_seguro)

REFERENCES tipo_seguro (cod_tipo_seguro) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT reftipo_seguro41 FOREIGN KEY (cod_tipo_seguro)

REFERENCES tipo_seguro (cod_tipo_seguro) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.37 TABLE rol

CREATE TABLE rol(

cod_rol serial NOT NULL,

nombre character varying(50) NOT NULL,

descripcion character varying(250),

estado integer NOT NULL DEFAULT 1,

fecha_registro date NOT NULL,

CONSTRAINT pk46 PRIMARY KEY (cod_rol))

II.1.9.2.2.2.38 TABLE rol_funcionalidad

CREATE TABLE rol_funcionalidad(

cod_rol integer NOT NULL,

cod_funcionalidad character varying(10) NOT NULL,

CONSTRAINT pk50 PRIMARY KEY (cod_rol, cod_funcionalidad),

CONSTRAINT fkb41e1e7d115fb12e FOREIGN KEY (cod_funcionalidad)

REFERENCES funcionalidad (cod_funcionalidad) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT fkb41e1e7d4387abf2 FOREIGN KEY (cod_rol)

REFERENCES rol (cod_rol) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT reffuncionalidad65 FOREIGN KEY (cod_funcionalidad)

REFERENCES funcionalidad (cod_funcionalidad) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refrol64 FOREIGN KEY (cod_rol)

REFERENCES rol (cod_rol) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.39 TABLE telefono_empresa

CREATE TABLE telefono_empresa(

cod_telefono_empresa serial NOT NULL,

estado integer NOT NULL,

numero character varying(30) NOT NULL,

fecha_registro date NOT NULL,

cod_empresa integer NOT NULL,

CONSTRAINT pk8 PRIMARY KEY (cod_telefono_empresa),

CONSTRAINT fk19dd1e3e39b26f66 FOREIGN KEY (cod_empresa)

REFERENCES empresa (cod_empresa) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refempresa8 FOREIGN KEY (cod_empresa)

REFERENCES empresa (cod_empresa) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION)

II.1.9.2.2.2.40 TABLE tipo_empresa

CREATE TABLE tipo_empresa(

cod_tipo_empresa serial NOT NULL,

nombre character varying(100) NOT NULL,

descripcion character varying(300),

estado integer NOT NULL DEFAULT 1,

fecha_registro date NOT NULL,

CONSTRAINT pk13 PRIMARY KEY (cod_tipo_empresa))

II.1.9.2.2.2.41 TABLE tipo_seguro

CREATE TABLE tipo_seguro(

cod_tipo_seguro serial NOT NULL,

estado integer NOT NULL DEFAULT 1,

nombre character varying(50) NOT NULL,

descripcion character varying(300),

fecha_registro date NOT NULL,

CONSTRAINT pk31 PRIMARY KEY (cod_tipo_seguro))

II.1.9.2.2.2.42 TABLE usuario

CREATE TABLE usuario(

cod_usuario serial NOT NULL,

username character varying(50) NOT NULL,

"password" character varying(50) NOT NULL,

estado integer NOT NULL DEFAULT 1,

fecha_registro date NOT NULL,

cod_rol integer NOT NULL,

cod_persona integer,

CONSTRAINT pk45 PRIMARY KEY (cod_usuario),

CONSTRAINT fkf814f32e4387abf2 FOREIGN KEY (cod_rol)

REFERENCES rol (cod_rol) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT fkf814f32eaa678dac FOREIGN KEY (cod_persona)

REFERENCES persona (cod_persona) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT refpersona67 FOREIGN KEY (cod_persona)

REFERENCES persona (cod_persona) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

```
CONSTRAINT refrol75 FOREIGN KEY (cod_rol)
REFERENCES rol (cod_rol) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION)
```

II.1.9.2.2.2.43 TABLE zona

```
CREATE TABLE zona
(  cod_zona serial NOT NULL,
  nombre character varying(100) NOT NULL,
  descripcion character varying(300),
  estado
 integer NOT NULL DEFAULT 1,
  fecha_registro date NOT NULL,
  CONSTRAINT pk6 PRIMARY KEY (cod_zona))
```

II.1.10 DISEÑO DE PANTALLAS DE USUARIO

II.1.10.1 Introducción

Trata de diseños que permiten al usuario tener una idea sobre las interfaces que proveerá el sistema.

II.1.10.1.1 Propósito

Presentar los diseños de pantallas para que el usuario tenga idea de la interfaz que le presenta el sistema.

II.1.10.1.2 Alcance

Mostrar los diseños de pantallas, sujeto a modificaciones a lo largo del desarrollo del sistema

II.1.10.2 Diagrama Navegacional

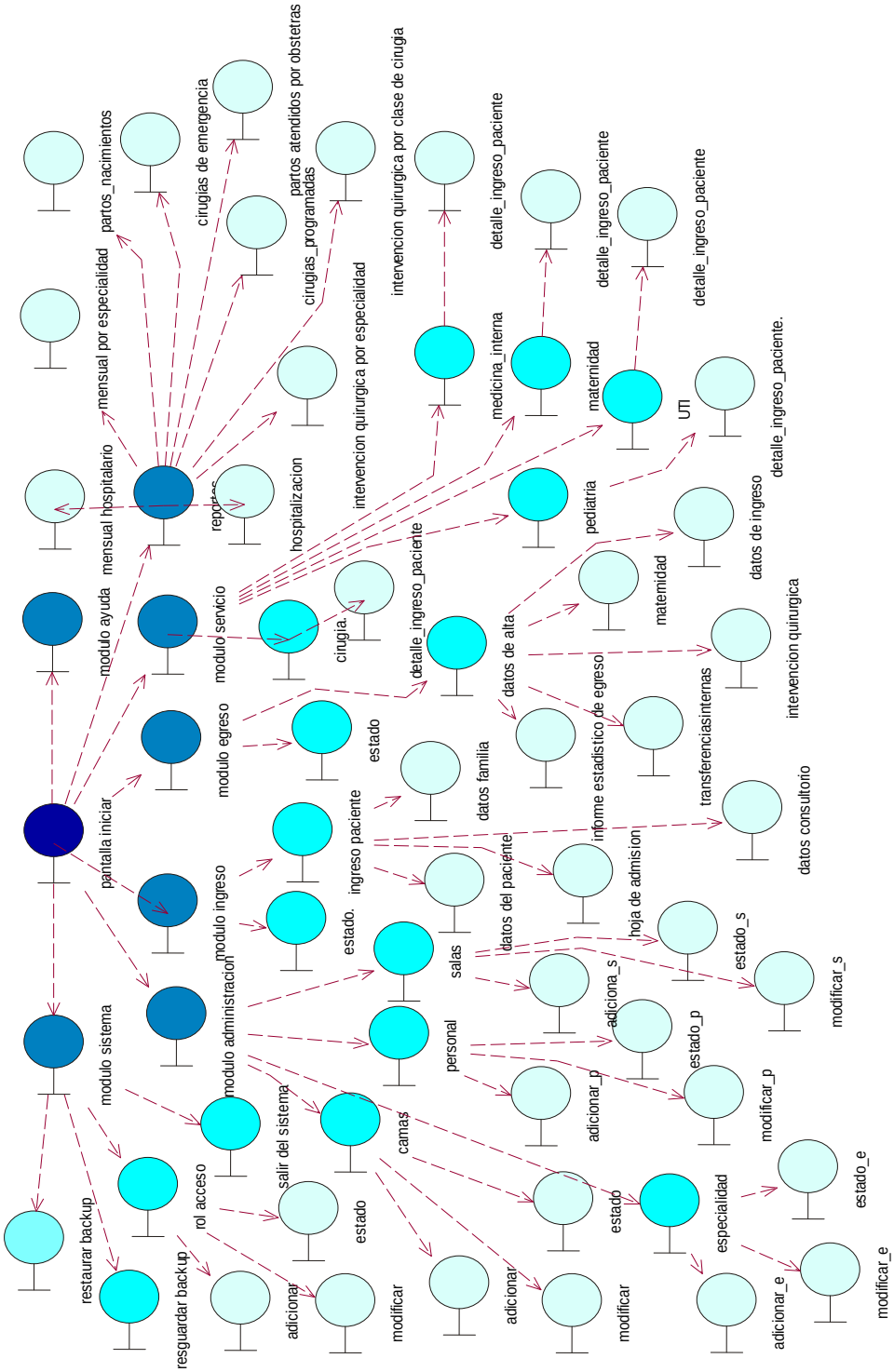


Figura 164. Diagrama Navegacional

II.1.10.3 Pantallas

II.1.10.3.1 pantalla iniciar

Figura 165.Pantalla iniciar

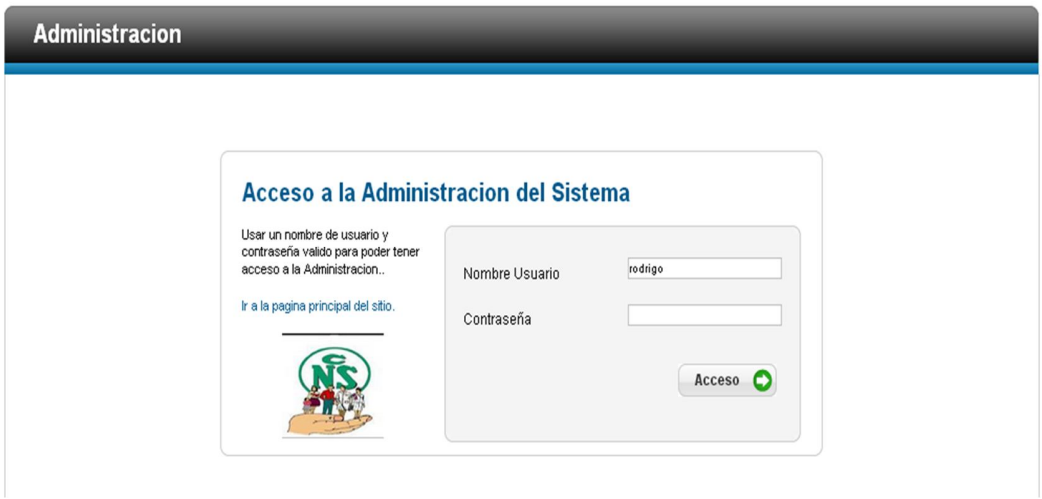


Figura 166. Pantalla iniciar2

Figura 167. Pantalla Principal

II.1.10.3.2 Modulo de Sistema

Figura 168. Modulo Sistema

II.1.10.3.3 Rol acceso

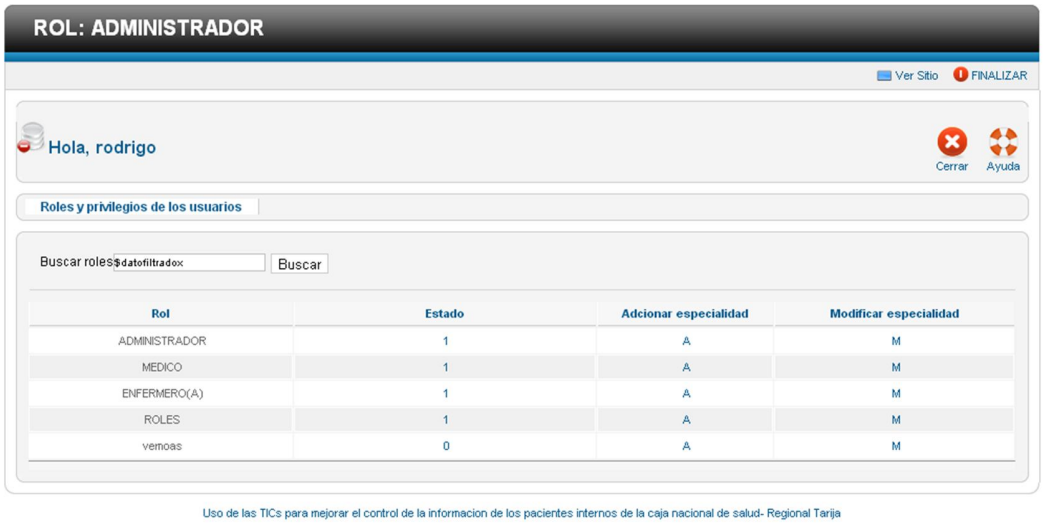


Figura 169. Pantalla Rol acceso

II.1.10.3.4 Adicionar rol

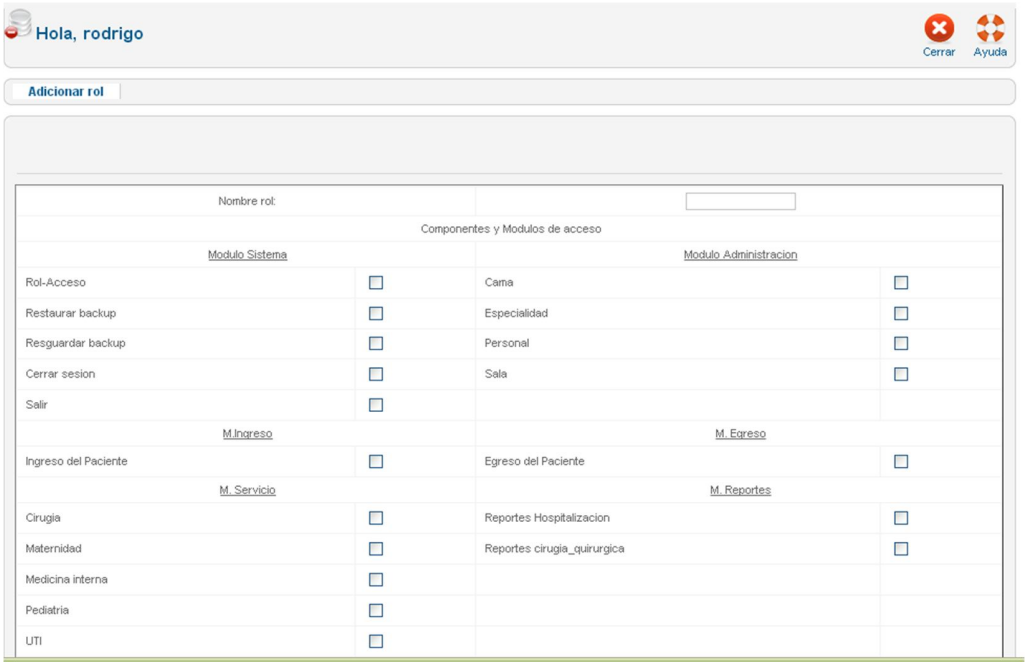


Figura 170. Pantalla Adicionar rol

II.1.10.3.5 Modificar rol

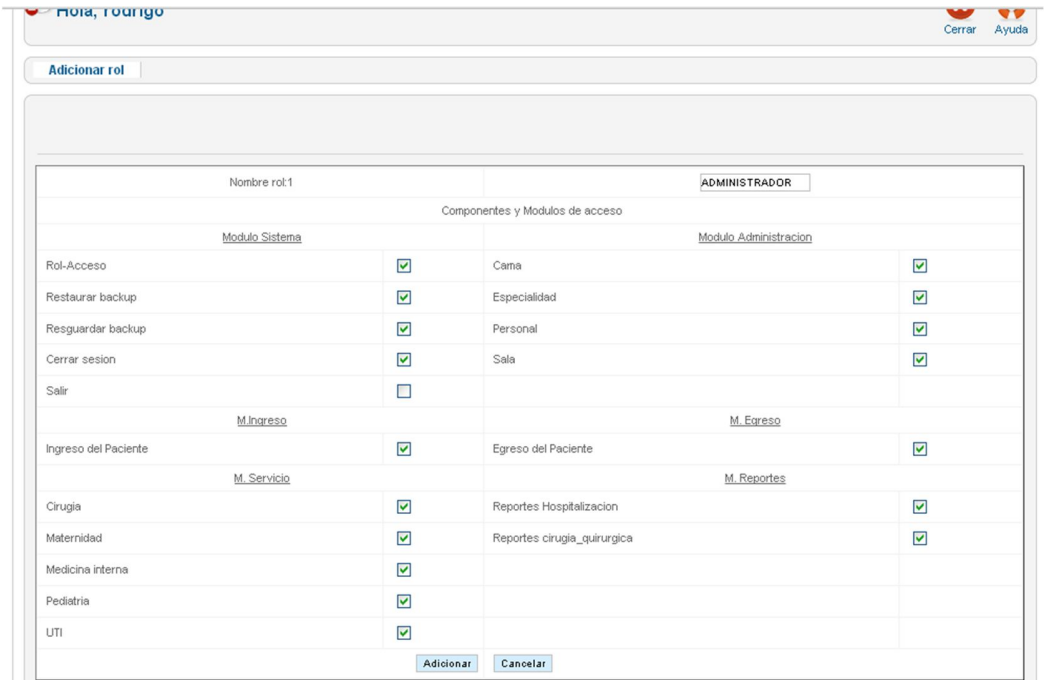


Figura 171. Pantalla Modificar rol

II.1.10.3.6 Obtener Estado (rol)

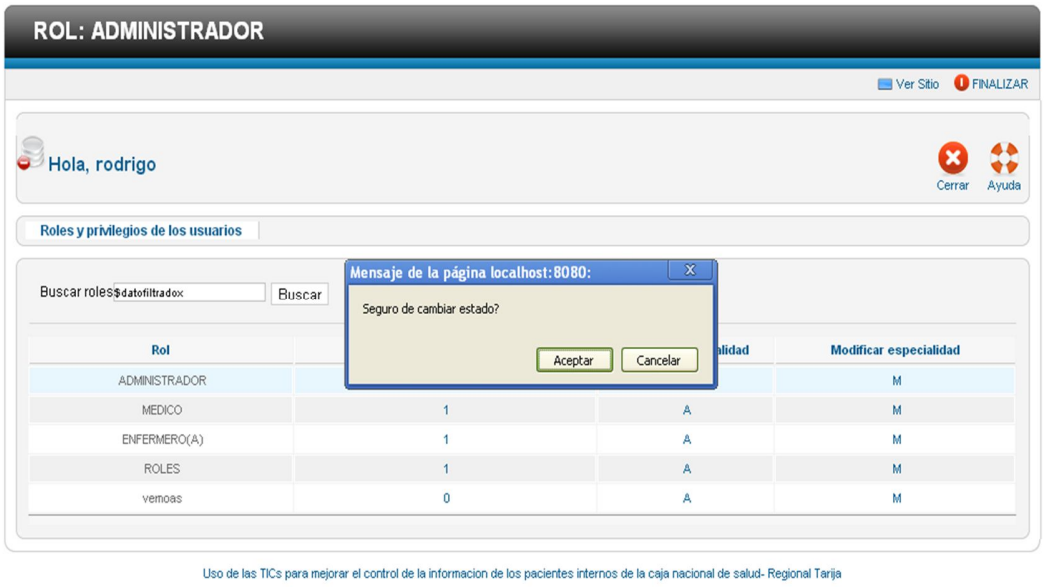


Figura 172. Pantalla Obtener Estado (rol)

II.1.10.3.7 Modulo administración

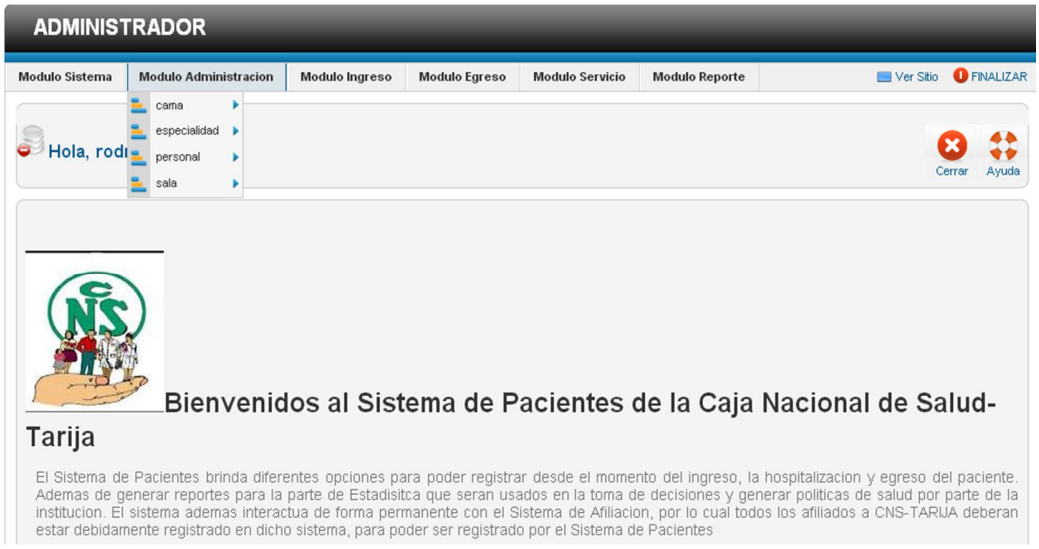


Figura 173. Modulo administración

II.1.10.3.8 Camas

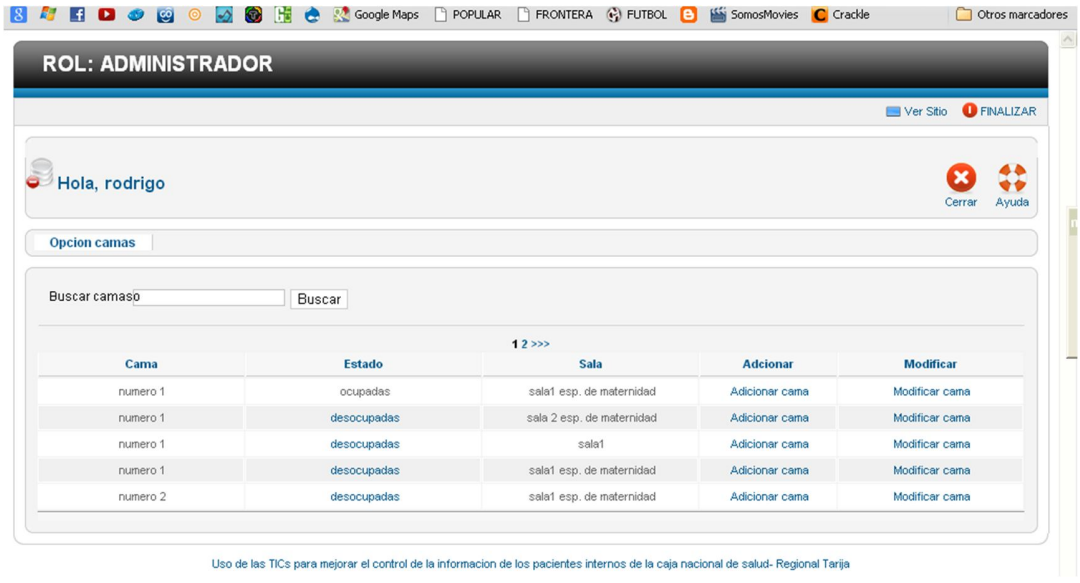


Figura 174. Pantalla Camas

II.1.10.3.9 Adicionar camas

Figura 175. Pantalla Adicionar camas

II.1.10.3.10 Modificar camas

Figura 176. Pantalla Modificar camas

II.1.10.3.11 Obtener Estado (cama)

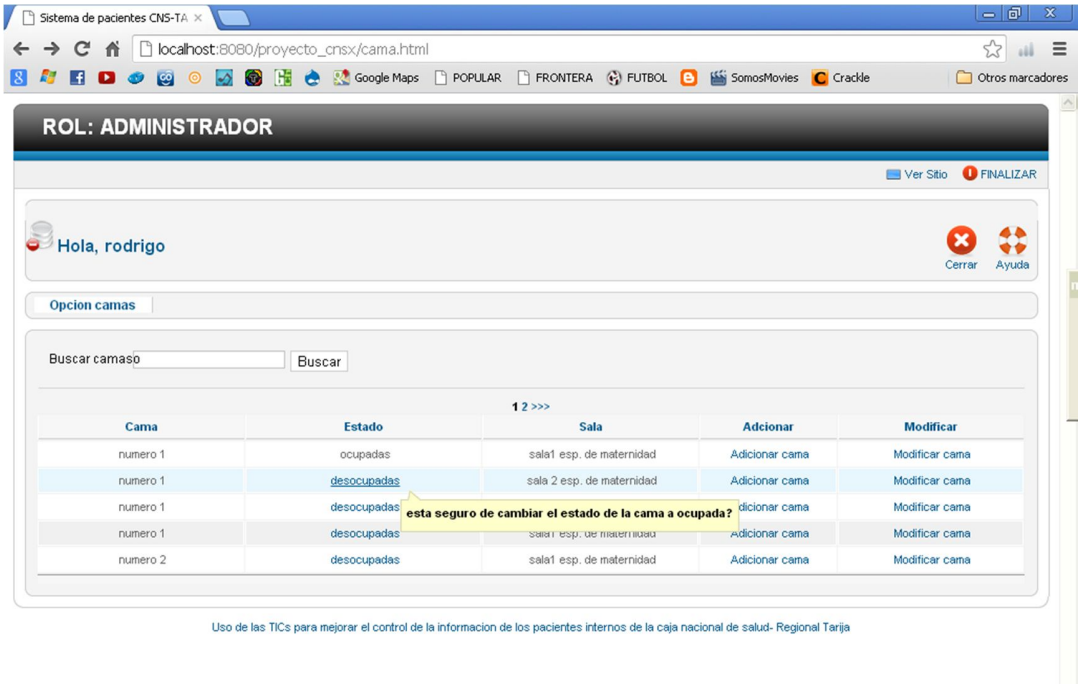


Figura 177. Pantalla Obtener Estado (cama)

II.1.10.3.12 Especialidad

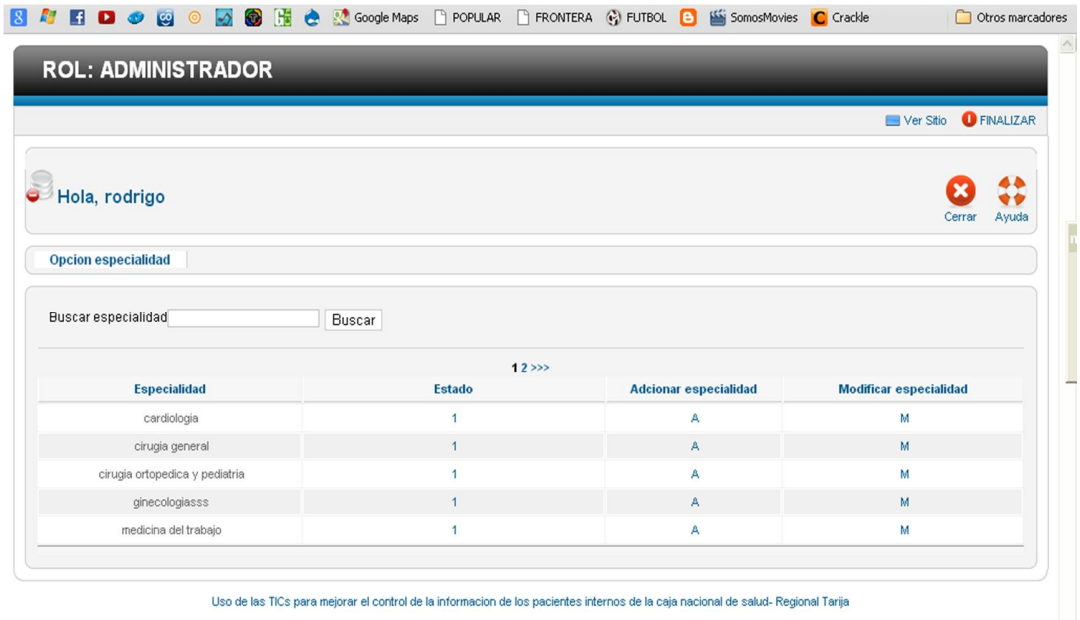


Figura 178. Pantalla Especialidad

II.1.10.3.13 Adicionar especialidad

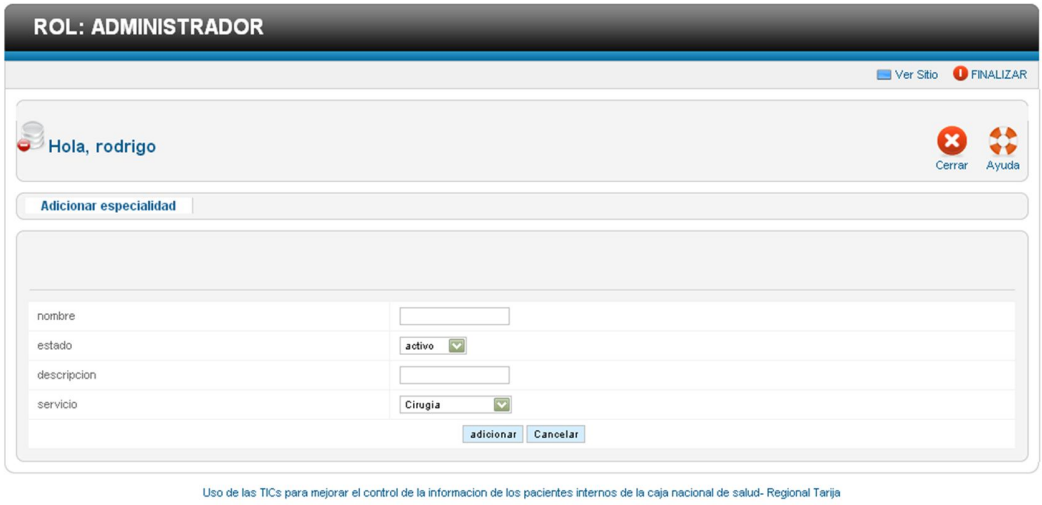


Figura 179. Pantalla Adicionar especialidad

II.1.10.3.14 Modificar especialidad

Figura 180. Pantalla Modificar especialidad

II.1.10.3.15 Obtener Estado (especialidad)

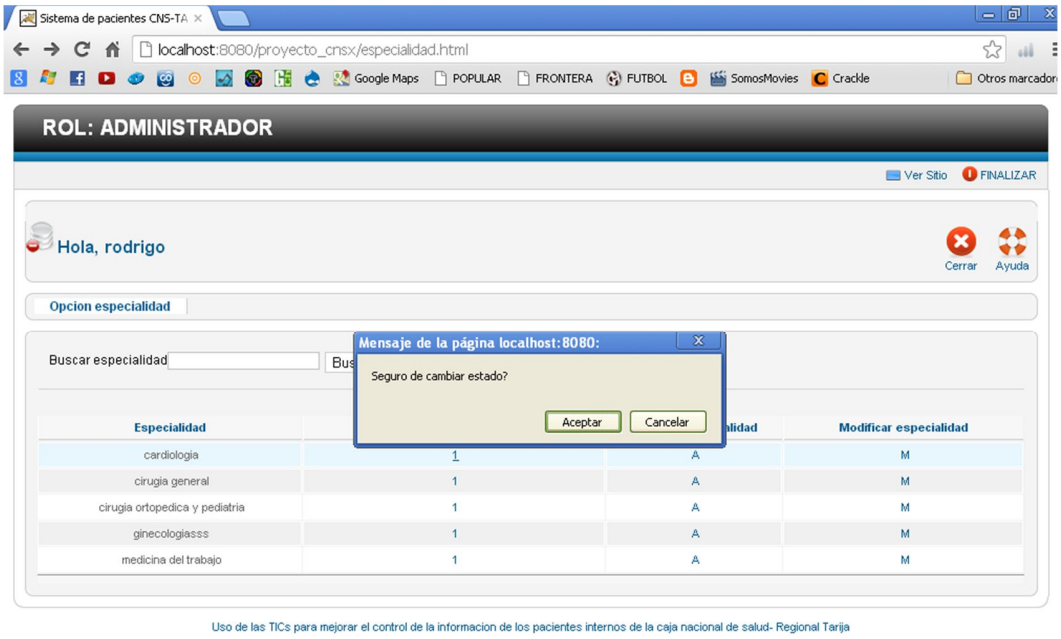


Figura 181. Pantalla Obtener Estado (especialidad)

II.1.10.3.16 Personal

Figura 182. Pantalla Personal

II.1.10.3.17 Adicionar personal

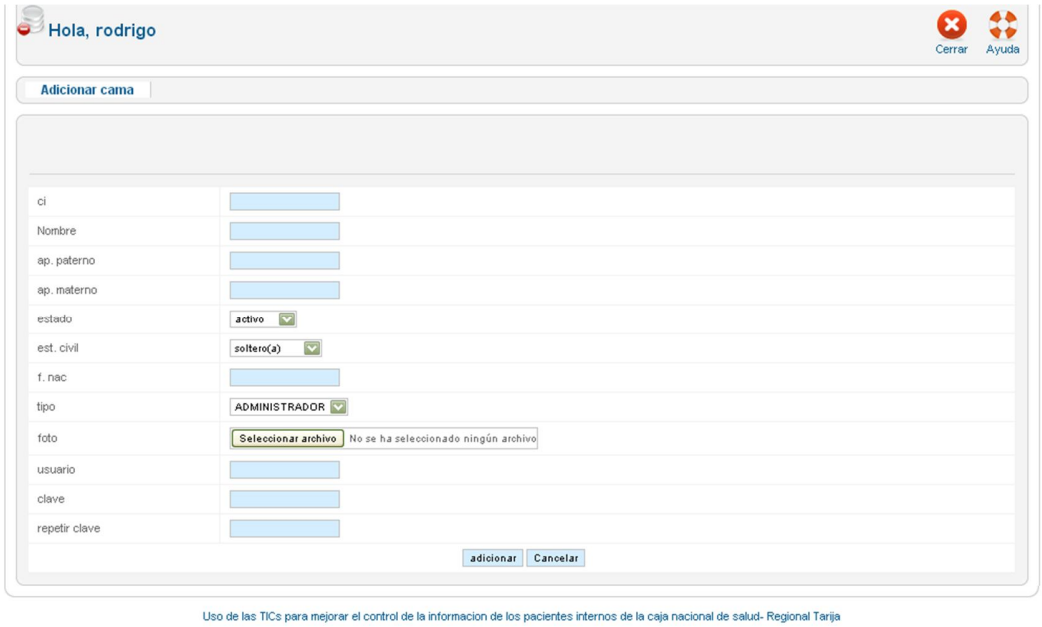


Figura 183. Pantalla Adicionar personal

II.1.10.3.18 Modificar personal

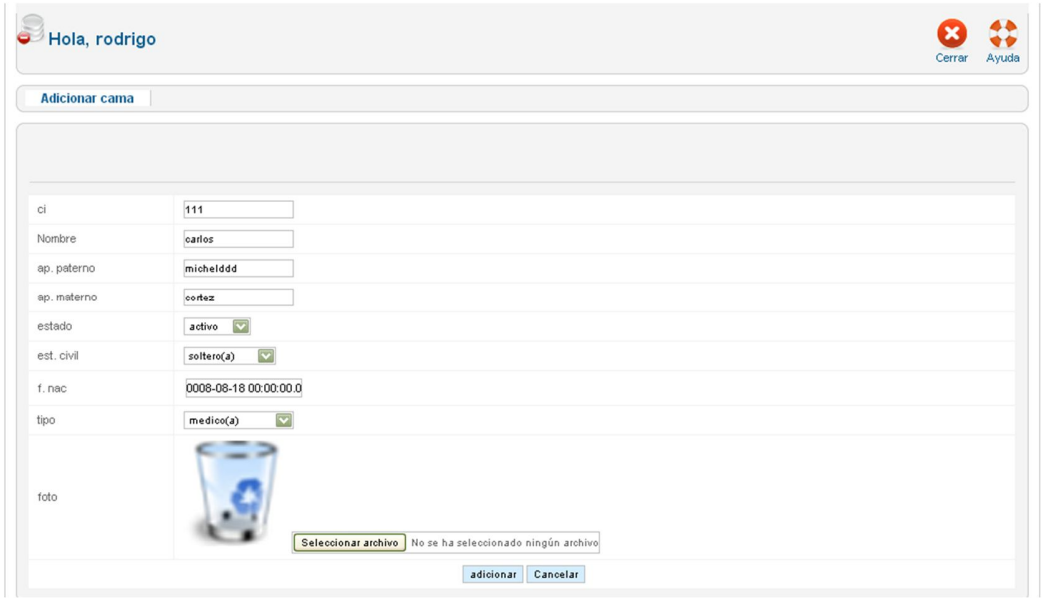


Figura 184. Pantalla Modificar personal

II.1.10.3.19 Obtener Estado (personal)

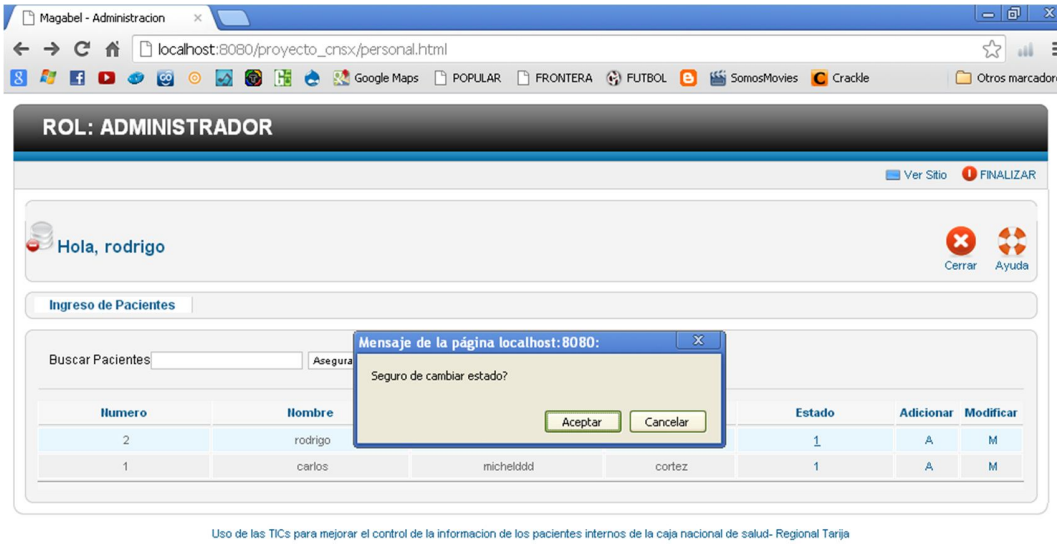


Figura 185. Pantalla Obtener Estado (personal)

II.1.10.3.20 Salas

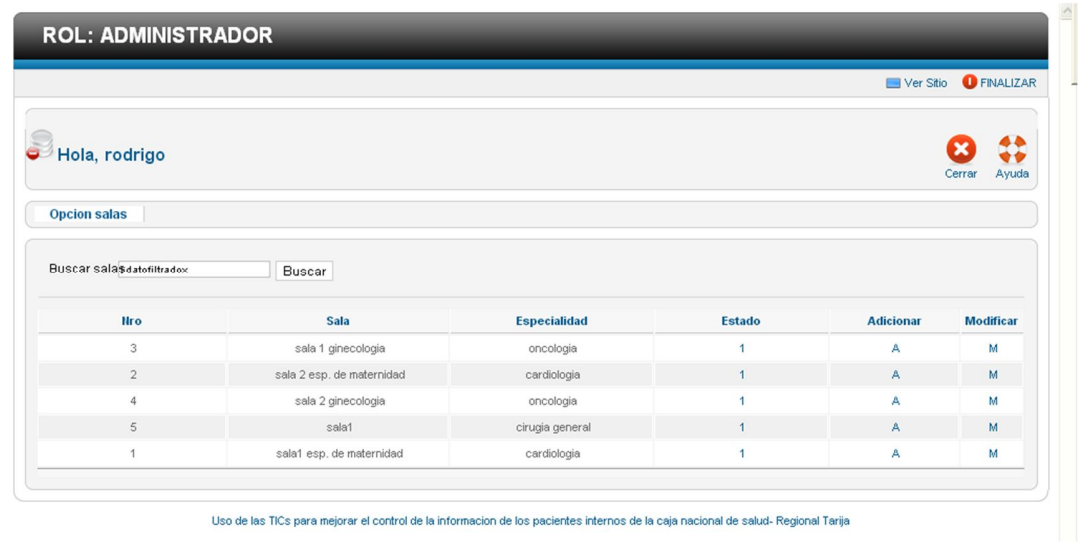


Figura 186. Pantalla Salas

II.1.10.3.21 Adicionar salas

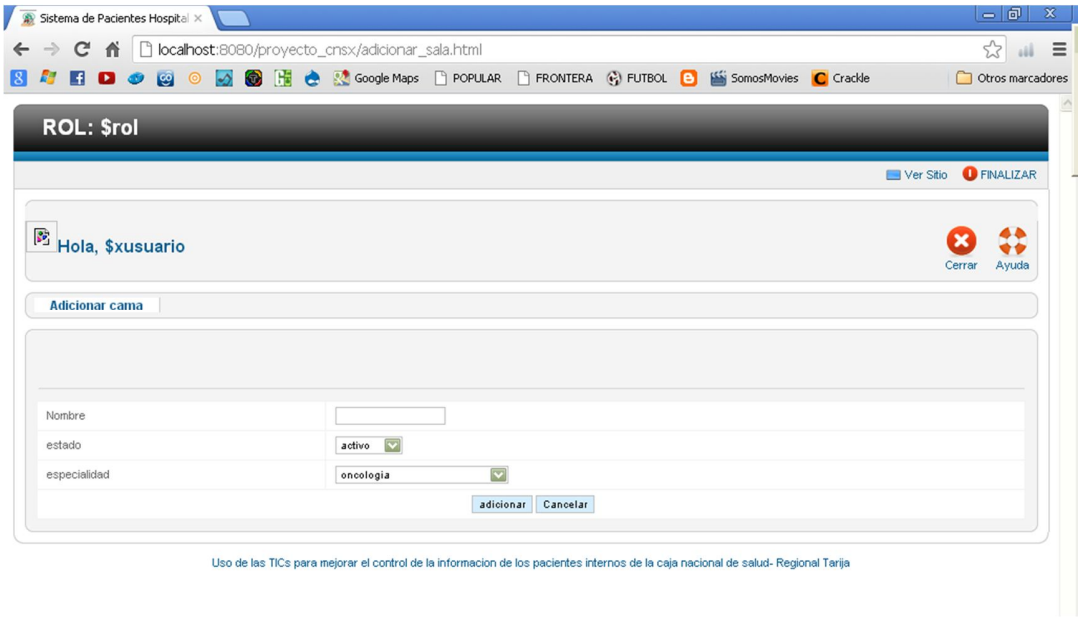


Figura 187. Pantalla Adicionar salas

II.1.10.3.22 Modificar salas

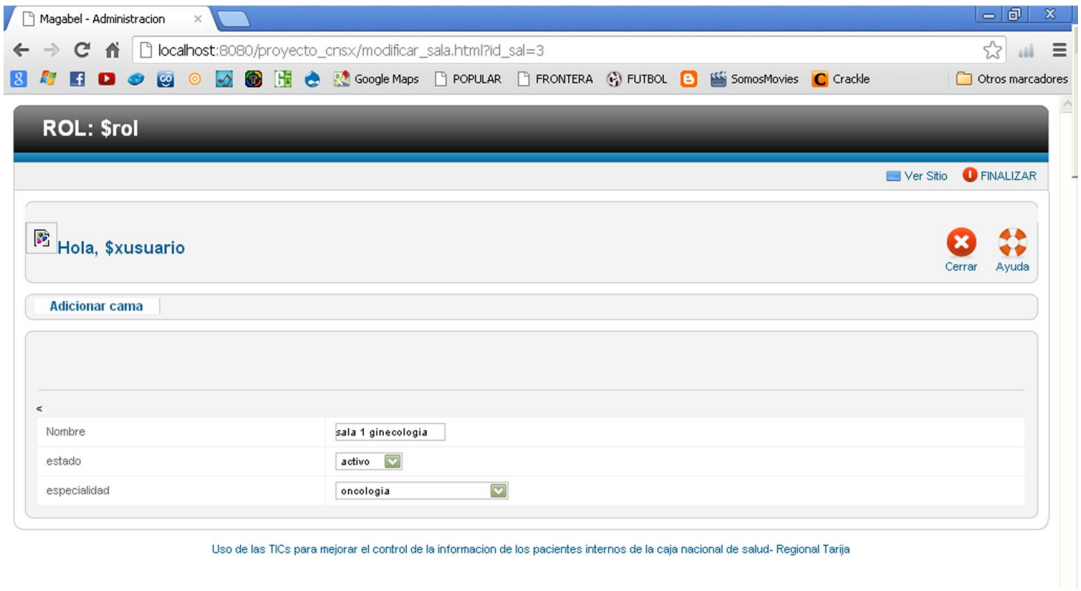


Figura 188. Pantalla Modificar salas

II.1.10.3.23 Obtener Estado (sala)

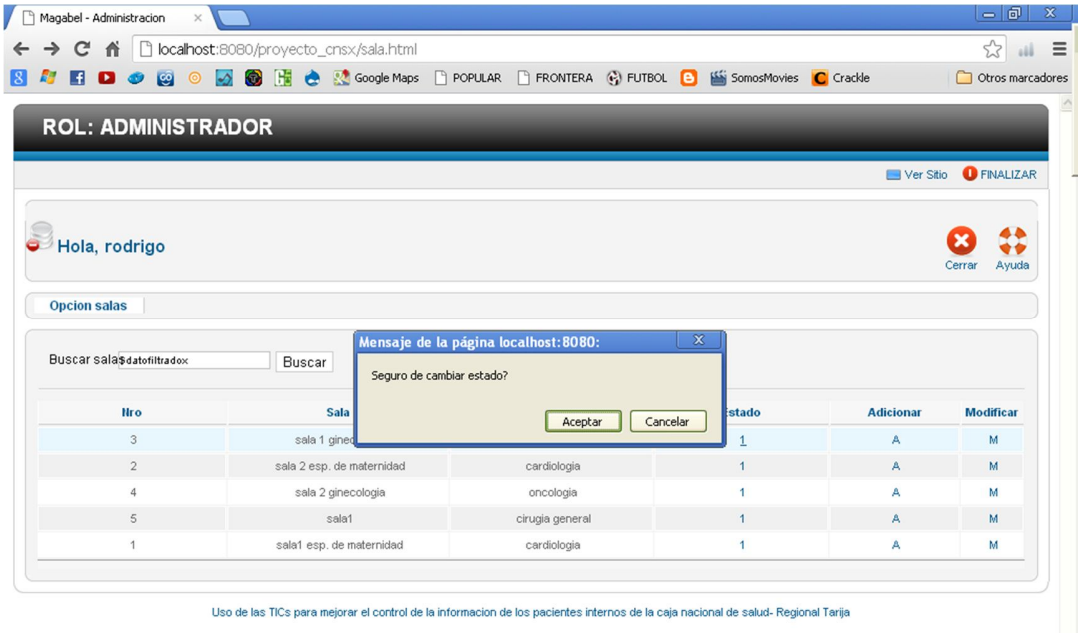


Figura 187. Pantalla Obtenerv Estado (sala)

II.1.10.3.24 Modulo Ingreso

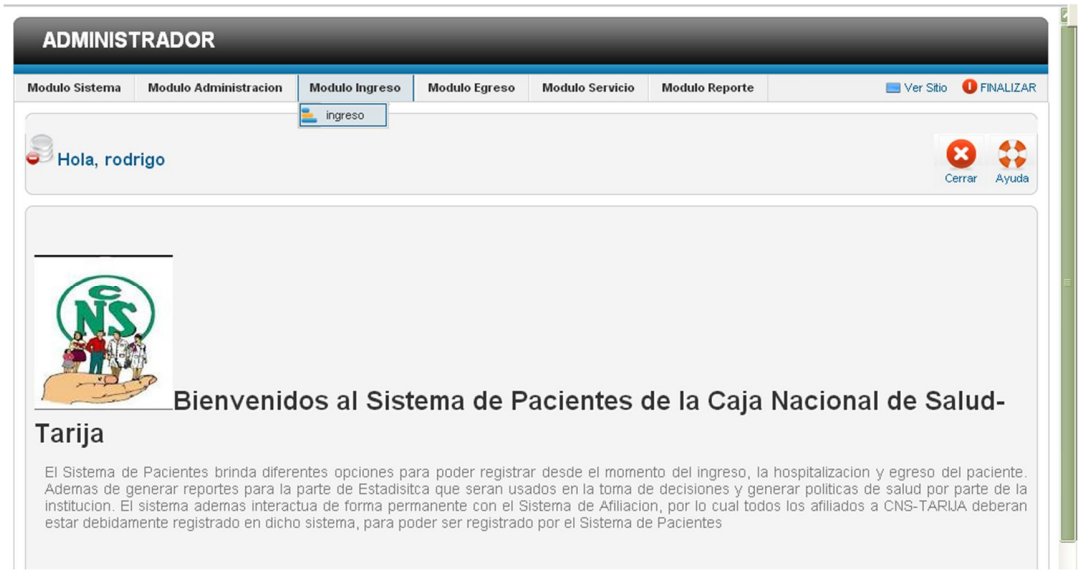


Figura 188. Pantalla Modulo Ingreso

II.1.10.3.25 Modulo Ingreso

Figura 189. Modulo ingreso

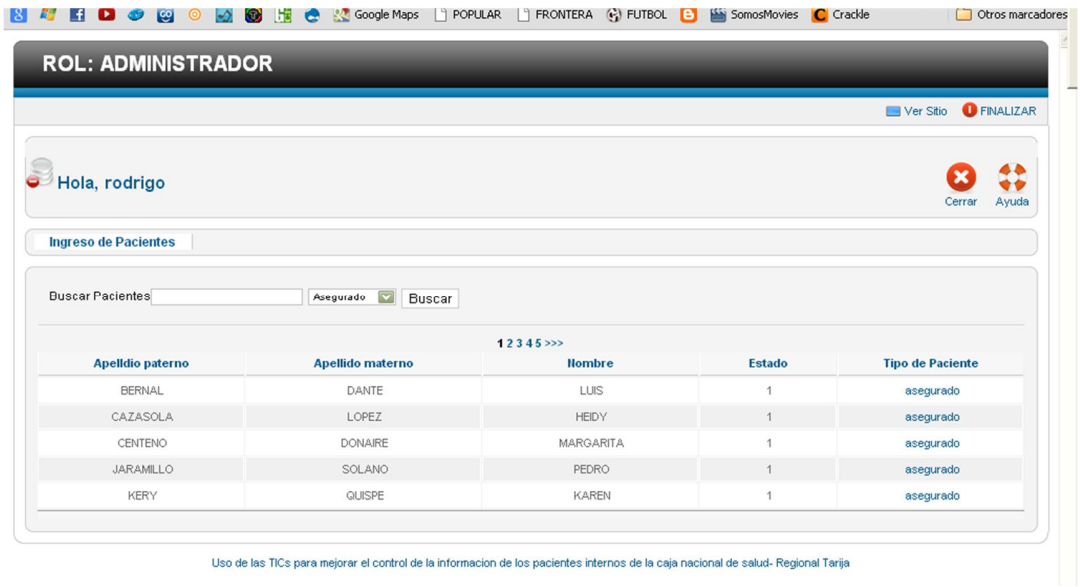


Figura 190. Ingreso Paciente

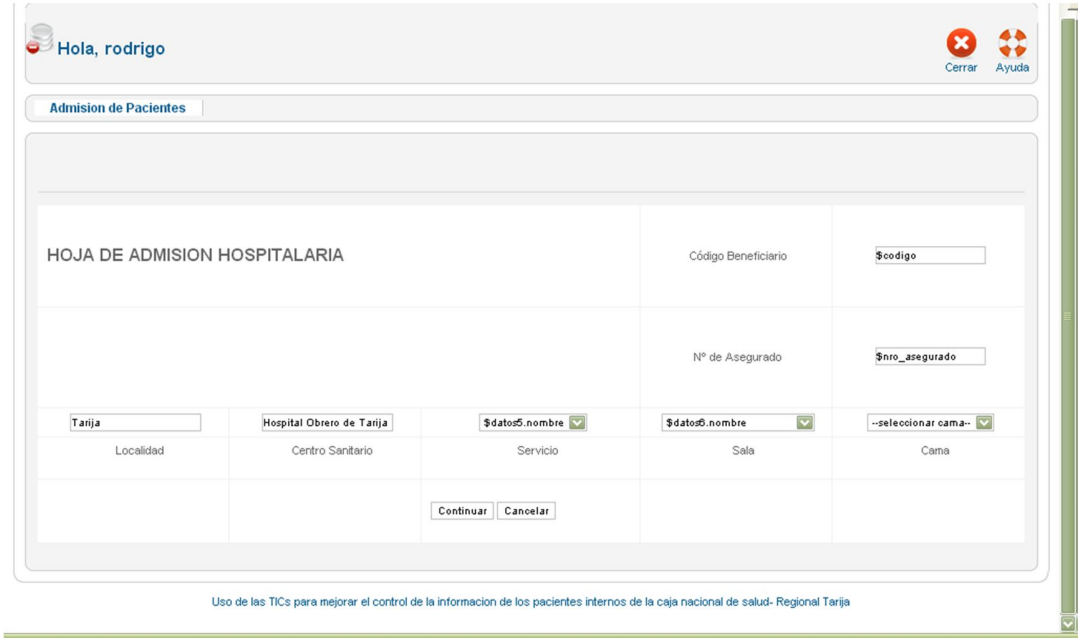


Figura 191. Pantalla llenar_hoja de admisión

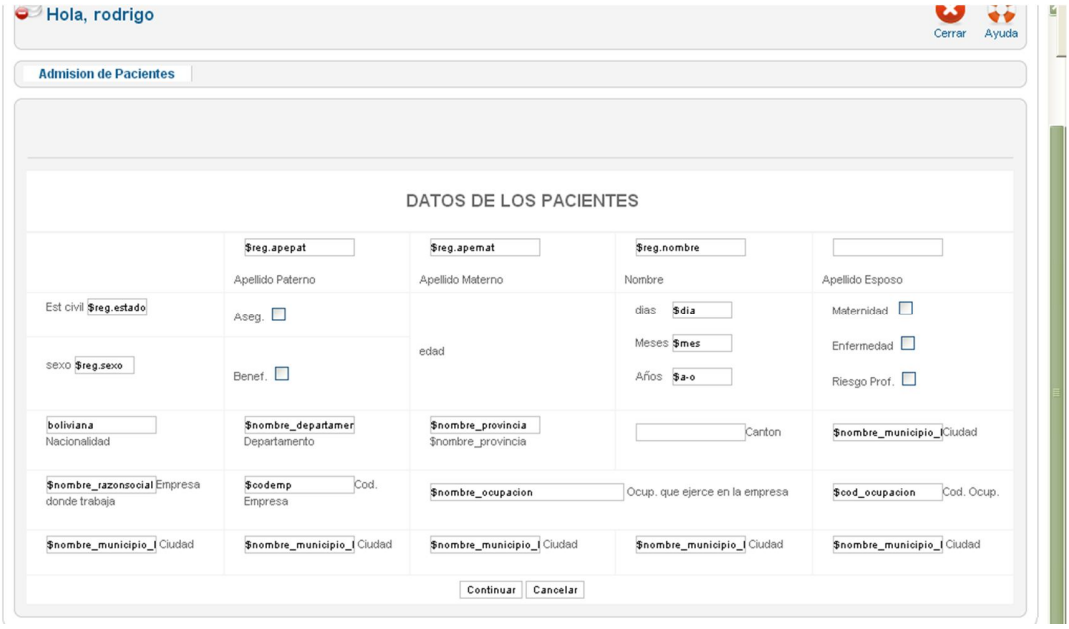


Figura 192. Pantalla llenar_datos paciente

Figura 193. Pantalla llenar_datos de consultorio

II.1.10.3.26 Modulo Egreso

Figura 194. Pantalla Modulo Egreso

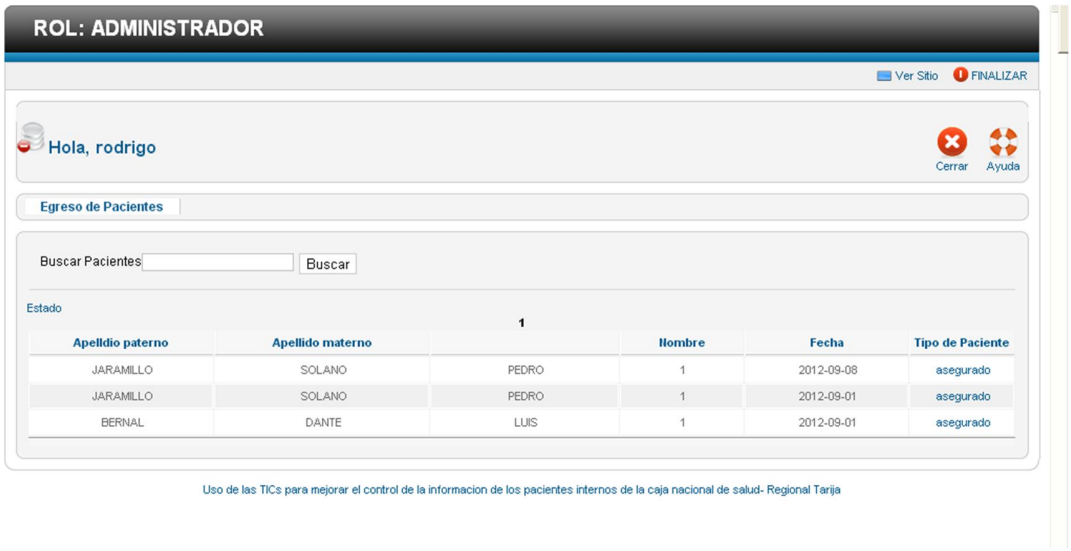


Figura 195. Pantalla Egreso Paciente

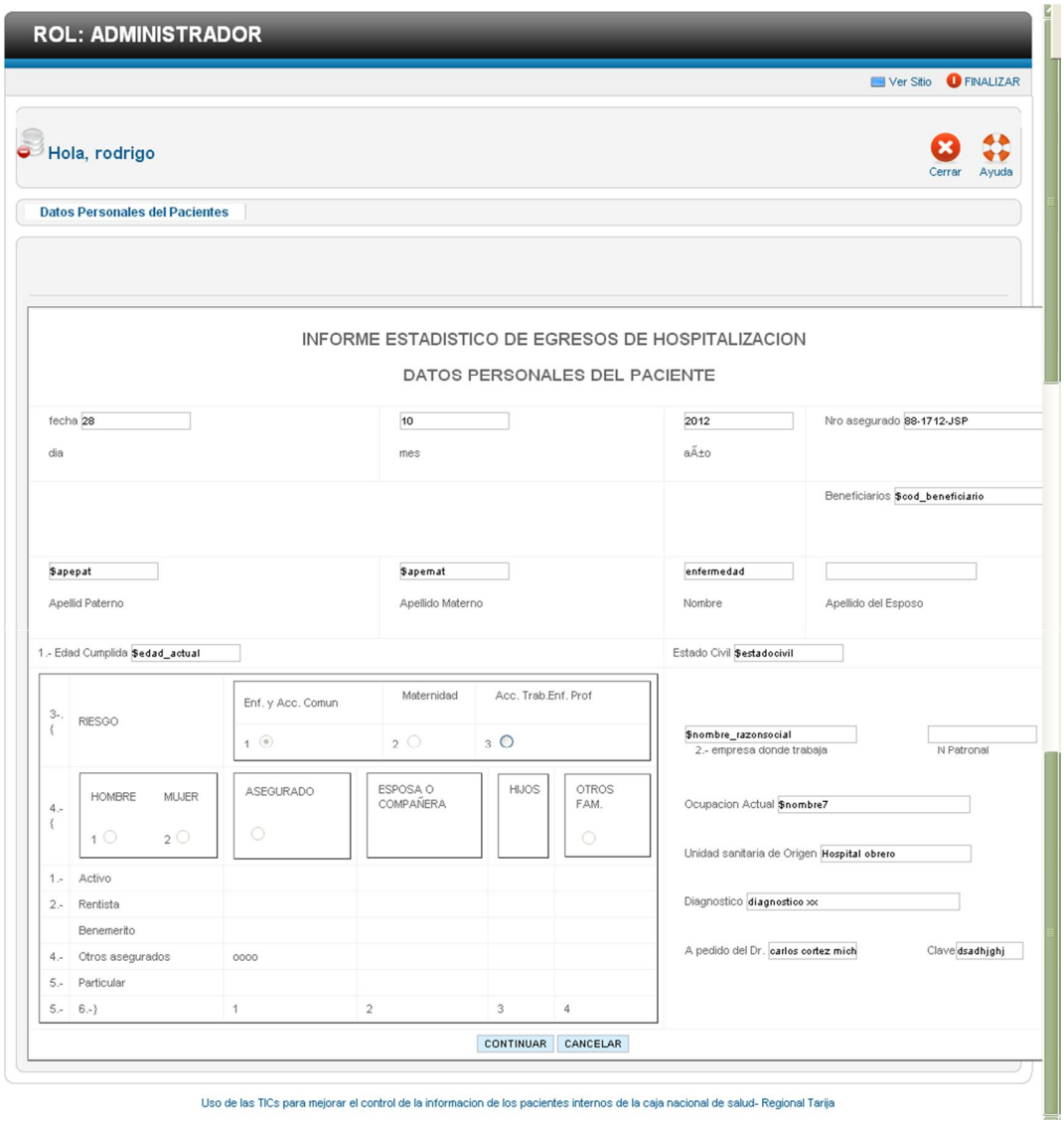


Figura 196. Pantalla informe estadístico de egreso

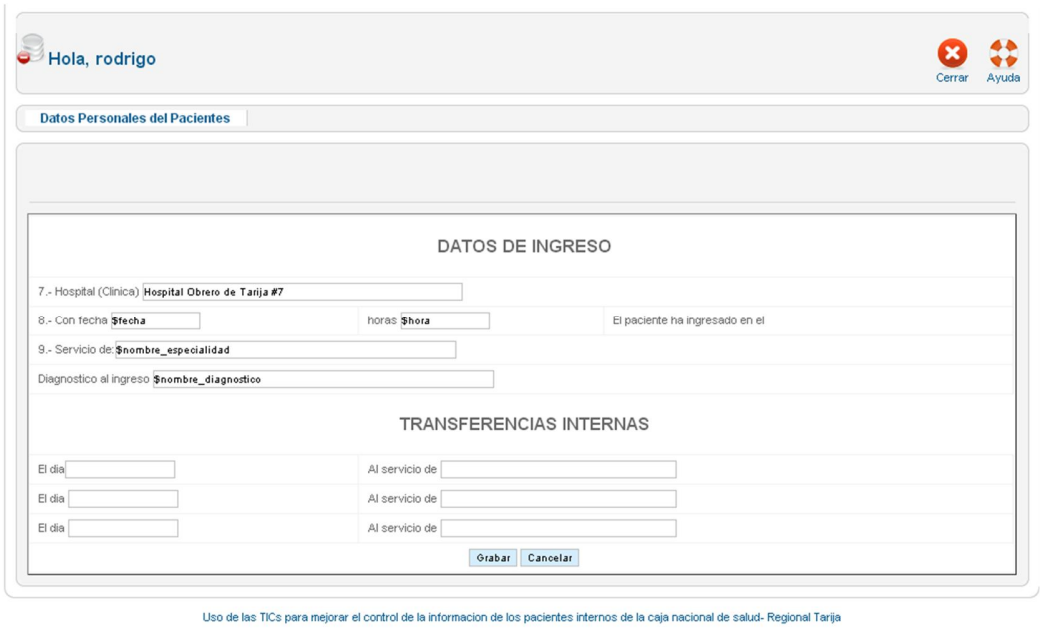


Figura 197. Pantalla datos de ingreso – transferencias internas

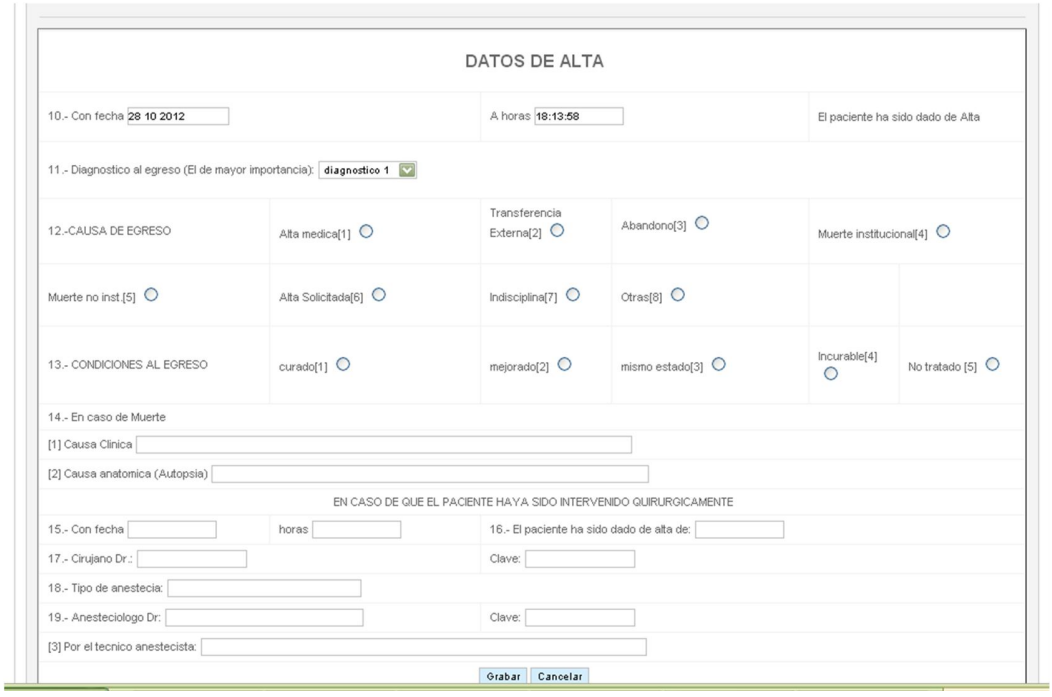


Figura 198. Pantalla Intervencion quirúrgica (datos de alta)

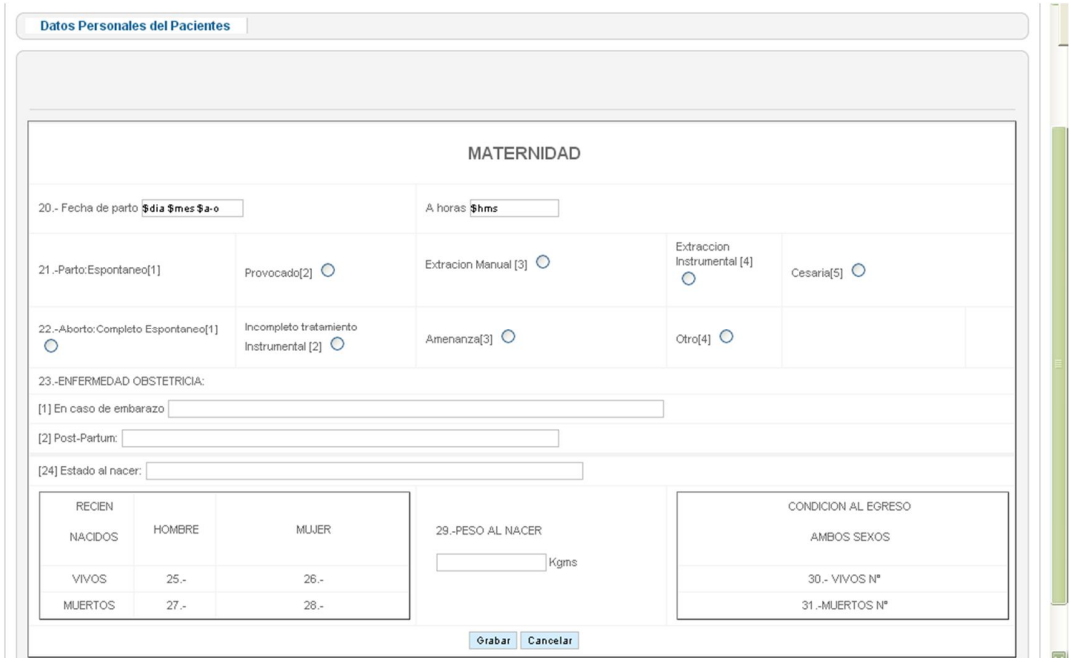


Figura 199. Pantalla Maternidad

II.1.10.3.27 Modulo Servicio

Figura 200. Pantalla Modulo Servicio

II.1.10.3.28 Cirugía

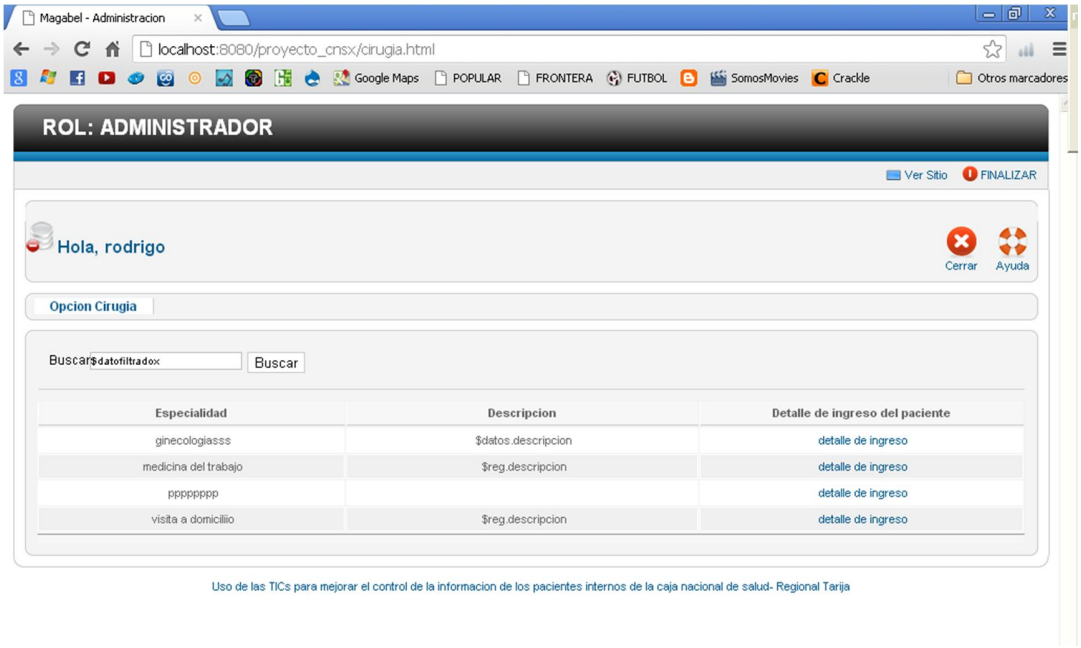


Figura 201. Pantalla Cirugía

Figura 202. Pantalla detalle ingreso paciente

II.1.10.3.29 Maternidad

Figura 203. Pantalla Maternidad

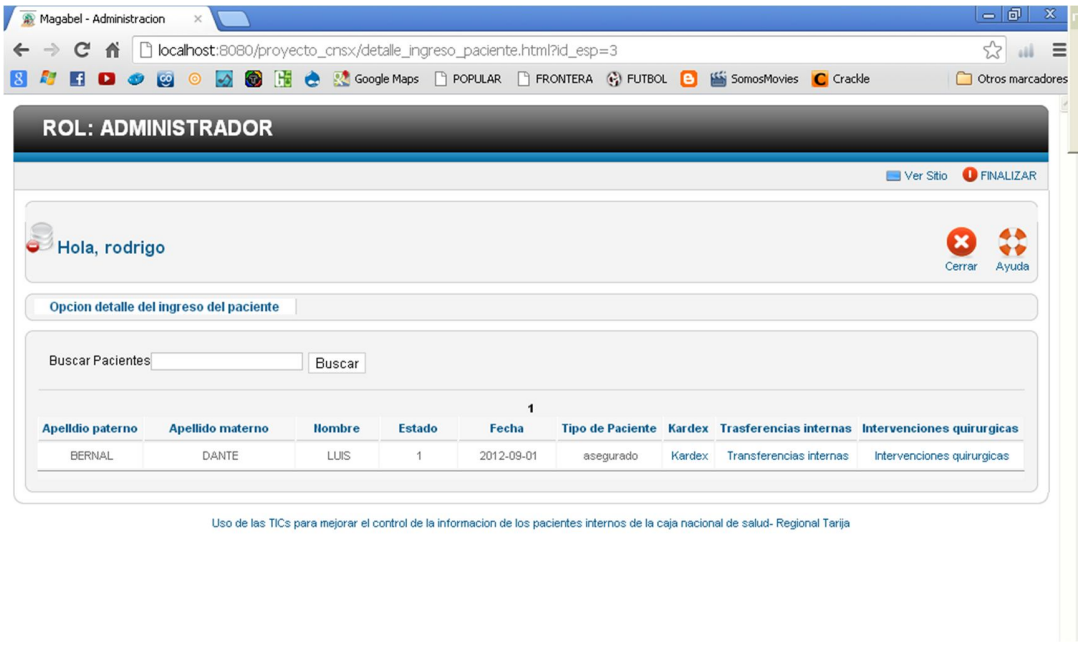


Figura 204. Pantalla Detalleingreso paciente

II.1.10.3.30 Medicina interna

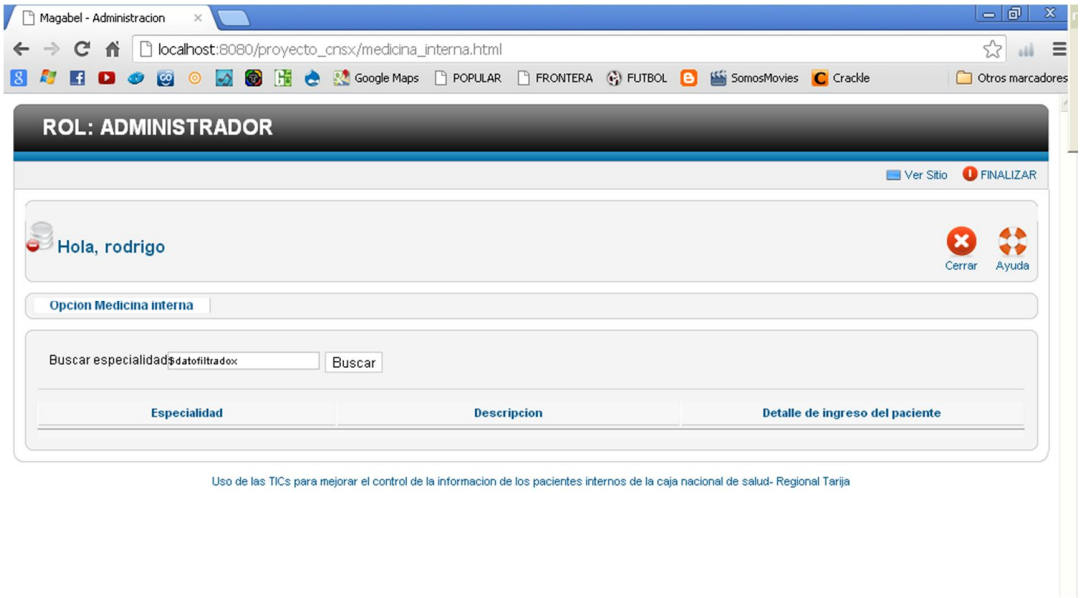


Figura 205. Pantalla Medicina interna

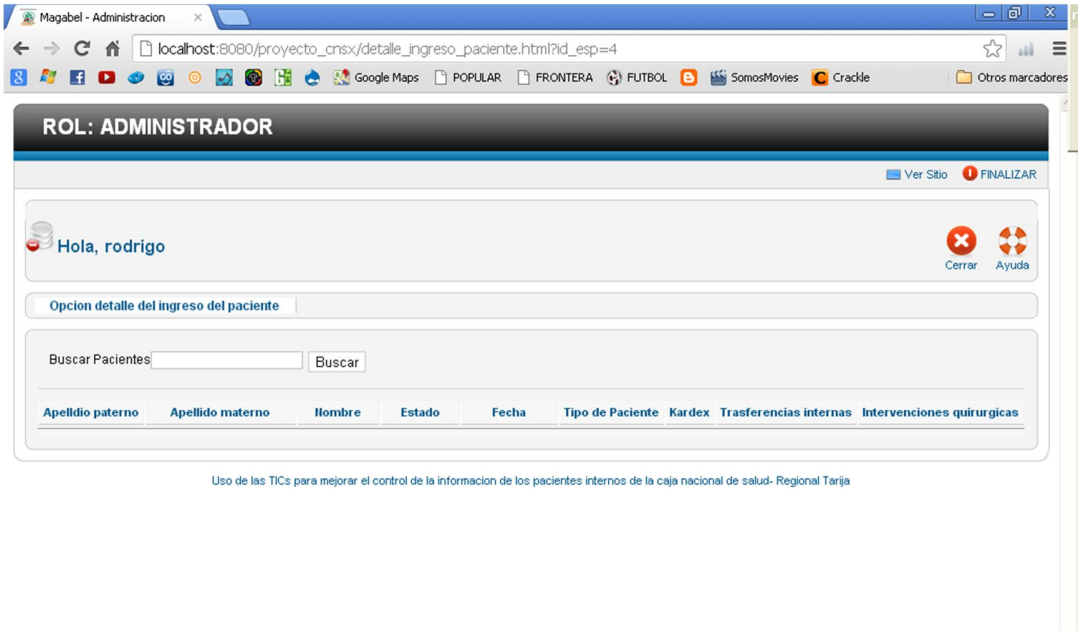


Figura 206. Pantalla Detalle ingreso paciente

II.1.10.3.31 Pediatria

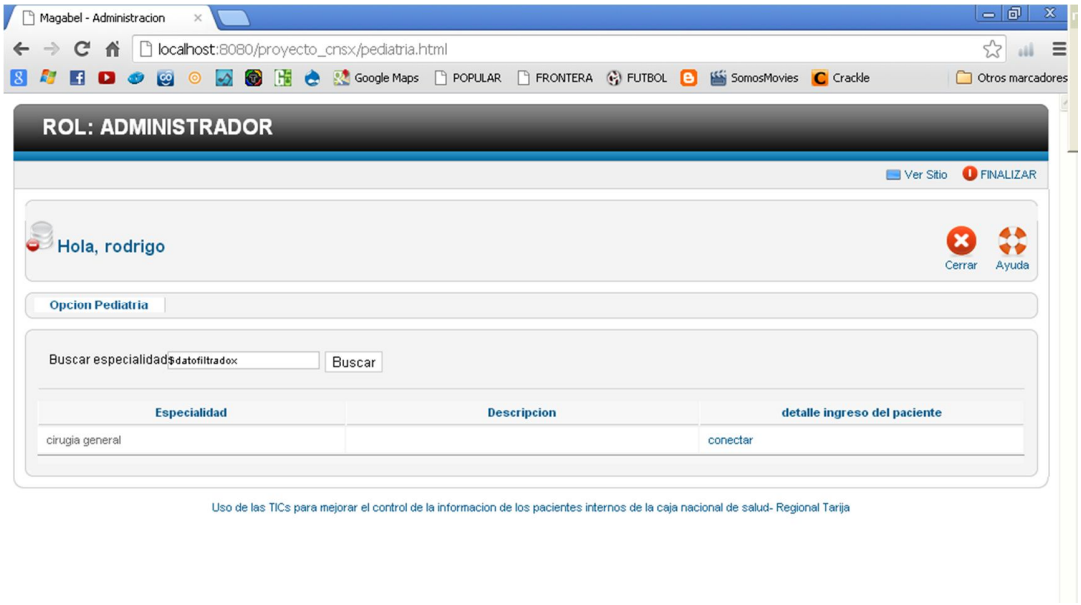


Figura 207. Pantalla Pediatria

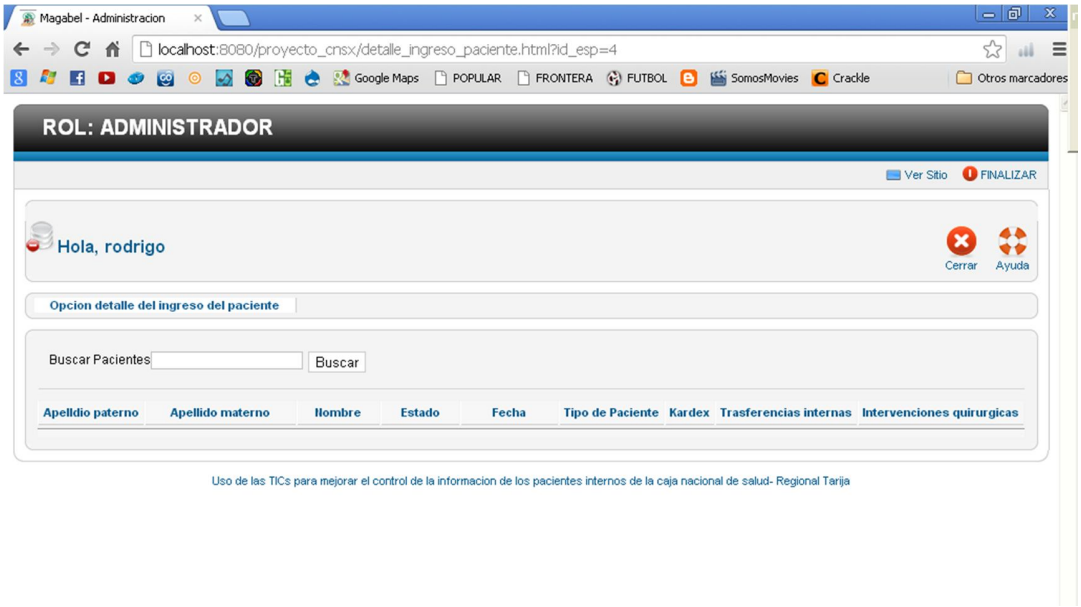


Figura 208. Pantalla Detalle ingreso Paciente

II.1.10.3.32 UTI

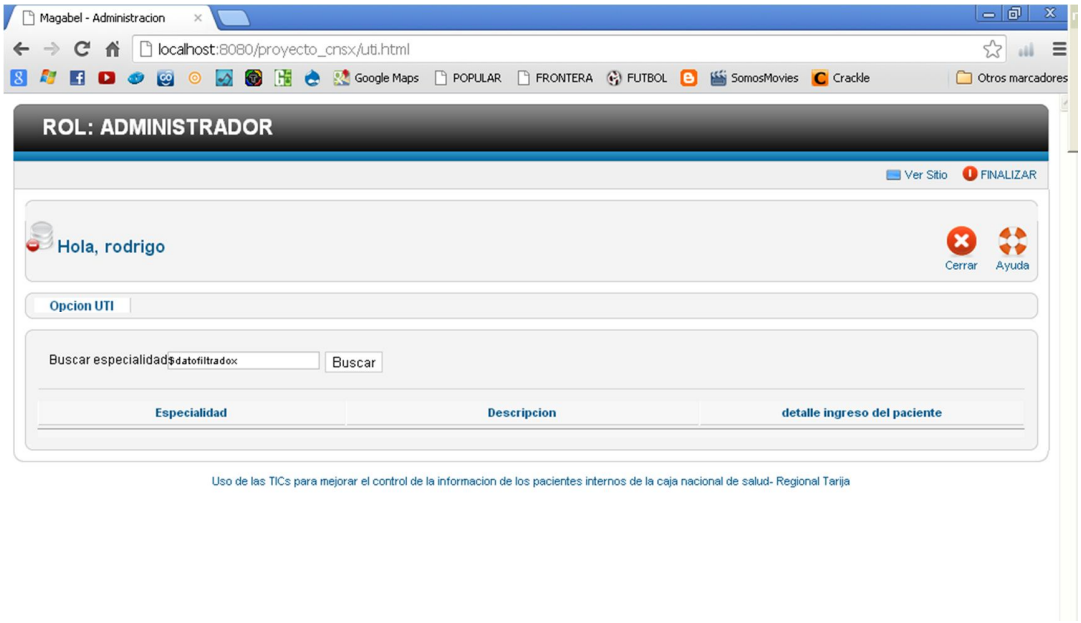


Figura 209. Pantalla UTI

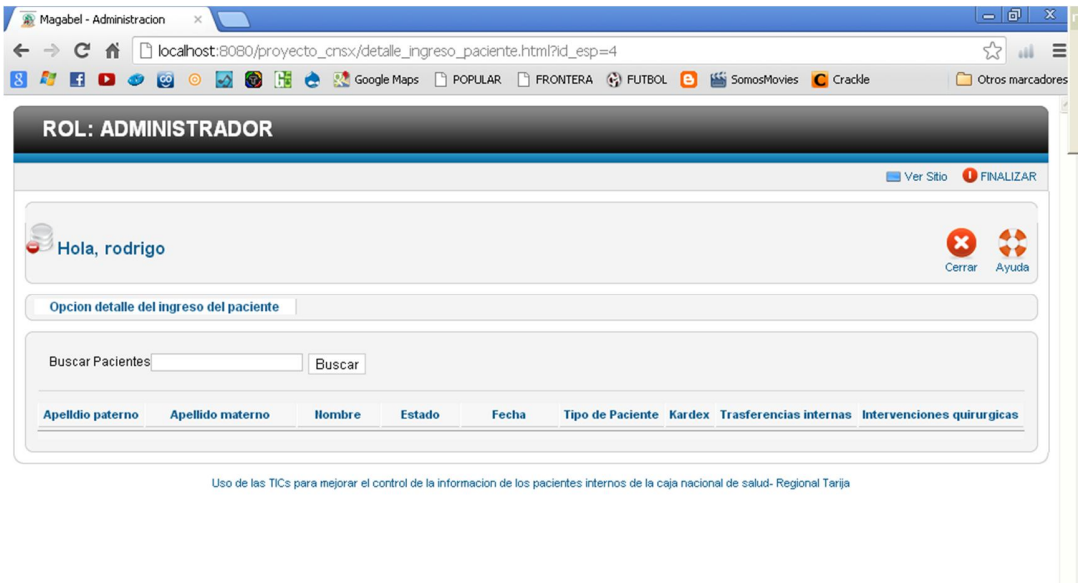


Figura 210. Pantalla Detalle ingreso paciente

II.1.10.3.33 Reportes

Figura 211. Pantalla Reportes

II.1.10.3.34 Ver Hospitalizacion

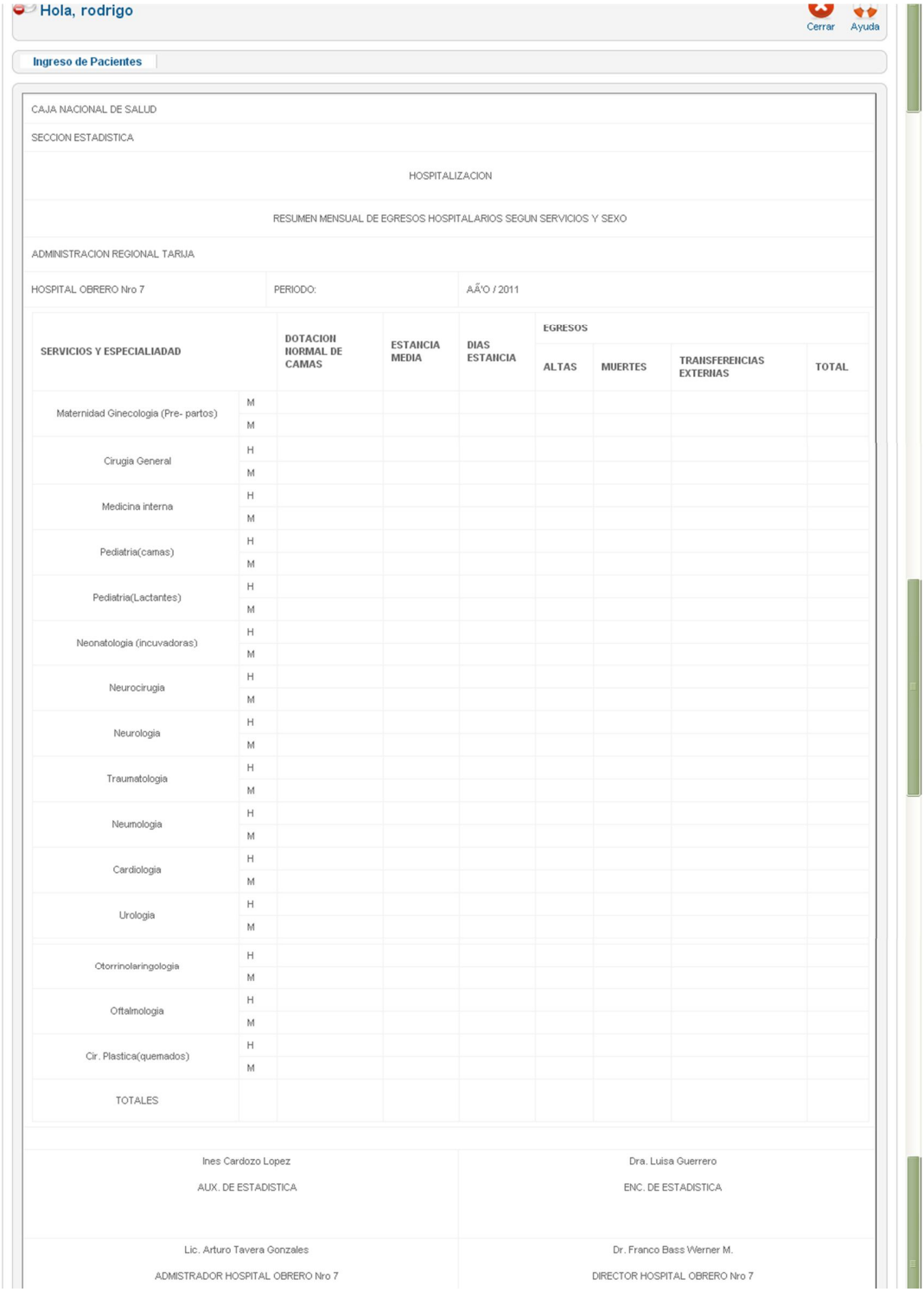


Figura 212. Pantalla Ver Hospitalizacion

II.1.10.3.35 Ver Intervencion quirúrgica por especialidad

Ingreso de Pacientes

CAJA NACIONAL DE SALUD

SECCION ESTADISTICA

RESUMEN REGIONAL DE INTERVENCIONES QUIRURGICAS

POR ESPECIALIDAD Y CLASE DE CIRUGIA

ADMINISTRACION REGIONAL TARIJA

UNIDAD SANITARIA:HOSPITAL OBRERO Nro 7

MES:SEPTIEMBREAÃ'O / 2011

Nro	ESPECIALIADAD	TOTAL	CLASE DE CIRUGIAS		
			MENOR	MEDIANA	MAYOR
1	Cirugia General				
2	Cardiologia				
3	Ginecologia				
4	Neurocirugia				
5	Obstetricia				
6	Oftalmologia				
7	Cirugia Pediatria				
8	Otorrinolaringologia				
9	Cirugia Plastica(Quemados)				
10	Traumatologia				
11	Urologia				
12	Dermatologia				
13	Gastroentologia				
14	Medicina				
15	Odontologia				
16	Cirugia Vascular				
17	Otras Especialidades				
	TOTALES				

CLASE DE ANESTECIA				
TOTAL DE ANESTECIA	RAQUIDEA	GENERAL	PERIDURAL	LOCAL

Ines Cardozo Lopez

AUX. DE ESTADISTICA

Dra. Luisa Guerrero

ENC. DE ESTADISTICA

Lic. Arturo Tavera Gonzales

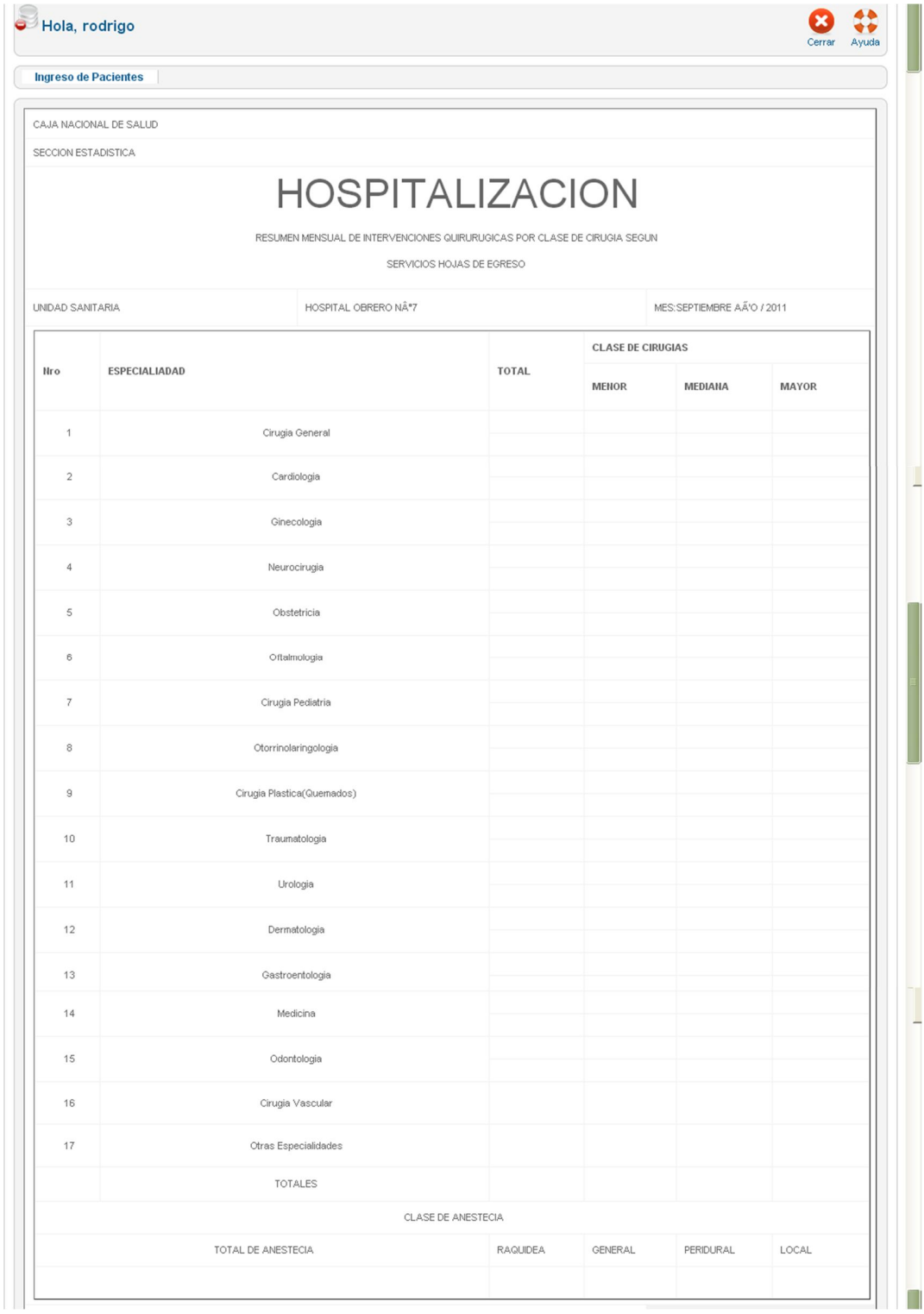
ADMISTRADOR HOSPITAL OBRERO Nro 7

Dr. Franco Bass Werner M.

DIRECTOR HOSPITAL OBRERO Nro 7

Figura 213. Pantalla Ver Intervencion quirúrgica por especialidad

II.1.10.3.36 Modulo Ver Intervencion quirúrgica por clase cirugía



Hola, rodrigo

Cerrar Ayuda

Ingreso de Pacientes

CAJA NACIONAL DE SALUD

SECCION ESTADISTICA

HOSPITALIZACION

RESUMEN MENSUAL DE INTERVENCIONES QUIRURGICAS POR CLASE DE CIRUGIA SEGUN

SERVICIOS HOJAS DE EGRESO

UNIDAD SANITARIA	HOSPITAL OBRERO NÂ°7	MES:SEPTIEMBRE AÃ'O / 2011

Nro	ESPECIALIADAD	TOTAL	CLASE DE CIRUGIAS		
			MENOR	MEDIANA	MAYOR
1	Cirugia General				
2	Cardiologia				
3	Ginecologia				
4	Neurocirugia				
5	Obstetricia				
6	Oftalmologia				
7	Cirugia Pediatria				
8	Otorrinolaringologia				
9	Cirugia Plastica(Quemados)				
10	Traumatologia				
11	Urologia				
12	Dermatologia				
13	Gastroentologia				
14	Medicina				
15	Odontologia				
16	Cirugia Vascular				
17	Otras Especialidades				
	TOTALES				

CLASE DE ANESTECIA				
TOTAL DE ANESTECIA	RAQUIDEA	GENERAL	PERIDURAL	LOCAL

UNIDAD SANITARIA: POLICLINICO Nro 36

Nro	ESPECIALIDAD	TOTAL	CLASE DE CIRUGIA		
			MENOR	MEDIANA	MAYOR

CLASE DE ANESTESIA REALIZADO EN POLICLINICO

TOTAL DE ANESTESIA	RAQUIDEA	GENERAL	PERIDURAL	LOCAL

Ines Cardozo Lopez
AUX. DE ESTADISTICA

Dra. Luisa Guerrero
ENC. DE ESTADISTICA

Lic. Arturo Tavera Gonzales
ADMISTRADOR HOSPITAL OBRERO Nro 7

Dr. Franco Bass Werner M.
DIRECTOR HOSPITAL OBRERO Nro 7

Figura 214. Pantalla Ver Intervencion quirúrgica por clase de cirugia

II.1.11 MODELO DE IMPLEMENTACION

II.1.11.1 Introducción

Este modelo es una colección de componentes y subsistemas que los contienen. Estos componentes incluyen ficheros ejecutables, ficheros de código fuente y de tipo accesorios para implantación y despliegue del sistema

II.1.11.2 Diagrama de Componentes

Sistema de Seguimiento de Documentación, lenguaje de desarrollo: java, los accesos a la información son a través de la base de datos.

II.1.11.2.1 Diagrama de Componentes Iniciar

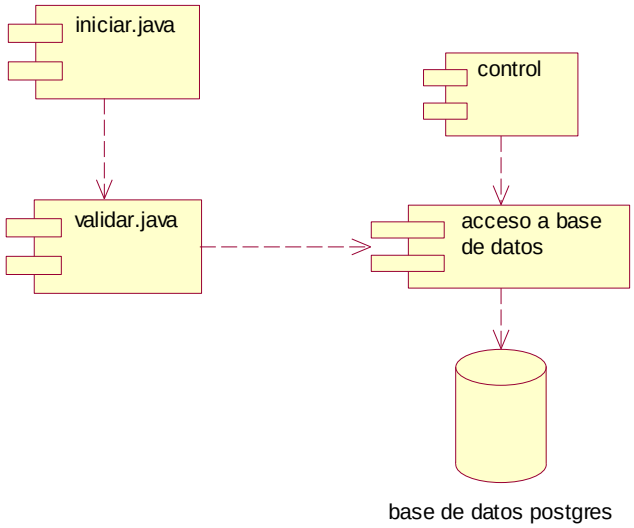


Figura 215. D. Componentes Iniciar

II.1.11.2.2 Diagrama de Componentes Pantalla Principal

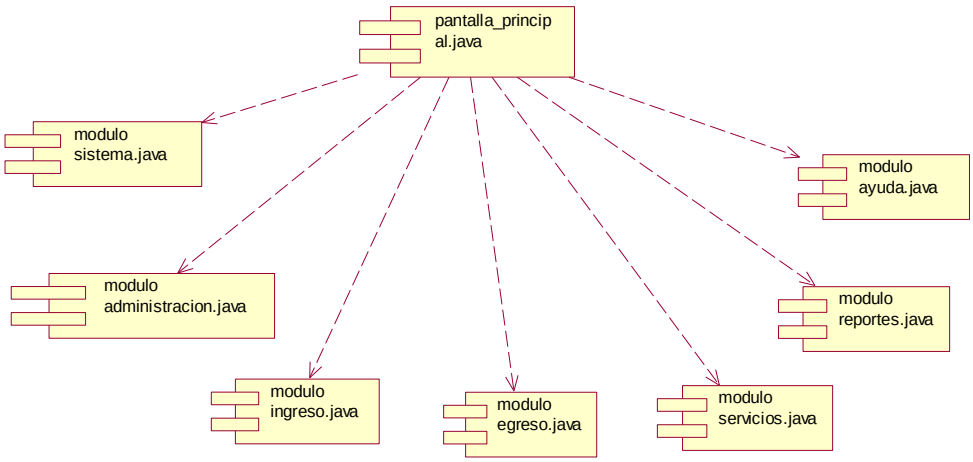


Figura 216. D. Componentes Pantalla Principal

II.1.11.2.3 Diagrama de Componentes modulo sistema

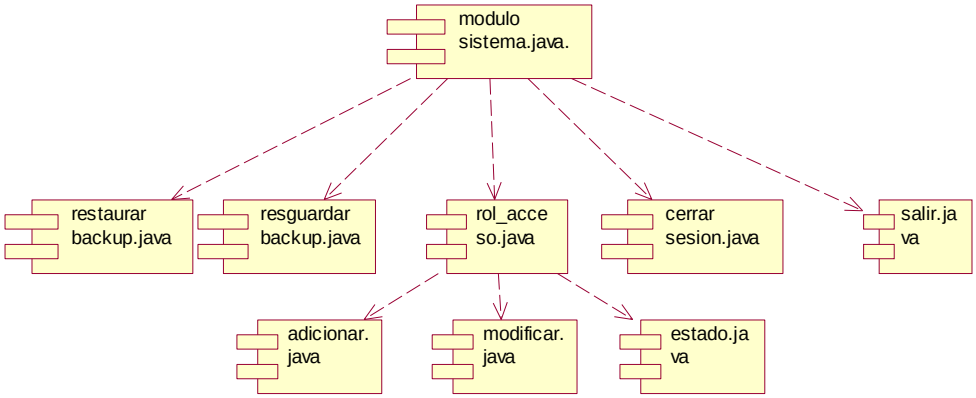


Figura 217. D. Componentes modulo sistema

II.1.11.2.4 Diagrama de Componentes modulo administracion

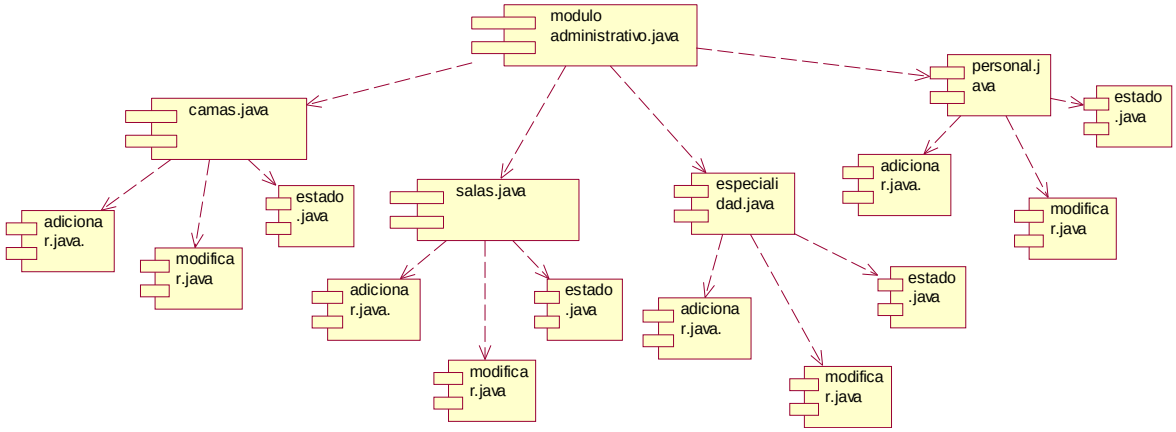


Figura 218. D. Componentes modulo administración

II.1.11.2.5 Diagrama de Componentes Modulo ingreso

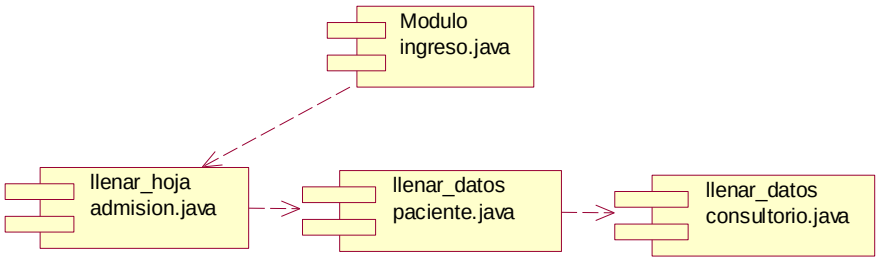


Figura 219. D. Componentes modulo ingreso

II.1.11.2.6 Diagrama de Componentes modulo egreso

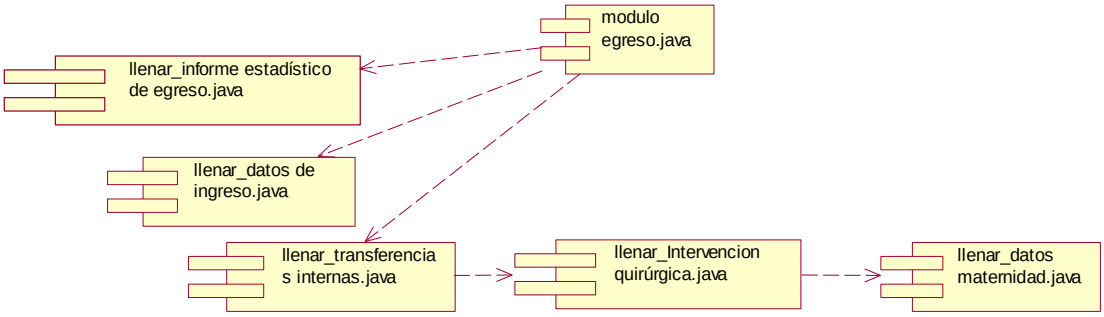


Figura 220. D. Componentes modulo egreso

II.1.11.2.7 Diagrama de Componentes Modulo servicio

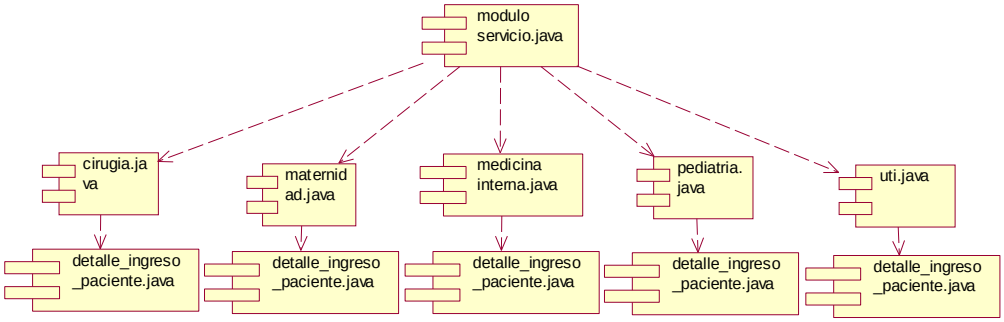


Figura 221. D. Componentes Modulo servicio

II.1.11.2.8 Diagrama de Componente reportes

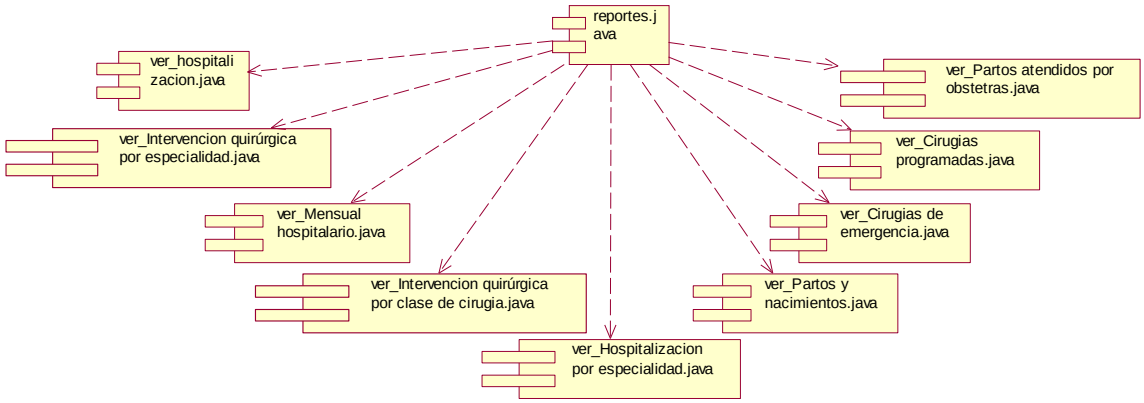


Figura 222. D. Componentes reportes

II.1.11.2.9 Diagrama de Componentes ayuda

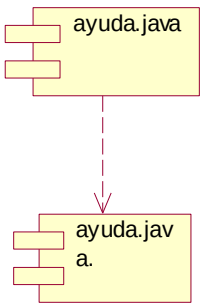


Figura 223. D. Componente ayuda

II.1.12 MODELO DE DESPLIEGUE

II.1.12.1 Introducción

El modelo de Despliegue provee un modelo detallado de la forma en la que los componentes se desplegaran a lo largo de la infraestructura del sistema.

II.1.12.2 Propósito

Mostrar la disposición física de los elementos que intervienen en el Administrador

II.1.12.2 Diagrama de Despliegue

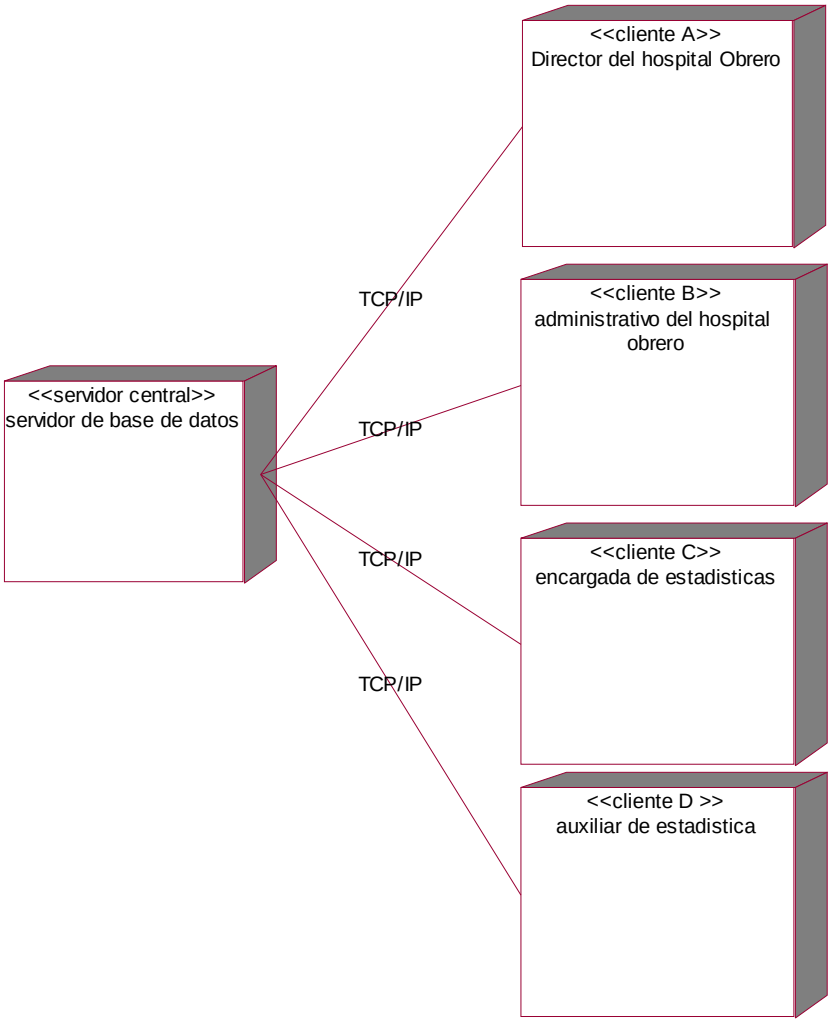


Figura 224. D. Despliegue

II.1.13 ESPECIFICACION DE METODOS

II.1.13.1 Introducción

Las especificaciones de métodos describen el comportamiento de un procedimiento en términos de sus precondiciones y pos condiciones, también referirse solamente a los campos de especificación, a los argumentos del método y a las variables globales, pero nunca a procedimientos concretos.

II.1.13.1.1 Propósito

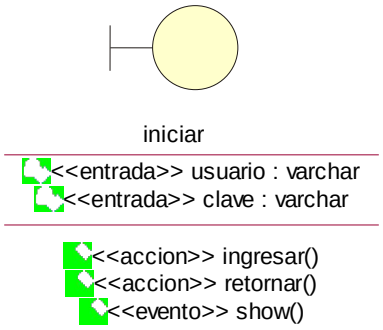
- Mostrar el código fuente concluido
- Comprender la estructura de la programacion

II.1.13.1.2 Alcance

- Se describen solo los métodos que implican la navegación en una pantalla específica.
- Identificar los algoritmos de los procedimientos.

II.1.13.2 Especificación de Métodos

II.1.13.2.1 INICIAR




```
package controlador;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;
```

```
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import modelo.manager.DatoManager;
import modelo.manager.MenuManager;
import modelo.manager.RolProcesoManager;
import modelo.manager.PersonalManager;
import modelo.manager.ProcesoManager;
import modelo.manager.UsuarioManager;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;
import encoder.Base64;
public class index extends MultiActionController implements InitializingBean{
public UsuarioManager usuarioManager;
```

```
public DatoManager datoManager;
public RolProcesoManager rolProcesoManager;
public MenuManager menuManager;
public ProcesoManager procesoManager;
public PersonalManager personalManager;             /**
* @return the personalManager         */
public PersonalManager getPersonalManager() {
return personalManager;        }         /**
* @param personalManager the personalManager to set
*/         public void setPersonalManager(PersonalManager personalManager) {
this.personalManager = personalManager;    }
/**        * @return the rolProcesoManager     */
public RolProcesoManager getRolProcesoManager() {
return rolProcesoManager;     }         /**
* @param rolProcesoManager the rolProcesoManager to set          */
public void setRolProcesoManager(RolProcesoManager rolProcesoManager) {
this.rolProcesoManager = rolProcesoManager;        }
/**        * @return the menuManager             */
public MenuManager getMenuManager() {
return menuManager; }
/**        * @param menuManager the menuManager to set  */
public void setMenuManager(MenuManager menuManager) {
this.menuManager = menuManager; }
```

```
/**       * @return the procesoManager         */
public ProcesoManager getProcesoManager() {
return procesoManager;         }
/**       * @param procesoManager the procesoManager to set       */
public void setProcesoManager(ProcesoManager procesoManager) {
this.procesoManager = procesoManager;      }
/**       * @return the datoManager   */
public DatoManager getDatoManager() {
return datoManager;   }
/**       * @param datoManager the datoManager to set       */
public void setDatoManager(DatoManager datoManager) {
this.datoManager = datoManager;     }
/**       * @return the usuarioManager         */
public UsuarioManager getUsuarioManager() {
return usuarioManager;         }
/**       * @param usuarioManager the usuarioManager to set        */
public void setUsuarioManager(UsuarioManager usuarioManager) {
this.usuarioManager = usuarioManager;       }
public void afterPropertiesSet() throws Exception {}
public index() {          }
public ModelAndView index(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
HttpSession session=request.getSession(true);
try{
```

```
if(request.getParameter("finalizar")!=null){
session.removeAttribute("xusuario");//creado en el metodo validacion
session.removeAttribute("xclave");//creado en el metodo validacion
session.removeAttribute("foto");//creado en el metodo validacion
session.removeAttribute("id_usu");//creado en el metodo validacion
session.removeAttribute("cod_persona");//creado en el metodo ingreso
session.removeAttribute("rol");//creado en en metodo validacion
session.removeAttribute("id_ser");
session.removeAttribute("especialidad");
session.removeAttribute("cod_beneficiario");
session.removeAttribute("id_dip");
                        session.removeAttribute("cod_asegurado");
                        session.removeAttribute("nro_asegurado");
                        session.removeAttribute("cod_beneficiario");
                        session.removeAttribute("codigo");
                        session.removeAttribute("booleano");
                        session.removeAttribute("bandera");
                        session.removeAttribute("estado");
                        session.removeAttribute("estado2");
                        session.removeAttribute("tipo");//session del metodo ingreso
session.removeAttribute("id_cam");
System.out.println("veamos uuuuuuuuuuuu");                  }
}catch (Exception e) {                          System.out.println("exception en el index
"+e.getLocalizedMessage()); }
```

```
return new ModelAndView("index");    }

public ModelAndView instructivo(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {

HttpSession session=request.getSession(true);

Map p = new HashMap();

return new ModelAndView("instructivo");    }

public ModelAndView informacion(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {

HttpSession session=request.getSession(true);

Map p = new HashMap();

return new ModelAndView("instructivo");    }

public ModelAndView noticia(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {

HttpSession session=request.getSession(true);

Map p = new HashMap();

return new ModelAndView("instructivo");    }

public ModelAndView comunicado(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {

HttpSession session=request.getSession(true);

Map p = new HashMap();

return new ModelAndView("instructivo");    }

public ModelAndView servicio(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {

HttpSession session=request.getSession(true);
```

```
Map p = new HashMap();
return new ModelAndView("instructivo");    }
public ModelAndView profesionales(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
HttpSession session=request.getSession(true);
Map p = new HashMap();
return new ModelAndView("instructivo");    }
public ModelAndView acceso(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
HttpSession session=request.getSession(true);
Map p = new HashMap();
return new ModelAndView("acceso");    }
public ModelAndView validacion(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
HttpSession session=request.getSession(true);
Map p = new HashMap();
String usuario=null,clave=null;
try {
if(request.getParameter("xusuario")!=null){
usuario=Base64.encode(request.getParameter("xusuario"));
clave=Base64.encode(request.getParameter("xclave"));
session.setAttribute("xusuario",usuario);
session.setAttribute("xclave",clave);                    }
```

```
} catch (Exception e) {System.out.println("ummmm
"+e.getMessage());if(session.getAttribute("xusuario")==null){
p.put("men","Negado el Acceso");
p.put("url","index.html");
return new ModelAndView( "mensaje",p);              }
try {     List x1 =
this.usuarioManager.getValidaUsuario(session.getAttribute("xusuario").toString(),ses
sion.getAttribute("xclave").toString());
int id_usu=0;
if(x1.size()==1){
for (Iterator i = x1.iterator(); i.hasNext();) {
Object[] row1 = (Object[]) i.next();
id_usu=(Integer)row1[0];
session.setAttribute("foto",row1[3]);
session.setAttribute("id_usu", id_usu);                          }
try{
List x2 = this.menuManager.getUsuarioMenu(id_usu);
int id_men=0;
String nombrem="",nombrep="",enlace="",rol;
int id_rol=0;
ArrayList lista1=new ArrayList();
for (Iterator i = x2.iterator(); i.hasNext();) {
Map map = new HashMap();
Object[] row1 = (Object[]) i.next();
```

```
id_men= (Integer) row1[0];
nombrem= (String) row1[1];
int id_pro=(Integer)row1[2];
nombrep= (String) row1[3];
enlace= (String) row1[4];
id_rol= (Integer) row1[5];
rol= (String) row1[6];
map.put("id_men",id_men);
map.put("nombrem",nombrem);
map.put("id_pro", id_pro);
map.put("nombrep",nombrep);
map.put("enlace",enlace);
map.put("id_rol",id_rol);
map.put("rol",rol);
session.setAttribute("rol", rol);
lista1.add(map);                    }
p.put("datos1",lista1);
}catch (Exception e) {
System.out.println("exception "+e.getLocalizedMessage());          }
System.out.println("..... 
"+Base64.decode(session.getAttribute("xusuario").toString())+"
"+session.getAttribute("xusuario").toString());
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
```

```
p.put("rol",session.getAttribute("rol"));
return new ModelAndView("validacion",p);                    }catch(Exception s){
System.out.println("MENSAJE error postgres "+s.getMessage());    }
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("men","No existe como usuario del sistema");
p.put("url","index.html");
return new ModelAndView("mensaje",p);              }
public ModelAndView lista_pacientes_internos(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
HttpSession session=request.getSession(true);
Connection conexion = null; //Objeto para la conexión a la BD
Statement sentencia = null; //Objeto que ejecuta sentencias
ResultSet resultados = null;//Objeto que guardar resultados
Map p = new HashMap();
if(session.getAttribute("xusuario")==null){
p.put("men","Negado el Acceso");
p.put("url","index.html");
return new ModelAndView( "mensaje",p);              }
ArrayList milistax = new ArrayList();
ArrayList milista = new ArrayList();
Map mapx = new HashMap();
String datofiltro="";
int rango=1;
if(request.getParameter("xdatofiltro")==null){
```

```
datofiltro="";
}else{datofiltro=request.getParameter("xdatofiltro");        }
if(request.getParameter("xinter")==null){
rango=1;
}else{
rango=Integer.parseInt(request.getParameter("xinter"));              }
try{
//generador de el paginador
//Leemos el driver de Postgresql
Class.forName("org.postgresql.Driver");
//Nos conectamos a la BD local
conexion = DriverManager.getConnection (
"jdbc:postgresql://localhost:5432/afiliacion",
postgres","123456");
//Creamos una sentencia a partir de la conexión
sentencia=conexion.createStatement();
double num=0;
resultados=sentencia.executeQuery("SELECT
s.cod_persona,s.cod_beneficiario,s.tipo,s.id_hfi,s.fecha,s.id_ser,s.nombres," +
"s.id_esp,s.nombre,s.id_cam,s.numero,p.apepat,p.apemat,p.nombre,p.estado FROM
persona p "+
"INNER JOIN dblink('dbname=cns port=5432 host=localhost "+
```

```
"user=postgres password=123456','SELECT
dip.cod_persona,dip.cod_beneficiario,dip.tipo,hfi.id_hfi,hfi.fecha,se.id_ser,se.nombre
,e.id_esp,e.nombre,c.id_cam,c.numero "+

"FROM detalleingresopacientes dip,horafechaingresos hfi,especialidades e,camas
c,servicios se where se.id_ser=e.id_ser and dip.id_cam=c.id_cam and
dip.id_hfi=hfi.id_hfi and dip.id_esp=e.id_esp "+

"order by hfi.fecha desc ') AS s(cod_persona int,cod_beneficiario int,tipo
varchar,id_hfi int,fecha date,id_ser int,nombres varchar,id_esp int,nombre
varchar,id_cam int,numero int) ON p.cod_persona=s.cod_persona");

while(resultados.next()) {

num++;System.out.println("----- "+num)                    }

double xx=Math.ceil(num / 5);

if(rango>1){

mapx = new HashMap();

mapx.put("rango","<a id='x' onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='lista_pacientes_internos.html?xdatofiltro="+datofiltro+"&xinter="+(rango-
1)+"">"+"<<<"+"</a>");

milistax.add(mapx);          }

for(int i=1;i<=xx;i++){

mapx = new HashMap();

if(i==rango){

mapx.put("rango",""+i+"");                    }else{

mapx.put("rango","<a id="'+i+'" onMouseOver='Over(this.id)'
onMouseOut='Out(this.id)'
href='lista_pacientes_internos.html?xdatofiltro="+datofiltro+"&xinter="+i+"">"+i+"<
/a>");                    }
```

```
milistax.add(mapx);                    }
                    if(rango<xx){
                        mapx = new HashMap();
                        mapx.put("rango","<a id='y'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='lista_pacientes_internos.html?xdatofiltro="+datofiltro+"&xinter="+(rango+1)+
"'>"+">>>"+"</a>");
                        milistax.add(mapx);          }
                    //lista de los datos en 5 filas
List mat =
this.detalleingresopacienteManager.getListaDetalleIngresoPacienteFiltro(datofiltro,ra
ngo);
                    //Creamos una sentencia a partir de la conexión
resultados=sentencia.executeQuery("SELECT
s.cod_persona,s.cod_beneficiario,s.tipo,s.id_hfi,s.fecha,s.id_ser,s.nombres," +

"s.id_esp,s.nombre,s.id_cam,s.numero,p.apepat,p.apemat,p.nombre,p.estado FROM
persona p "+
                        "INNER JOIN dblink('dbname=cns port=5432
host=localhost "+
                        "user=postgres password=123456','SELECT
dip.cod_persona,dip.cod_beneficiario,dip.tipo,hfi.id_hfi,hfi.fecha,se.id_ser,se.nombre
,e.id_esp,e.nombre,c.id_cam,c.numero "+
                        "FROM detalleingresopacientes
dip,horafechaingresos hfi,especialidades e,camas c,servicios se where
```

```
se.id_ser=e.id_ser and dip.id_cam=c.id_cam and dip.id_hfi=hfi.id_hfi and
dip.id_esp=e.id_esp "+
                                        "order by hfi.fecha desc ') AS s(cod_persona
int,cod_beneficiario int,tipo varchar,id_hfi int,fecha date,id_ser int,nombres
varchar,id_esp int,nombre varchar,id_cam int,numero int) ON
p.cod_persona=s.cod_persona "+
                                        "limit 5 offset (("+rango+"-1)*5) ");
                                        while(resultados.next()) {
                                        Map map = new HashMap();
                                        map.put("cod_persona", resultados.getInt(1));
                                        /*map.put("cod_beneficiario",
resultados.getInt(2));
                map.put("apepat", resultados.getString(3));
                map.put("apemat", resultados.getString(4));
                map.put("nombre", resultados.getString(5));
                map.put("id_hfi",resultados.getInt(6));
                session.setAttribute("id_dip",resultados.getInt(7));
                map.put("fecha",resultados.getDate(8));
                map.put("estado", resultados.getInt(9));
                map.put("tipo", resultados.getString(10));*/
milista.add(map);               }
try {
if (conexion!=null)
conexion.close();
```

```
}catch (SQLException e3) {
System.out.println("ERROR:Fallo al desconectar de la BD: "+
e3.getMessage());                }
}catch (Exception e) {
                        System.out.println("que fue "+e.getLocalizedMessage());   }
//envio los intervalos
p.put("datos", milista);
p.put("intervalo",milistax);
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("xclave", session.getAttribute("xclave"));
p.put("datofiltradox",datofiltro);
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                p.put("foto",session.getAttribute("foto"));
                p.put("rol",session.getAttribute("rol"));
            return new ModelAndView("lista_pacientes_internos",p);    }}
```

II.1.13.2.2 Modulo sistema

```
package controlador.ModuloSistema;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
```

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import modelo.dominio.Menu;
import modelo.dominio.Rol;
import modelo.dominio.RolProceso;
import modelo.manager.MenuManager;
import modelo.manager.PersonalManager;
import modelo.manager.RolManager;
import modelo.manager.RolProcesoManager;
import modelo.manager.RolProcesoManager;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;
import encoder.Base64;
public class sistema extends MultiActionController implements InitializingBean{
        PersonalManager personalManager;
        RolManager rolManager;
        RolProcesoManager rolProcesoManager;
        MenuManager menuManager;
        public void afterPropertiesSet() throws Exception {        }
        public sistema() {}
```

```
/**       * @return the personalManager        */
public PersonalManager getPersonalManager() {
return personalManager;       }
/**       * @param personalManager the personalManager to set     */
public void setPersonalManager(PersonalManager personalManager) {
this.personalManager = personalManager;   }
/**       * @return the rolManager      */
public RolManager getRolManager() {
                return rolManager;     }
/**       * @param rolManager the rolManager to set           */
public void setRolManager(RolManager rolManager) {
                this.rolManager = rolManager;         }
public RolProcesoManager getRolProcesoManager() {
                return rolProcesoManager;     }
public void setRolProcesoManager(RolProcesoManager rolProcesoManager) {
this.rolProcesoManager = rolProcesoManager;}
public MenuManager getMenuManager() {
return menuManager; }
public void setMenuManager(MenuManager menuManager) {
                this.menuManager = menuManager; }}
```

II.1.13.2.3 Resguadar backup

```
public ModelAndView resguardar(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
HttpSession session=request.getSession(true);
```

```
Map p = new HashMap();
if(session.getAttribute("xusuario")==null){
p.put("men","Negado el Acceso");
p.put("url","index.html");
return new ModelAndView( "mensaje",p);   }
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
p.put("rol",session.getAttribute("rol"));
return new ModelAndView("mensaje",p);}
```

II.1.13.2.4 Restaurar backup

```
public ModelAndView restaurar(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
HttpSession session=request.getSession(true);
Map p = new HashMap();
if(session.getAttribute("xusuario")==null){
p.put("men","Negado el Acceso");
p.put("url","index.html");
return new ModelAndView( "mensaje",p)  }
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
        p.put("foto",session.getAttribute("foto"));
        p.put("rol",session.getAttribute("rol"));
        return new ModelAndView("mensaje",p);
```

II.1.13.2.4 Rol_acceso

```
public ModelAndView rol_acceso(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
Map p = new HashMap();
HttpSession session=request.getSession(true);
if(session.getAttribute("xusuario")==null){
p.put("men","Negado el Acceso");
p.put("url","index.html");
return new ModelAndView( "mensaje",p);              }
List lista=this.rolManager.getListaRol();
p.put("datos", lista);
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
p.put("rol",session.getAttribute("rol"));
return new ModelAndView("sistema/rol_acceso",p);    }
```

II.1.13.2.5 Adicionar rol

```
public ModelAndView adicionar_rol(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
Map p = new HashMap();
                HttpSession session=request.getSession(true);
                ArrayList mi_lisra=new ArrayList();
                if(session.getAttribute("xusuario")==null){
p.put("men","Negado el Acceso");
p.put("url","index.html");
```

```
return new ModelAndView( "mensaje",p);
}p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
p.put("rol",session.getAttribute("rol"));
return new ModelAndView("sistema/adicionar_menu",p);    }
public ModelAndView adicionar_rol2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
                HttpSession session=request.getSession(true);
                if(session.getAttribute("xusuario")==null){
                   p.put("men","Negado el Acceso");
                p.put("url","index.html");
return new ModelAndView( "mensaje",p);            }
List lista=this.rolManager.getListaRol();
Rol r=new Rol();
                r.setId_rol(lista.size()+1);
                r.setNombre(request.getParameter("nombre"));
                this.rolManager.saveRol(r);
                RolProceso rp=new RolProceso();
                try{
String d[]=request.getParameterValues("proceso");
//System.out.println("---------- "+d.length);
for (int i = 0; i < d.length; i++) {
rp.setId_rol(r.getId_rol());
```

```
rp.setId_pro(Integer.parseInt(d[i]));
//System.out.println("_____ "+r.getId_rol()+"   "+d[i]);
this.rolProcesoManager.saveRolProceso(rp);          }
//System.out.println("..... "+d[0]+"   "+d[1]+"  "+d[2]+"  "+d[3]);          }catch
(Exception e) {
                        p.put("url","rol_acceso.html");
                        p.put("men","lo sentimos no adicionado correctamente");
put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
p.put("rol",session.getAttribute("rol"));        }
                p.put("url","rol_acceso.html");
                p.put("men","adicionado correctamente");
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
p.put("rol",session.getAttribute("rol"));
return new ModelAndView("mensaje",p);    }
```

II.1.13.2.6 Modificar rol

```
public ModelAndView modificar_rol(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
Map p = new HashMap();
HttpSession session=request.getSession(true);
try{                    int i=1;                    List
lista=this.rolManager.getListaRol2(Integer.parseInt(request.getParameter("id_rol")));
for (Iterator iterator = lista.iterator(); iterator.hasNext();) {
```

```
Object[] name = (Object[]) iterator.next();
.put("id_pro"+(Integer)name[0],(Integer)name[0]);
p.put("id_rol",(Integer)name[2]);
p.put("rol",(String)name[3]);                    }
}catch (Exception e) {
System.out.println("::::::--------- ::: "+e.getLocalizedMessage());                    }
p.put("chequeo","checked");
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                p.put("foto",session.getAttribute("foto"));
                p.put("rol",session.getAttribute("rol"));
        return new ModelAndView("sistema/modificar_menu",p);    }
                public ModelAndView modificar_rol2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
                HttpSession session=request.getSession(true);
                ArrayList mi_lista=new ArrayList();
                try{
this.rolManager.deleteRol(Integer.parseInt(request.getParameter("id_rol")));
}catch (Exception e) {
//System.out.println(".miercoles "+e.getLocalizedMessage());       }
try{            String d[]=request.getParameterValues("proceso");
                Rol r=new Rol();
                r.setId_rol(Integer.parseInt(request.getParameter("id_rol")));
                r.setNombre(request.getParameter("rol"));
```

```
//r.setDescripcion("---------");
this.rolManager.saveRol(r);
RolProceso rp=new RolProceso();
for (int i = 0; i < d.length; i++) {
rp.setId_rol(Integer.parseInt(request.getParameter("id_rol")));
rp.setId_pro(Integer.parseInt(d[i]));
this.rolProcesoManager.saveRolProceso(rp);          }
}catch (Exception e) {
                        p.put("url","rol_acceso.html");
                        p.put("men","adicionado
correctamente");p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toS
tring()));
                        p.put("foto",session.getAttribute("foto"));
                        p.put("rol",session.getAttribute("rol"));       }
                        p.put("url","rol_acceso.html");
p.put("men","se modifico correctamente");
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
p.put("rol",session.getAttribute("rol"));
return new ModelAndView("mensaje",p);    }
```

II.1.13.2.7 Obtener Estado (rol)

```
public ModelAndView estado_rol(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
Map p = new HashMap();
```

```
HttpSession session=request.getSession(true);
if(session.getAttribute("xusuario")==null){
p.put("men","Negado el Acceso");
p.put("url","index.html");
return new ModelAndView( "mensaje",p);              }
try{     this.rolManager.deleteRol(Integer.parseInt(request.getParameter("id_rol")));
}catch (Exception e) {
System.out.println("::::::::::: "+e.getLocalizedMessage());    }
                    p.put("url","rol_acceso.html");
                    p.put("men","se elimino correctamente");
return new ModelAndView("mensaje",p);    }
```

II.1.13.2.8 Modulo de administración

```
package controlador.ModuloAdministracion;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import javax.servlet.http.HttpSession;
import modelo.dominio.Cama;
import modelo.dominio.Dato;
import modelo.dominio.Especialidad;
import modelo.dominio.Personal;
import modelo.dominio.Sala;
import modelo.dominio.Usuario;
import modelo.manager.*;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;
import encoder.Base64;
public class administracion extends MultiActionController implements
InitializingBean{
        UsuarioManager usuarioManager;
        DatoManager datoManager;
        PersonalManager personalManager;
        RolManager rolManager ;
        SalaManager salaManager;
        CamaManager camaManager;
        EspecialidadManager especialidadManager;
        ServicioManager servicioManager;
public void afterPropertiesSet() throws Exception {
// TODO Auto-generated method stub                    }
```

```
public administracion() {
// TODO Auto-generated constructor stub    }
public UsuarioManager getUsuarioManager() {
                return usuarioManager;          }
public void setUsuarioManager(UsuarioManager usuarioManager) {
                this.usuarioManager = usuarioManager;       }
public DatoManager getDatoManager() {
                return datoManager;   }
public void setDatoManager(DatoManager datoManager) {
                this.datoManager = datoManager;     }
public PersonalManager getPersonalManager() {
                return personalManager;         }
public void setPersonalManager(PersonalManager personalManager) {
                this.personalManager = personalManager;    }
        public RolManager getRolManager() {
                return rolManager;     }
        public void setRolManager(RolManager rolManager) {
                this.rolManager = rolManager;          }
        public SalaManager getSalaManager() {
                return salaManager;    }
        public void setSalaManager(SalaManager salaManager) {
                this.salaManager = salaManager;       }
        public CamaManager getCamaManager() {
```

```
                return camaManager; }
public void setCamaManager(CamaManager camaManager) {
                this.camaManager = camaManager;  }
public EspecialidadManager getEspecialidadManager() {
                return especialidadManager;  }
        public void setEspecialidadManager(EspecialidadManager
especialidadManager) {
                this.especialidadManager = especialidadManager;   }
        public ServicioManager getServicioManager() {
                return servicioManager; n      }
//*****************
public void setServicioManager(ServicioManager servicioManager) {
.servicioManager = servicioManager;          }}
```

II.1.13.2.9 Camas

```
public ModelAndView cama(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
        Map p = new HashMap();
        HttpSession session=request.getSession(true);
        if(session.getAttribute("xusuario")==null){
p.put("men","Negado el Acceso");
p.put("url","index.html");
return new ModelAndView( "mensaje",p);   }
try{
if(request.getParameter("estado")!=null){
```

```
Cama m =
this.camaManager.getCama(Integer.parseInt(request.getParameter("id_cam")));
String xestadox=request.getParameter("estado");
try {
if(xestadox.equals("1")){
xestadox="0";
}else{
xestadox="1";                                   }
                                        //se modifica la instancia recuperada
                                        m.setEstado(Integer.parseInt(xestadox));
                                        //Se modifica en la Base de datos
                                        this.camaManager.updateCama(m);
                                        }  catch(Exception s){
                                                System.out.println("MENSAJE error estado
"+s.getLocalizedMessage());                    }                   }
ArrayList milistax = new ArrayList();
ArrayList milista2 = new ArrayList();
Map mapx = new HashMap();
int datofiltro=0;
int rango=1;
                                        if(request.getParameter("xdatofiltro")==null){
                                                datofiltro=0;
                                        }else{
datofiltro=Integer.parseInt(request.getParameter("xdatofiltro"));                       }
```

```
if(request.getParameter("xinter")==null){
rango=1;
}else{
                    rango=Integer.parseInt(request.getParameter("xinter"));     }
try {
//generador de el paginador
List Total = this.camaManager.getListaCama(datofiltro);
double num=Total.size();
double xx=Math.ceil(num / 5);
if(rango>1){
                        mapx = new HashMap();
                        mapx.put("rango","<a id='x'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='cama.html?xdatofiltro="+datofiltro+"&xinter="+(rango-
1)+"'>"+"<<<"+"</a>");
milistax.add(mapx);                 }
for(int i=1;i<=xx;i++){
mapx = new HashMap();
if(i==rango){
mapx.put("rango",""+i+"");
}else{  mapx.put("rango","<a id="'+i+'" onMouseOver='Over(this.id)'
onMouseOut='Out(this.id)'
href='cama.html?xdatofiltro="+datofiltro+"&xinter="+i+"'>"+i+"</a>");          }
milistax.add(mapx);                 }
```

```
if(rango<xx){
mapx = new HashMap();
mapx.put("rango","<a id='y' onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='cama.html?xdatofiltro="+datofiltro+"&xinter="+(rango+1)+"'>"+">>>"+"</a>
");
                                    milistax.add(mapx);                    }
                            //lista de los datos en 5 filas
                            List mat = this.camaManager.getListaCama2(datofiltro,rango);
                            for (Iterator iterator = mat.iterator(); iterator.hasNext();) {
Object name[] = (Object[]) iterator.next();
Map pp = new HashMap();
pp.put("id_cam",(Integer)name[0]);
        pp.put("numero",(Integer)name[1]);
        pp.put("estado",(Integer)name[2]);
        pp.put("id_sal",(Integer)name[3]);
        pp.put("nombre",(String)name[4]);
        milista2.add(pp);
p.put("datos", milista2);
//envio los intervalos
p.put("intervalo",milistax);            }  catch(Exception
s){System.out.println("MENSAJE error  " +s.getLocalizedMessage());}
//System.out.println("id usu sesionn "+session.getAttribute("id_usu"));
p.put("datofiltradox",datofiltro);p.put("xusuario",Base64.decode(session.getAttribute(
"xusuario").toString()));
```

```
p.put("foto",session.getAttribute("foto"));
p.put("rol",session.getAttribute("rol"));
} catch (Exception e) {
System.out.println("ummmmmmmm "+e.getLocalizedMessage());
return new ModelAndView("administracion/cama",p);}
```

II.1.13.2.10 Adicionar camas

```
public ModelAndView adicionar_cama(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
        Map p = new HashMap();
        HttpSession session=request.getSession(true);
        if(session.getAttribute("xusuario")==null){
p.put("men","Negado el Acceso");
p.put("url","index.html");
return new ModelAndView( "mensaje",p);   }
        List lista=this.salaManager.getSala();
        p.put("datos", lista);
        p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
        p.put("foto",session.getAttribute("foto"));
        p.put("rol",session.getAttribute("rol"));
        return new ModelAndView("administracion/adicionar_cama",p);}
public ModelAndView adicionar_cama2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
        Map p = new HashMap();
        HttpSession session=request.getSession(true);
```

```
if(session.getAttribute("xusuario")==null){
p.put("men","Negado el Acceso");
p.put("url","index.html");
return new ModelAndView( "mensaje",p);   }
try{
        List lista=this.camaManager.getCama();
        Cama sa=new Cama();
        sa.setId_cam(lista.size()+1);
        sa.setNumero(Integer.parseInt(request.getParameter("numero")));
        sa.setEstado(Integer.parseInt(request.getParameter("estado")));
        sa.setId_sal(Integer.parseInt(request.getParameter("sala")));
        this.camaManager.saveCama(sa);
        }catch (Exception e) {
        .put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
p.put("rol",session.getAttribute("rol"));
p.put("men", "los sentimos no se adiciono correctamente");
                        p.put("url","http://localhost:8080/proyecto_cnsx/cama.html");
                        System.out.println("ummmm,m "+e.getLocalizedMessage());}
        p.put("men", "se adiciono correctamente");
        p.put("url","http://localhost:8080/proyecto_cnsx/cama.html");
        p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
        p.put("foto",session.getAttribute("foto"));
```

```
p.put("rol",session.getAttribute("rol"));
return new ModelAndView("mensaje",p);              }
```

II.1.13.2.11 Modificar camas

```
public ModelAndView modificar_cama(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
        Map p = new HashMap();
        ArrayList mi_lista=new ArrayList();
        HttpSession session=request.getSession(true);
        if(session.getAttribute("xusuario")==null){
        p.put("men","Negado el Acceso");
        p.put("url","index.html");
        return new ModelAndView( "mensaje",p);   }
        try{     Cama
ca=this.camaManager.getCama(Integer.parseInt((request.getParameter("id_cam"))));
        List lista=this.salaManager.getSala();
        p.put("datos", ca);
        p.put("datos2", lista);
        }catch (Exception e) {
                System.out.println("modper "+e.getLocalizedMessage());   }
        p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
        p.put("foto",session.getAttribute("foto"));
        p.put("rol",session.getAttribute("rol"));
        return new ModelAndView("administracion/modificar_cama",p);}
```

```
public ModelAndView modificar_cama2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
Map p = new HashMap();
HttpSession session=request.getSession(true);
if(session.getAttribute("xusuario")==null){
p.put("men","Negado el Acceso");
p.put("url","index.html");
return new ModelAndView( "mensaje",p);   }
        try{     Cama s=new Cama();
        s.setId_cam(Integer.parseInt(request.getParameter("id_cam")));
        s.setEstado(Integer.parseInt(request.getParameter("estado")));
        s.setId_sal(Integer.parseInt(request.getParameter("sala")));
        s.setNumero(Integer.parseInt(request.getParameter("numero")));
        this.camaManager.updateCama(s);
        }catch (Exception e) {
        p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                p.put("foto",session.getAttribute("foto"));
                p.put("rol",session.getAttribute("rol"));
                p.put("url", "http://localhost:8080/proyecto_cnsx/cama.html");
                p.put("men", "lo sentimos no se modifico correctamente");
                System.out.println("mod perso2 "+e.getLocalizedMessage());       }
        p.put("url", "http://localhost:8080/proyecto_cnsx/cama.html");
        p.put("men", "se modifico correctamente");
        p.put("rol",session.getAttribute("rol"));
```

```
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
return new ModelAndView("mensaje",p);}
```

II.1.13.2.12 Obtener Estado (cama)

```
public ModelAndView estado_cama(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
        Map p = new HashMap();
        HttpSession session=request.getSession(true);
        if(session.getAttribute("xusuario")==null){
        p.put("men","Negado el Acceso");
        p.put("url","index.html");
        return new ModelAndView( "mensaje",p);   }
        try{
this.camaManager.deleteCama(Integer.parseInt(request.getParameter("id_cam")));
}catch (Exception e) {
                System.out.println("eli sala "+e.getLocalizedMessage());    }
        p.put("url", "cama.html");
        p.put("men", "se elimino correctamente");
        p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                p.put("foto",session.getAttribute("foto"));
                p.put("rol",session.getAttribute("rol"));
        return new ModelAndView("mensaje",p);
```

II.1.13.2.13 Especialidad

```
public ModelAndView especialidad(HttpServletRequest request,HttpServletResponse response) throws Exception,ServletException {
Map p = new HashMap();
HttpSession session=request.getSession(true);
if(session.getAttribute("xusuario")==null){
        p.put("men","Negado el Acceso");
            p.put("url","index.html");
            return new ModelAndView( "mensaje",p);          }
            try{
            if(request.getParameter("estado")!=null){
                        Especialidad m =(Especialidad) this.especialidadManager.getEspecialidadEstado(Integer.parseInt(request.getParameter("cod_esp")));
                        String xestadox=request.getParameter("estado");
                        try {   if(xestadox.equals("1")){
                                    xestadox="0";
                                }else{  xestadox="1";          }
//se modifica la instancia recuperada
m.setEstado(Integer.parseInt(xestadox));
//Se modifica en la Base de datos
this.especialidadManager.updateEspecialidad(m);
}  catch(Exception s){
System.out.println("MENSAJE error estado "+s.getLocalizedMessage());}        }
```

```
ArrayList milistax = new ArrayList();
ArrayList milista2 = new ArrayList();
Map mapx = new HashMap();
String datofiltro="";
int rango=1;

                                if(request.getParameter("xdatofiltro")==null){
                                        datofiltro="";
                                }else{
                                datofiltro=request.getParameter("xdatofiltro");}
                                if(request.getParameter("xinter")==null){
                                        rango=1;
}else{rango=Integer.parseInt(request.getParameter("xinter"));                }
                                try {
                                //generador de el paginador
                                List Total =
this.especialidadManager.getListaEspecialidad(datofiltro);
                                double num=Total.size();
                                double xx=Math.ceil(num / 5);
                                if(rango>1){
                                        mapx = new HashMap();
                                        mapx.put("rango","<a id='x'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
```

```
href='especialidad.html?xdatofiltro="+datofiltro+"&xinter="+(rango-
1)+'">"+"<<<"+"</a>");
milistax.add(mapx);                    }
                                for(int i=1;i<=xx;i++){
                                        mapx = new HashMap();
                                        if(i==rango){
                                        mapx.put("rango",""+i+"");
                                        }else{
                                        mapx.put("rango","<a id='"+i+"'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='especialidad.html?xdatofiltro="+datofiltro+"&xinter="+i+'">"+i+"</a>");  }
                                        milistax.add(mapx);             }
                                       if(rango<xx){
                                        mapx = new HashMap();
                                        mapx.put("rango","<a id='y'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='especialidad.html?xdatofiltro="+datofiltro+"&xinter="+(rango+1)+'">"+">>>"
+"</a>");
                                        milistax.add(mapx);                    }
                                //lista de los datos en 5 filas
                                List mat =
this.especialidadManager.getListaEspecialidad2(datofiltro,rango);
                                p.put("datos", mat);
                                //envio los intervalos
                                p.put("intervalo",milistax);
```

```
} catch(Exception s){System.out.println("MENSAJE error  "
+s.getLocalizedMessage());}
//System.out.println("id usu sesionn "+session.getAttribute("id_usu"));
p.put("datofiltradox",datofiltro);
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                    p.put("foto",session.getAttribute("foto"));
                    p.put("rol",session.getAttribute("rol"));
                    } catch (Exception e) {
                    System.out.println("ummmmmmmm
"+e.getLocalizedMessage()); }
return new ModelAndView("administracion/especialidad",p);    }
```

II.1.13.2.14 Adicionar especialidad

```
public ModelAndView adicionar_especialidad(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
                HttpSession session=request.getSession(true);
                if(session.getAttribute("xusuario")==null){
                p.put("men","Negado el Acceso");
                p.put("url","index.html");
                return new ModelAndView( "mensaje",p);            }
                List lista=this.servicioManager.getListaServicio();
                p.put("datos",
lista);p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
```

```
p.put("rol",session.getAttribute("rol"));
return new ModelAndView("administracion/adicionar_especialidad",p);    }
public ModelAndView adicionar_especialidad2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                System.out.println("ohhhhhhh     h");
                Map p = new HashMap();
                HttpSession session=request.getSession(true);
                if(session.getAttribute("xusuario")==null){
                p.put("men","Negado el Acceso");
                p.put("url","index.html");
                return new ModelAndView( "mensaje",p);              }
                try{List lista=this.especialidadManager.getEspecialidad();
                System.out.println("veamos wey "+lista.size()+"
"+request.getParameter("id_ser"));
                Especialidad sa=new Especialidad();
                sa.setId_esp(lista.size()+1);
                sa.setNombre(request.getParameter("nombre"));
                sa.setEstado(Integer.parseInt(request.getParameter("estado")));
                sa.setDescripcion(request.getParameter("descripcion"));
                sa.setId_ser(Integer.parseInt(request.getParameter("id_ser")));
                this.especialidadManager.saveEspecialidad(sa);
                }catch (Exception e) {
p.put("men", "lo sentimos no se adiciono correctamente");
p.put("url","http://localhost:8080/proyecto_cnsx/especialidad.html");
```

```
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
p.put("rol",session.getAttribute("rol"));
System.out.println("ummmm,m "+e.getLocalizedMessage());                }
                p.put("men", "se adiciono correctamente");
                p.put("url","http://localhost:8080/proyecto_cnsx/especialidad.html");
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                p.put("foto",session.getAttribute("foto"));
                p.put("rol",session.getAttribute("rol"));
return new ModelAndView("mensaje",p);            }
```

II.1.13.2.15 Modificar especialidad

```
public ModelAndView modificar_especialidad(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
Map p = new HashMap();
ArrayList mi_lista=new ArrayList();
                HttpSession session=request.getSession(true);
                if(session.getAttribute("xusuario")==null){
p.put("men","Negado el Acceso");
p.put("url","index.html");
return new ModelAndView( "mensaje",p);            }
                try{    Especialidad
ca=this.especialidadManager.getEspecialidadEstado(Integer.parseInt((request.getPara
meter("id_esp"))));
List ser=this.servicioManager.getListaServicio();
```

```
p.put("datos", ca);
p.put("datos1", ser);
}catch (Exception e) {
System.out.println("modper "+e.getLocalizedMessage());            }
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                    p.put("foto",session.getAttribute("foto"));
                    p.put("rol",session.getAttribute("rol"));
return new ModelAndView("administracion/modificar_especialidad",p);    }
public ModelAndView modificar_especialidad2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
Map p = new HashMap();
                    HttpSession session=request.getSession(true);
                    if(session.getAttribute("xusuario")==null){
                    p.put("men","Negado el Acceso");
                    p.put("url","index.html");
                    return new ModelAndView( "mensaje",p);            }
                    try{
                    Especialidad s=new Especialidad();
                    s.setId_esp(Integer.parseInt(request.getParameter("id_esp")));
                    s.setNombre(request.getParameter("nombre"));
                    s.setEstado(Integer.parseInt(request.getParameter("estado")));
                    s.setDescripcion(request.getParameter("descripcion"));
                    s.setId_ser(Integer.parseInt(request.getParameter("id_ser")));
```

```
this.especialidadManager.updateEspecialidad(s);
}catch (Exception e) {
System.out.println("mod esp "+e.getLocalizedMessage()+"
"+request.getParameter("id_esp"));
p.put("url", "http://localhost:8080/proyecto_cnsx/especialidad.html");
p.put("men", "lo sentimos no se modifico correctamente");
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
p.put("rol",session.getAttribute("rol"));        }
p.put("url", "http://localhost:8080/proyecto_cnsx/especialidad.html");
p.put("men", "se modifico correctamente");
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                p.put("foto",session.getAttribute("foto"));
                p.put("rol",session.getAttribute("rol"));
                return new ModelAndView("mensaje",p);    }
```

II.1.13.2.16 Obtener Estado (especialidad)

```
public ModelAndView estado_especialidad(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
                HttpSession session=request.getSession(true);
                if(session.getAttribute("xusuario")==null){
                p.put("men","Negado el Acceso");
                p.put("url","index.html");
                return new ModelAndView( "mensaje",p);            }
```

```
try{
this.especialidadManager.deleteEspecialidad(Integer.parseInt(request.getParameter("id_esp")));
}catch (Exception e) {
System.out.println("eli sala "+e.getLocalizedMessage());              }
                    p.put("url", "especialidad.html");
                    p.put("men", "se elimino correctamente");
                    return new ModelAndView("mensaje",p);    }
```

II.1.13.2.17 Personal

```
public ModelAndView personal(HttpServletRequest request,HttpServletResponse response) throws Exception,ServletException {
Map p = new HashMap();
                    HttpSession session=request.getSession(true);
                    if(session.getAttribute("xusuario")==null){
                    p.put("men","Negado el Acceso");
                    p.put("url","index.html");
                    return new ModelAndView( "mensaje",p);              }
                    try{
                              if(request.getParameter("xestado")!=null){
Personal m = this.personalManager.getPersonal(Integer.parseInt((request.getParameter("xid_per"))));
String xestadox=request.getParameter("xestado");                                        try {
if(xestadox.equals("1")){
```

```
xestadox="0";                    }else{
xestadox="1";          }
//se modifica la instancia recuperada
m.setEstado(Integer.parseInt(xestadox));
//Se modifica en la Base de datos
this.personalManager.updatePersonal(m);    }  catch(Exception s){
System.out.println("MENSAJE error estado ");      }        }
ArrayList milistax = new ArrayList();
                        Map mapx = new HashMap();
                        String datofiltro="";
                        int rango=1;
if(request.getParameter("xdatofiltro")==null){
datofiltro="";
}else{  datofiltro=request.getParameter("xdatofiltro");       }
if(request.getParameter("xinter")==null){    rango=1;        }else{
rango=Integer.parseInt(request.getParameter("xinter"));}
try {
//generador de el paginador
List Total = this.personalManager.getListaUsuariosFiltroTotal(datofiltro);
                                double num=Total.size();
                                double xx=Math.ceil(num / 5);
                                if(rango>1){
                                mapx = new HashMap();
```

```
mapx.put("rango","<a id='x' onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='personal.html?xdatofiltro="+datofiltro+"&xinter="+(rango-
1)+"'>"+"<<<"+"</a>");
milistax.add(mapx);             }
for(int i=1;i<=xx;i++){
mapx = new HashMap();
if(i==rango){
mapx.put("rango",""+i+"");
}else{
mapx.put("rango","<a id="'+i+'" onMouseOver='Over(this.id)'
onMouseOut='Out(this.id)'
href='personal.html?xdatofiltro="+datofiltro+"&xinter="+i+"'>"+i+"</a>");}
milistax.add(mapx);                     }
if(rango<xx){
                                        mapx = new HashMap();
                                        mapx.put("rango","<a id='y'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='personal.html?xdatofiltro="+datofiltro+"&xinter="+(rango+1)+"'>"+">>>"+"</
a>");
milistax.add(mapx);   }
                                //lista de los datos en 5 filas
                                List mat =
this.personalManager.getListaUsuariosFiltro(datofiltro,rango);
                                p.put("datos", mat);
                                //envio los intervalos
```

```
p.put("intervalo",milistax);
}  catch(Exception s){System.out.println("MENSAJE error  ");}
//System.out.println("id usu sesionn "+session.getAttribute("id_usu"));
p.put("datofiltradox",datofiltro);
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                        p.put("foto",session.getAttribute("foto"));
                        p.put("rol",session.getAttribute("rol"));
} catch (Exception e) {
System.out.println("ummmmmmmm "+e.getLocalizedMessage());          }
return new ModelAndView("administracion/personal",p);    }
```

II.1.13.2.18 Adicionar personal

```
public ModelAndView adicionar_personal(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
                HttpSession session=request.getSession(true);
                if(session.getAttribute("xusuario")==null){
                p.put("men","Negado el Acceso");
                p.put("url","index.html");
                return new ModelAndView( "mensaje",p);            }
                List lista=this.rolManager.getListaRol();
p.put("datos",lista);p.put("xusuario",Base64.decode(session.getAttribute("xusuario").
toString()));
                p.put("foto",session.getAttribute("foto"));
                p.put("rol",session.getAttribute("rol"));
```

```
return new ModelAndView("administracion/adicionar_personal",p);    }
public ModelAndView adicionar_personal2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
                HttpSession session=request.getSession(true);
                if(session.getAttribute("xusuario")==null){
                p.put("men","Negado el Acceso");
                p.put("url","index.html");
                return new ModelAndView( "mensaje",p);   }
                try{
                List lista=this.personalManager.getListaPersonal();
                Date d;
                SimpleDateFormat fechay = new SimpleDateFormat("dd-MM-yyyy");
                d = fechay.parse(request.getParameter("fecha_nacimiento"));
                Personal per=new Personal();
                per.setAm(request.getParameter("apellido_materno"));
                per.setAp(request.getParameter("apellido_paterno"));
                per.setCi(Integer.parseInt(request.getParameter("xci")));
                per.setEstado(Integer.parseInt(request.getParameter("estado")));
                per.setEstado_civil(request.getParameter("estado_civil"));
                per.setFecha_nacimiento(d);
                per.setFoto(request.getParameter("foto"));
                per.setId_per(lista.size()+1);
                per.setNombre(request.getParameter("nombre"));
```

```
this.personalManager.savePersonal(per);
Usuario us=new Usuario();
List lista3=this.usuarioManager.getListaUsuario();
Usuario u=new Usuario();
u.setId_per(per.getId_per());
u.setId_usu(lista3.size()+1);
u.setId_rol(Integer.parseInt(request.getParameter("tipo")));
this.usuarioManager.saveUsuario(u);
                Dato da=new Dato();
                List lista2=this.datoManager.getDato();
if(request.getParameter("clave1").equals(request.getParameter("clave1"))){
                        try{
                        da.setClave(Base64.encode(request.getParameter("clave1")));
                        da.setId_dat(lista2.size()+1);
                        da.setId_usu(u.getId_usu());
da.setUsuario(Base64.encode(request.getParameter("usuario")));
                        //System.out.println(" .....
"+request.getParameter("clave1")+" "+(lista2.size()+1)+" "+(u.getId_usu())+"
"+request.getParameter("usuario"));
this.datoManager.saveDato(da);
}catch (Exception e) {
System.out.println("ummmm,m "+e.getLocalizedMessage());}}
```

```
}catch (Exception e) {System.out.println("ddddddddd "+e.getLocalizedMessage());}
p.put("url","personal.html");
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                    p.put("foto",session.getAttribute("foto"));
                    p.put("rol",session.getAttribute("rol"));
return new ModelAndView("mensaje",p);              }
```

II.1.13.2.19 Modificar personal

```
public ModelAndView modificar_personal(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
Map p = new HashMap();
                    ArrayList mi_lista=new ArrayList();
                    HttpSession session=request.getSession(true);
                    if(session.getAttribute("xusuario")==null){
                    p.put("men","Negado el Acceso");
                    p.put("url","index.html");
                    return new ModelAndView( "mensaje",p);   }
                    try{
                    Personal
per=this.personalManager.getPersonal(Integer.parseInt((request.getParameter("id_per
")))); List
dat=this.datoManager.getListaDato(Integer.parseInt(request.getParameter("id_per")))
;
for (Iterator iterator = dat.iterator(); iterator.hasNext();) {
```

```
Object []name = (Object[]) iterator.next();
Map pp = new HashMap();
int id_dat=(Integer)name[0];
int id_usu=(Integer)name[1];
String usuario=(String)name[2];
String clave=(String)name[3];
pp.put("id_dat", id_dat);
pp.put("id_usu", id_usu);
pp.put("usuario", usuario);
pp.put("clave", clave);
mi_lista.add(pp);        }
                    //System.out.println("vemaos cghe "+per.getId_per()+"
"+per.getAm());
                    p.put("datos2", mi_lista);
                    p.put("datos", per);
                    }catch (Exception e) {
                    System.out.println("modper "+e.getLocalizedMessage());   }
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                    p.put("foto",session.getAttribute("foto"));
                    p.put("rol",session.getAttribute("rol"));
return new ModelAndView("administracion/modificar_personal",p);    }
public ModelAndView modificar_personal2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
```

```
Map p = new HashMap();
HttpSession session=request.getSession(true);
if(session.getAttribute("xusuario")==null){
p.put("men","Negado el Acceso");
                    p.put("url","index.html");
                    return new ModelAndView( "mensaje",p);   }
                try{
                Personal per=new Personal();
                Date d;
                SimpleDateFormat fechay = new SimpleDateFormat("dd-MM-yyyy");
                d = fechay.parse(request.getParameter("fecha_nacimiento"));
                per.setAm(request.getParameter("apellido_materno"));
                per.setAp(request.getParameter("apellido_paterno"));
                per.setCi(Integer.parseInt(request.getParameter("ci")));
                per.setEstado(Integer.parseInt(request.getParameter("estado")));
                per.setEstado_civil(request.getParameter("estado_civil"));
                per.setFecha_nacimiento(d);
                per.setId_per(Integer.parseInt(request.getParameter("id_per")));
                per.setFoto(request.getParameter("foto"));
                per.setNombre(request.getParameter("nombre"));
                this.personalManager.updatePersonal(per);
                try {
                Dato dat=new Dato();
```

```
dat.setId_dat(Integer.parseInt(request.getParameter("id_dat")));
dat.setClave(request.getParameter("clave1"));
dat.setId_usu(Integer.parseInt(request.getParameter("id_usu")));
                    dat.setUsuario(request.getParameter("usuario"));
                    this.datoManager.updateDato(dat);
} catch (Exception e) {
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                    p.put("foto",session.getAttribute("foto"));
                    p.put("rol",session.getAttribute("rol"));
                    System.out.println("  ... "+e.getLocalizedMessage());        }
}catch (Exception e) {
System.out.println("mod perso2 "+e.getLocalizedMessage());                }
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                p.put("foto",session.getAttribute("foto"));
                p.put("rol",session.getAttribute("rol"));
                p.put("url", "personal.html");
                return new ModelAndView("mensaje",p);    }
```

II.1.13.2.20 Obtener Estado (personal)

```
public ModelAndView estado personal (HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
                HttpSession session=request.getSession(true);
                if(session.getAttribute("xusuario")==null){
                p.put("men","Negado el Acceso");
```

```
p.put("url","index.html");
return new ModelAndView( "mensaje",p);            }
                try{
this.salaManager.deleteSala(Integer.parseInt(request.getParameter("id_sal")));
}catch (Exception e) {
System.out.println("eli sala "+e.getLocalizedMessage());            }
p.put("url", "sala.html");
                p.put("men", "se elimino correctamente");
                return new ModelAndView("mensaje",p);    }
```

II.1.13.2.21 Salas

```
public ModelAndView sala(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
                Map p = new HashMap();
                HttpSession session=request.getSession(true);
                if(session.getAttribute("xusuario")==null){
                p.put("men","Negado el Acceso");
                p.put("url","index.html");
                return new ModelAndView( "mensaje",p);            }
                try{
                        if(request.getParameter("estado")!=null){
                        Sala m =
this.salaManager.getSala2(Integer.parseInt(request.getParameter("id_sal")));
                                String xestadox=request.getParameter("estado");
                                try {
```

```
if(xestadox.equals("1")){
xestadox="0";
}else{
xestadox="1";          }
                              //se modifica la instancia recuperada
                              m.setEstado(Integer.parseInt(xestadox));
                              //Se modifica en la Base de datos
                              this.salaManager.updateSala(m);
}  catch(Exception s){
                                   System.out.println("MENSAJE error estado
"+s.getLocalizedMessage());                    }                    }
                         ArrayList milistax = new ArrayList();
                         ArrayList milista2 = new ArrayList();
                         Map mapx = new HashMap();
                         String datofiltro="";
                         int rango=1;
if(request.getParameter("xdatofiltro")==null){
datofiltro="";
}else{
datofiltro=request.getParameter("xdatofiltro");                                   }
                              if(request.getParameter("xinter")==null){
                                   rango=1;
                              }else{
rango=Integer.parseInt(request.getParameter("xinter"));              }
```

```
try {
        //generador de el paginador
        List Total = this.salaManager.getListaSala(datofiltro);
        double num=Total.size();
        double xx=Math.ceil(num / 5);
        if(rango>1){
mapx = new HashMap();
mapx.put("rango","<a id='x' onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='sala.html?xdatofiltro="+datofiltro+"&xinter="+(rango-1)+"">"+"<<<"+"</a>");
milistax.add(mapx);             }
for(int i=1;i<=xx;i++){
                                        mapx = new HashMap();
                                        if(i==rango){
mapx.put("rango",""+i+"");   }else{
                                                mapx.put("rango","<a id='"+i+"'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='sala.html?xdatofiltro="+datofiltro+"&xinter="+i+"">"+i+"</a>");             }
milistax.add(mapx);             }
                                        if(rango<xx){
                                        mapx = new HashMap();
                                        mapx.put("rango","<a id='y'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='sala.html?xdatofiltro="+datofiltro+"&xinter="+(rango+1)+"'"+">>>"+"</a>");
milistax.add(mapx);                     }
```

```
//lista de los datos en 5 filas
List mat = this.salaManager.getListaSala2(datofiltro,rango);
for (Iterator iterator = mat.iterator(); iterator.hasNext();) {
                                        Object name[] = (Object[]) iterator.next();
                                        Map pp = new HashMap();
                                        pp.put("id_sal",(Integer)name[0]);
                                        pp.put("sala",(String)name[1]);
                                        pp.put("id_esp",(Integer)name[2]);
                                        pp.put("estado",(Integer)name[3]);
                                        pp.put("especialidad",(String)name[4]);
                                        milista2.add(pp);}
                                    p.put("datos", milista2);
                                    //envio los intervalos
                                    p.put("intervalo",milistax);
} catch(Exception s){System.out.println("MENSAJE error  "
+s.getLocalizedMessage());}
//System.out.println("id usu sesionn "+session.getAttribute("id_usu"));
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
p.put("rol",session.getAttribute("rol"));
} catch (Exception e) {
System.out.println("ummmmmmmm "+e.getLocalizedMessage());          }
```

```
return new ModelAndView("administracion/sala",p);    }
```

II.1.13.2.22 Adicionar salas

```
public ModelAndView adicionar_sala(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
Map p = new HashMap();
HttpSession session=request.getSession(true);
                    if(session.getAttribute("xusuario")==null){
                    p.put("men","Negado el Acceso");
                    p.put("url","index.html");
                    return new ModelAndView( "mensaje",p);              }
                    List lista=this.especialidadManager.getEspecialidad();
                    p.put("datos", lista);
return new ModelAndView("administracion/adicionar_sala",p);    }
public ModelAndView adicionar_sala2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
Map p = new HashMap();
                    HttpSession session=request.getSession(true);
                    if(session.getAttribute("xusuario")==null){
                    p.put("men","Negado el Acceso");
                    p.put("url","index.html");
                    return new ModelAndView( "mensaje",p);   }
                    try{
                    List lista=this.salaManager.getSala();
Sala sa=new Sala();
```

```
sa.setId_sal(lista.size()+1);
sa.setNombre(request.getParameter("nombre"));
sa.setId_esp(Integer.parseInt(request.getParameter("especialidad")));
sa.setEstado(Integer.parseInt(request.getParameter("estado")));
this.salaManager.saveSala(sa);
}catch (Exception e) {
System.out.println("ummmm,m "+e.getLocalizedMessage());       }
                    p.put("men", "sala adicionada correctamente");
                    p.put("url","sala.html");
return new ModelAndView("mensaje",p);              }
```

II.1.13.2.23 Modificar salas

```
public ModelAndView modificar_sala(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                    Map p = new HashMap();
                    ArrayList mi_lista=new ArrayList();
                    HttpSession session=request.getSession(true);
                    if(session.getAttribute("xusuario")==null){
                    p.put("men","Negado el Acceso");
                    p.put("url","index.html");
                    return new ModelAndView( "mensaje",p);            }
                    try{
                    Sala
sa=this.salaManager.getSala2(Integer.parseInt((request.getParameter("id_sal"))));
List lista=this.especialidadManager.getEspecialidad();
```

```
p.put("datos", sa);
p.put("datos2", lista);
}catch (Exception e) {
System.out.println("modper "+e.getLocalizedMessage());            }
return new ModelAndView("administracion/modificar_sala",p);    }
public ModelAndView modificar_sala2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
                HttpSession session=request.getSession(true);
                if(session.getAttribute("xusuario")==null){
                p.put("men","Negado el Acceso");
                p.put("url","index.html");
                return new ModelAndView( "mensaje",p);            }
                try{
                Sala s=new Sala();
                s.setId_sal(Integer.parseInt(request.getParameter("id_sal")));
                s.setEstado(Integer.parseInt(request.getParameter("estado")));
                s.setId_esp(Integer.parseInt(request.getParameter("especialidad")));
                s.setNombre(request.getParameter("nombre"));
                this.salaManager.updateSala(s);
}catch (Exception e) {
System.out.println("mod perso2 "+e.getLocalizedMessage());       }
```

```
p.put("url", "sala.html");
p.put("men", "se modifico correctamente");
return new ModelAndView("mensaje",p);    }
```

II.1.13.2.24 Obtener Estado (sala)

```
public ModelAndView estado_sala(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
                Map p = new HashMap();
                HttpSession session=request.getSession(true);
                if(session.getAttribute("xusuario")==null){
                p.put("men","Negado el Acceso");
                p.put("url","index.html");
                return new ModelAndView( "mensaje",p);            }
                try{
this.salaManager.deleteSala(Integer.parseInt(request.getParameter("id_sal")));
}catch (Exception e) {
System.out.println("eli sala "+e.getLocalizedMessage());    }
p.put("url", "sala.html");
                p.put("men", "se elimino correctamente");
                return new ModelAndView("mensaje",p);    }
```

II.1.13.2.25 Modulo Ingreso

```
package controlador.ModuloIngreso;
import java.text.SimpleDateFormat;
import encoder.Base64;
```

```
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.TimeZone;
import java.util.Vector;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import modelo.dominio.*;
import modelo.manager.*;
import encoder.Base64;
import org.hibernate.type.AnyType;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;
import encoder.Base64;
import controlador.*;
```

```
public class ingreso extends MultiActionController implements InitializingBean{
///inicio base afiliacion
        PersonaManager personaManager;
        AseguradoManager aseguradoManager;
        ServicioManager servicioManager;
        SalaManager salaManager;
        CamaManager camaManager;
        TipoDiagnosticoManager tipoDiagnosticoManager;
        TipoPrescripcionManager tipoPrescripcionManager;
        TipoEstadoManager tipoEstadoManager;
        TipoSeguroManager tipoSeguroManager;
        ControlPacienteManager controlPacienteManager;
        DetalleIngresoPacienteManager detalleIngresoPacienteManager;
        HoraFechaIngresoManager horaFechaIngresoManager;
        public ControlPacienteManager getControlPacienteManager() {
                return controlPacienteManager;        }
        public void setControlPacienteManager(
        ControlPacienteManager controlPacienteManager) {
        this.controlPacienteManager = controlPacienteManager;    }
        public DetalleIngresoPacienteManager getDetalleIngresoPacienteManager() {
        return detalleIngresoPacienteManager;        }
        public void setDetalleIngresoPacienteManager(
DetalleIngresoPacienteManager detalleIngresoPacienteManager) {
```

```
this.detalleIngresoPacienteManager = detalleIngresoPacienteManager;     }
public HoraFechaIngresoManager getHoraFechaIngresoManager() {
return horaFechaIngresoManager;     }
public void setHoraFechaIngresoManager(
HoraFechaIngresoManager horaFechaIngresoManager) {
this.horaFechaIngresoManager = horaFechaIngresoManager;        }
public TipoSeguroManager getTipoSeguroManager() {
return tipoSeguroManager;    }
public void setTipoSeguroManager(TipoSeguroManager tipoSeguroManager) {
                this.tipoSeguroManager = tipoSeguroManager;       }
public TipoDiagnosticoManager getTipoDiagnosticoManager() {
return tipoDiagnosticoManager;       }
public void setTipoDiagnosticoManager(
TipoDiagnosticoManager tipoDiagnosticoManager) {
this.tipoDiagnosticoManager = tipoDiagnosticoManager;   }
public TipoPrescripcionManager getTipoPrescripcionManager() {
                return tipoPrescripcionManager;       }
public void setTipoPrescripcionManager(
TipoPrescripcionManager tipoPrescripcionManager) {
                this.tipoPrescripcionManager = tipoPrescripcionManager;}
public TipoEstadoManager getTipoEstadoManager() {
                return tipoEstadoManager;    }
public void setTipoEstadoManager(TipoEstadoManager tipoEstadoManager) {
```

```
this.tipoEstadoManager = tipoEstadoManager;        }
public CamaManager getCamaManager() {
                return camaManager; }
public void setCamaManager(CamaManager camaManager) {
                this.camaManager = camaManager;  }
public SalaManager getSalaManager() {       return salaManager;    }
public void setSalaManager(SalaManager salaManager) {
                this.salaManager = salaManager;}
public ServicioManager getServicioManager() {
                return servicioManager;}
public void setServicioManager(ServicioManager servicioManager) {
                this.servicioManager = servicioManager;     }
public AseguradoManager getAseguradoManager() {
                return aseguradoManager;     }
public void setAseguradoManager(AseguradoManager aseguradoManager) {
                this.aseguradoManager = aseguradoManager;}
///final clase afiliacion
public PersonaManager getPersonaManager() {
                return personaManager;}
public void setPersonaManager(PersonaManager personaManager) {
                this.personaManager = personaManager;     }
public void afterPropertiesSet() throws Exception {}
public ingreso() {       }
```

```
@SuppressWarnings("unchecked")
public ModelAndView ingreso(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                //String usuario=(String) session.getAttribute("usuario");
                Map p = new HashMap();
                if(session.getAttribute("xusuario")==null){
                  p.put("men","Negado el Acceso");
                p.put("url","index.html");
                return new ModelAndView( "mensaje",p);          }
                ArrayList milistax = new ArrayList();
                Map mapx = new HashMap();
                String datofiltro="";
                String tipo="";
                int rango=1;
if(request.getParameter("xdatofiltro")==null){
                        datofiltro="";
                        }else{  datofiltro=request.getParameter("xdatofiltro");}
                        if(request.getParameter("xinter")==null){
                        rango=1;
}else{  rango=Integer.parseInt(request.getParameter("xinter"));     }
try {
                        //generador de el paginador
                        List Total = null;
```

```
try{
if(request.getParameter("tipo")==null) {//cuando entra por primera vez, es decir por defecto                                                Total = this.personaManager.getListaUsuariosFiltroTotalAsegurado(datofiltro);
//session.setAttribute("tipo","asegurado");                }
                                                if(request.getParameter("tipo")!=null) {
if (request.getParameter("tipo").equalsIgnoreCase("asegurado")) {
Total = this.personaManager.getListaUsuariosFiltroTotalAsegurado(datofiltro);
}else{//de momento solo sera con asegurados y no beneficiarios
Total = this.personaManager.getListaUsuariosFiltroTotalAsegurado(datofiltro);
//Total = this.personaManager.getListaUsuariosFiltroTotalBeneficiario(datofiltro);
}
//session.setAttribute("tipo",request.getParameter("tipo"));            }
}catch (Exception e) {                    System.out.println("MENSAJE 1 "+e.getLocalizedMessage()); }
double num=Total.size();
double xx=Math.ceil(num / 5);
if(rango>1){
                                        mapx = new HashMap();
                                        mapx.put("rango","<a id='x' onMouseOver='Over(this.id)' onMouseOut='Out(this.id)' href='ingreso.html?tipo="+tipo+"&xdatofiltro="+datofiltro+"&xinter="+(rango-1)+"'>"+"<<<"+"</a>");
                                        milistax.add(mapx);              }
for(int i=1;i<=xx;i++){
```

```
mapx = new HashMap();
if(i==rango){
mapx.put("rango",""+i+"");
}else{
mapx.put("rango","<a id='"+i+"' onMouseOver='Over(this.id)'
onMouseOut='Out(this.id)'
href='ingreso.html?tipo="+tipo+"&xdatofiltro="+datofiltro+"&xinter="+i+"'>"+i+"</
a>");              }
milistax.add(mapx);              }
if(rango<xx){
                                    mapx = new HashMap();
                                    mapx.put("rango","<a id='y'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='ingreso.html?tipo="+tipo+"&xdatofiltro="+datofiltro+"&xinter="+(rango+1)+"
'>"+">>>"+"</a>");                    milistax.add(mapx);    }
                          //lista de los datos en 5 filas
                          List asegurado;
                          List beneficiario;
                          ArrayList lista_asegurado=new ArrayList();
                          ArrayList lista_beneficiario=new ArrayList();
                          if(request.getParameter("tipo")==null) {//cuando sea asegurado
asegurado = this.personaManager.getListaUsuariosFiltroAsegurado(datofiltro,rango);
for (Iterator iterator = asegurado.iterator(); iterator.hasNext();) {
Object[] object1 = (Object[]) iterator.next();
HashMap listita = new HashMap();
```

```
listita.put("cod_persona",object1[0]);
                                    listita.put("nombre",object1[1]);
                                    listita.put("apepat",object1[2]);
                                    listita.put("apemat",object1[3]);
                                    listita.put("estado",object1[4]);
                                    listita.put("tipo","asegurado");
                                    lista_asegurado.add(listita);                    }
p.put("datos",lista_asegurado);          }
                                try{
if(request.getParameter("tipo")!=null) {
if (request.getParameter("tipo").equalsIgnoreCase("asegurado")) {
asegurado = this.personaManager.getListaUsuariosFiltroAsegurado(datofiltro,rango);
for (Iterator iterator = asegurado.iterator(); iterator.hasNext();) {
Object[] object1 = (Object[]) iterator.next();
HashMap mierda1 = new HashMap();
mier1.put("cod_persona",object1[0]);
session.setAttribute("cod_persona", object1[0].toString());
mier1.put("nombre",object1[1]);
mier1.put("apepat",object1[2]);
mier1.put("apemat",object1[3]);
mier1.put("estado",object1[4]);
System.out.println("para eliminar "+object1[4]);
mier1.put("tipo","asegurado");
```

```
lista_asegurado.add(mierda1);                                    }
p.put("datos",lista_asegurado);
                                                                 }else{//por el momento
solo sera con asegurado
beneficiario =
this.personaManager.getListaUsuariosFiltroBeneficiario(datofiltro,rango);
for (Iterator iterator = beneficiario.iterator(); iterator.hasNext();) {
Object[] object2 = (Object[]) iterator.next();
HashMap mier2 = new HashMap();
mier2.put("cod_persona",object2[0]);
session.setAttribute("cod_beneficiario",object2[0].toString());
mier2.put("nombre",object2[1]);
mier2.put("apepat",object2[2]);
mier2.put("tipo","beneficiario");
lista_beneficiario.add(mier2);                                   }
.put("datos", lista_beneficiario);       */
asegurado = this.personaManager.getListaUsuariosFiltroAsegurado(datofiltro,rango);
for (Iterator iterator = asegurado.iterator(); iterator.hasNext();) {
Object[] object1 = (Object[]) iterator.next();
HashMap mier1 = new HashMap();
mier1.put("cod_persona",object1[0]);
session.setAttribute("cod_persona", object1[0].toString());
mier1.put("nombre",object1[1]);
          mier1.put("apepat",object1[2]);
```

```
mier1.put("apemat",object1[3]);

mier1.put("estado",object1[4]);

mier1.put("tipo","asegurado");

lista_asegurado.add(mier1);

p.put("datos",lista_asegurado);        }        }

}catch (Exception e) {                                System.out.println("MENSAJE
2222222  "+e.getLocalizedMessage());                        }

//envio los intervalos

                }  catch(Exception s){System.out.println("MENSAJE error
"+s.getLocalizedMessage());}

                p.put("foto",session.getAttribute("foto").toString());

                p.put("intervalo",milistax);

                p.put("rol",session.getAttribute("rol"));

                p.put("xusuario",Base64.decode((String)
session.getAttribute("xusuario")));

//p.put("xclave", session.getAttribute("xclave"));

p.put("datofiltradox",datofiltro);

return new ModelAndView("ingreso/ingreso",p);   }

public ModelAndView adicionar_ingreso(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {

                HttpSession session=request.getSession(true);

                HashMap p = new HashMap();

                Date d;

                ControlPaciente cp=new ControlPaciente();
```

```
try{
List lista1=this.controlPacienteManager.getControlPaciente();
cp.setId_cop(lista1.size()+1);
cp.setEstatura(Integer.parseInt(request.getParameter("estatura")));
cp.setPeso(Integer.parseInt(request.getParameter("peso")));
cp.setPresion(Integer.parseInt(request.getParameter("presion")));
cp.setPulso(Integer.parseInt(request.getParameter("pulso")));
cp.setTemperatura(Integer.parseInt(request.getParameter("temperatura")));
                this.controlPacienteManager.saveControlPaciente(cp);
                }catch (Exception e) {
System.out.println("----- control paciente ----- "+e.getLocalizedMessage());}
 HoraFechaIngreso hfi=new HoraFechaIngreso();
                try {
                        List
lista2=this.horaFechaIngresoManager.getHoraFechaIngreso();
                        hfi.setId_hfi(lista2.size()+1);
                        hfi.setHora((request.getParameter("hora")));
SimpleDateFormat fechay = new SimpleDateFormat("dd-MM-yyyy");
                        d = fechay.parse(request.getParameter("fecha"));
                        hfi.setFecha(d);
                        this.horaFechaIngresoManager.saveHoraFechaIngreso(hfi);
} catch (Exception e) {System.out.println("----- horafecha ingreso-----
"+e.getLocalizedMessage()); }
TipoEstado tie=new TipoEstado();
```

```
try {     List lista_tie=this.tipoEstadoManager.getTipoEstado();
tie.setId_tie(lista_tie.size()+1);
tie.setNombre(request.getParameter("estado"));
tie.setDescripcion(request.getParameter("descripcion"));
                    this.tipoEstadoManager.saveTipoEstado(tie);
                } catch (Exception e) {
System.out.println("----- tipo estado----- "+e.getLocalizedMessage()        }
                TipoPrescripcion tip=new TipoPrescripcion();
                try {
                List lista_tip=this.tipoPrescripcionManager.getTipoPrescripcion();
                tip.setId_tip(lista_tip.size()+1);
                tip.setNombre(request.getParameter("prescripcion"));
                ///tip.setDescripcion(request.getParameter("descripcion"));
                this.tipoPrescripcionManager.saveTipoPrescripcion(tip);
                } catch (Exception e) {
                    System.out.println("----- tipo  prescripcion-----
"+e.getLocalizedMessage());           }
                TipoDiagnostico tid=new TipoDiagnostico();
                try {     List
lista_tid=this.tipoDiagnosticoManager.getTipoDiagnostico();
                    tid.setId_tid(lista_tid.size()+1);
                    tid.setNombre(request.getParameter("diagnostico"));
                    //tid.setDescripcion(request.getParameter("descripcion"));
                    this.tipoDiagnosticoManager.saveTipoDiagnostico(tid);
```

```
} catch (Exception e) {
System.out.println("----- tipo diagnostico----- "+e.getLocalizedMessage());}
//*****************************************
                        DetalleIngresoPaciente dip=new DetalleIngresoPaciente();
                        List
lista3=this.detalleIngresoPacienteManager.getListaDetalleIngresoPaciente();
                        try{     dip.setId_dip(lista3.size()+1);
}catch (Exception e) {  System.out.println(" 1 "+e.getLocalizedMessage());}
try{//dip.setId_esp(Integer.parseInt((String)session.getAttribute("especialidad")));
}catch (Exception e) {
System.out.println("2 "+e.getLocalizedMessage());          }
try{dip.setId_hfi(hfi.getId_hfi());
}catch (Exception e) {System.out.println(" 3 "+e.getLocalizedMessage());
try {
int x=((Integer)session.getAttribute("id_usu"));
                                dip.setId_per((x));
                                } catch (Exception e) {
System.out.println(" 4 "+e.getLocalizedMessage());                                }
try {
dip.setId_tie(tie.getId_tie());
} catch (Exception e) {
System.out.println(" 5 "+e.getLocalizedMessage());                          }
try{dip.setId_tid(tid.getId_tid());
}catch (Exception e) {
```

```
System.out.println(" 6 "+e.getLocalizedMessage());          }
try {
dip.setId_tip(tip.getId_tip());
                        } catch (Exception e) {
System.out.println(" 7 "+e.getLocalizedMessage());                          }
try{
String xx=((String)session.getAttribute("id_cam"));
dip.setId_cam(Integer.parseInt(xx));
}catch (Exception e) {
System.out.println("9  "+e.getLocalizedMessage());                  }
try {
dip.setId_cop(cp.getId_cop());
                        } catch (Exception e) {
System.out.println("10  "+e.getLocalizedMessage());         }
try{
                        String x=((String)session.getAttribute("cod_asegurado"));
                        dip.setCod_persona(Integer.parseInt(x));
                        }catch (Exception e) {
System.out.println(" 11 "+e.getLocalizedMessage());         }
try {dip.setPersona_condujo(request.getParameter("persona_condujo"));
} catch (Exception e) {
System.out.println(" 12 "+e.getLocalizedMessage());                  }
try {
```

```
String x=((String)session.getAttribute("cod_beneficiario"));
dip.setCod_beneficiario(Integer.parseInt(x));
} catch (Exception e) {
System.out.println(" 13 "+e.getLocalizedMessage());         }
try{
this.detalleIngresoPacienteManager.saveDetalleIngresoPaciente(dip);
                    }catch (Exception e) {
                                System.out.println(" el detalle
"+e.getLocalizedMessage()); }
                    p.put("xusuario",session.getAttribute("xusuario"));
                    p.put("xclave",session.getAttribute("xclave"));
/*{
Cama
ca=this.camaManager.getCama(Integer.parseInt((String)session.getAttribute("id_cam
")));ca.setEstado(1);
//ca.setEstado(Integer.parseInt((String)session.getAttribute("id_cam")));
this.camaManager.updateCama(ca);
                    }catch (Exception e) {
                                p.put("men","no registrado correctamente");
                                p.put("url","ingreso.html");
                                System.out.println(" exception cama
"+e.getLocalizedMessage());}*/
                    p.put("xusuario",session.getAttribute("xusuario") );
                    p.put("xclave", session.getAttribute("xclave"));
```

```
p.put("men","ingreso registrado correctamente");
p.put("url","ingreso.html");
return new ModelAndView("mensaje",p);   }  }
```

II.1.13.2.26 Llenar_Hoja admisión

```
public ModelAndView admision(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                HashMap p = new HashMap();
                p.put("rol",session.getAttribute("rol"));
p.put("xusuario",Base64.decode((String) session.getAttribute("xusuario")));
                p.put("foto",session.getAttribute("foto"));
                List lista1 = null;
                int tamano=0;
                try{                            try{
if(request.getParameter("cod_persona")!=null){//inicioooooo cuando es asegurado
if(request.getParameter("tipo").toString().equalsIgnoreCase("asegurado")){
//session.setAttribute("cod_persona",request.getParameter("cod_persona"));
lista1 =
this.aseguradoManager.getTipoPersona(Integer.parseInt(request.getParameter("cod_p
ersona")));
tamano=lista1.size();
if(tamano==0){
this.aseguradoManager.getTipoPersonaSinBeneficiario(Integer.parseInt(request.getPa
rameter("cod_persona")));                                   }
```

```
if(tamano==0){//
for (Iterator iterator = lista1.iterator(); iterator.hasNext();) {
Object object[] = (Object[]) iterator.next();
System.out.println("cuando es sin beneficiaio");
session.setAttribute("cod_asegurado", object[0].toString());
session.setAttribute("nro_asegurado",object[1].toString() );
session.setAttribute("cod_beneficiario", "");
session.setAttribute("codigo","");
session.setAttribute("booleano","false");
session.setAttribute("bandera","false");
session.setAttribute("estado","disabled");
session.setAttribute("estado2","disabled");                    }                    }
if(tamano==1){//cuando es asegurado
for (Iterator iterator = lista1.iterator(); iterator.hasNext();) {
object[] = (Object[]) iterator.next();
System.out.println("no se mete");
        session.setAttribute("cod_asegurado", object[0].toString());
        session.setAttribute("nro_asegurado",object[1].toString() );
        session.setAttribute("cod_beneficiario", object[2].toString());
        session.setAttribute("codigo",object[3].toString());
        session.setAttribute("booleano","false");
        session.setAttribute("bandera","false");
        session.setAttribute("estado","disabled");
```

```
session.setAttribute("estado2","disabled");   }          }}  }

////finallll

}catch (Exception e) {

System.out.println("que sera "+e.getLocalizedMessage());                    }

/*try{

if(request.getParameter("cod_persona")!=null){  /// cuando es
beneficiariooooooooooooo

if(session.getAttribute("tipo").toString().equalsIgnoreCase("beneficiario")){

lista1 =
this.aseguradoManager.getTipoPersonaBeneficiario(Integer.parseInt(request.getPara
meter("cod_persona")));

tamano=lista1.size();

for (Iterator iterator = lista1.iterator(); iterator.hasNext();) {

System.out.println(".,.,., "+tamano);

Object object[] = (Object[]) iterator.next();

System.out.println("no se mete 2");

session.setAttribute("cod_asegurado", object[0].toString());

session.setAttribute("nro_asegurado",object[1].toString() );

session.setAttribute("cod_beneficiario", object[2].toString());

session.setAttribute("codigo",object[3].toString());

session.setAttribute("booleano","false");

session.setAttribute("bandera","false");

session.setAttribute("estado","disabled");

session.setAttribute("estado2","disabled");   }          }                    }
```

```
}catch (Exception e) {
System.out.println("que sera otra vez "+e.getLocalizedMessage());          }
*/List lista2=this.servicioManager.getListaServicio();
            List lista3=this.salaManager.getSala();
            List lista4=this.camaManager.getCama();
try{
if(request.getParameter("id_ser")!=null){
session.setAttribute("id_ser", request.getParameter("id_ser"));
//System.out.println("ummmm "+request.getParameter("id_ser")+"
"+Integer.parseInt(request.getParameter("id_ser"))+1000);
lista3=this.servicioManager.getConsultaSala(Integer.parseInt(request.getParameter("i
d_ser")));
lista4=this.servicioManager.getConsultaCama(Integer.parseInt(request.getParameter(
"id_ser")));
Servicio
ser=this.servicioManager.getFilaServicio(Integer.parseInt(request.getParameter("id_s
er")));
p.put("datos5",ser);
session.setAttribute("booleano","true");
        session.setAttribute("bandera","false");
        session.setAttribute("estado","enabled");     }
if(request.getParameter("id_sal")!=null){
Servicio
ser=this.servicioManager.getFilaServicio(Integer.parseInt((session.getAttribute("id_s
er").toString())));
```

```
Sala
sal=this.salaManager.getFilaSala(Integer.parseInt(request.getParameter("id_sal")));
lista3=this.servicioManager.getConsultaSala(Integer.parseInt((session.getAttribute("id_ser").toString())));
lista4=this.camaManager.getCamaSala(Integer.parseInt(request.getParameter("id_sal")));
Sala
lista_sala=this.salaManager.getSala2(Integer.parseInt(request.getParameter("id_sal"));
///session.setAttribute("id_sal",request.getParameter("id_sal"));
                                        //p.put("datos1", lista1);
                                        System.out.println("no se mete 3");
                                        p.put("datos2", lista2);
                                        p.put("datos3",lista3);
                                        p.put("datos4", lista4);
                                        p.put("datos5",ser);
                                        p.put("datos6", lista_sala);
                                        p.put("booleano","true");
                                        p.put("bandera", "true");

        p.put("cod_asegurado",session.getAttribute("cod_asegurado").toString());
        p.put("nro_asegurado",session.getAttribute("nro_asegurado").toString());
        p.put("cod_beneficiario",session.getAttribute("cod_beneficiario").toString());
        p.put("codigo",session.getAttribute("codigo").toString());
session.setAttribute("estado2","enabled");
```

```
return new ModelAndView("ingreso/admision",p);                    }
 }catch (Exception e) {System.out.println("que mierda sera
"+e.getLocalizedMessage());}
                    p.put("datos2",lista2);
                    p.put("datos3", lista3);
                    p.put("datos4",lista4);
                    p.put("estado",session.getAttribute("estado").toString());
                    p.put("estado2",session.getAttribute("estado2").toString());
                    p.put("booleano",session.getAttribute("booleano").toString());
                    p.put("bandera",session.getAttribute("bandera").toString());
p.put("cod_asegurado",session.getAttribute("cod_asegurado").toString());
p.put("nro_asegurado",session.getAttribute("nro_asegurado").toString());
p.put("cod_beneficiario",session.getAttribute("cod_beneficiario").toString());
p.put("codigo",session.getAttribute("codigo").toString());
                    //System.out.println("no se mete
nooooo"+session.getAttribute("cod_beneficiario")+"
"+session.getAttribute("codigo"));
                    }catch (Exception e) {
System.out.println("haber q pasa"+e.getLocalizedMessage());}
return new ModelAndView("ingreso/admision",p);    }
```

II.1.13.2.27 Llenar Datos Paciente

```
public ModelAndView dato_paciente(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
HttpSession session=request.getSession(true);
```

```
HashMap p = new HashMap();
Persona dato_persona;
                Beneficiario dato_beneficiario;
                session.setAttribute("id_cam", request.getParameter("id_cam"));
try{if(session.getAttribute("tipo").equals("asegurado")){
dato_persona=this.personaManager.getDatoPersonaAsegurado(Integer.parseInt(sessi
on.getAttribute("cod_asegurado").toString()));
p.put("reg",dato_persona);
String fecha[]=dato_persona.getFnac().toString().split("-");
//System.out.println("mmm "+dato_persona.getFnac().toString());
                        p.put("dia",fecha[2]);
                        p.put("mes",fecha[1]);
                        p.put("a-o",fecha[0]);
                        p.put("x","CHECKED");}else{
dato_beneficiario=this.personaManager.getDatoPersonaBeneficiario(Integer.parseInt(
session.getAttribute("cod_beneficiario").toString()));
                        p.put("reg",dato_beneficiario);
                        p.put("y","CHECKED");                    }
String nombre=null;
                List lista_tipoSeguro=null;
                if(session.getAttribute("tipo").equals("asegurado")){
lista_tipoSeguro=this.tipoSeguroManager.getTipoSeguroAsegurado(Integer.parseInt(
session.getAttribute("cod_asegurado").toString()));
```

```
}else{
lista_tipoSeguro=this.tipoSeguroManager.getTipoSeguroBeneficiario(Integer.parseInt(session.getAttribute("cod_beneficiario").toString()));
System.out.println("ya vas "+lista_tipoSeguro.size());                    }
                    for (Iterator iterator = lista_tipoSeguro.iterator();
iterator.hasNext();) {
                        Object object[] = (Object[]) iterator.next();
                        nombre=(String)object[1];}
                    if(nombre.equalsIgnoreCase("MATERNIDAD")) {
                        p.put("x","CHECKED");}
                    if(nombre.equalsIgnoreCase("enfermedad")) {
                        p.put("Y","CHECKED");}
                    if(nombre.equalsIgnoreCase("riesgo profesional")) {
                        p.put("Z","CHECKED");              }
                    try {List
lista_dato_paciente1=this.personaManager.getListaDatoPaciente1(Integer.parseInt(session.getAttribute("cod_asegurado").toString()));
System.out.println("hjashfghj "+lista_dato_paciente1.size());
for (Iterator iterator = lista_dato_paciente1.iterator(); iterator.hasNext();) {
Object object[] = (Object[]) iterator.next();
//d.cod_departamento,d.nombre,p.cod_provincia,p.nombre,e.cod_empresa,
//e.nombre_razonsocial,e.codemp,o.cod_ocupacion,o.nombre
                        p.put("cod_departamento",object[0]);
                        p.put("nombre_departamento",object[1]);
```

```
p.put("cod_provincia",object[2]);
p.put("nombre_provincia",object[3]);
p.put("cod_empresa",object[4]);
                    p.put("nombre_razonsocial",object[5]);
                    p.put("codemp",object[6]);
                    p.put("cod_ocupacion",object[7]);
                    p.put("nombre_ocupacion",object[8]);
                    p.put("num_minicipio_localidad1",object[9]);
p.put("nombre_municipio_localidad1",object[10]);
//System.out.println("....."+object[0]+object[1]+object[2]+object[3]+object[4]+obj
ect[5]+object[6]+object[7]+object[8]);         }
List
lista_dato_paciente2=this.personaManager.getListaDatoPaciente2(Integer.parseInt(se
ssion.getAttribute("cod_asegurado").toString()));
for (Iterator iterator = lista_dato_paciente2.iterator(); iterator.hasNext();) {
Object object[] = (Object[]) iterator.next();
//z.cod_zona,z.nombre,da.cod_direccion_asegurado,da.calle,da.numero,ml.num_muni
cipio_localidad,ml.nombre
                                        p.put("cod_zona",object[0]);
                                        p.put("nombre_zona",object[1]);
                                        p.put("direccion_asegurado",object[2]);
                                        p.put("calle",object[3]);
                                        p.put("numero",object[4]);
                                        p.put("num_municipio_localidad2",object[5]);
```

```
p.put("nombre_municipio_localidad2",object[6]);              }
} catch (Exception e) {                                    System.out.println("........:::::..
"+e.getLocalizedMessage());           }
}catch (Exception e) {
System.out.println("..........  ummm "+e.getLocalizedMessage());  }
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                    p.put("foto",session.getAttribute("foto"));
                    p.put("rol",session.getAttribute("rol"));
return new ModelAndView("ingreso/dato_paciente",p);    }
```

II.1.13.2.30 Llenar Datos consultorio

```
public ModelAndView dato_consultorio(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                    HttpSession session=request.getSession(true);
                    HashMap p = new HashMap();
try {                  List
lista_diagnostico=this.tipoDiagnosticoManager.getTipoDiagnostico();
List lista_prescripcion=this.tipoPrescripcionManager.getTipoPrescripcion();
List lista_estado=this.tipoEstadoManager.getTipoEstado();
p.put("datos",lista_diagnostico);
                              p.put("datos1", lista_estado);
                              p.put("datos2",lista_prescripcion);
                              Calendar calendario = new GregorianCalendar();
int hora =calendario.get(Calendar.HOUR_OF_DAY);
int minutos = calendario.get(Calendar.MINUTE);
```

```
int segundos = calendario.get(Calendar.SECOND);
String
hms=String.valueOf(hora)+":"+String.valueOf(minutos)+":"+String.valueOf(segundo
s);
p.put("hms",hms);
                } catch (Exception e) {
System.out.println("excepcion de "+e.getLocalizedMessage());     }
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                p.put("foto",session.getAttribute("foto"));
                p.put("rol",session.getAttribute("rol"));
                return new ModelAndView("ingreso/dato_consultorio",p);    }
```

II.1.13.2.31 Modulo Egreso

```
package controlador.ModuloIngreso;
import java.text.SimpleDateFormat;
import encoder.Base64;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.TimeZone;
```

```
import java.util.Vector;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import modelo.dominio.*;
import modelo.manager.*;
import encoder.Base64;
import org.hibernate.type.AnyType;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;
import encoder.Base64;
import controlador.*;
public class ingreso extends MultiActionController implements InitializingBean{
///inicio base afiliacion
        PersonaManager personaManager;
        AseguradoManager aseguradoManager;
        ServicioManager servicioManager;
        SalaManager salaManager;
        CamaManager camaManager;
        TipoDiagnosticoManager tipoDiagnosticoManager;
        TipoPrescripcionManager tipoPrescripcionManager;
```

```
TipoEstadoManager tipoEstadoManager;
TipoSeguroManager tipoSeguroManager;
ControlPacienteManager controlPacienteManager;
DetalleIngresoPacienteManager detalleIngresoPacienteManager;
HoraFechaIngresoManager horaFechaIngresoManager;
public ControlPacienteManager getControlPacienteManager() {
return controlPacienteManager;        }
        public void setControlPacienteManager(
        ControlPacienteManager controlPacienteManager) {
        this.controlPacienteManager = controlPacienteManager;    }
        public DetalleIngresoPacienteManager getDetalleIngresoPacienteManager() {
        return detalleIngresoPacienteManager;        }
        public void setDetalleIngresoPacienteManager(
DetalleIngresoPacienteManager detalleIngresoPacienteManager) {
this.detalleIngresoPacienteManager = detalleIngresoPacienteManager;     }
public HoraFechaIngresoManager getHoraFechaIngresoManager() {
return horaFechaIngresoManager;    }
public void setHoraFechaIngresoManager(
HoraFechaIngresoManager horaFechaIngresoManager) {
this.horaFechaIngresoManager = horaFechaIngresoManager;        }
public TipoSeguroManager getTipoSeguroManager() {
return tipoSeguroManager;    }
public void setTipoSeguroManager(TipoSeguroManager tipoSeguroManager) {
```

```
this.tipoSeguroManager = tipoSeguroManager;       }
public TipoDiagnosticoManager getTipoDiagnosticoManager() {
return tipoDiagnosticoManager;        }
public void setTipoDiagnosticoManager(
TipoDiagnosticoManager tipoDiagnosticoManager) {
this.tipoDiagnosticoManager = tipoDiagnosticoManager;   }
public TipoPrescripcionManager getTipoPrescripcionManager() {
                return tipoPrescripcionManager;       }
public void setTipoPrescripcionManager(
TipoPrescripcionManager tipoPrescripcionManager) {
                this.tipoPrescripcionManager = tipoPrescripcionManager;}
public TipoEstadoManager getTipoEstadoManager() {
                return tipoEstadoManager;    }
public void setTipoEstadoManager(TipoEstadoManager tipoEstadoManager) {
this.tipoEstadoManager = tipoEstadoManager;       }
public CamaManager getCamaManager() {
                return camaManager; }
public void setCamaManager(CamaManager camaManager) {
                this.camaManager = camaManager;  }
public SalaManager getSalaManager() {      return salaManager;   }
public void setSalaManager(SalaManager salaManager) {
                this.salaManager = salaManager;}
public ServicioManager getServicioManager() {
```

```
return servicioManager;}
public void setServicioManager(ServicioManager servicioManager) {
                this.servicioManager = servicioManager;     }
public AseguradoManager getAseguradoManager() {
                return aseguradoManager;     }
public void setAseguradoManager(AseguradoManager aseguradoManager) {
                this.aseguradoManager = aseguradoManager;}
///final clase afiliacion
public PersonaManager getPersonaManager() {
                return personaManager;}
public void setPersonaManager(PersonaManager personaManager) {
                this.personaManager = personaManager;     }
public void afterPropertiesSet() throws Exception {}
public ingreso() {       }
@SuppressWarnings("unchecked")
public ModelAndView ingreso(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                //String usuario=(String) session.getAttribute("usuario");
                Map p = new HashMap();
                if(session.getAttribute("xusuario")==null){
                  p.put("men","Negado el Acceso");
                p.put("url","index.html");
                return new ModelAndView( "mensaje",p);             }
```

```
ArrayList milistax = new ArrayList();
Map mapx = new HashMap();
String datofiltro="";
String tipo="";
int rango=1;
if(request.getParameter("xdatofiltro")==null){
                        datofiltro="";
                        }else{  datofiltro=request.getParameter("xdatofiltro");}
                        if(request.getParameter("xinter")==null){
                        rango=1;
}else{  rango=Integer.parseInt(request.getParameter("xinter"));     }
try {
                        //generador de el paginador
                        List Total = null;
try{
if(request.getParameter("tipo")==null) {//cuando entra por primera vez, es decir por
defecto                                                 Total =
this.personaManager.getListaUsuariosFiltroTotalAsegurado(datofiltro);
//session.setAttribute("tipo","asegurado");             }
                                        if(request.getParameter("tipo")!=null) {
if (request.getParameter("tipo").equalsIgnoreCase("asegurado")) {
Total = this.personaManager.getListaUsuariosFiltroTotalAsegurado(datofiltro);
}else{//de momento solo sera con asegurados y no beneficiarios
Total = this.personaManager.getListaUsuariosFiltroTotalAsegurado(datofiltro);
```

```
//Total = this.personaManager.getListaUsuariosFiltroTotalBeneficiario(datofiltro);

}

//session.setAttribute("tipo",request.getParameter("tipo"));           }

}catch (Exception e) {                    System.out.println("MENSAJE 1
"+e.getLocalizedMessage()); }

double num=Total.size();

double xx=Math.ceil(num / 5);

if(rango>1){

                                    mapx = new HashMap();

                                    mapx.put("rango","<a id='x'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='ingreso.html?tipo="+tipo+"&xdatofiltro="+datofiltro+"&xinter="+(rango-
1)+"'>"+"<<<"+"</a>");

                                    milistax.add(mapx);             }

for(int i=1;i<=xx;i++){

mapx = new HashMap();

if(i==rango){

mapx.put("rango",""+i+"");

}else{

mapx.put("rango","<a id='"+i+"' onMouseOver='Over(this.id)'
onMouseOut='Out(this.id)'
href='ingreso.html?tipo="+tipo+"&xdatofiltro="+datofiltro+"&xinter="+i+"'>"+i+"</
a>");              }

milistax.add(mapx);             }

if(rango<xx){
```

```
mapx = new HashMap();
mapx.put("rango","<a id='y' onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='ingreso.html?tipo="+tipo+"&xdatofiltro="+datofiltro+"&xinter="+(rango+1)+"
'>"+">>>"+"</a>");                    milistax.add(mapx);    }
                        //lista de los datos en 5 filas
                        List asegurado;
                        List beneficiario;
                        ArrayList lista_asegurado=new ArrayList();
                        ArrayList lista_beneficiario=new ArrayList();
                        if(request.getParameter("tipo")==null) {//cuando sea asegurado
asegurado = this.personaManager.getListaUsuariosFiltroAsegurado(datofiltro,rango);
for (Iterator iterator = asegurado.iterator(); iterator.hasNext();) {
Object[] object1 = (Object[]) iterator.next();
HashMap listita = new HashMap();
listita.put("cod_persona",object1[0]);
                                listita.put("nombre",object1[1]);
                                listita.put("apepat",object1[2]);
                                listita.put("apemat",object1[3]);
                                listita.put("estado",object1[4]);
                                listita.put("tipo","asegurado");
                                lista_asegurado.add(listita);                    }
p.put("datos",lista_asegurado);         }
                        try{if(request.getParameter("tipo")!=null) {
if (request.getParameter("tipo").equalsIgnoreCase("asegurado")) {
```

```
asegurado = this.personaManager.getListaUsuariosFiltroAsegurado(datofiltro,rango);
for (Iterator iterator = asegurado.iterator(); iterator.hasNext();) {
Object[] object1 = (Object[]) iterator.next();
HashMap mierda1 = new HashMap();
mier1.put("cod_persona",object1[0]);
session.setAttribute("cod_persona", object1[0].toString());
mier1.put("nombre",object1[1]);
mier1.put("apepat",object1[2]);
mier1.put("apemat",object1[3]);
mier1.put("estado",object1[4]);
System.out.println("para eliminar "+object1[4]);
mier1.put("tipo","asegurado");
lista_asegurado.add(mierda1);                                    }
p.put("datos",lista_asegurado);
                                                                 }else{//por el momento
solo sera con asegurado
beneficiario =
this.personaManager.getListaUsuariosFiltroBeneficiario(datofiltro,rango);
for (Iterator iterator = beneficiario.iterator(); iterator.hasNext();) {
Object[] object2 = (Object[]) iterator.next();
HashMap mier2 = new HashMap();
mier2.put("cod_persona",object2[0]);
session.setAttribute("cod_beneficiario",object2[0].toString());
mier2.put("nombre",object2[1]);
```

```
mier2.put("apepat",object2[2]);
mier2.put("tipo","beneficiario");
lista_beneficiario.add(mier2);                                    }
.put("datos", lista_beneficiario);       */
asegurado = this.personaManager.getListaUsuariosFiltroAsegurado(datofiltro,rango);
for (Iterator iterator = asegurado.iterator(); iterator.hasNext();) {
Object[] object1 = (Object[]) iterator.next();
HashMap mier1 = new HashMap();
mier1.put("cod_persona",object1[0]);
session.setAttribute("cod_persona", object1[0].toString());
mier1.put("nombre",object1[1]);
          mier1.put("apepat",object1[2]);
mier1.put("apemat",object1[3]);
mier1.put("estado",object1[4]);
mier1.put("tipo","asegurado");
lista_asegurado.add(mier1);
p.put("datos",lista_asegurado);         }         }
}catch (Exception e) {                                System.out.println("MENSAJE
2222222  "+e.getLocalizedMessage());                          }
//envio los intervalos
                  }  catch(Exception s){System.out.println("MENSAJE error
"+s.getLocalizedMessage());}
                  p.put("foto",session.getAttribute("foto").toString());
                  p.put("intervalo",milistax);
```

```
p.put("rol",session.getAttribute("rol"));
p.put("xusuario",Base64.decode((String) session.getAttribute("xusuario")))
//p.put("xclave", session.getAttribute("xclave"));
p.put("datofiltradox",datofiltro);
return new ModelAndView("ingreso/ingreso",p);   }
public ModelAndView adicionar_ingreso(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                HashMap p = new HashMap();
                Date d;
                ControlPaciente cp=new ControlPaciente();
try{
List lista1=this.controlPacienteManager.getControlPaciente();
cp.setId_cop(lista1.size()+1);
cp.setEstatura(Integer.parseInt(request.getParameter("estatura")));
cp.setPeso(Integer.parseInt(request.getParameter("peso")));
cp.setPresion(Integer.parseInt(request.getParameter("presion")));
cp.setPulso(Integer.parseInt(request.getParameter("pulso")));
cp.setTemperatura(Integer.parseInt(request.getParameter("temperatura")));
                this.controlPacienteManager.saveControlPaciente(cp);
                }catch (Exception e) {
System.out.println("----- control paciente ----- "+e.getLocalizedMessage());}
 HoraFechaIngreso hfi=new HoraFechaIngreso();
                try {
```

```
List lista2=this.horaFechaIngresoManager.getHoraFechaIngreso();
hfi.setId_hfi(lista2.size()+1);
                    hfi.setHora((request.getParameter("hora")));
SimpleDateFormat fechay = new SimpleDateFormat("dd-MM-yyyy");
                    d = fechay.parse(request.getParameter("fecha"));
                    hfi.setFecha(d);
                    this.horaFechaIngresoManager.saveHoraFechaIngreso(hfi);
} catch (Exception e) {System.out.println("----- horafecha ingreso-----
"+e.getLocalizedMessage()); }
TipoEstado tie=new TipoEstado();
try {    List lista_tie=this.tipoEstadoManager.getTipoEstado();
tie.setId_tie(lista_tie.size()+1);
tie.setNombre(request.getParameter("estado"));
tie.setDescripcion(request.getParameter("descripcion"));
                    this.tipoEstadoManager.saveTipoEstado(tie);
                } catch (Exception e) {
System.out.println("----- tipo estado----- "+e.getLocalizedMessage()        }
                TipoPrescripcion tip=new TipoPrescripcion();
                try {
                List lista_tip=this.tipoPrescripcionManager.getTipoPrescripcion();
                tip.setId_tip(lista_tip.size()+1);
                tip.setNombre(request.getParameter("prescripcion"));
                ///tip.setDescripcion(request.getParameter("descripcion"));
                this.tipoPrescripcionManager.saveTipoPrescripcion(tip);
```

```
} catch (Exception e) {
System.out.println("----- tipo  prescripcion----- "+e.getLocalizedMessage());       }
TipoDiagnostico tid=new TipoDiagnostico();
                    try {     List
lista_tid=this.tipoDiagnosticoManager.getTipoDiagnostico();
                              tid.setId_tid(lista_tid.size()+1);
                              tid.setNombre(request.getParameter("diagnostico"));
                              //tid.setDescripcion(request.getParameter("descripcion"));
                              this.tipoDiagnosticoManager.saveTipoDiagnostico(tid);
} catch (Exception e) {
System.out.println("----- tipo diagnostico----- "+e.getLocalizedMessage());}
//****************************************
                              DetalleIngresoPaciente dip=new DetalleIngresoPaciente();
                              List
lista3=this.detalleIngresoPacienteManager.getListaDetalleIngresoPaciente();
                              try{      dip.setId_dip(lista3.size()+1);
}catch (Exception e) {  System.out.println(" 1 "+e.getLocalizedMessage());}
try{//dip.setId_esp(Integer.parseInt((String)session.getAttribute("especialidad")));
}catch (Exception e) {
System.out.println("2 "+e.getLocalizedMessage());            }
try{dip.setId_hfi(hfi.getId_hfi());
}catch (Exception e) {System.out.println(" 3 "+e.getLocalizedMessage());
try {
int x=((Integer)session.getAttribute("id_usu"));
```

```
dip.setId_per((x));
} catch (Exception e) {
System.out.println(" 4 "+e.getLocalizedMessage());                    }
try {
dip.setId_tie(tie.getId_tie());
} catch (Exception e) {
System.out.println(" 5 "+e.getLocalizedMessage());                    }
try{dip.setId_tid(tid.getId_tid());
}catch (Exception e) {
System.out.println(" 6 "+e.getLocalizedMessage());          }
try {
dip.setId_tip(tip.getId_tip());
                        } catch (Exception e) {
System.out.println(" 7 "+e.getLocalizedMessage());                    }
try{
String xx=((String)session.getAttribute("id_cam"));
dip.setId_cam(Integer.parseInt(xx));
}catch (Exception e) {
System.out.println("9  "+e.getLocalizedMessage());                }
try {
dip.setId_cop(cp.getId_cop());
                        } catch (Exception e) {
System.out.println("10  "+e.getLocalizedMessage());         }
```

```
try{
                        String x=((String)session.getAttribute("cod_asegurado"));
                        dip.setCod_persona(Integer.parseInt(x));
                        }catch (Exception e) {
System.out.println(" 11 "+e.getLocalizedMessage());         }
try {dip.setPersona_condujo(request.getParameter("persona_condujo"));
} catch (Exception e) {
System.out.println(" 12 "+e.getLocalizedMessage());                    }
try {
String x=((String)session.getAttribute("cod_beneficiario"));
dip.setCod_beneficiario(Integer.parseInt(x));
} catch (Exception e) {
System.out.println(" 13 "+e.getLocalizedMessage());         }
try{
this.detalleIngresoPacienteManager.saveDetalleIngresoPaciente(dip);
                    }catch (Exception e) {
                                System.out.println(" el detalle
"+e.getLocalizedMessage()); }
                    p.put("xusuario",session.getAttribute("xusuario"));
                    p.put("xclave",session.getAttribute("xclave"));
/*{
Cama
ca=this.camaManager.getCama(Integer.parseInt((String)session.getAttribute("id_cam
")));ca.setEstado(1);
```

```
//ca.setEstado(Integer.parseInt((String)session.getAttribute("id_cam")));
this.camaManager.updateCama(ca);
                }catch (Exception e) {
                        p.put("men","no registrado correctamente");
                        p.put("url","ingreso.html");
                        System.out.println(" exception cama
"+e.getLocalizedMessage());}*/
                p.put("xusuario",session.getAttribute("xusuario") );
                p.put("xclave", session.getAttribute("xclave"));
p.put("men","ingreso registrado correctamente");
p.put("url","ingreso.html");
return new ModelAndView("mensaje",p);   }  }
```

II.1.13.2.32 Llenar Informe estadístico de egreso

```
public ModelAndView informe_estadistico_egreso(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                HashMap p = new HashMap();
                Date d = new Date();
                SimpleDateFormat fordat = new SimpleDateFormat("dd-MM-yyyy");
                String fechax = fordat.format(d);
                String x[]=fechax.split("-");
                p.put("dia",x[0] );
                p.put("mes", x [1]);
                p.put("a-o", x[2]);
```

```
try{
List
lista1=this.aseguradoManager.getListaDetalleEgresoPaciente(Integer.parseInt(request
.getParameter("cod_persona")));
                for (Iterator iterator = lista1.iterator(); iterator.hasNext();) {
                        Object[] object = (Object[]) iterator.next();
                        p.put("cod_persona",object[0]);
                        p.put("apepat",object[1]);
                        p.put("apemat",object[2]);
                        p.put("nombre_persona",object[3]);
                        p.put("sexo",object[4]);
                        System.out.println(".....eliminar
"+object[4].toString().toLowerCase());
if ((object[4].toString().toLowerCase()).equalsIgnoreCase("Masculino")) {
p.put("sexo1", "checked");
                        }else{
p.put("sexo2", "checked");              }
                        p.put("estadocivil",object[5]);
                        Date date=(Date)object[6];
                        String cadena_date=String.valueOf(date);
                        String cadena_date2[]=cadena_date.split("-");
                        System.out.println("aqui   "+x[2]+"   "+cadena_date2[0]);
p.put("edad_actual", Integer.parseInt(x[2])-Integer.parseInt(cadena_date2[0]));
p.put("cod_empresa",object[7]);
```

```
p.put("nombre_razonsocial",object[8]);        }
List
lista2=this.aseguradoManager.getListaDetalleEgresoPacienteAsegurado(Integer.parse
Int(request.getParameter("cod_persona")));
for (Iterator iterator = lista2.iterator(); iterator.hasNext();) {
                        Object[] object = (Object[]) iterator.next();
                        p.put("cod_asegurado", object[0]);
                        p.put("nro_asegurado", object[1]);              }
List
lista3=this.aseguradoManager.getListaDetalleEgresoPacienteBeneficiario(Integer.par
seInt(request.getParameter("cod_persona")));
for (Iterator iterator = lista3.iterator(); iterator.hasNext();) {
                        Object[] object = (Object[]) iterator.next();
                        p.put("cod_beneficiario", object[0]);
                        p.put("codigo", object[1]);
                        //System.out.println("ummmmmm "+object[3].toString()+"
"+object[3].toString().trim());
p.put("ESPOSO", "checked");
                        }else{
                                if(object[3].toString().equalsIgnoreCase("HIJO
(A)")){
p.put("HIJO", "checked");
                                }else{
p.put("OTROS", "checked");                      }       }       }
```

```
List
lista4=this.aseguradoManager.getListaDetalleEgresoPacienteTipoSeguro(Integer.pars
eInt(request.getParameter("cod_persona")));
for (Iterator iterator = lista4.iterator(); iterator.hasNext();) {
                        Object[] object = (Object[]) iterator.next();
                        p.put("cod_tipo_seguro", object[0]);
p.put("nombre", object[1].toString().toLowerCase());
if (object[1].toString().equalsIgnoreCase("enfermedad")) {
p.put("x", "CHECKED");              }
if (object[1].toString().equalsIgnoreCase("maternidad")) {
p.put("y", "CHECKED");                        }
if (object[1].toString().equalsIgnoreCase("riesgo profesional")) {
p.put("z", "CHECKED");     }
//System.out.println("....."+object[1].toString().toUpperCase());              }
List
lista5=this.detalleIngresoPacienteManager.getListaDetalleIngresoPaciente(Integer.pa
rseInt(request.getParameter("cod_persona")));
for (Iterator iterator = lista5.iterator(); iterator.hasNext();) {
                        Object[] object = (Object[]) iterator.next();
                        p.put("id_per", object[0]);
                        p.put("nombre_personal", object[1]);
                        p.put("am", object[2]);
                        p.put("ap", object[3]);
                        p.put("clave", object[4]);
```

```
p.put("id_tid", object[5]);
p.put("nombre_diagnostico", object[6]);                }
                }catch (Exception e) {
System.out.println("..... "+e.getLocalizedMessage());        }
p.put("cod_persona",request.getParameter("cod_persona"));
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                p.put("foto",session.getAttribute("foto"));
                p.put("rol",session.getAttribute("rol"));
return new ModelAndView("egreso/dato_paciente_egreso",p);    }
```

II.1.13.2.33 Llenar Datos de ingreso

```
public ModelAndView dato_ingreso(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                HashMap p=new HashMap();
                try{
                List
lista=this.detalleIngresoPacienteManager.getListaDetalleIngresoPacienteDatoIngreso
(Integer.parseInt(request.getParameter("cod_persona")));
for (Iterator iterator = lista.iterator(); iterator.hasNext();) {
                        Object[] name = (Object[]) iterator.next();
//hfi.id_hfi,fecha,hora,es.id_esp,es.id_ser,es.nombre,tid.id_tid,tid.nombre
                        p.put("id_hfi", name[0]);
                        p.put("fecha", name[1]);
                        p.put("hora", name[2]);
```

```
p.put("id_esp", name[3]);
p.put("id_ser", name[4]);
p.put("nombre_especialidad",name[5]);
p.put("id_tid", name[6]);
p.put("nombre_diagnostico", name[7]);       }
}catch (Exception e) {
System.out.println("....dato ingreso "+e.getLocalizedMessage());   }
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
p.put("foto",session.getAttribute("foto"));
p.put("rol",session.getAttribute("rol"));
return new ModelAndView("egreso/dato_ingreso",p);   }
```

II.1.13.2.34 Llenar Transferencias internas

```
public ModelAndView transferencias_internas(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                Map p = new HashMap();
                try {    } catch (Exception e) {
System.out.println(",,,,,,,, "+e.getLocalizedMessage());              }
p.put("foto",session.getAttribute("foto").toString());
p.put("rol",session.getAttribute("rol"));
p.put("xusuario",Base64.decode((String) session.getAttribute("xusuario")));
return new ModelAndView("servicio/transferencias_internas",p);    }
```

II.1.13.2.35 Llenar Intervencion quirúrgica

```
public ModelAndView intervenciones_quirurgicas(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
Map p = new HashMap();     try {
} catch (Exception e) {
System.out.println(",,,,,,,, "+e.getLocalizedMessage());      }
p.put("foto",session.getAttribute("foto").toString());
                p.put("rol",session.getAttribute("rol"));
                p.put("xusuario",Base64.decode((String)
session.getAttribute("xusuario")));
return new ModelAndView("servicio/intervenciones_quirurgicas",p);    }
```

II.1.13.2.36 Llenar datos de Maternidad

```
public ModelAndView maternidades(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                HashMap p=new HashMap();
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                p.put("foto",session.getAttribute("foto"));
                p.put("rol",session.getAttribute("rol"));
return new ModelAndView("egreso/maternidades",p);    }
```

II.1.13.2.37 Modulo Servicio

```
package controlador.ModuloServicio;
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import modelo.dominio.DetalleIngresoPaciente;
import modelo.manager.DetalleIngresoPacienteManager;
import modelo.manager.EspecialidadManager;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;
import encoder.Base64;
public class servicio extends MultiActionController implements InitializingBean{
EspecialidadManager especialidadManager;
```

```
public void afterPropertiesSet() throws Exception {}

public servicio() {      }

public ModelAndView kardex(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {

                HttpSession session=request.getSession(true);

                Map p = new HashMap();

                try {               } catch (Exception e) {

System.out.println(",,,,,,,, "+e.getLocalizedMessage());      }

p.put("foto",session.getAttribute("foto").toString());

p.put("rol",session.getAttribute("rol"));

p.put("xusuario",Base64.decode((String) session.getAttribute("xusuario")));

return new ModelAndView("servicio/kardex",p);    }

public ModelAndView detalle_ingreso_paciente(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {

HttpSession session=request.getSession(true);

Connection conexion = null; //Objeto para la conexión a la BD

                Statement sentencia = null; //Objeto que ejecuta sentencias

                ResultSet resultados = null;//Objeto que guardar resultados

                Map p = new HashMap();

                if(session.getAttribute("xusuario")==null){

                 p.put("men","Negado el Acceso");

                p.put("url","index.html");

                return new ModelAndView( "mensaje",p);            }

                ArrayList milistax = new ArrayList();
```

```
ArrayList milista = new ArrayList();
Map mapx = new HashMap();
String datofiltro="";
int rango=1;
if(request.getParameter("xdatofiltro")==null){
datofiltro="";
}else{
datofiltro=request.getParameter("xdatofiltro");                    }
                        if(request.getParameter("xinter")==null){
                                rango=1;
                        }else{
                        rango=Integer.parseInt(request.getParameter("xinter"));}
                try{    //generador de el paginador



//Leemos el driver de Postgresql
Class.forName("org.postgresql.Driver");
//Nos conectamos a la BD local
conexion = DriverManager.getConnection (
"jdbc:postgresql://localhost:5432/afiliacion",
"postgres","123456");
//Creamos una sentencia a partir de la conexión
sentencia=conexion.createStatement();
```

```
int id_esp=Integer.parseInt(request.getParameter("id_esp"));
System.out.println("vvvvvvvvv "+id_esp);
double num=0;
resultados=sentencia.executeQuery("SELECT
s.cod_persona,s.cod_beneficiario,s.id_hfi,s.fecha,p.apepat,p.apemat,p.nombre,p.estad
o FROM persona p "+
"INNER JOIN dblink('dbname=cns port=5432 host=localhost "+
"user=postgres password=123456','SELECT
dip.cod_persona,dip.cod_beneficiario,hfi.id_hfi,hfi.fecha "+
"FROM detalleingresopacientes dip,horafechaingresos hfi where dip.id_hfi=hfi.id_hfi
and dip.id_esp="+id_esp+" order by hfi.fecha desc ') "+
"AS s(cod_persona int,cod_beneficiario int,id_hfi int,fecha date) ON
p.cod_persona=s.cod_persona "+
"and upper(p.apepat || p.apemat||p.nombre) like upper('%"+datofiltro+"%')");
while(resultados.next()) {
num++;
System.out.println("----- "+num);                    }
double xx=Math.ceil(num / 5);
if(rango>1){
                                mapx = new HashMap();
                                mapx.put("rango","<a id='x'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='egreso.html?xdatofiltro="+datofiltro+"&xinter="+(rango-
1)+"'>"+"<<<"+"</a>");
                                milistax.add(mapx);   }
```

```
for(int i=1;i<=xx;i++){
mapx = new HashMap();
if(i==rango){
mapx.put("rango",""+i+"");
}else{
mapx.put("rango","<a id="'+i+'" onMouseOver='Over(this.id)'
onMouseOut='Out(this.id)'
href='egreso.html?xdatofiltro="+datofiltro+"&xinter="+i+"'>"+i+"</a>");}
milistax.add(mapx);                    }
if(rango<xx){
                                mapx = new HashMap();
                                mapx.put("rango","<a id='y'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='egreso.html?xdatofiltro="+datofiltro+"&xinter="+(rango+1)+"'>"+">>>"+"</a
>");                    milistax.add(mapx);                    }
                    //lista de los datos en 5 filas
                    //List mat =
this.detalleingresopacienteManager.getListaDetalleIngresoPacienteFiltro(datofiltro,ra
ngo);          //Creamos una sentencia a partir de la conexión
                    resultados=sentencia.executeQuery("SELECT
s.cod_persona,s.cod_beneficiario,p.apepat,p.apemat,p.nombre,s.id_hfi,s.id_dip,s.fech
a,p.estado,s.tipo from persona p "+
"INNER JOIN dblink('dbname=cns port=5432 host=localhost "+
"user=postgres password=123456','SELECT
dip.cod_persona,dip.cod_beneficiario,hfi.id_hfi,dip.id_dip,hfi.fecha,dip.tipo "+
```

```
"FROM detalleingresopacientes dip,horafechaingresos hfi where dip.id_hfi=hfi.id_hfi
and dip.id_esp="+id_esp+" order by hfi.fecha desc ') "+
"AS s(cod_persona int,cod_beneficiario int,id_hfi int,id_dip int,fecha date,tipo
varchar) ON p.cod_persona=s.cod_persona "+
"and upper(p.apepat || p.apemat||p.nombre) like upper('%"+datofiltro+"%') "+
"limit 5 offset (("+rango+"-1)*5) ");
                        while(resultados.next()) {
                                Map map = new HashMap();
                                map.put("cod_persona", resultados.getInt(1));
                                map.put("cod_beneficiario",
resultados.getInt(2));
                                map.put("apepat", resultados.getString(3));
                                map.put("apemat", resultados.getString(4));
                                map.put("nombre", resultados.getString(5));
                        map.put("id_hfi",resultados.getInt(6));
                        session.setAttribute("id_dip",resultados.getInt(7));
                                map.put("fecha",resultados.getDate(8));
                                map.put("estado", resultados.getInt(9));
                                map.put("tipo", resultados.getString(10));
                                milista.add(map);       }
try {
                        if (conexion!=null)
                        conexion.close();
                        }catch (SQLException e3) {
```

```
System.out.println("ERROR:Fallo al desconectar de la BD: "+
                                    e3.getMessage());                }
}catch (Exception e) {
System.out.println("que fue "+e.getLocalizedMessage());             }
                    //envio los intervalos
                    p.put("datos", milista);
                    p.put("intervalo",milistax);
          p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                    p.put("xclave", session.getAttribute("xclave"));
                    p.put("datofiltradox",datofiltro);
          p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                    p.put("foto",session.getAttribute("foto"));
                    p.put("rol",session.getAttribute("rol"));
               return new ModelAndView("servicio/detalle_ingreso_paciente",p);  }
          public ModelAndView adicionar_kardex(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                    HttpSession session=request.getSession(true);
                    Map p = new HashMap();
                    try {
                    } catch (Exception e) {
                    System.out.println(",,,,,,,, "+e.getLocalizedMessage());      }
                    p.put("foto",session.getAttribute("foto").toString());
p.put("rol",session.getAttribute("rol"));
p.put("xusuario",Base64.decode((String) session.getAttribute("xusuario")));
```

```
return new ModelAndView("servicio/adicionar_kardex",p);   }
/*** @return the especialidadManager*/
public EspecialidadManager getEspecialidadManager() {
return especialidadManager;}
/*** @param especialidadManager the especialidadManager to set          */
public void setEspecialidadManager(EspecialidadManager especialidadManager) {
this.especialidadManager = especialidadManager;   }         }
```

II.1.13.2.38 Cirugía

```
public ModelAndView cirugia(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                Map p = new HashMap();
                try {              List
lista1=this.especialidadManager.getServicioEspecialidad(1);
p.put("datos",lista1);
                } catch (Exception e) {
System.out.println(",,,,,,,, "+e.getLocalizedMessage());              }
                p.put("servicio","Cirugia");
                p.put("foto",session.getAttribute("foto").toString());
                p.put("rol",session.getAttribute("rol"));
                p.put("xusuario",Base64.decode((String)
session.getAttribute("xusuario")));
```

```
return new ModelAndView("servicio/cirugia",p);    }
```

II.1.13.2.39 Maternidad

```
public ModelAndView maternidad(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                Map p = new HashMap();
                try {                    List
lista1=this.especialidadManager.getServicioEspecialidad(2);
                        p.put("datos",lista1);
                } catch (Exception e) {
                        System.out.println(",,,,,,,, "+e.getLocalizedMessage());      }
                p.put("servicio","Maternidad");
                p.put("foto",session.getAttribute("foto").toString());
                p.put("rol",session.getAttribute("rol"));
                p.put("xusuario",Base64.decode((String)
session.getAttribute("xusuario")));
        return new ModelAndView("servicio/maternidad",p);    }
```

II.1.13.2.40 Medicina interna

```
public ModelAndView medicina_interna(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                Map p = new HashMap();
                try {            List
lista1=this.especialidadManager.getServicioEspecialidad(3);
```

```
p.put("datos",lista1);
} catch (Exception e) {
System.out.println(",,,,,,,, "+e.getLocalizedMessage());      }
p.put("servicio","Medicina interna");
                p.put("foto",session.getAttribute("foto").toString());
                p.put("rol",session.getAttribute("rol"));
                p.put("xusuario",Base64.decode((String)
session.getAttribute("xusuario")));
return new ModelAndView("servicio/medicina_interna",p);   }
```

II.1.13.2.41 Pediatria

```
public ModelAndView pediatria(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                Map p = new HashMap();
                try {
List lista1=this.especialidadManager.getServicioEspecialidad(4);
p.put("datos",lista1);
                } catch (Exception e) {
System.out.println(",,,,,,,, "+e.getLocalizedMessage());      }
                p.put("servicio","Pediatria");
                p.put("foto",session.getAttribute("foto").toString());
                p.put("rol",session.getAttribute("rol"));
                p.put("xusuario",Base64.decode((String)
session.getAttribute("xusuario")));
```

```
return new ModelAndView("servicio/pediatria",p);    }
```

II.1.13.2.42 UTI

```
public ModelAndView uti(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                Map p = new HashMap();
                try {                    List
lista1=this.especialidadManager.getServicioEspecialidad(5);
                        p.put("datos",lista1);
                } catch (Exception e) {
System.out.println(",,,,,,,, "+e.getLocalizedMessage());              }
                p.put("servicio","UTI");
                p.put("foto",session.getAttribute("foto").toString());
                p.put("rol",session.getAttribute("rol"));
                p.put("xusuario",Base64.decode((String)
session.getAttribute("xusuario")));
return new ModelAndView("servicio/uti",p);    }
```

II.1.13.2.43 Reportes

```
package controlador.ModuloReporte;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
```

```
import java.util.List;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import modelo.dominio.*;
import modelo.manager.*;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;
import controlador.*;
import encoder.Base64;
public class reporte extends MultiActionController implements InitializingBean{
public void afterPropertiesSet() throws Exception {}
public reporte() {}
public ModelAndView hospitalizacion(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                HttpSession session=request.getSession(true);
                //String usuario=(String) session.getAttribute("usuario");
                Map p = new HashMap();
                if(session.getAttribute("xusuario")==null){
                p.put("men","Negado el Acceso");
                p.put("url","index.html");
```

```
return new ModelAndView( "mensaje",p);   }
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                p.put("foto",session.getAttribute("foto"));
                p.put("rol",session.getAttribute("rol"));
                return new ModelAndView( "reporte/hospitalizacion",p);}}
```

II.1.13.2.44 Ver Hospitalizacion

```
public ModelAndView hospitalizacion(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
HttpSession session=request.getSession(true);
                //String usuario=(String) session.getAttribute("usuario");
                Map p = new HashMap();
                if(session.getAttribute("xusuario")==null){
                 p.put("men","Negado el Acceso");
                p.put("url","index.html");
                return new ModelAndView( "mensaje",p);   }
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                p.put("foto",session.getAttribute("foto"));
                p.put("rol",session.getAttribute("rol"));
return new ModelAndView( "reporte/hospitalizacion",p);   }
```

II.1.13.2.45 Ver Intervencion quirúrgica por especialidad

```
public ModelAndView
intervencion_quirurgica_por_especialidad(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
HttpSession session=request.getSession(true);
```

```
//String usuario=(String) session.getAttribute("usuario");
                    Map p = new HashMap();
                    if(session.getAttribute("xusuario")==null){
                     p.put("men","Negado el Acceso");
                    p.put("url","index.html");
                    return new ModelAndView( "mensaje",p);   }
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                    p.put("foto",session.getAttribute("foto"));
                    p.put("rol",session.getAttribute("rol"));
return new ModelAndView( "reporte/intervencion_quirurgica",p);}
```

II.1.13.2.46 Ver Intervencion quirúrgica por clase de cirugía

```
public ModelAndView
intervencion_quirurgica_por_clase_de_cirugia(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                    HttpSession session=request.getSession(true);
                    //String usuario=(String) session.getAttribute("usuario");
                    Map p = new HashMap();
                    if(session.getAttribute("xusuario")==null){
                     p.put("men","Negado el Acceso");
                    p.put("url","index.html");
                    return new ModelAndView( "mensaje",p);   }
p.put("xusuario",Base64.decode(session.getAttribute("xusuario").toString()));
                    p.put("foto",session.getAttribute("foto"));
                    p.put("rol",session.getAttribute("rol"));
```

return new ModelAndView(
"reporte/intervencion_quirurgica_por_clase_de_cirugia",p); }

II.1.13.2.47 Ayuda

/* * Java Base64 - A pure Java library for reading and writing Base64

* encoded streams. *

* Copyright (C) 2007-2009 Carlo Pelliccia (www.sauronsoftware.it) *

* This program is free software: you can redistribute it and/or modify

* it under the terms of the GNU Lesser General Public License version

* 2.1, as published by the Free Software Foundation. *

* This program is distributed in the hope that it will be useful,

* but WITHOUT ANY WARRANTY; without even the implied warranty of

* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

* GNU General Public License for more details. *

* You should have received a copy of the GNU Lesser General Public

* License version 2.1 along with this program.

* If not, see <http://www.gnu.org/licenses/>. */

package encoder;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

```
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
/** * <p>
 * Base64 encoding and decoding utility methods, both for binary and textual
 * informations.
 * </p> * */
public class Base64 {
/**
	* <p>
	* Encodes a string.
	* </p>
	* <p>
	* Before the string is encoded in Base64, it is converted in a binary
	* sequence using the system default charset.
	* </p> *
	*		If an unexpected error occurs.	*/
public static String encode(String str) throws RuntimeException {
byte[] bytes = str.getBytes();
byte[] encoded = encode(bytes);
try {
return new String(encoded, "ASCII");
} catch (UnsupportedEncodingException e) {
throw new RuntimeException("ASCII is not supported!", e);	}}
```

```
public static String encode(String str, String charset)
throws RuntimeException {
                byte[] bytes;
                try {
                bytes = str.getBytes(charset);
                } catch (UnsupportedEncodingException e) {
throw new RuntimeException("Unsupported charset: " + charset, e;}
byte[] encoded = encode(bytes);
try {
return new String(encoded, "ASCII");
} catch (UnsupportedEncodingException e) {
throw new RuntimeException("ASCII is not supported!", e);}}
public static String decode(String str) throws RuntimeException {
                byte[] bytes;
                try {
                bytes = str.getBytes("ASCII");
                } catch (UnsupportedEncodingException e) {
                throw new RuntimeException("ASCII is not supported!", e);}
                byte[] decoded = decode(bytes);
                return new String(decoded);  }
public static String decode(String str, String charset)
throws RuntimeException {
                byte[] bytes;
```

```
try {
bytes = str.getBytes("ASCII");
} catch (UnsupportedEncodingException e) {
throw new RuntimeException("ASCII is not supported!", e);         }
                    byte[] decoded = decode(bytes);
                    try {     return new String(decoded, charset);
                    } catch (UnsupportedEncodingException e) {
throw new RuntimeException("Unsupported charset: " + charset, e);         }         }
/**        * <p>   Encodes a binary sequence.  * </p>
* <p>   * If data are large, i.e. if you are working with large binary files,
* consider to use a {@link Base64OutputStream} instead of loading too much
* data in memory.       * </p>
*          * @param bytes         *   The source sequence.
* @return The encoded sequence.
* @throws RuntimeException
*            If an unexpected error occurs.
* @since 1.2    */
public static byte[] encode(byte[] bytes) throws RuntimeException {
return encode(bytes, 0);         }
public static byte[] encode(byte[] bytes, int wrapAt)
throws RuntimeException {
ByteArrayInputStream inputStream = new ByteArrayInputStream(bytes);
ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
```

```
try {
encode(inputStream, outputStream, wrapAt);
} catch (IOException e) {
throw new RuntimeException("Unexpected I/O error", e);
} finally {
try {
inputStream.close();
} catch (Throwable t) {         ;         }
try {
outputStream.close();
} catch (Throwable t) {         ;                   }         }
return outputStream.toByteArray();  }
/**       * <p>  * Decodes a binary sequence.          * </p>
* <p>   * If data are large, i.e. if you are working with large binary files,
* consider to use a {@link Base64InputStream} instead of loading too much
* data in memory.       * </p>
*  * @param bytes*    The encoded sequence.          * @return The decoded
sequence.
* @throws RuntimeException
*  If an unexpected error occurs. * @since 1.2         */
public static byte[] decode(byte[] bytes) throws RuntimeException {
ByteArrayInputStream inputStream = new ByteArrayInputStream(bytes);
ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
try {
```

```
decode(inputStream, outputStream);
} catch (IOException e) {
throw new RuntimeException("Unexpected I/O error", e);
} finally {
                        try {
                                inputStream.close();
                        } catch (Throwable t) {         ;         }
                        try {                       outputStream.close();
                        } catch (Throwable t) {;        }         }
                return outputStream.toByteArray();  }
public static void encode(InputStream inputStream,
OutputStream outputStream, int wrapAt) throws IOException {
Base64OutputStream aux = new Base64OutputStream(outputStream, wrapAt);
                copy(inputStream, aux);
                aux.commit(); }
/**       * <p>  * Decodes data from the given input stream and writes them in the
given    * output stream.         * </p>
 * <p>  * The supplied input stream is read until its end is reached, but it's not
 * closed by this method.       * </p>
 * <p>  * The supplied output stream is nor flushed neither closed by this method.
 * </p> *
 * @param inputStream          *          The input stream from which encoded data are
read.
 * @param outputStream
```

```
*           The output stream in which decoded data are written.
* @throws IOException
*           If an I/O error occurs.
*/
public static void decode(InputStream inputStream, OutputStream outputStream)
                        throws IOException {
copy(new Base64InputStream(inputStream), outputStream);         }
public static void encode(File source, File target, int wrapAt)
throws IOException {
InputStream inputStream = null;
                OutputStream outputStream = null;
                try {
                        inputStream = new FileInputStream(source);
                        outputStream = new FileOutputStream(target);
                        Base64.encode(inputStream, outputStream, wrapAt);
                } finally {
                        if (outputStream != null) {
                                try {    outputStream.close();
                                } catch (Throwable t) {         ;         }}
                        if (inputStream != null) {
                                try {                           inputStream.close();
                                } catch (Throwable t) {         ;         }}}}
```

```
public static void encode(File source, File target) throws IOException {
InputStream inputStream = null;
OutputStream outputStream = null;
                try {
                        inputStream = new FileInputStream(source);
                        outputStream = new FileOutputStream(target);
                        Base64.encode(inputStream, outputStream);
                } finally {
                        if (outputStream != null) {
                                try {                   outputStream.close();
                                } catch (Throwable t) {         ;         }         }
                        if (inputStream != null) {
                                try {inputStream.close();
                                } catch (Throwable t) {         ;}
}}}
```

II.1.14 CASOS DE PRUEBA

II.1.14.1 INTRODUCCION

Cada prueba es especificada mediante un documento que establece las condiciones de ejecución como las entradas y los resultados esperados. Estos casos son aplicados como pruebas de regresión en cada iteración. Cada caso llevara asociado un procedimiento de prueba con las instrucciones para realizar las mismas y dependiendo del tipo que se utilice, este procedimiento podrá ser automatizable mediante un script.

II.1.14.2 DEFINICION

La prueba es un proceso de ejecución de un programa con la intención de descubrir errores.

Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error desconocido hasta entonces.

Todos los productos de Software son probados de dos formas:

- Conociendo la función específica para la que fue diseñado el producto, se puede llevar a cabo pruebas que demuestren que cada función es completamente operativa denominada prueba de caja negra.
- Conociendo el funcionamiento del producto, se pueden realizar pruebas que aseguren que la especificación interna se ajuste a las especificaciones y que todos los componentes hospitalizados se hayan comprobado de forma adecuada, esta forma se denomina prueba de caja blanca.

Para la prueba al sistema se utilizara prueba de caja negra.

II.1.14.3 PRUEBA DE CAJA NEGRA.

La prueba funcional o de caja negra se centra en el estudio de la especificación del software, del análisis de las funciones que debe realizar de las entradas y de las salidas.

Los errores que pretenden detectar mediante las pruebas de caja negra son:

- Funcionalidad incorrecta o ausente.
- Errores de rendimiento.
- Errores de interfaces.
- Errores en la estructura de datos.
- Errores de inicialización y terminación.

Para realizar las pruebas al sistema, se utilizara el método de particiones o clases de equivalencia a todos aquellos formularios que sean necesarios y tengan entrada de datos. El diseño de casos de este método consiste en:

- Identificación de clases de equivalencia.
- Creación de los casos de prueba correspondientes.

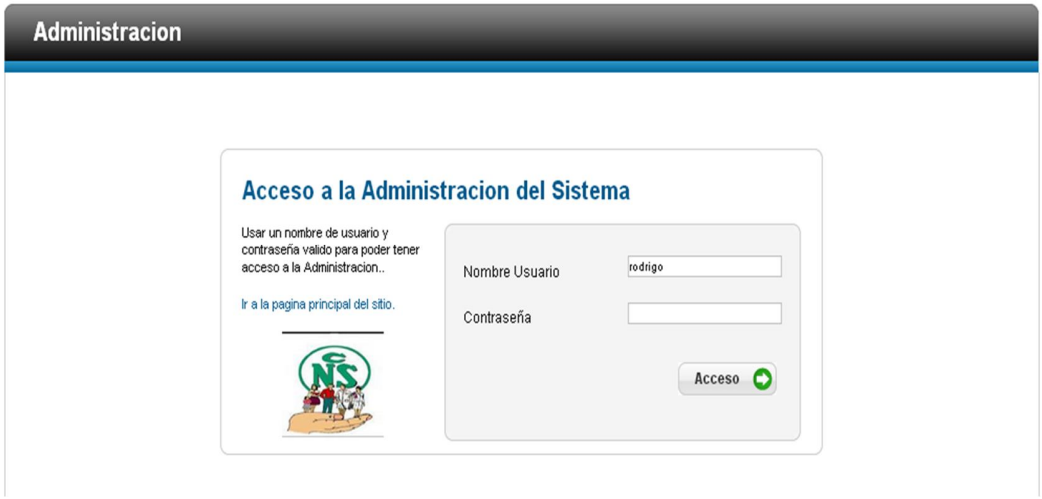
Para identificar las posibles clases de equivalencia de un programa a partir de su especificación se deben seguir los siguientes pasos:

- Identificación de las condiciones de las entradas del programa, es decir, restricciones de formato o contenido de los datos de entrada.
- A partir de ellas se identifican las clases de equivalencia que pueden ser:
    - De datos validos
    - De datos no validos o erróneos.

Técnica: Partición Equivalente

II.1.14.4 PRUEBA DE CAJA NEGRA AL SISTEMA

II.1.14.4.1 iniciar



Datos contenidos en una aplicación de automatización, el software captura los datos de la siguiente forma:

Datos: **Usuario:** letra, tamaño 10

Clave: letra, tamaño 10

Condiciones de Entrada	Clases Equivalencia Validas	Clases Equivalencia Invalidas
Usuario especificado	**1** Si	**2.** No
Usuario	**3** Letra	**4.** Cualquier otro
Tamaño de Usuario	**5** Valor <=10	**6.** <=0
Clave especificada	**7** Si	**8.** No
Contraseña	**9** Letra	**10.** Cualquier otro carácter

Tamaño de Contraseña	**1** Valor <=10	**12.** <=0

Identificación de los Casos de Prueba que sean uno o más clases de equivalencia

Clases Válidas

Cp.	**Usuario**	**Clave**
	juan	jurad

Cubre las clases de equivalencia válidas: 1-3-5-7-9-11

Clases No Válidas

Cp.	**Usuario**	**Clave**
		jurad

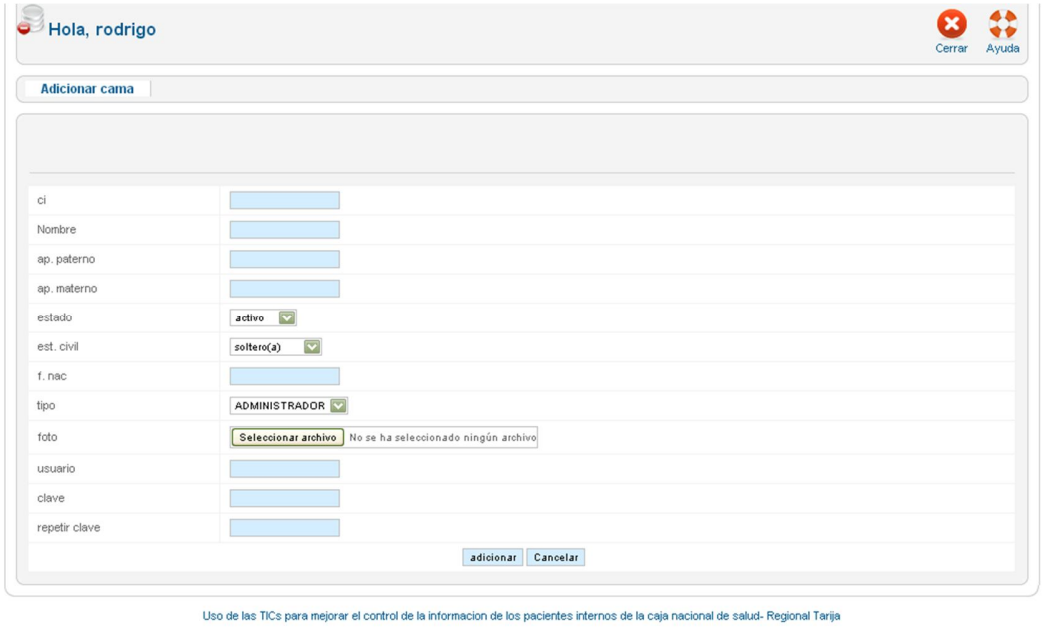
Cubre las clases de equivalencia no válida:2-4-6

Cp.	**Usuario**	**Clave**
	--..:) ☺	++ -☺☺

Clases: 4-6-10-12

II.1.14.4.2 modulo administracion

II.1.14.4.2.1 personal



Datos contenidos en una aplicación de automatización de Administración de personal, el software captura los datos de la siguiente forma:

Datos:

CI: letra, de 8 dígitos

Nombre: letra, tamaño 20

Ap.: letra, tamaño 20

Ap.: letra, tamaño 20

Condiciones de Entrada	**Clases Equivalencia Validas**	**Clases Equivalencia Invalidas**
CI	**1.** Número	**2** Cualquier otro carácter
Tamaño de CI	**3.** 8	**4** ¡!=7

Nombre	**5.** Letra	**6** Cualquier otro carácter
Tamaño de Nombre	**7.** Valor <=20	**8** >20
Ap.	**9.** Letra	**1** Cualquier otro carácter
Tamaño de Ap.	**11.** Valor <=20	**1** >20
Am.	**13.** Letra	**1** Cualquier otro carácter
Tamaño de Am.	**15.** Valor <=20	**1** >20
Domicilio	**17.** Numero	**1** Cualquier otro carácter
Tamaño de Nº	**19.** Valor <=10	**2** >10
Calle	**21.** Letra	**2** Cualquier otro carácter
Tamaño de Calle	**23.** Valor <=30	**2** >30
Zona	**25.** Letra	**2** Cualquier otro carácter
Tamaño de Zona	**27.** Valor <=30	**2** >30
Teléfono	**29.** Letra y numero	**3** Cualquier otro carácter
Tamaño de teléfono	**31.** Valor <=20	**3** >20

Identificación de los Casos de Prueba que sean uno o más clases de equivalencia Clases Válidas

CI	**nombre**	**Ap**	**Am.**	**Domicilio**	**Calle**	**Zona**	**Teléfono**
7654321	rodrigo	garcia		07654	celedonio	El Tejar	66-61668

Cubre las clases válidas: 1-3-5-7-9-11-13-15-17-19-21-23-25-27-29-31-33

Clases No Válidas

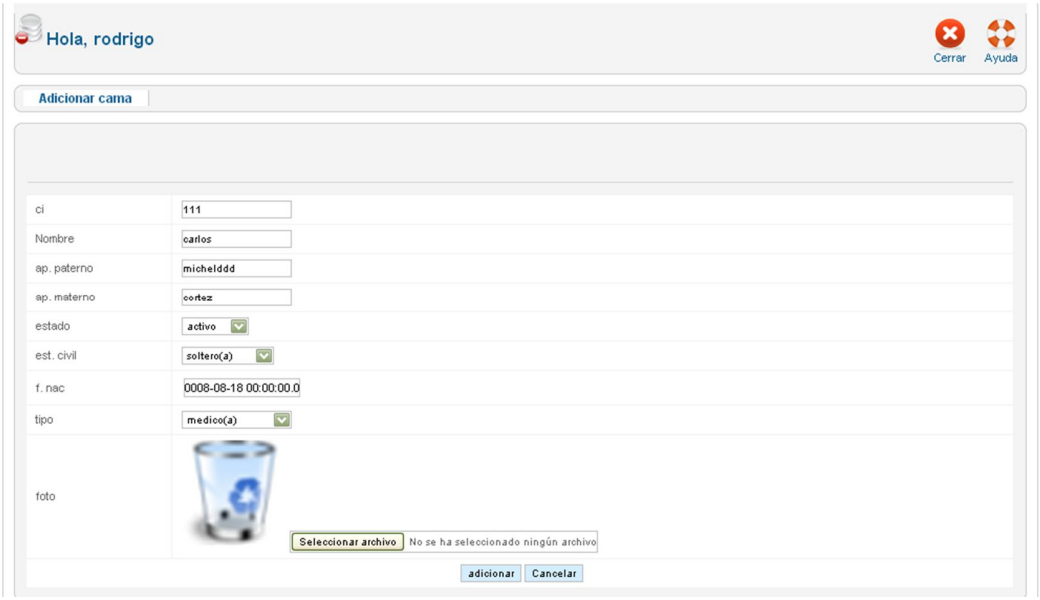
rude	CI	nombre	Ap	Am	Nº Domicilio	Calle	Zona	Teléfono

Cubre las clases de equivalencia no válida:2-4-6-8-10-12-14-20-22-28-30-32-34

	hola	…☺,..	*****be	☺☺☺☺****	No se	(>☺<)	;) ;;;;	***{{,,,

Clases: 4-6-8-10-12-14-16-18-20-22-24-26-28-30-32-34

II.1.14.4.2.2 Modificar



Datos contenidos en una aplicación de automatización de modificar personal, el software captura los datos de la siguiente forma:

Datos:

Nombre: letra, tamaño 20

Ap.: letra, tamaño 20

Ap.: letra, tamaño 20

Direccion:

Nº número, tamaño 10

Calle: letra, tamaño 30

Zona: letra, tamaño 30

Teléfono: letra, tamaño 20

Condiciones de Entrada	Clases Equivalencia Validas	Clases Equivalencia Invalidas
Tamaño de Ap. Paterno	**1.** Valor <=20	**2.** >20
Ap. Materno	**3.** Letra	**4.** Cualquier otro carácter
Tamaño de Ap. Materno	**5.** Valor <=20	**6.** >20
Nº Domicilio	**7.** Numero	**8.** Cualquier otro carácter
Tamaño de Nº Domicilio	**9.** Valor <=5	**10** >5
Calle	**11** Letra	**12** Cualquier otro carácter
Tamaño de Calle	**13** Valor <=30	**14** >30
Zona	**15** Letra	**16** Cualquier otro carácter
Tamaño de Zona	**17** Valor <=50	**18** >50
Teléfono	**19** Letra y numero	**20** Cualquier otro carácter
Tamaño de teléfono	**21** Valor <=30	**22** >30

Identificación de los Casos de Prueba que sean uno o más clases de equivalencia Clases Válidas

Cp.	nombre	Ap	Am.	Domicilio	Calle	Zona	Teléfono
	rodrigo	garcia	zambrana	07654	celedonio	El Tejar	66-61668

Cubre las clases de equivalencia válidas: 1-3-5-7-9-11-13-15-17-19-21

Clases No Válidas

Cp.	nombre	Ap	Am	Nº Domicilio	Calle	Zona	Teléfono
			zambrana		celedonio		

Cubre las clases de equivalencia no válida:2-4-6-8-10-12-14-16-18-20-22

Cp.	rude	CI	nombre	Ap	Am	Nº Domicilio	Calle	Zona	Teléfono
		hola	…☺,..	;) *****be	☺☺☺☺****	No se	(>☺<)	;) ;;;;	***{{,,,

Clases: 2-4-6-8-10-12-14-16-18-20-22

II.1.14.4.3 camas

II.1.14.4.3.1 adicionar cama

Datos contenidos en una aplicación de automatización de creación de cama, el software captura los datos de la siguiente forma:

nombre: letra, tamaño 40

Condiciones de Entrada	Clases Equivalencia Validas	Clases Equivalencia Invalidas
nombre	**1.** si	**2.** no
nombre	**3.** Letra	**4.** Cualquier otro carácter
Tamaño de nombre	**5.** Valor <=40	**6.** >40

Identificación de los Casos de Prueba que sean uno o más clases de equivalencia Clases Válidas

Cp.	nombre
	derecha

Cubre las clases de equivalencia válidas: 1-3-5

Clases No Válidas

Cp.	nombre

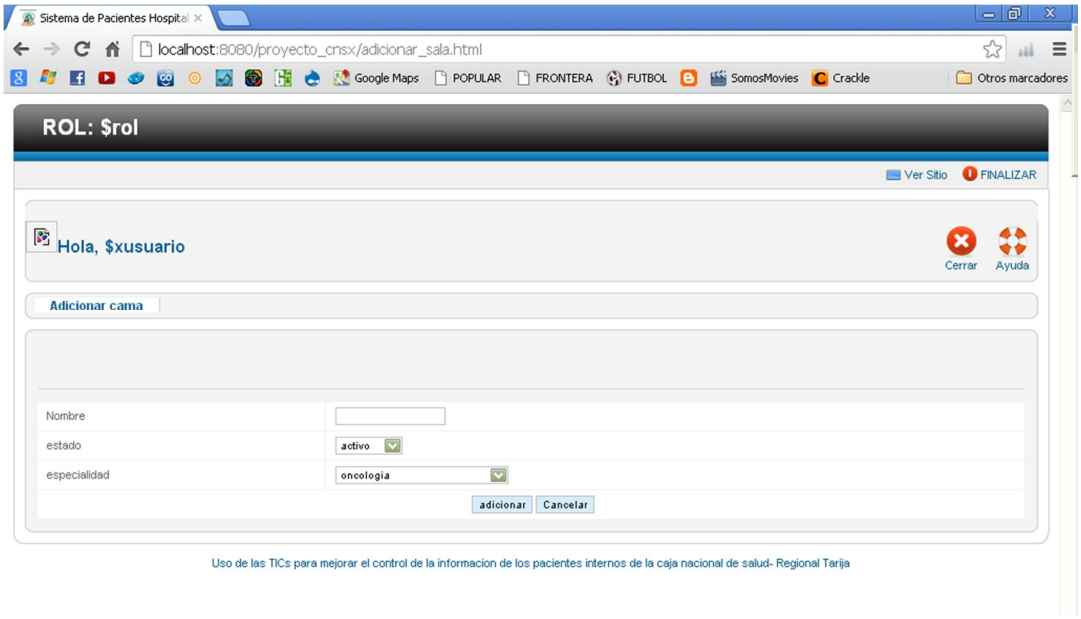
Cubre las clases de equivalencia no válida: 2-6

Cp.	nombre
	;) ;) ☺☺☺***☺☺☺ ;) ;)

Clases: 2-4

II.1.14.4.4 sala

II.1.14.4.4.1 adicionar sala



Datos contenidos en una aplicación de automatización de creación de sala, el software captura los datos de la siguiente forma:

nombre: letra, tamaño 40

Condiciones de Entrada	Clases Equivalencia Validas	Clases Equivalencia Invalidas
nombre	**7.** si	**8.** no
nombre	**9.** Letra	**10** Cualquier otro carácter
Tamaño de nombre	**11** Valor <=40	**12** >40

Identificación de los Casos de Prueba que sean uno o más clases de equivalencia Clases Válidas

Cp.	nombre
	juegos

Cubre las clases de equivalencia válidas: 1-3-5

Clases No Válidas

Cp.	nombre

Cubre las clases de equivalencia no válida: 2-6

Cp.	nombre
	;) ;) ☺☺☺***☺☺☺ ;) ;)

Clases: 2-4

II.1.15 LISTA DE RIESGOS

II.1.15.1 Introducción

Son los riesgos identificados a lo largo del desarrollo del sistema que se podrá minimizar si se sigue con las estrategias de atenuación.

II.1.15.1.1 Propósito

Identificar los posibles riesgos en el desarrollo e implementación del sistema.

II.1.15.1.2 Alcance

Garantizar que los riesgos se minimicen durante el periodo de desarrollo del Sistema.

II.1.15.2 Lista de Riesgos

Riesgo	**Acción**
Vago conocimiento de uno de las herramientas utilizadas a lo largo del proyecto.	Por ejemplo: No manejar la notación U.M.L. resultado: Retraso en la presentación.
Falta de coordinación en los nombres de los métodos y los casos de uso	Métodos no coordinan a la hora de ejecutar el sistema
No terminar todos los componentes, no entregar el proyecto en tiempo estimado	No es posible presentar en el tiempo previsto el proyecto. Proyecto incompleto
Errores en el producto	Errores encontrados en el código del software.

Tabla 58. Lista de Riesgos

II.1.15.2.1 Gestion de riesgos

Riesgo	**Descripción**	**Probabi-lidad %**	**Impacto**	**Plan frente al riesgo**	**Supervisión**
Vago conocimiento	Por ejemplo: el no manejar	20%	Retraso en la	Buscar capacitación	Director de

de uno de las herramientas utilizadas a lo largo del proyecto.	correctamente la notación U.M.L.		presentación.	acerca del manejo de la herramienta.	proyecto.
Falta de coordinación en los nombres de los métodos y los casos de uso	Métodos no coordinan a la hora de ejecutar el sistema	50%	Proyecto incompleto	Mejorar coordinación de los nombres de los métodos.	Director de proyecto.
No terminar todos los componentes, no entregar el proyecto en tiempo estimado	No es posible presentar en el tiempo previsto el proyecto. Proyecto incompleto	20%	Retraso en la presentación.	Coordinar mejor el tiempo de desarrollo del proyecto	Director de proyecto.
Errores en el producto	Se presenta errores en el código del sistema	30%	Bajo nivel en el rendimiento del producto.	Realizar pruebas constantes sistema	Director de proyecto.

Tabla 59. Gestion de riesgos

II.1.15.2.2 Análisis de Riesgos

Riesgo y Descripción	**Ponderación y Efecto**	**Estrategias de Atenuación**	**Estrategias de Anulación**	**Plan de Contingencia**	**Políticas de Supervisión y Seguimiento**
Vago conocimiento de uno de las herramientas utilizadas a lo largo del proyecto. **Tipo** : Negocio	50% Moderado	Actualización de los conocimientos		mejorar los conocimientos respeto al rol a desarrollar	Capacitar constantemente
Falta de coordinación en los nombres de los métodos y los casos de uso **Tipo**: Negocio	65% Alto	Reducción al mínimo los problemas de comunicación	Mejorar la coordinación de los diagramas	Capacitación en el rol que se maneja	Mejorar la organización
No terminar todos los componentes, no entregar el proyecto en	90% Catastrófico	Minimizar complejidad de Componen	Establecer prioridad de compone	Reutilizar componentes ya creados	Controlar constantemente la calendarización para el desarrollo

tiempo estimado **Tipo**: Proyecto		tes a desarrollar	ntes		de componentes.
Errores en el producto **Tipo**: Producto	20% Bajo	Utilizar software que realiza corrección de ciertos errores	Realizando iteraciones en el proyecto	Realizar pruebas.	Revisiones periódicas en el producto, pruebas

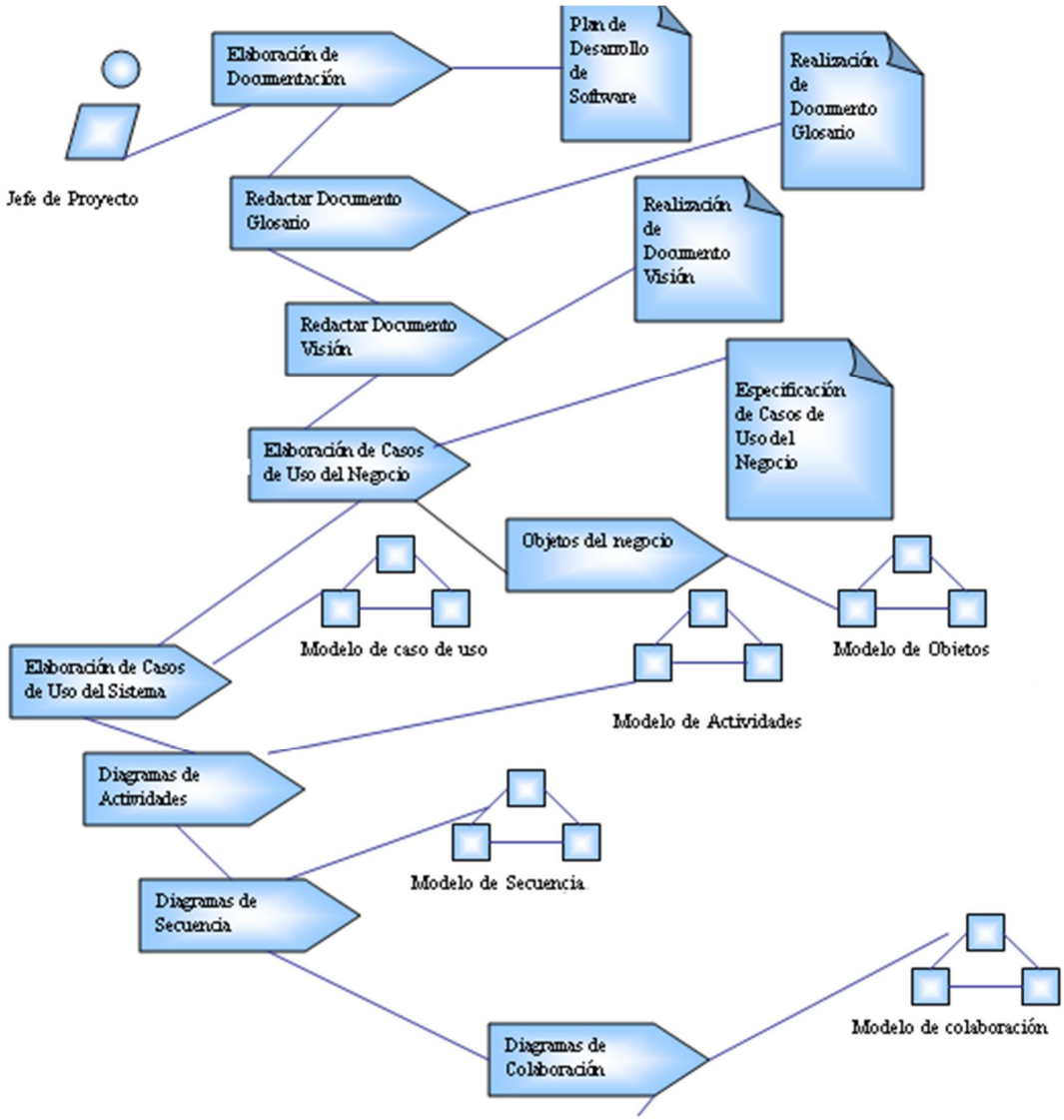
Tabla 60. Analisis de Riesgos

II.1.16 PLAN DE ITERACION

II.1.16.1 Introducción

Es un conjunto de actividades y tareas ordenadas temporalmente, con recursos asignados, dependencias entre ellas. Se realiza para cada iteración, y para todas las fases.

II.1.16.2 Iteración



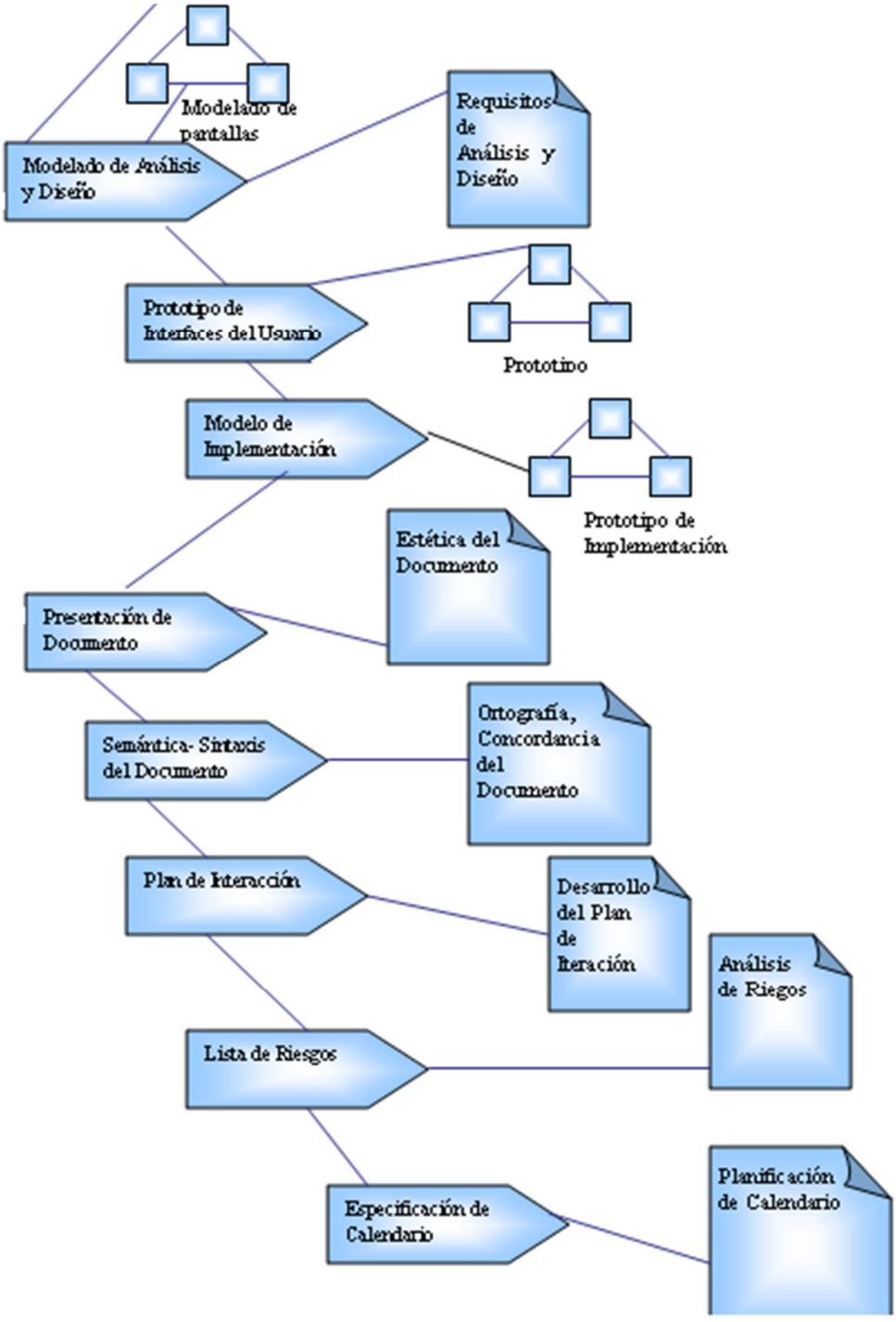


Figura 184. Plan de Iteracion

II.1.16.3 Evaluación de Iteración

Este documento incluye le evaluación de los resultados de cada iteración, el grado en el cual se han conseguido los objetivos de la iteración, las lecciones aprendidas y los cambios a ser realizados.

Rol	**Nombre de Actividad**	**Artefacto**	**Estado**	**Detalle de Cumplimiento**
Director de Proyecto	Elaboración de Documentación	Plan de desarrollo de software	En desarrollo	Se cumplio de acuerdo a lo establecido
Director de Proyecto	Redacción de Documento Glosario	Realización de Documento Glosario	En desarrollo	Se cumplio de acuerdo a lo establecido
Director de Proyecto	Redacción de Documento Visión	Realización de Documento Visión	En desarrollo	Se cumplio de acuerdo a lo establecido
Director de Proyecto	Casos de Uso de Negocio	Especificación de Casos de Uso del Negocio	Terminado sujeto a modificaciones	Se cumplio de acuerdo a lo establecido
Director de Proyecto	Objetos del Negocio	Modelo de Objetos	Terminado sujeto a modificaciones	Se cumplio de acuerdo a lo establecido
Director de Proyecto	Diagramas de Secuencia	Modelado de secuencia	En desarrollo	Se realizo modificaciones a lo planeado

				debido a cambios en los nombres de los métodos porque no concordaban
Director de Proyecto	Diagramas de colaboración	Modelado de Colaboración	En desarrollo	Se realizo modificaciones a lo planeado debido a cambios en los nombres de los métodos porque no concordaban
Director de Proyecto	Modelado de Análisis y Diseño	Modelado de Pantallas	En Desarrollo	Se realizo modificaciones a lo planeado debido a cambios en los nombres de los métodos porque no concordaban
Director de Proyecto	Modelado de Análisis y Diseño	Requisitos de Análisis y Diseño	En desarrollo	Se está cumpliendo de acuerdo a lo establecido
Director de	Prototipo de Interfaz del	Prototipo	En Desarrollo	Se realizo modificaciones a

Proyecto	Usuario			lo planeado debido a cambios en los nombres de los métodos porque no concordaban
Director de Proyecto	Modelado de Implementación	Prototipo de Implementación	En Desarrollo	Se cumplio de acuerdo a lo establecido
Director de Proyecto	Presentación del Documento	Estética del Documento	En Desarrollo	Se modifico la fecha
Director de Proyecto	Semántica-Sintaxis del Documento	Ortografía, Concordancia del Documento	En Desarrollo	Se modifico la fecha
Director de Proyecto	Plan de Interacción	Desarrollo del Plan de Iteración	En Desarrollo	Se a realizado modificaciones a lo planeado debido a cambios realizados en los elementos de control.
Director de Proyecto	Lista de Riesgos	Análisis de Riegos	En Desarrollo	Se esta cumpliendo de acuerdo a lo establecido

Director de Proyecto	Especificación de Calendario	Planificación de Calendario	En Desarrollo	Se modifico la fecha de acuerdo a lo establecido

Tabla 61. Evaluacion de iteracion

II.2 Personal capacitado en el uso del software

II.2.1 Capacitación

En la actualidad la capacitación de los recursos humanos es la respuesta a la necesidad que tienen las empresas o instituciones de contar con un personal calificado y productivo.

La obsolescencia también es una de las razones por la cual, las instrucciones se preocupan por capacitar a sus recursos humanos, pues esta procura actualizar sus conocimientos con las nuevas técnicas y métodos de trabajo que garantizan eficiencia.

Los procesos de capacitación son recibidos en muchas ocasiones como una oportunidad de crecimiento y de aprendizaje con el fin de, no sólo de mejorar la tarea y el desempeño para el cual hemos sido contratados, sino también para crecer como personas, para interiorizar contenidos que quizá no tenga aplicación inmediata pero que dan temple y seguridad para las oportunidades futuras.

II.2.2 Importancia de la Capacitación

A mayor desarrollo tecnológico en la sociedad, mayor necesidad de talento, de personas competentes técnica y emocionalmente capaces de crear, innovar, crear valor, afrontar retos en los negocios, elaborar bienes y servicios de calidad y contribuyan a que la organización aprenda a mantenerse en un mercado globalizado, la tendencia es que las organizaciones se conviertan en comunidades de aprendizaje que lo generen, lo conserven y lo traduzcan en acciones de valor agregado, la sobrevivencia en el mundo global y competitivo depende, en estos momentos, de la inversión que hagan las empresas en intangibles, como innovación tecnológica, organización flexible y desarrollo de capital humano. Así, la utilización del conocimiento apropiado se convierte en la principal fuente de ventaja competitiva para una organización en la época actual.

II.2.3 Metodología DICE

DICE es una metodología que permite fortalecer las competencias. Esta metodología integra cuatro elementos importantes que son Diagnostico, Intervención, Comprobación, Evaluación.

II.2.3.1 Diagnostico

Es el análisis de necesidades de capacitación, permite al responsable de capacitación encontrar no solamente los temas sobre los cuales se debe trabajar, sino también permite definir la profundidad requerida para cada uno de los temas, establecer la intensidad horaria de cada intervención, definir los grupos de participantes, es decir nos permite estructurar la capacitación.

En base a los requerimientos expresados por el encargado de sistemas se realizo el presente proyecto, además se pudo constatar que no existe un buen conocimiento sobre el uso de los equipos de computación, que en realidad se tienen pero no se utilizan por lo mismo.

II.2.3.1.1 Estructura de la capacitación

II.2.3.1.1.1 Objetivo

El objetivo central de la capacitación es capacitar a las autoridades administrativas de sistemas delhospital Obrero en el Uso del sistema y las aplicaciones que se requieren para el mismo.

Permitiendo a los participantes de la capacitación conocer nuevas tecnologías que faciliten su trabajo en lo que se refiere a la administración.

II.2.3.1.2 Cronograma

Presentación del proyecto al hospital Obrero pertenecientes a Region Tarija

Jueves 25 de octubre de 2012 16:00 p.m.

Area de Sistemasdel Policlinico

Tiempos de Ejecución: 3 horas

Temas

- Introducción al internet
- Exposicion del Sistema Mejoramiento del Control de la Información de los Pacientes Hospitalizados en la Caja Nacional de Salud Regional Tarija
- Manejo del sistema por parte de los participantes de la capacitación, aplicación de la metodología DICE.

II.2.3.1.3 Resultados de la capacitación:

La capacitación se llevo a cabo a la hora establecida, el proyecto fue aceptado conforme a lo acordado y se pudo evidenciar el interés que tienen los administrativos en implementar y hacer uso del sistema, sobre todo por la facilidad con el que lo manejaban y la generación de reportes, que viene a facilitar en gran manera su trabajo.(Anexos)

II.2.3.1.4 Dirigido a:

La capacitación estuvo dirigida a la Autoridades representante de sistemas del hospital Obrero, a los directores y secretarias como también a los profesores que quisieran asistir sin limitación de la participación de otras personas. Como bien se dijo la capacitación estuvo dirigida a profesionales pero no se limito la participación de otras personas.

II.2.3.2 Intervención

La intervención entonces es la ejecución de los programas de capacitación que abarca diferentes escenarios de acción, En el tiempo de intervención es preciso tener en cuenta también la disponibilidad de los participantes.

En el proceso de intervención o desarrollo de la capacitación se tomo en cuenta la disponibilidad de tiempo de los asistentes, por lo que se solicito permiso para la asistencia del personal administrativo.

III. Conclusiones y Recomendaciones

III.1 Conclusiones

Al finalizar el proyecto, Mejoramiento del Control de la Información de los Pacientes Hospitalizados en la Caja Nacional de Salud Regional Tarija.

Se cumplió con los objetivos propuestos al obtener información precisa y oportuna acerca de los pacientes, información importante dentro de la CNS, además se pudo observar la satisfacción al obtener la información de manera eficiente y eficaz que realmente permite un control de la información de los pacientes hospitalizados situación muy distinta al manejo manual que se realizo hasta ahora, que sí lleva un control pero algo tedisoso e impresiso.

La disponibilidad del personal al realizar la capacitación fue algo muy importante ya que permitió explicar el sistema en pleno, y aclarar dudas, a su vez se pudo observar la facilidad con que los usuarios manejaron el sistema, un factor negativo fue la falta de equipos necesarios para ejecutar el sistema.

A la conclusión del mismo se pudo evidenciar lo siguiente

- Los resultados obtenidos permitirán una buena toma de decisiones y de manera oportuna
- La información se mantendrá actualizada

Las autoridades al notar la facilidad del uso del sistema y además de evidenciar la rapidez de la obtención de información mostraron bastante interés en implementarlo en la siguiente gestión. Comprometiéndose en solicitar el hardware necesario a las autoridades pertinentes para un mejor uso del sistema.

En conclusión el sistema cumplió las expectativas, el desarrollo del proyecto para la administración fue satisfactorio por la automatización de la información.

Finalmente las tecnologías utilizadas para el desarrollo del sistema, fueron empleadas por ser las mas disponibles en el momento de iniciar la etapa de planificación, luego de uso, se constata que fueron suficientes para el sistema desarrollado.

III.2 Recomendaciones

Se recomienda realizar capacitaciones constantes acerca del uso del equipo de computacion, para optimizar el uso del sistema.

Se recomienda también lo siguiente requisitos para la implementación del proyecto

Requisitos de Plataforma hardware

- 2.4 GHz Microprocesador.
- 500 MB mínimo de espacio en disco duro para la instalación de programas.
- 512 MB RAM (1 GB RAM recomendado).
- CD-ROM drive (para la instalación desde el CD).

Requisitos de Plataforma software

- Sistema Operativo: Microsoft® Windows® 2000, XP Home, o XP Professional.
- Java en su versión 6 Update 2 (jdk-1.6.0_02).
- Postgres 8.4

Es necesario que un profesional informatico sea el encargado de realizar la conversión del manejo manual al nuevo sistema(conversión de datos manuales a datos digitales, cargar la base de datos y otros), pero se recomienda en un futuro, implementar un módulo de actualización, que permita que no solo el encargado del area de informática sea el único capaz de actualizarlo, sino darle lugar a un tercer usuario que lo pueda realizar sin complicaciones.

Y en caso de dudas al utilizar el sistema consulta el manual de usuario.

- ✓ Finalmente se recomienda a los docentes fortalecer a los estudiantes en el manejo de las herramientas CASE que muy importante en el anilisis y diseño del sistema y la posterior codificación.