INSTITUTO TECNOLOGICO
"CORDILLERA"

INSTITUTO TECNOLOGICO SUPERIOR "CORDILLERA"

CARRERA DE ANALISIS DE SISTEMAS

TÉCNICAS DE PROGRAMACIÓN TDD Y ATDD APLICADOS AL DESARROLLO DEL SISTEMA DE CONTROL DEL HISTORIAL CLÍNICO Y LA RESERVACIÓN DE TURNOS PARA LA CLÍNICA SAN JOAQUÍN OCCIDENTAL.

Proyecto previo a la obtención del Título de Tecnólogo Analista de Sistemas

Autor: Chanchay Guanoluisa Diego Iván

Tutor: Ing. Richard Mafla, MSc.

Quito, Noviembre 2013

Declaración de Autoría del estudiante

DECLARATORIA

Declaro que la investigación es absolutamente original, autentica, personal, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes. Las ideas, doctrinas resultados y conclusiones a los que he llegado son de mi absoluta responsabilidad.

Diego Iván Chanchay Guanoluisa

CC 172335042-5

CONTRATO DE CESIÓN SOBRE DERECHOS PROPIEDAD INTELECTUAL

Comparecen a la celebración del presente contrato de cesión y transferencia de derechos de propiedad intelectual, por una parte, el estudiante **Chanchay Guanoluisa Diego Iván**, por sus propios y personales derechos, a quien en lo posterior se le denominará el "CEDENTE"; y, por otra parte, el INSTITUTO SUPERIOR TECNOLÓGICO CORDILLERA, representado por su Rector el Ingeniero Ernesto Flores Córdova, a quien en lo posterior se lo denominará el "CESIONARIO". Los comparecientes son mayores de edad, domiciliados en esta ciudad de Quito Distrito Metropolitano, hábiles y capaces para contraer derechos y obligaciones, quienes acuerdan al tenor de las siguientes cláusulas:

PRIMERA: ANTECEDENTE.- a) El Cedente dentro del pensum de estudio en la carrera de análisis de sistemas que imparte el Instituto Superior Tecnológico Cordillera, y con el objeto de obtener el título de Tecnólogo en Análisis de Sistemas, el estudiante participa en el proyecto de grado denominado "**Técnicas de Programación TDD y ATDD Aplicados al Desarrollo del Sistema de Control del Historial Clínico y la Reservación de Turnos para la Clínica San Joaquín Occidental** ", el cual incluye la creación y desarrollo del programa de ordenador o software, para lo cual ha implementado los conocimientos adquiridos en su calidad de alumno. **b)** Por iniciativa y responsabilidad del Instituto Superior Tecnológico Cordillera se desarrolla la creación del programa de ordenador, motivo por el cual se regula de forma clara la cesión de los derechos de autor que genera la obra literaria y que es producto del proyecto de grado, el mismo que culminado es de plena aplicación técnica, administrativa y de reproducción.

SEGUNDA: CESIÓN Y TRANSFERENCIA.- Con el antecedente indicado, el Cedente libre y voluntariamente cede y transfiere de manera perpetua y gratuita todos los derechos patrimoniales del programa de ordenador descrito en la cláusula anterior a favor del Cesionario, sin reservarse para sí ningún privilegio especial

(código fuente, código objeto, diagramas de flujo, planos, manuales de uso, etc.). El Cesionario podrá explotar el programa de ordenador por cualquier medio o procedimiento tal cual lo establece el Artículo 20 de la Ley de Propiedad Intelectual, esto es, realizar, autorizar o prohibir, entre otros: a) La reproducción del programa de ordenador por cualquier forma o procedimiento; b) La comunicación pública del software; c) La distribución pública de ejemplares o copias, la comercialización, arrendamiento o alquiler del programa de ordenador; d) Cualquier transformación o modificación del programa de ordenador; e) La protección y registro en el IEPI el programa de ordenador a nombre del Cesionario; f) Ejercer la protección jurídica del programa de ordenador; g) Los demás derechos establecidos en la Ley de Propiedad Intelectual y otros cuerpos legales que normen sobre la cesión de derechos de autor y derechos patrimoniales.

TERCERA: OBLIGACIÓN DEL CEDENTE.- El cedente no podrá transferir a ningún tercero los derechos que conforman la estructura, secuencia y organización del programa de ordenador que es objeto del presente contrato, como tampoco emplearlo o utilizarlo a título personal, ya que siempre se deberá guardar la exclusividad del programa de ordenador a favor del Cesionario.

CUARTA: CUANTIA.- La cesión objeto del presente contrato, se realiza a título gratuito y por ende el Cesionario ni sus administradores deben cancelar valor alguno o regalías por este contrato y por los derechos que se derivan del mismo.

QUINTA: PLAZO.- La vigencia del presente contrato es indefinida.

SEXTA: DOMICILIO, JURISDICCIÓN Y COMPETENCIA.- Las partes fijan como su domicilio la ciudad de Quito. Toda controversia o diferencia derivada de éste, será resuelta directamente entre las partes y, si esto no fuere factible, se solicitará la asistencia de un Mediador del Centro de Arbitraje y Mediación de la Cámara de Comercio de Quito. En el evento que el conflicto no fuere resuelto mediante este procedimiento, en el plazo de diez días calendario desde su inicio,

pudiendo prorrogarse por mutuo acuerdo este plazo, las partes someterán sus controversias a la resolución de un árbitro, que se sujetará a lo dispuesto en la Ley de Arbitraje y Mediación, al Reglamento del Centro de Arbitraje y Mediación de la Cámara de comercio de Quito, y a las siguientes normas: a) El árbitro será seleccionado conforme a lo establecido en la Ley de Arbitraje y Mediación; b) Las partes renuncian a la jurisdicción ordinaria, se obligan a acatar el laudo arbitral y se comprometen a no interponer ningún tipo de recurso en contra del laudo arbitral; c) Para la ejecución de medidas cautelares, el árbitro está facultado para solicitar el auxilio de los funcionarios públicos, judiciales, policiales y administrativos, sin que sea necesario recurrir a juez ordinario alguno; d) El procedimiento será confidencial y en derecho; e) El lugar de arbitraje serán las instalaciones del centro de arbitraje y mediación de la Cámara de Comercio de Quito; f) El idioma del arbitraje será el español; y, g) La reconvención, caso de haberla, seguirá los mismos procedimientos antes indicados para el juicio principal.

SÉPTIMA: ACEPTACIÓN.- Las partes contratantes aceptan el contenido del presente contrato, por ser hecho en seguridad de sus respectivos intereses.

En aceptación firman a los 14 días del mes de Noviembre del dos mil doce.

f)_____ f)_____

C.C. Nº Instituto Superior Tecnológico Cordillera

CEDENTE **CESIONARIO**

Índice General

Contenido

Página

Índice de Tablas

Contenido **Páginas**

Índice de Figuras

Contenido Páginas

Índice de Anexos

Contenido Página

Resumen ejecutivo

El sistema informático para médicos fue realizado con el proposito de brindar a los pacientes una mejor atencion y un mayor control en las enfermedades y la atención recibida por parte de los médicos.

A su vez fue realizado para poder aplicar las pruebas de unidad Test Driven Developement (TDD) que ayudan al desarrollador a crear un codigo de calidad y que se pueda ser de utilidad no solo para un determinado proceso.

Para su desarrollo se utilizo el motor de base de datos PosgreSQL, que consta de datos para el manejo de los esquemas de la clinica y de la administración del sistema.

La implementación de código se lo realizo en el ambiente Jboss developer Studio, código fuente JAVA utilizando un servidor de aplicaciones Jboss-as-1.0.0.final y el repositorio Maven que es el que sigue el patrón de convención sobre configuración. Esto significa que debemos situar nuestras clases, tests o recursos en un lugar en concreto, para que luego Maven sea capaz de tratarlos correctamente; Primefaces se utiliza para hacer el diseño de las páginas Web, el código JAVA está dividido entre los Controller, DataManger, DAO, Servicios y Model. Que son las principales carpetas que alojan las clases e interfaces.

Los test forman parte de otra capa que el sistema en si no los utilizará pero facilita la realización de pruebas de unidad y que el código que se está realizando sea el correcto y realice una buena funcionalidad.

Abstract

The computer system for doctors was made with the purpose of providing patients with better care and better diseases control and care provided by doctors.

A turn was made to apply unit tests Test Driven Development (TDD) that help the developer to create a quality code that can be useful not only for a particular process.

For its development engine use the PostgreSQL database, this consists of data for managing the clinic schemes and administration system.

The code implementation we perform in the environment Jboss developer studio, JAVA source code using an application server Jboss-as-1.0.0.final and Maven repository is following the pattern of convention over configuration. This means that we must place our classes, tests or resources in a particular location, then have Maven is able to treat them right, PrimeFaces is used to design Web pages, Java code is divided between the Controller, Data Manger, DAO, Services and Model. What are the main folders that host classes and interfaces.

The tests are part of another layer that the system itself will not use but facilitates unit testing and code that is being done is correct and make a good functionality.

Introducción

EL presente fue elaborado con el propósito de realizar el proyecto final de grado de especialización en la carrera de Análisis de Sistemas “Técnicas de programación TDD y ATDD aplicados a un sistema informático para médicos de la clínica San Joaquín Occidental” en donde detallamos los diferentes procesos, técnicas y herramientas utilizadas para poder realizar la implementación del mismo.

Se llevaron a cabo investigaciones sobre cómo gestionar una clínica mediante una aplicación, así como, los diferentes procesos que conllevan el mismo ya sea las asignaciones y controles.

Además, se realizó la investigación sobre TDD (Test Driven Developement) “Desarrollo Dirigido por Test”, que es la técnica de diseño e implementación del software incluida dentro de la metodología XP.

Capítulo I: Antecedentes

1.01 Contexto

La clínica San Joaquín Occidental fue inaugurada hace más de un año y desde entonces su atención ha sido ininterrumpida en el poco tiempo de existencia ha tenido una buena acogida la cual ha hecho que se vayan implementando varios cambios en su infraestructura y en el servicio que brinda, esto ha generado algunos problemas y uno de ellos es que no se cuenta con infraestructura tecnológica la cual permita tener un control sobre los pacientes, sus síntomas y enfermedades que presente, a su generar un historial clínico que ayude a los médicos en su atención.

Los antecedentes médicos de los pacientes no existen y solo se tiene referencias detallados por los mismos pacientes los cuales no son claros y no cuentan con un respaldo de los tratamientos, medicamentos y análisis realizados con anterioridad.

El poco control sobre los pacientes que están recibiendo atención, los médicos no tienen detalles de sus atenciones anteriores, si las indicaciones se están siguiendo de acuerdo a las dadas por los mismos.

No existe un registro de los recursos asignados a los pacientes que se encuentran internados dentro de la clínica esto puede provocar pérdidas de los mismos, deterioro, una asignación inútil, si los recursos dados se encuentran en buen estado y que se hayan limpiado correctamente.

1.02 Justificación

Con la finalidad de brindar una mejor atención y tener un buen control de los médicos sobre las enfermedades y medicamentos dados a los pacientes se implementará un sistema de control médico el cual tiene previsto ordenar la atención

de cada paciente desde su ingreso hasta el final de su tratamiento o su atención, en la cual se tiene que tomar en cuenta los síntomas que presenten y realizar un diagnóstico inicial para que pueda ser atendido acorde a la gravedad de su enfermedad. Para luego llevar un control minucioso sobre el tratamiento a seguir para su recuperación.

Por esta razón la implementación de este sistema informático, permitirá mejorar las actividades que se vinculan con dicho proceso, representando una gran evolución en el servicio, ya que serían sustituidas las hojas de registro de los pacientes por un sistema encargado del almacenamiento y búsqueda de toda información vinculada en el servicio médico.

Logrando disminuir la carga que representa para los médicos y asistentes, todo el mecanismo que lleva a cabo desde el ingreso de formularios hasta que se logre una historia clínica, alcanzando el mejor provecho de las herramientas tecnológicas con las que cuentan.

Por lo anteriormente expuesto se justifica total y plenamente la realización de este proyecto, ya que el mismo deberá ser de gran beneficio para una buena atención, permitiéndoles un mayor rendimiento en el campo competitivo y laboral. Y con el fin de poder desarrollar un buen sistema donde se pueda emplear todos los conocimientos adquiridos a lo largo de la carrera poniendo en uso lo teórico, práctico y la investigación realizada sobre la Metodología de Programación TDD en el Desarrollo de Sistemas.

1.03 Matriz de Análisis de Fuerzas (Matriz T)

EL presente cuadro nos permite analizar la situación empeorada actual y mejorada incluyendo fuerzas impulsadoras y bloqueadoras a las cuales se les calificará de 1 a 4 donde se demostrará la factibilidad en la implementación del sistema.

Tabla 1.

Matriz T

Análisis de la Fuerza T					
Situación Empeorada	**Situación Actual**				**Situación Mejorada**
El mal registro de los datos del cliente al momento de realizar una consulta médica y asignación de los recursos clínicos	Falta de Control de los procesos de Historial Clínico y atención al cliente.				Un mejor control de los diferentes servicios que se ofrecen dentro de la clínica San Joaquín Occidental
Fuerza Impulsadora	**I**	**PC**	**I**	**PC**	**Fuerzas bloqueadoras**
Registro de Pacientes.	3	3	1	2	Los registros son llevados en hojas y realizados a mano
Distribución de recursos de acuerdo a las necesidades de los pacientes	3	4	1	3	La mala administración de los recursos de la clínica.
Asignación de Turnos acorde al diagnóstico del paciente	3	4	1	2	La mala asignación de los turnos para los pacientes
Registro de Hospitalización.	3	3	1	4	El control de las Hospitalizaciones es deficiente
Registro de labores de los empleados	2	3	1	2	No existe un control sobre el horario de trabajo da cada uno de

					los empleados.
Control de tratamientos	3	4	1	3	No existe control de los tratamientos a seguir.

Nota: Significado de iniciales I= Intensidad PC=Potencial de cambio

1.03.01 Análisis de la matriz T

La matriz T nos ayuda a definir cómo se encuentra la situación actual de la empresa, muestra los problemas que viene a ser la situación actual: "Falta de Control de los procesos de Historial Clínico y atención al cliente" ya que por los problemas que se presentan en la clínica como son: Los registros son llevados en hojas y realizados a mano, La mala administración de los recursos de la clínica y La mala administración de los turnos para los pacientes, no existe una mejor atención hacia el paciente. Al llegar a implementar el sistema se tendrá como logros: Control de los registro en forma automática, Una mejor asignación de los recursos clínicos y Asignación de turnos acorde a sus necesidades, se les asignó un valor a las fuerzas impulsadoras como a las bloqueadoras para que puedan ser medidos de acuerdo a su valor dentro del problema central la escala va entre el 1 y 5 y se clasifica entre la intensidad y el Potencial Cambio.

Capítulo II: Análisis De Involucrados

2.01 Mapeo de Involucrados

Para la ejecución del proyecto se realizará el mapeo y tipo de involucrados de quienes intervienen, las organizaciones opositores, neutros, y favorecedores que intervienen en el proyecto, se los describe a continuación.

- Clínica
- Usuarios
- Comunidad

Lista de Involucrados

Se enlista los diferentes usuarios que se verán involucrados directa e indirectamente con la ejecución del sistema y los puntos de vista de acuerdo a sus intereses y sus necesidades dentro de la clínica.

Tabla 2.

Mapeo de Involucrados

GRUPOS	INTERÉS	PROBLEMAS PERCIBIDOS	RECURSOS Y MANDATOS
Dueños de la clínica	Apoyo a la ejecución	No hay problemas	Los propietarios de la clínica
Doctores	Apoyo a la ejecución	No hay problemas	Los encargados de generar los historiales clínicos.
Enfermeras	Apoyo a la ejecución.	No hay problemas	Encargadas de cuidar de los pacientes de acuerdo a las indicaciones de los doctor.
Pacientes	No hay opinión	No hay problemas	Clientes del negocio
Recepcionistas	Apoyo a la ejecución.	No hay problemas	Son las personas que asignan los turnos a los pacientes.
Usuarios	Que se ejecute	Apoyan a la implementación del sistema.	Ingresan datos en el sistema

2.01.01 Análisis De Involucrados

Se procedió a realizar el análisis de los involucrados en los diferentes procesos y registros que se ejecutan dentro de una clínica, en los cuales participaron los dueños, médicos, recepcionista y demás empleados, así como los pacientes, los cuales son los principales involucrados, estos favorecen en la implementación del sistema ya que, al realizar el control de la clínica mediante un software trae varios beneficios hacia los mismos.

Los registros de los pacientes tendrán un buen respaldo y serán consistentes, además de estar a la mano de los médicos que necesiten realizar una consulta de los antecedentes eso es el punto a favor por el cual los médicos están de acuerdo con el proyecto.

El beneficio que se le otorga a los dueños del negocio, es fundamentalmente el control sobre los recursos médicos así como de las instalaciones esto les permitirá ofrecer un mejor servicio y así obtener mejor réditos económicos, con lo cual apoyan al proyecto.

Las personas encargadas de receptar los datos de los pacientes así como de asignar el turno se verán beneficiados en el caso de que un paciente ya se encuentre registrado en el sistema con lo que la asignación del turno será rápida y oportuna, al agilizar su trabajo también muestran su interés en que el proyecto se ejecute.

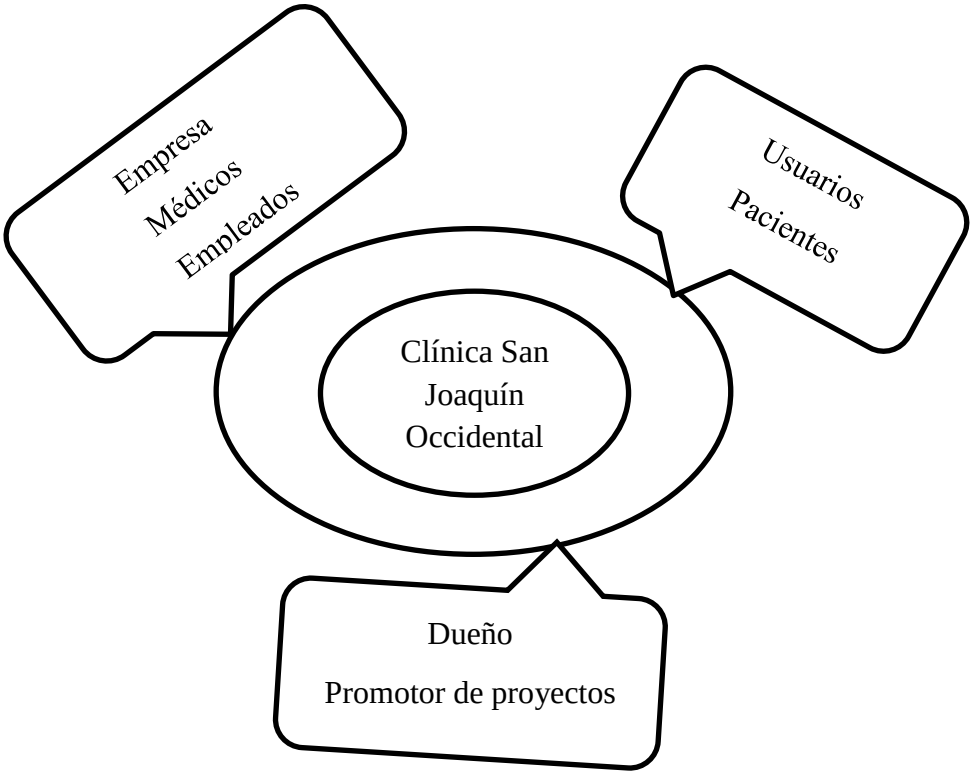


Figura 1. Se agrupa a los diferentes usuarios sean personas, instituciones o grupos sociales afectados por el problema o giran en torno al mismo. Caso recepción, registro de historial clínico y asignación de recursos médicos

2.01.02 Mapa de Involucrados

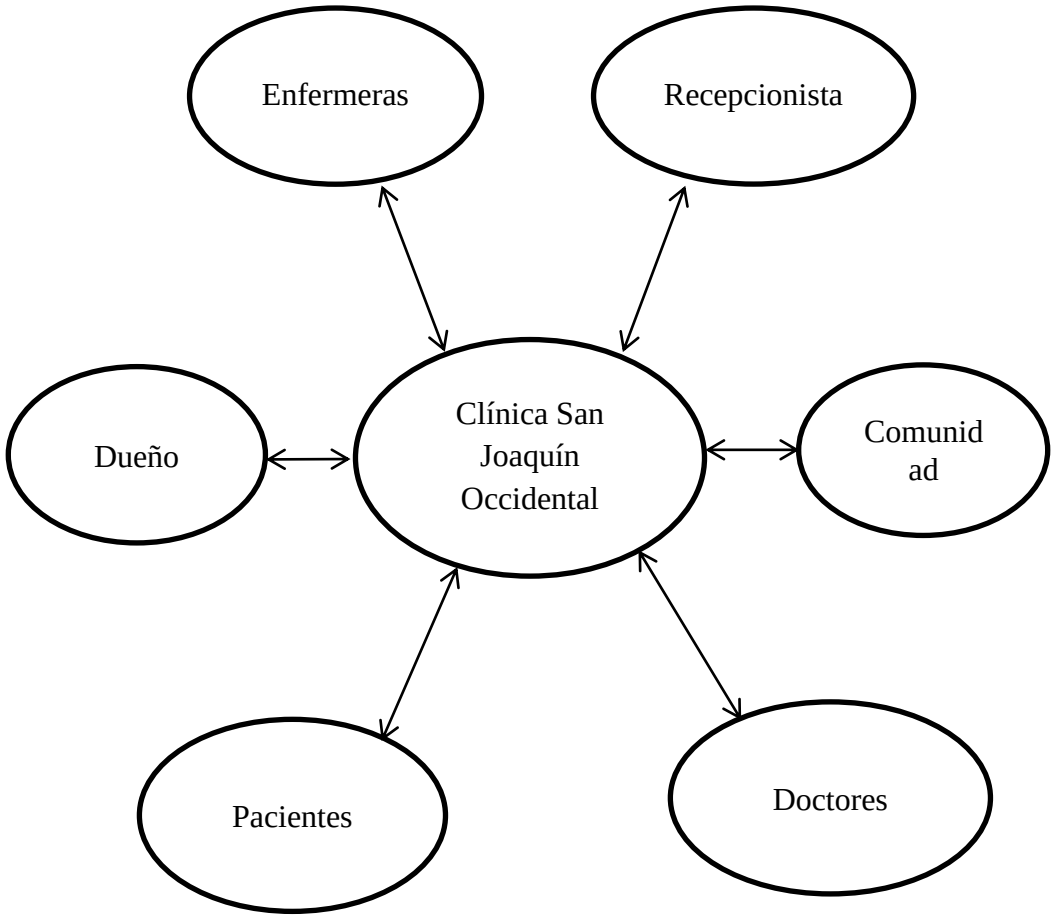


Figura 2. Se muestra gráficamente los involucrados del problema en forma global, todos los que se ven afectados directa e indirectamente

2.01.03 Mapa de análisis de involucrados.

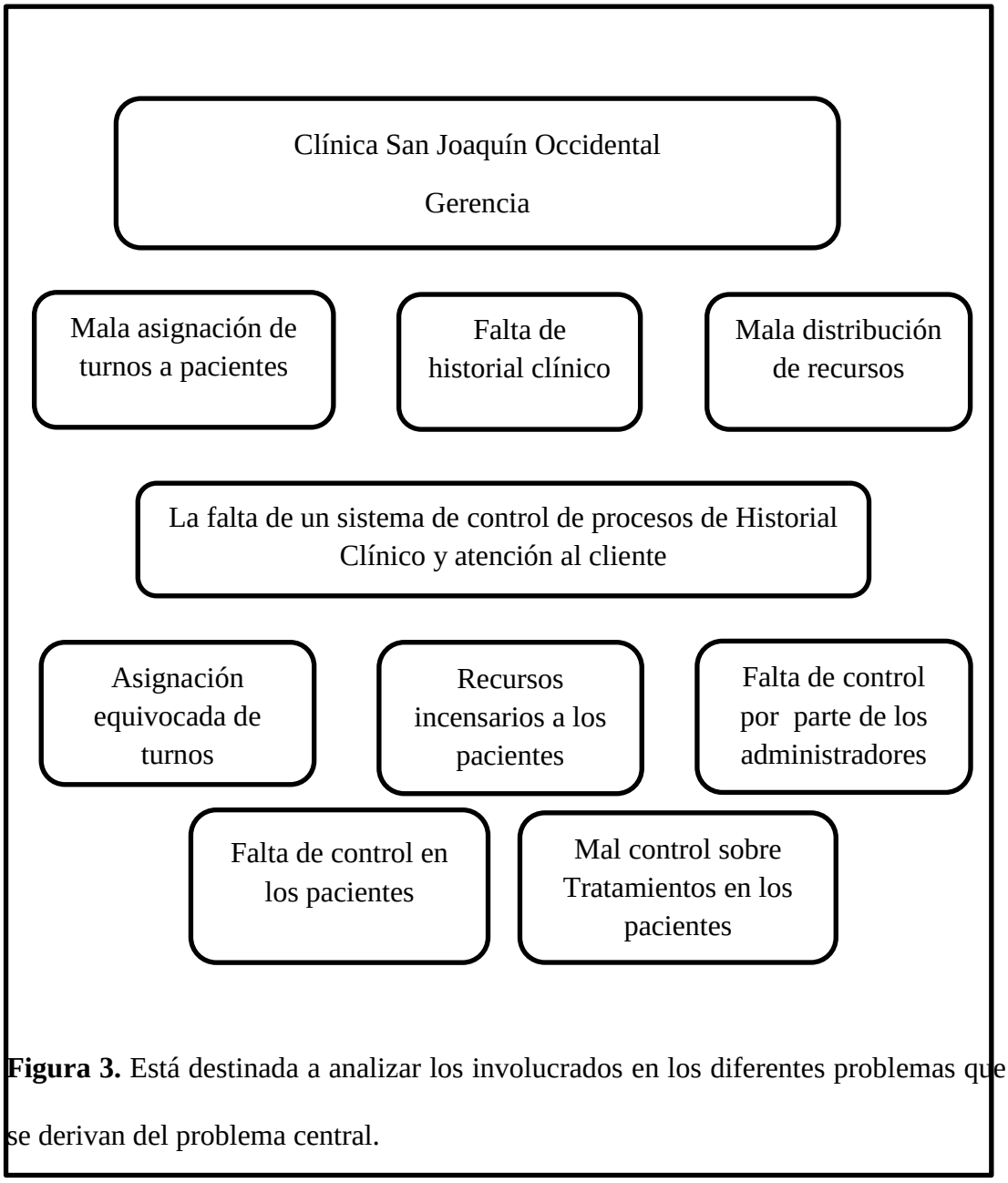


Figura 3. Está destinada a analizar los involucrados en los diferentes problemas que se derivan del problema central.

2.02 Matriz de Análisis de Involucrados

En esta matriz medimos el interés y el apoyo que se da al proyecto de parte de todos los involucrados y se los da una puntuación de 1 a 5, este se lo multiplica entre la Expectativa y fuerza para el resultante.

Tabla 3.

Matriz de Análisis de Involucrados

Expectativa	*	Fuerza	=	Resultante
5	*	5	=	25
5	*	5	=	25
4	*	4	=	16
4	*	4	=	16
3	*	4	=	12
1	*	2	=	2

2.02.01 Matriz de valoración caso Clínica San Joaquín Occidental

En la matriz damos a conocer la expectativa por cada involucrado definiendo cuales están a favor y en contra de la implementación del Sistema Clínico.

Tabla 4.

Matriz de Valores

Involucrados	Expectativa		Fuerza		Resultante	Posición Potencial
Ejecutador del proyecto	5	*	5	=	25	Favorecedor
Dueño	5	*	5	=	25	Favorecedor
Medico	4	*	4	=	16	Favorecedor
Recpcionista	4	*	4	=	16	Favorecedor
Enfermeras	4	*	4	=	16	Favorecedor
Pacientes	3	*	2	=	6	Indiferentes

Capítulo III: Problemas Y Objetivos

3.01 Árbol de Problemas

El árbol de problemas nos permite analizar los efectos y causas de los diferentes problemas y dificultades que existen en los procesos que se desarrollan dentro del control y registro que se realizan en la clínica.

Los cuales tienen los siguientes puntos:

- Problema central
- Causas directas del problema central
- Causas indirectas
- Causas estructurales

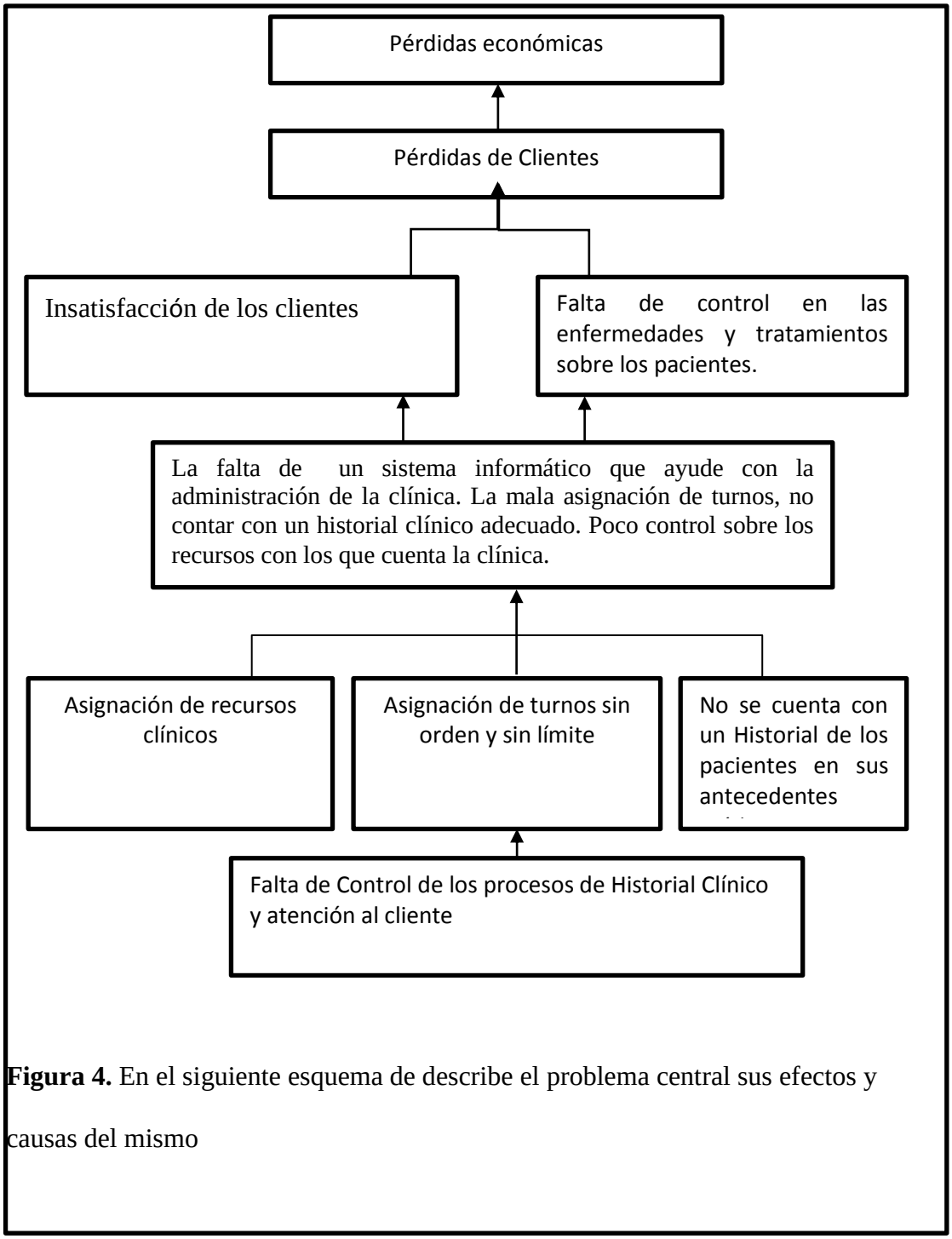


Figura 4. En el siguiente esquema de describe el problema central sus efectos y causas del mismo

3.01.01 Análisis del Árbol de Problemas

El árbol de problemas ha permitido identificar los mayores problemas presentados por la falta de un sistema informático que ayude con la administración de la clínica, así como la mala asignación de turnos, el poco control sobre los recursos con los que cuenta la clínica, y el no contar con un historial clínico adecuado de los pacientes que recibieron atención con anterioridad, ha generado una insatisfacción en los pacientes porque no tienen una atención oportuna, además que no se cuenta con un control sobre las principales enfermedades y tratamientos sobre los mismos, esto ha generado que se pierda algunos clientes, por ende son perdidas económicas hacia los propietarios.

3.02 Árbol de Objetivos

El árbol de objetivos nos ayuda a realizar el análisis de los componentes que nos ayudara a dar solución a los diferentes procesos que se llevan a cabo dentro de la clínica, logrando así poder controlar a los pacientes, asignación de turnos recursos.

Se van a tratar los siguientes puntos:

- Objetivos resultantes
- Objetivos componentes
- Objetivos finalidad
- Objetivos propósitos

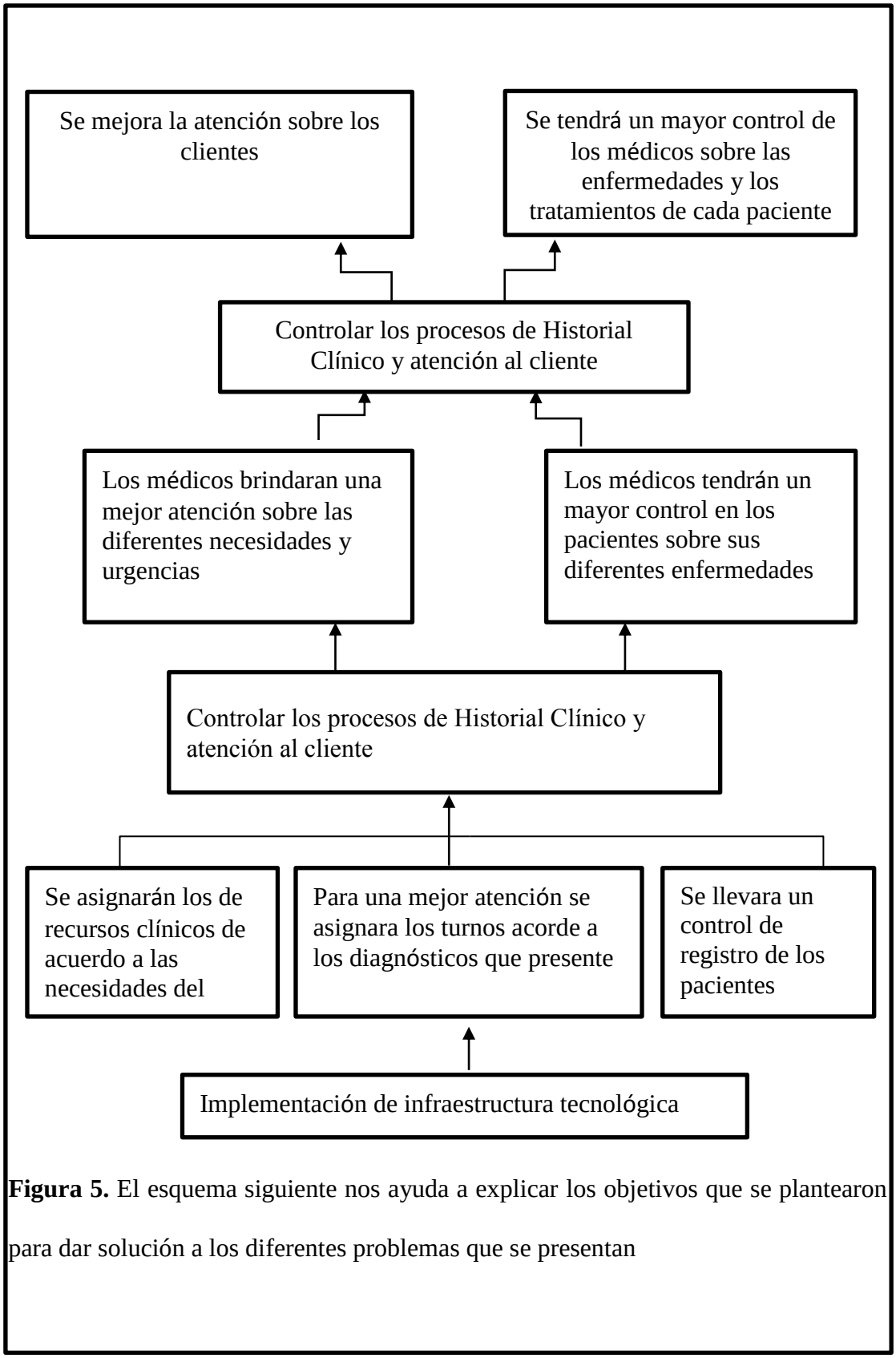


Figura 5. El esquema siguiente nos ayuda a explicar los objetivos que se plantearon para dar solución a los diferentes problemas que se presentan

Capítulo IV: Análisis De Alternativas

4.01 Matriz de análisis de Alternativas

Se establece un análisis de la disponibilidad de los recursos humanos financieros técnicos y materiales necesarios para llevar a cabo la ejecución del proyecto.

Tabla 5.

Matriz de Análisis de Alternativas

Matriz de Análisis de Alternativas

Objetivos	Impacto sobre el propósito	Factibilidad Técnica	Factibilidad Financiera	Factibilidad Social	Factibilidad Política	Total	Categ
Control de los médicos sobre las enfermedades	4	4	3	4	2	17	Alt
Asignación del turno a pacientes	2	3	3	2	2	12	Me. Alt.
Control materiales clínicos.	2	3	4	3	2	14	Me. Alt.
Mayor evolución en el tratamiento al paciente.	3	2	4	3	2	14	Me. Alt.
Total	**11**	**12**	**14**	**12**	**8**	**57**	

Nota: Alt.= Alto Me. Alt.= Media Alta

4.01.01 Análisis De Matriz de Análisis de Alternativas

El presente nos ayuda a medir la solución planteada para poder llevar un control sobre los pacientes en todas las atenciones que recibe en la clínica, los diferentes recetas y exámenes serán registrados para tener en claro su estado de salud, controlar las asignaciones de los recursos que tiene el centro médico. Así identificar el método para realizar la entrega de un turno a los pacientes para que puedan ser atendidos con un total orden.

Se ha generado una calificación a los diferentes objetivos trazados con el fin de calificar la factibilidad de la ejecución del proyecto.

4.02 Matriz de Análisis de Impacto de los Objetivos

En la matriz de análisis de impacto de los objetivos se examina los diferentes puntos que dan a conocer si los objetivos planteados son viables, cuerpo, relevancia, sostenibilidad, género y realizables en la aplicación del proyecto.

Tabla 6.

Matriz de Análisis de Impactos de los Objetivos

	Factibilidad de Lograrse *(Alta.Media-Baja) (4 - 2 - 1)*	**Impacto en genero** *(Alta.Media-Baja) (4 - 2 - 1)*	**Impacto ambiental** *(Alta.Media-Baja) (4 - 2 - 1)*	**Relevancia** *(Alta.Media-Baja) (4 - 2 - 1)*	**Sostenibilidad** *(Alta.Media-Baja) (4 - 2 - 1)*	**Tot**
CBJETIVCS	-Beneficio son mayores que la inversión -Cuenta con apoyo de la empresa -existes tecnología adecuada para la ejecución -se cuenta con políticas que favorece el desempeño de la clínica -es aceptable, conveniente por el fácil manejo que se presenta ante los usuarios	- incrementa participación de hombre y mujer en el manejo del sistema -incrementa un mayor uso de tecnología en hombres y mujeres -capacitación para el registro y control de la clínica	-control de los diferentes recursos utilizados y asignados en la clínica	-responde a las necesidades de los beneficiados -el uso del sistema agilizara la atención a los pacientes -la entrega de turnos será correctos	-fortaleza la participación de cliente -empresa -la población está de acuerdo con la implementación -se puede mejorar a futuro el sistema de control clínico	76
	20	16	8	12	20	

4.02.01 Análisis de Matriz de Análisis de Impacto de los Objetivos

El análisis realizado a los objetivos a tenido un buen resultado ya que se pueden tener varios beneficios los cuales ayudan en la parte económica, en la participación por parte del personal masculino y femenino en la implementación de la tecnología en el control de la clínica, los controles sobre los recurso e instalaciones, que el sistema responda a las necesidades que se presentan al realizar el control sobre los pacientes que sus antecedentes médicos se encuentren claros y con respaldos y así se logren cumplir con los objetivos que hicieron que sea necesario el realizar el sistema.

El fácil manejo ayuda a que se pueda utilizar al 100% en sistema y a su vez que se pueda realizar una actualización al mismo y que ayude a mejorar la atención cada día.

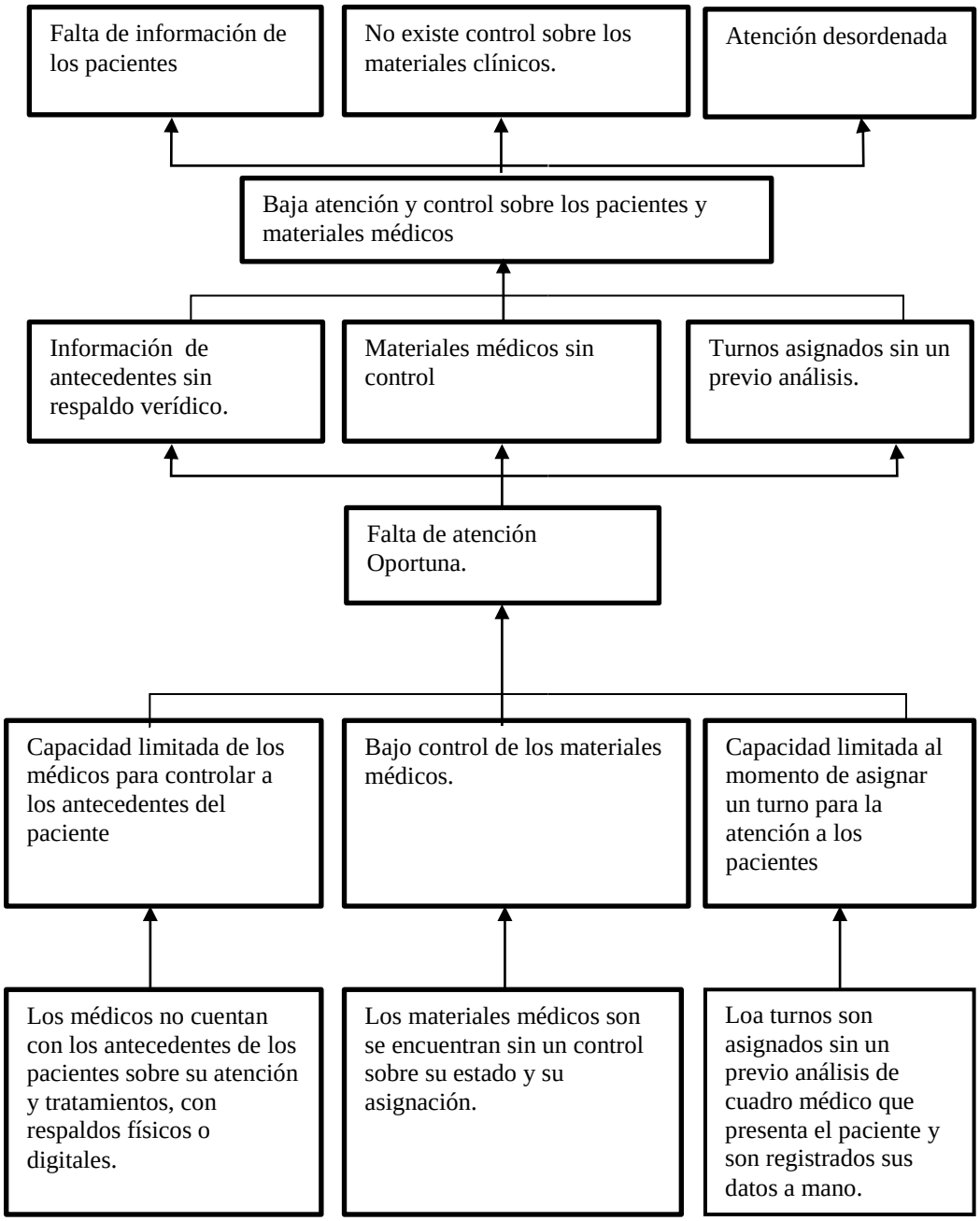


Figura 6. El presente analiza las diferentes finalidades, propósitos, componentes y objetivos de los resultados que se verán en el momento de ejecutar el sistema.

4.03 Diagrama de Estrategias

Es tomado como referencia el árbol de problemas y objetivos, sus relevancias para la implementación del proyecto.

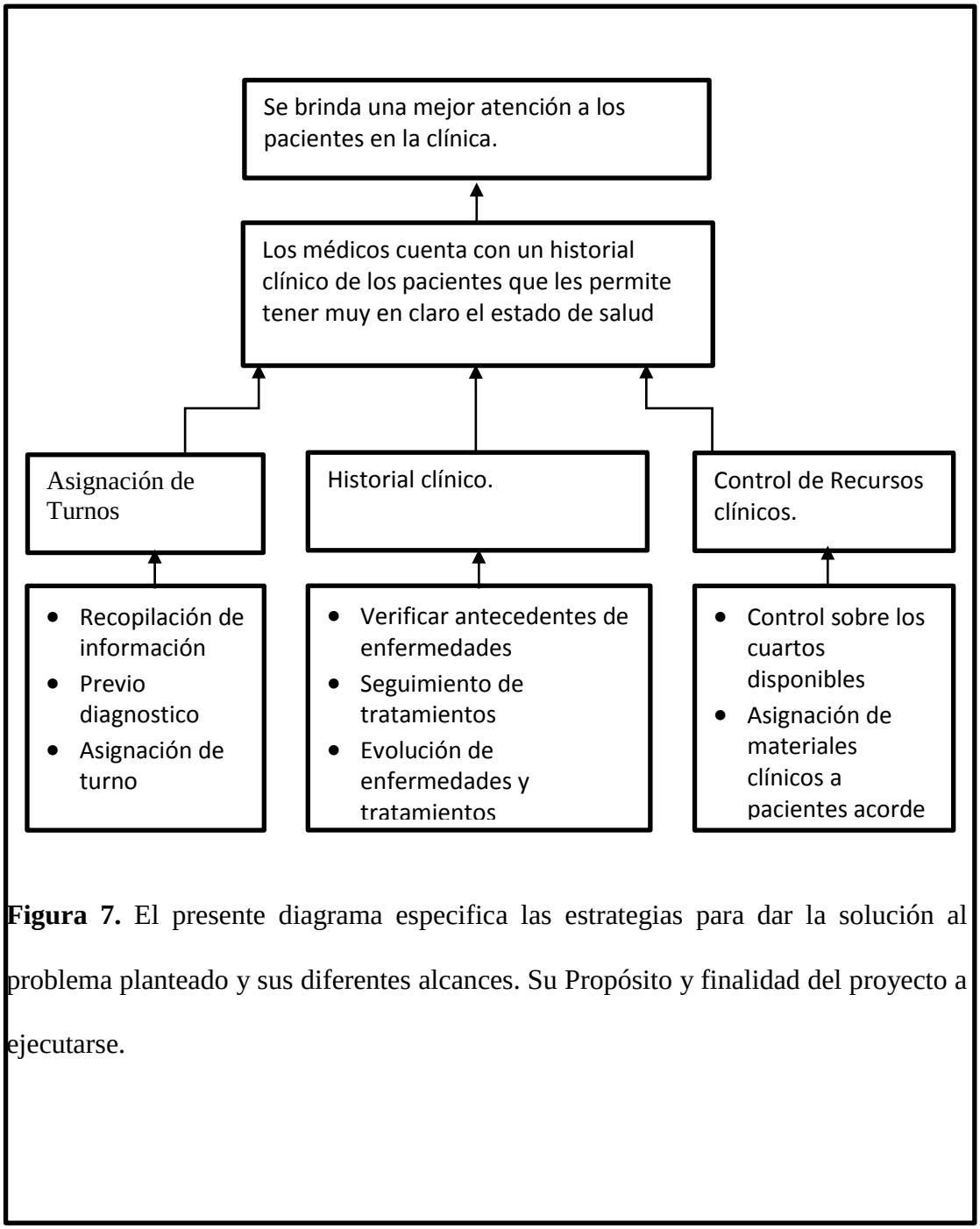


Figura 7. El presente diagrama especifica las estrategias para dar la solución al problema planteado y sus diferentes alcances. Su Propósito y finalidad del proyecto a ejecutarse.

4.04 Matriz de Marco Lógico

Permite resumir lo que se desea lograr en el proyecto, Propósitos y componentes que se alcanzaran del proyecto, factores externos necesarios, como obtener la información necesaria para una evolución del proyecto en ejecución.

Figura 7.

Matriz de Marco Lógico

RESUMEN NARRATIVO	Indicadores	Medios de Verificación	Supuestos
FINALIDAD DEL PROYECTO **Ayudar a los médicos a controlar la asignación de turno y materiales clínicos, Historial Clínico de los pacientes**	Los pacientes son atendidos acorde a sus necesidades o síntomas. Control minucioso sobre los pacientes.	Resultados de los pacientes que fueron atendido. El registro de las diferentes atenciones recibidas por los pacientes.	Detalles de atenciones médicas
PROPOSITO DEL PROYECTO **Mejor atención, oportuna y control de procesos.**	Menos quejas de los pacientes. Los pacientes tienen una mejor recuperación.	Estadísticas sobre los pacientes. Estadística en la evolución y recuperación de los pacientes.	
COMPONETES DEL PROYECTO **Asignación de turnos** **Historial Clínico** **Hospitalización** **Control de los materiales clínicos**	Turnos asignados con un previo análisis del paciente. Los antecedentes de enfermedades y tratamientos en orden. Materiales clínicos organizados y en un buen estado.	Estadística sobre los turnos asignados. Estadísticas sobre los pacientes. Datos actualizados de los materiales clínicos.	Registros se mantiene guardados y disponibles.
ACTIVIDADES DEL PROYECTO **Levantamiento de requerimientos** **Diseño de casos de uso** **Desarrollo del sistema** **Pruebas sobre el sistema** **Ejecución del sistema en producción**	Casos de uso, Diagramas de Secuencia, Diagramas de Iteración, Diagramas de Colaboración	Casos de uso y Diagramas.	

4.04.01 Análisis Matriz de Marco Lógico

La finalidad del proyecto que consiste en controlar la asignación de turnos, historial clínico, una hospitalización, y la distribución de materiales clínicos, tiene el propósito de mejorar la atención y que esta sea oportuna, a cada literal se ple planteo un indicador de cómo deben funcionar los procesos y mediante una verificación se establece que factibilidad tiene el proyecto en su ejecución, además se estableció unos supuestos que confirman lo expuesto.

Mediante diferentes actividades se procederá a poner en marcha el proyecto y este ira tomando forma de acuerdo a la realización de las mismas.

Los indicadores no son más que el detalle cómo se realizarán los procesos ya sea en la finalidad o en el propósito y los medios de verificación son poniendo en práctica los objetivos planteados con anterioridad,

Capítulo V: Propuesta

5.01 Justificación

En el presente capítulo se procederá a explicar la justificación del software a implementar los diferentes recursos utilizados para el desarrollo del mismo, los diferentes procesos mediante diagramas de Casos de Uso, Diagramas de Componentes, Diagramas de Secuencia, Diagramas de Colaboración y Diagramas de Clase.

Además se especificará los diferentes estándares de Programación y Base de Datos, así como la estructura del sistema, sus diferentes capas y clases.

La última parte del capítulo está destinada a documentar las diferentes pruebas que se realizan sobre el software en diferentes cuadros muy bien detallados.

Ubuntu

Ubuntu es un sistema operativo basado en Debian y que se distribuye como software libre y gratuito, el cual incluye su propio entorno de escritorio denominado Unity. Su nombre proviene de la ética de Ubuntu, en la que se habla de la existencia de uno mismo como cooperación de los demás.

Está orientado al usuario novel y promedio, con un fuerte enfoque en la facilidad de uso y en mejorar la experiencia de usuario. Está compuesto de múltiple software normalmente distribuido bajo una licencia libre o de código abierto. Estadísticas web sugieren que la cuota de mercado de Ubuntu dentro de las "distribuciones Linux" es, aproximadamente, del 49%, y con una tendencia a aumentar como servidor web. Y un importante incremento activo de 20 millones de usuarios para fines del 2011.

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

JBoss Developer Studio

Entorno open source de desarrollo integrado y basado en Eclipse, que combina las herramientas con el tiempo de ejecución.

JBoss Developer Studio ofrece una solución sencilla y potente para los desarrolladores, utilizando el middleware de JBoss. La solución proporciona un entorno de ciclo de vida estable y completo para el desarrollo y despliegue de aplicaciones.

JBoss es un servidor de aplicaciones Java EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible la máquina virtual de Java. JBoss Inc., empresa fundada por Marc Fleury y que desarrolló inicialmente JBoss, fue adquirida por Red Hat en abril del 2006. El proyecto se nutre de una red mundial de colaboradores. JBoss implementa todo el paquete de servicios de J2EE.

Las características destacadas de JBoss incluyen:

- Producto de licencia de código abierto sin coste adicional.

- Cumple los estándares.
- Confiable a nivel de empresa
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente
- Servicios del middleware para cualquier objeto de Java.
- Soporte completo para JMX.

PrimeFaces

PrimeFaces es un componente para JavaServer Faces (JSF) de código abierto que cuenta con un conjunto de componentes enriquecidos que facilitan la creación de las aplicaciones web. PrimeFaces está bajo la licencia de Apache License V2. Una de las ventajas de utilizar PrimeFaces, es que permite la integración con otros componentes como por ejemplo RichFaces.

Maven

Maven es una herramienta de software para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002. Es similar en funcionalidad a Apache Ant (y en menor medida a PEAR de PHP y CPAN de Perl), pero tiene un modelo de configuración de construcción más simple, basado en un formato XML. Estuvo integrado inicialmente dentro del proyecto Jakarta pero ahora ya es un proyecto de nivel superior de la Apache Software Foundation

Maven utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar

ciertas tareas claramente definidas, como la compilación del código y su empaquetado

JUnit

Es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java

JUnit es un conjunto de clases (framework) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

JUnit es también un medio de controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

El propio framework incluye formas de ver los resultados (runners) que pueden ser en modo texto, gráfico (AWT o Swing) o como tarea en Ant.

En la actualidad las herramientas de desarrollo como NetBeans y Eclipse cuentan con plug-ins que permiten que la generación de las plantillas necesarias para la creación de las pruebas de una clase Java se realice de manera automática,

facilitando al programador enfocarse en la prueba y el resultado esperado, y dejando a la herramienta la creación de las clases que permiten coordinar las pruebas.

El Desarrollo Dirigido por Tests (Test Driven Development), al cual se refiere como TDD, es una técnica de diseño e implementación de software incluida dentro de la metodología XP.. TDD es una técnica para diseñar software que se centra en tres pilares fundamentales:

- La implementación de las funciones justas que el cliente necesita y no más.
- La minimización del número de defectos que llegan al software en fase de producción.
- La producción de software modular, altamente reutilizable y preparado para el cambio.

Fuente: Carlos Blé Jurado y colaboradores (2010). Prólogo de José Manuel Beas Diseño Ágil con TDD, 48

Alternativas

La aplicación como tal se puede levantar y desarrollar en un ambiente Windows utilizando los mismos aplicativos que varían en la configuración de las variables JAVA_HOME y M2_HOME que donde se hace referencia de la instalación del JAVA y MAVEN.

En el caso de no obtener Jboss Developer Studio, se puede utilizar Eclipse como entorno de desarrollo.

5.02 Análisis y Diseño

En el presente ítem se detallaran el caso de uso, diagramas de secuencia, diagramas de secuencia, iteración y colaboración, Modelo Lógico y Físico.

5.02.01 Diagrama de Caso De Uso

En este literal se establece los casos de uso que comprenden el funcionamiento de registros que realizan los médicos sobre los pacientes y control de la clínica de sus instalaciones y materiales, con la que se cuenta actualmente y a los cuales se les va a automatizar dentro del sistema informático para médicos.

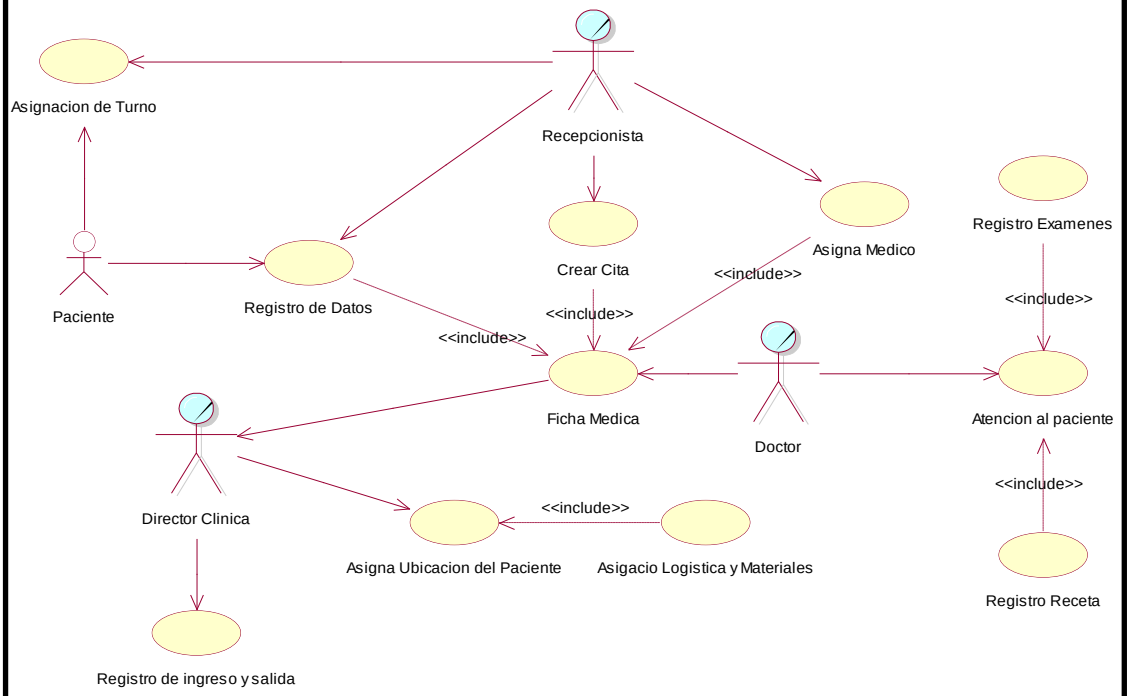


Figura 8. Diagrama de casos de uso general de donde se procederá a realizar el análisis y diseño.

Especificaciones de caso de uso

Tabla 8.

Asignación de Turno

ID	**U.C.1**
Nombre	Asignación de Turno
Actores: **Recepcionista**	
Recomendaciones:	
Flujo de encuesta:	
Flujo Alternativo: **El recepcionista recibe al paciente y le asigna un turno con el cual va ser atendido.**	
Post condiciones: **Se procede a realizar el registro de sus datos.**	

Tabla 9.

Registro de Datos

ID	**U.C.1**
Nombre	Registro de Datos
Actores: **Recepcionista**	
Recomendaciones:	
Flujo de encuesta:	
Flujo Alternativo: **El recepcionista registra los datos de los pacientes que van a ser atendidos en la clínica**	
Post condiciones: **Podrá ser atendido por el médico y se generará un historial clínico.**	

Tabla 10.

Crear Cita

ID	**U.C.1**
Nombre	Crear Cita
Actores: **Recepcionista**	

Recomendaciones:
Flujo de encuesta: **Se podrá generar una cita al paciente para que pueda ser atendido en un día determinado.**
Flujo Alternativo:
Post condiciones: **Se podrá cambiar de médico en caso de que se necesite una rápido atención.**

Tabla 11.

Ficha Médica

ID	**U.C.1**
Nombre	Ficha Médica
Actores: **Doctor**	
Recomendaciones:	
Flujo de encuesta:	
Flujo Alternativo: **Por cada atención que reciba se deberá registrar cada detalle sobre la misma.**	
Post condiciones: **Se podrá realizar consultas de una atención anterior.**	

Tabla 12.

Asignar Ubicación del Paciente

ID	**U.C.1**
Nombre	Asignar Ubicación del Paciente
Actores: **Director Clínico**	
Recomendaciones:	
Flujo de encuesta:	
Flujo Alternativo: **El director se encargara de asignar el lugar donde el paciente será hospitalizado y podrá recibir la atención**	
Post condiciones: **Se deberá generar la asignación de logística y materiales médicos.**	

Tabla 13.

Atención al Paciente

ID	**U.C.1**
Nombre	Atención al Paciente
Actores: **Doctor**	
Recomendaciones:	
Flujo de encuesta:	
Flujo Alternativo: **El doctor procederá a recibir al paciente y realizará un chequeo esto deberá ser registrado y será una historia más.**	
Post condiciones: **Podrá solicitar una cita médica o se podrá realizar una hospitalización.**	

Tabla 14.

Asignar Médico

ID	**U.C.1**
Nombre	Asignar Médico
Actores: **Recepcionista**	
Recomendaciones:	
Flujo de encuesta:	
Flujo Alternativo: **El recepcionista realiza la asignación del médico al paciente de acuerdo a la atención que necesite y los síntomas que presente**	
Post condiciones: **Se podrá cambiar de médico en caso de que se necesite una rápido atención.**	

Diagramas de Caso de Uso: **Asignar Médico**

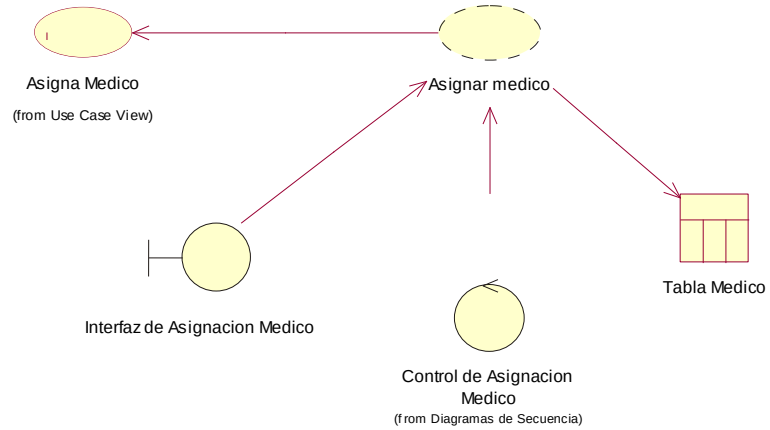


Figura 9. Permite identificar cuáles interfaces y controladores va a interactuar en la implementación del proceso dentro del sistema al momento de asignar a un médico a un paciente.

Diagramas de Caso de Uso: **Asignar Ubicación de Paciente**

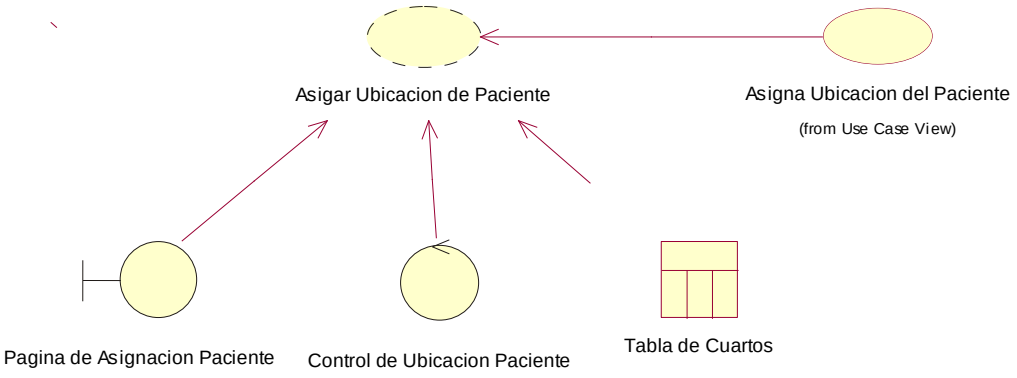


Figura 10. Muestra que el caso de uso Asignación de Pacientes va a interactuar mediante una interfaz que le permita asignar un cuarto al paciente y registrar el mismo.

Diagramas de Caso de Uso: **Asignación de Turno**

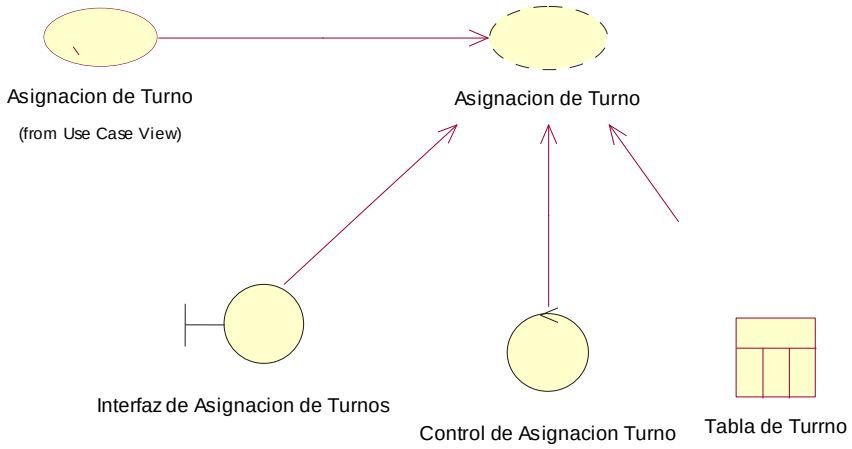


Figura 11. Para el proceso de Asignación de Turnos el sistema contara con una interfaz de asignación, un control de asignación y una tabla de turnos.

Diagramas de Caso de Uso: **Atención Paciente**

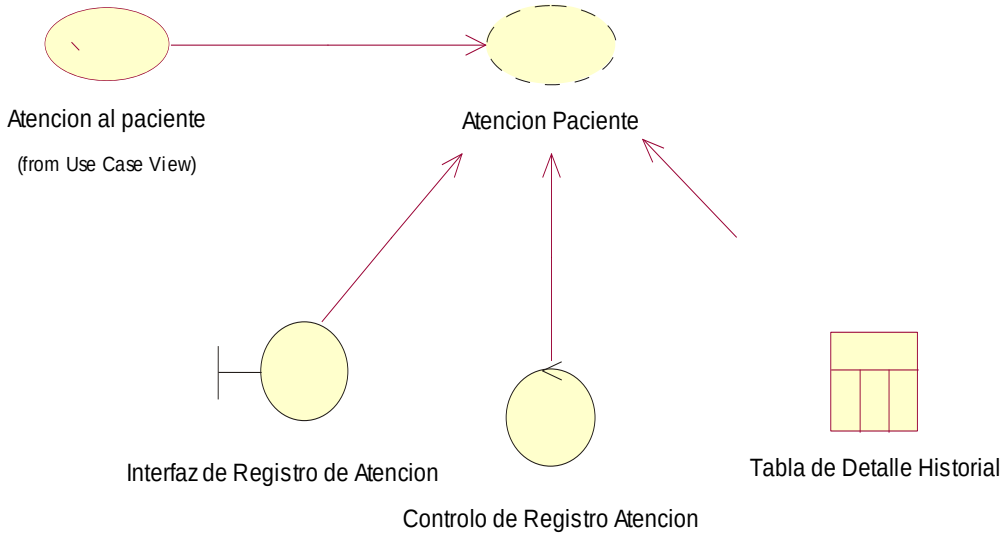


Figura 12 En el proceso de Atención Paciente se va a utilizar las interfaces de Historial Clínico, y control de Detalle de Historial.

Diagramas de Caso de Uso: **Crear Cita**

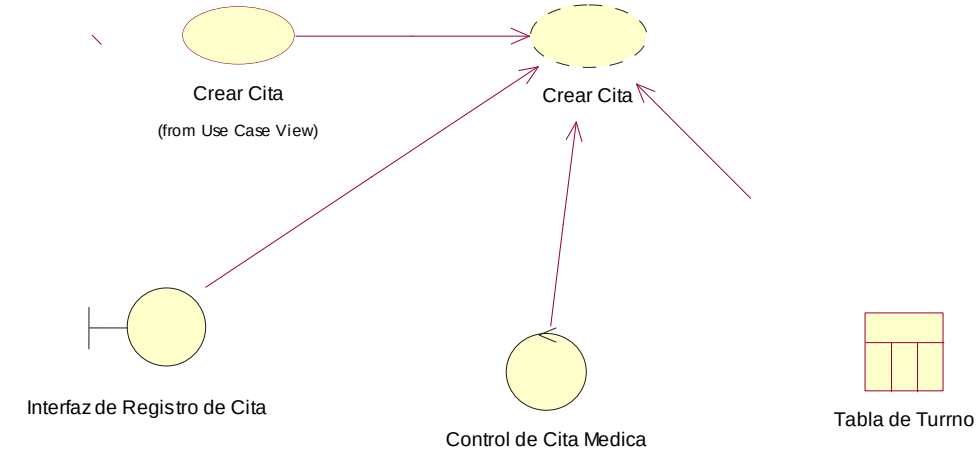


Figura 13. Para Crear una Cita se deberá utilizar la interfaz de Registro de Citas y utilizar un control de citas los cuales van a interactuar con la tabla Cita Médica.

Diagramas de Caso de Uso: **Ficha Médica**

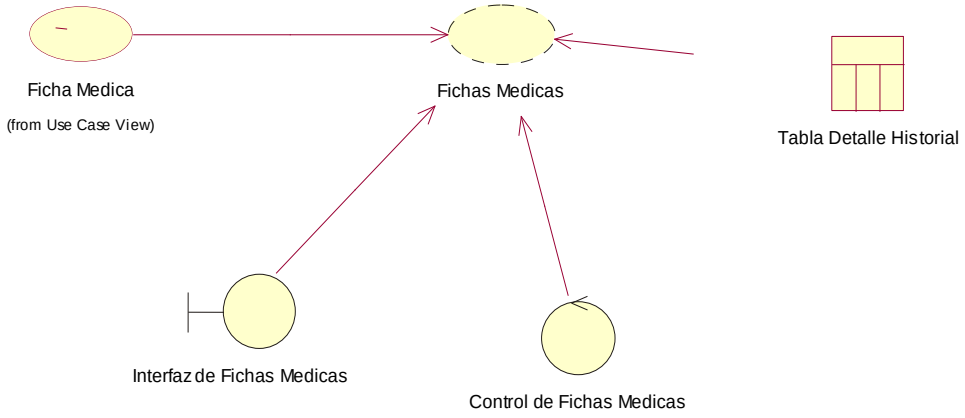


Figura 14. En el caso de la Ficha Médica o Historial utilizara la interfaz de registro de pacientes e Historial Clínico, Control de pacientes, control de historial y las tablas paciente e historial paciente.

Diagramas de Caso de Uso: **Registro de Datos**

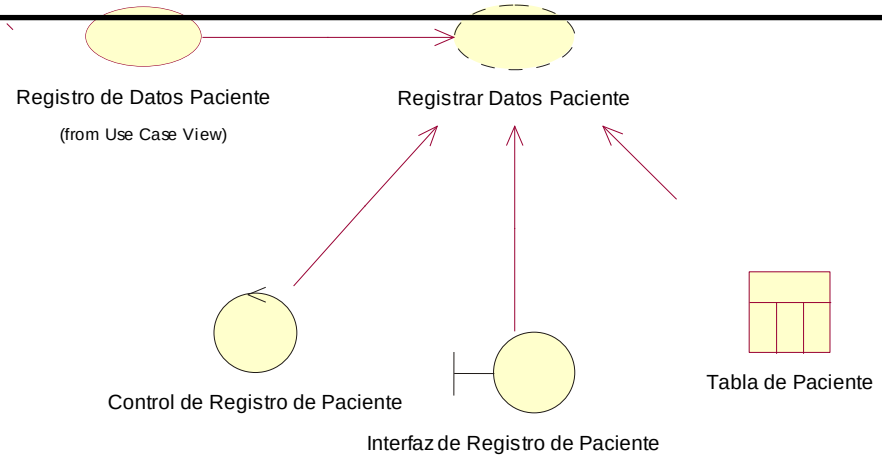


Figura 15. Se podrá ingresar nuevo pacientes al sistema mediante una interfaz el cual tendrá un control para poderlos registrar.

Diagramas de Caso de Uso: **Registro de Ingreso y Salida**

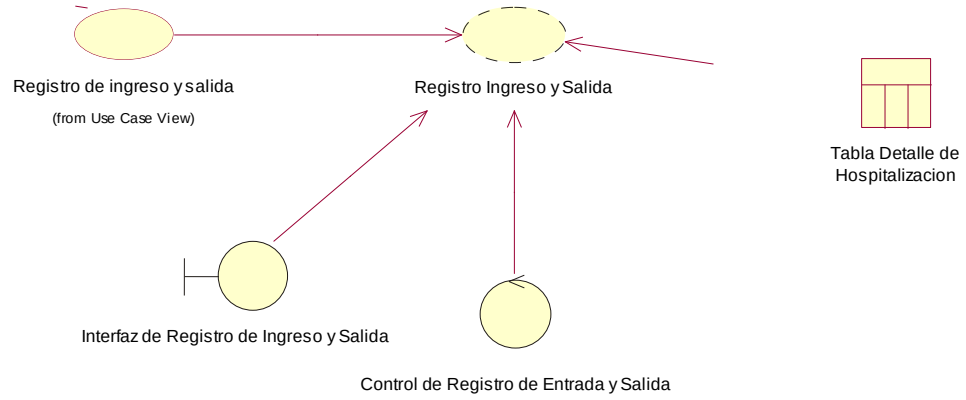


Figura 16. El sistema contara con una interfaz de hospitalización, su respectivo control y las tablas hospitalización y detalle hospitalización.

5.02.02 Diagramas de Componentes

Muestra los diferentes componentes que se utilizara para la implementación del sistema su extensión, las herramientas del desarrollo del código fuente el motor de base de datos, el servidor que se va utilizar para levantar la aplicación y el repositorio donde va a ubicar las librerías que servirán para el correcto funcionamiento de la aplicación.

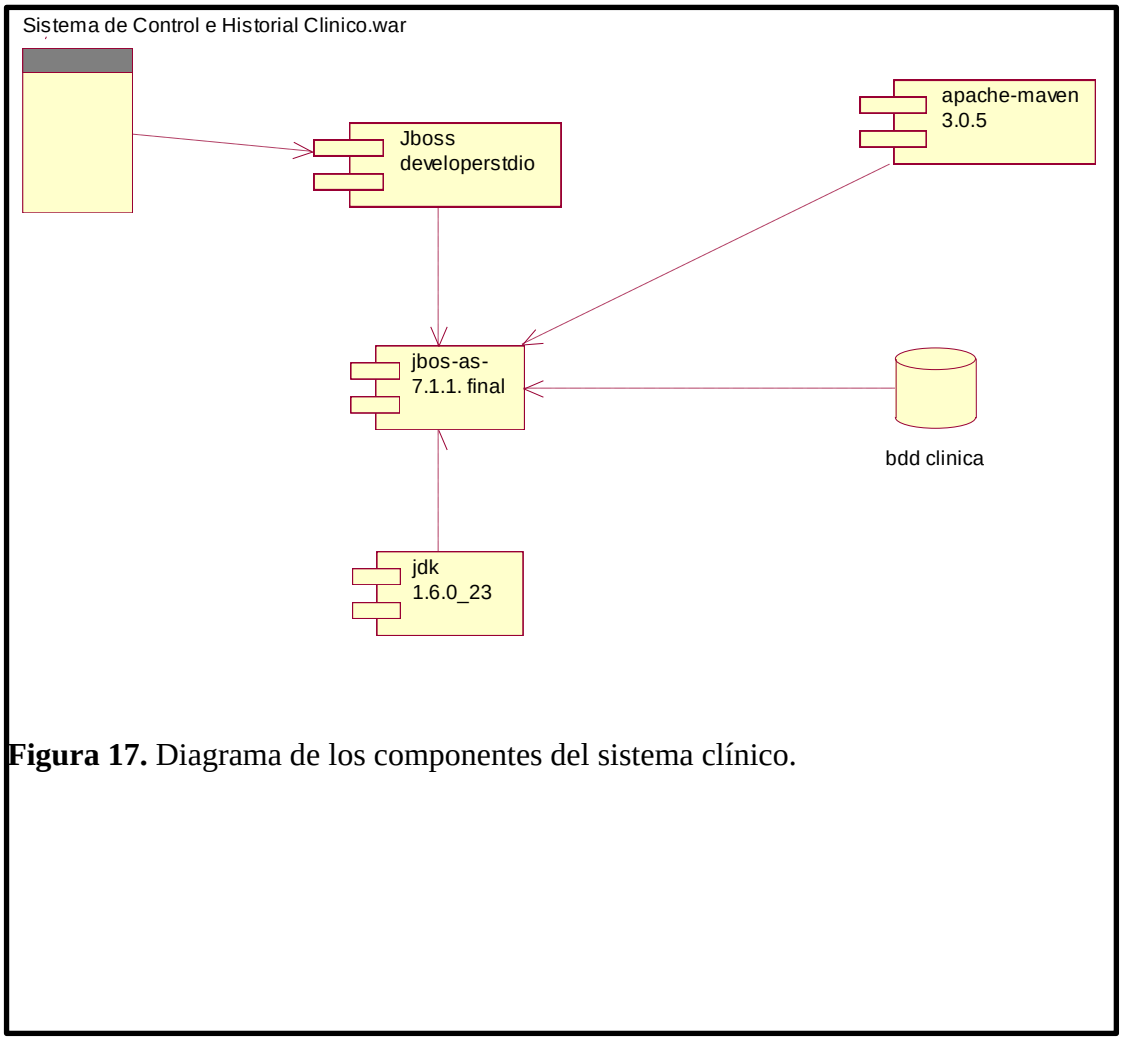


Figura 17. Diagrama de los componentes del sistema clínico.

5.02.03 Diagrama de Secuencia

Diagrama de Secuencia: **Login**

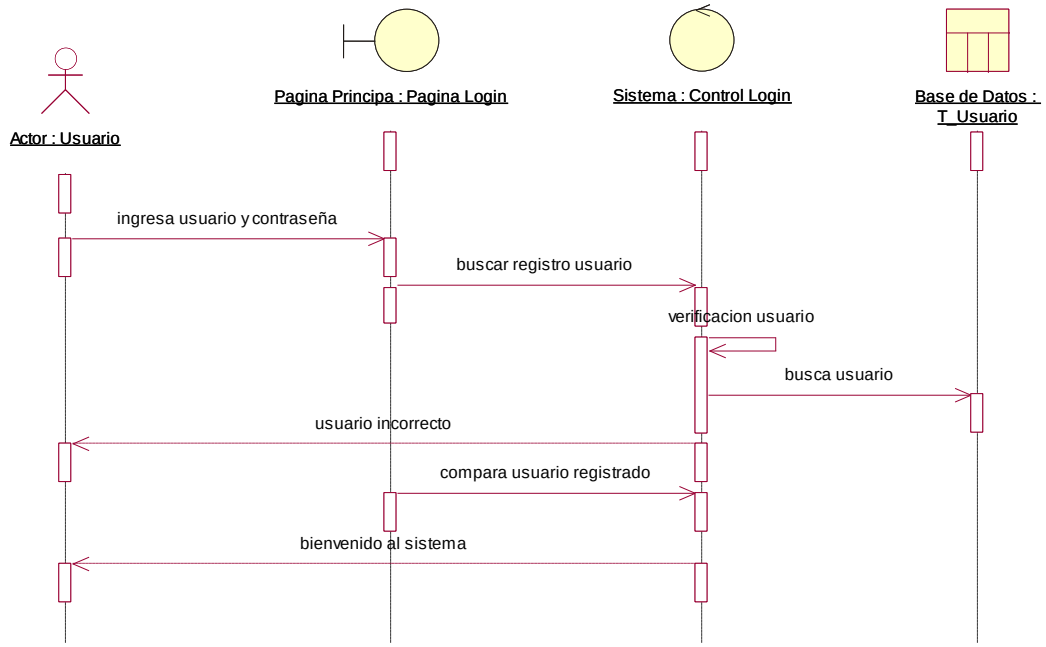


Figura 18. Para poder ingresar en el sistema médico el usuario deberá ingresar su nombre de Usuario y Contraseña que serán sus credenciales dentro del mismo, el grafico nos muestra cómo se deberá hacer el proceso de Login.

Diagrama de Secuencia: **Nuevo Usuario**

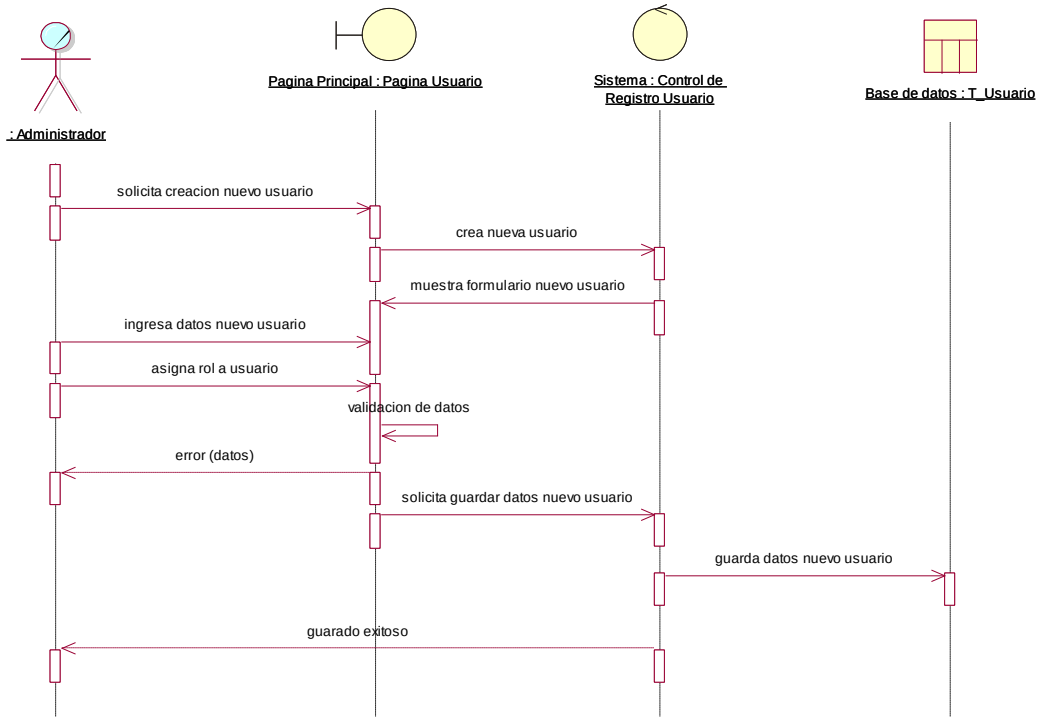


Figura 19. El administrador del sistema podrá crear nuevo usuarios en el sistema y asignara su respectivo rol de acuerdo a las funciones que realiza en la clínica.

Diagrama de Secuencia: **Nuevo Médico.**

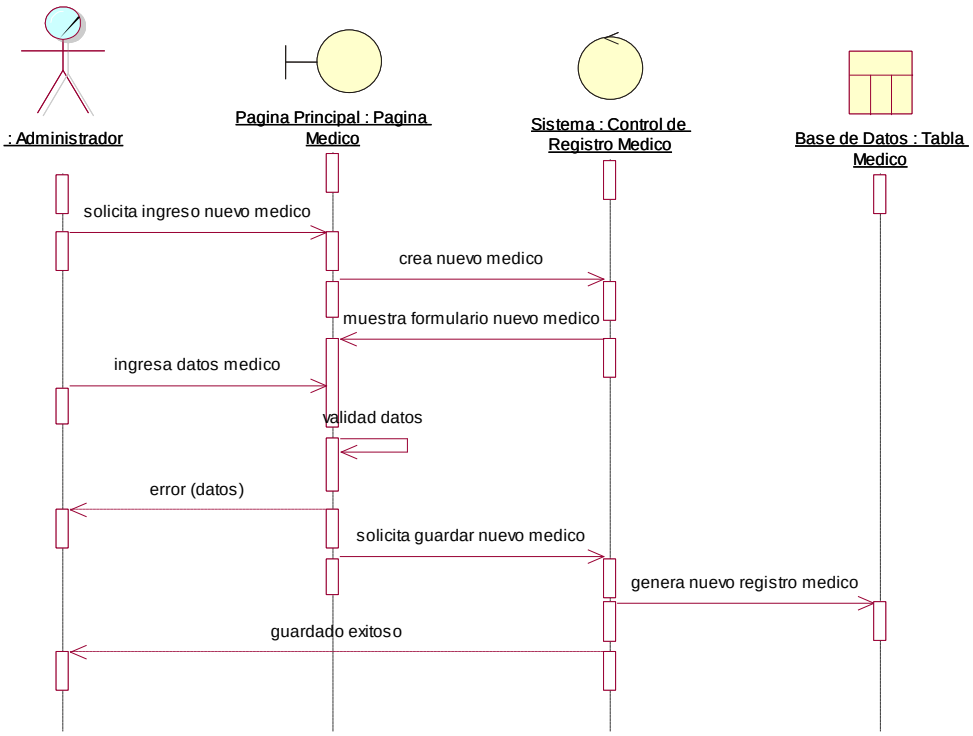


Figura 20. Muestra la secuencia que se realiza al momento de registrar un nuevo médico, además se especifica validaciones que tendrá esta pantalla, como se va a realizar el proceso dentro del sistema y los mensajes que se van a mostrar.

Diagrama de Secuencia: **Administración Cuarto**

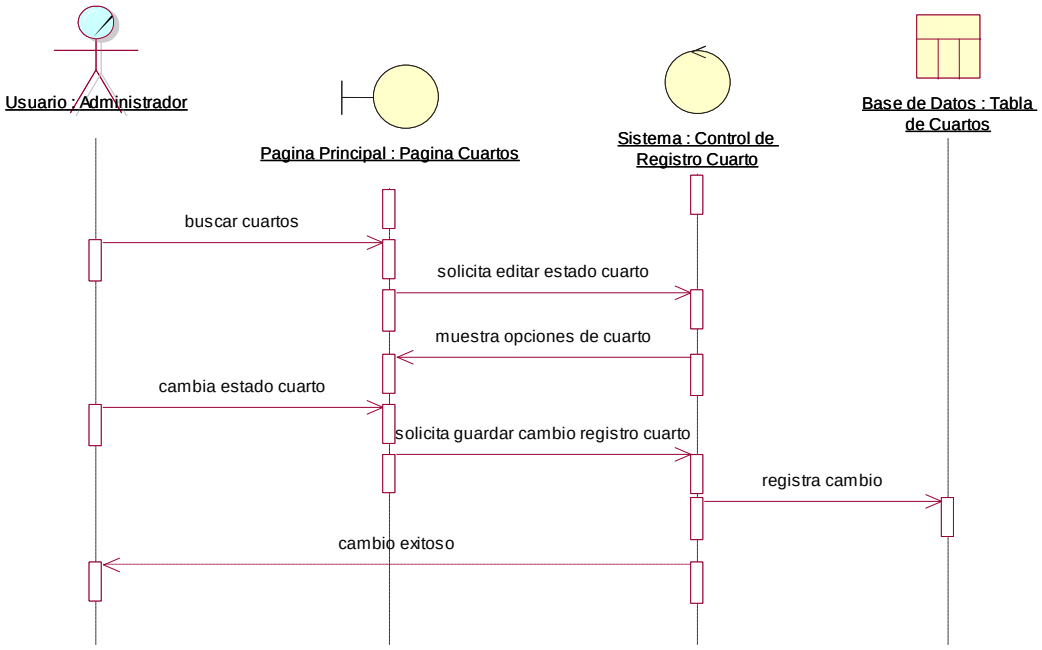


Figura 21. El diagrama detalla cuando se va a administrar los cuartos de los pacientes ya que estos deberán ser controlados que se encuentren en buen estado y disponibles para que puedan ser asignados en caso de que se registre una hospitalización.

Diagrama de Secuencia: **Nuevo Recurso**

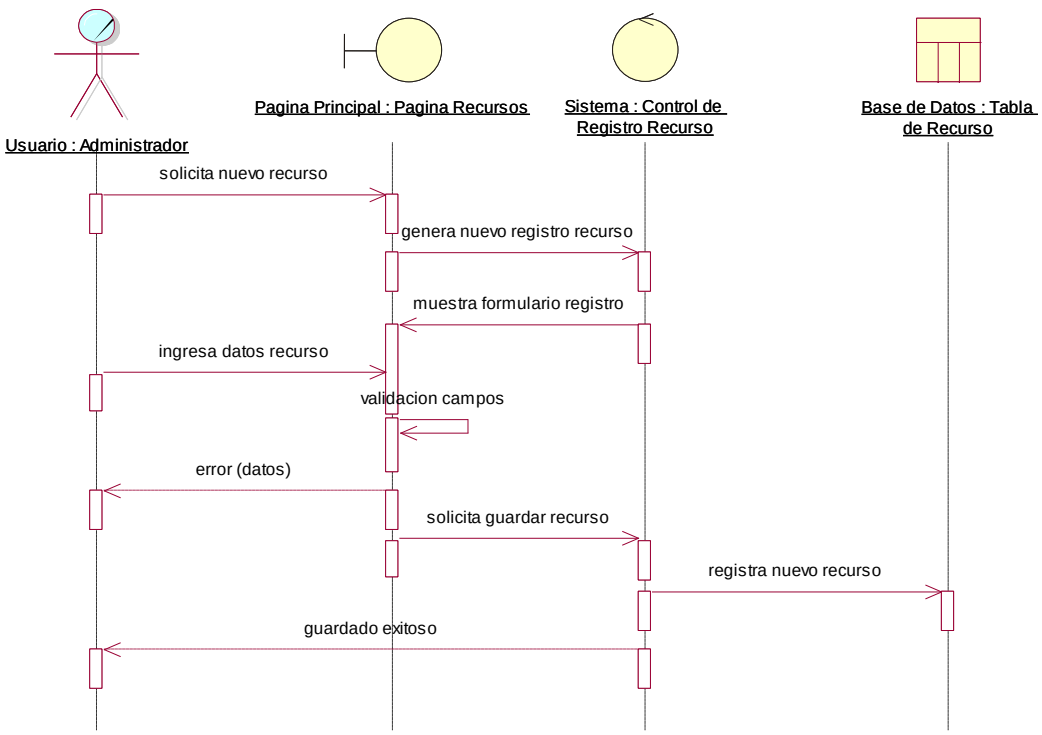


Figura 22. El diagrama muestra la secuencia al registrar un nuevo recurso en el sistema, se puede mostrar las validaciones, mensajes y el proceso en si del registro.

Diagrama de Secuencia: **Nuevo paciente**

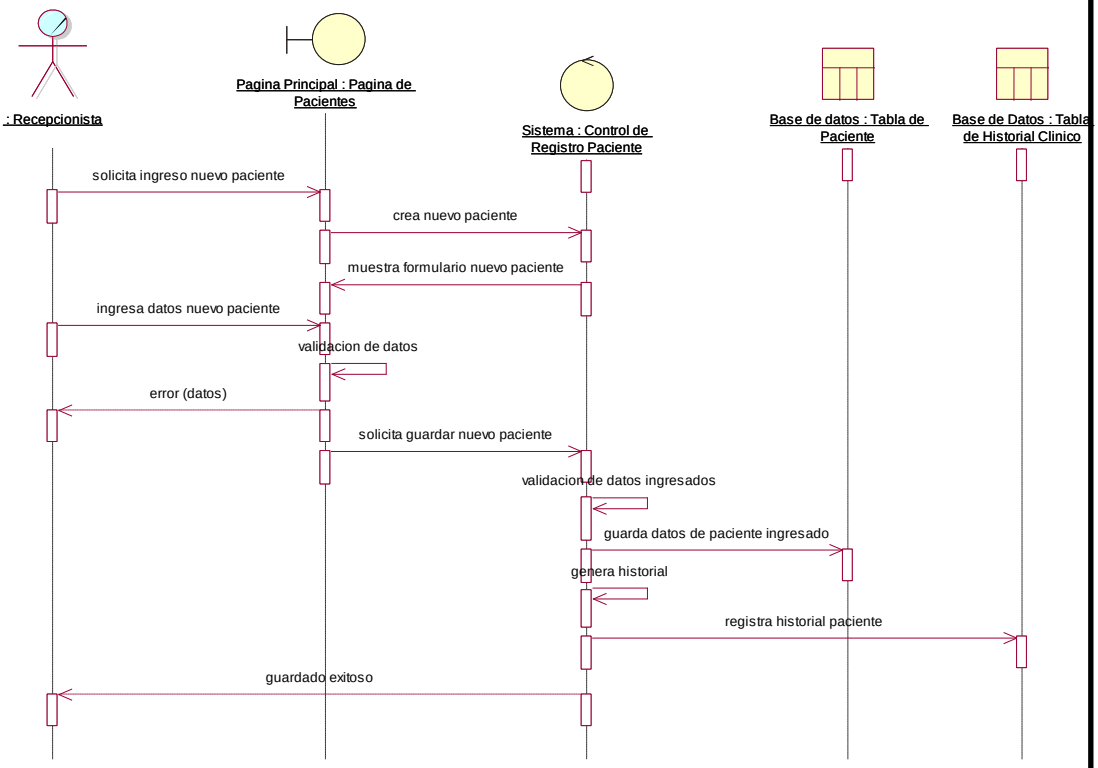


Figura 23. El proceso de registro de pacientes muestra en secuencia el detalle los procesos, mensajes y validaciones, el sistema el proceso interior deberá generar un número de historial automáticamente.

Diagrama de Secuencia: **Nuevo Turno**

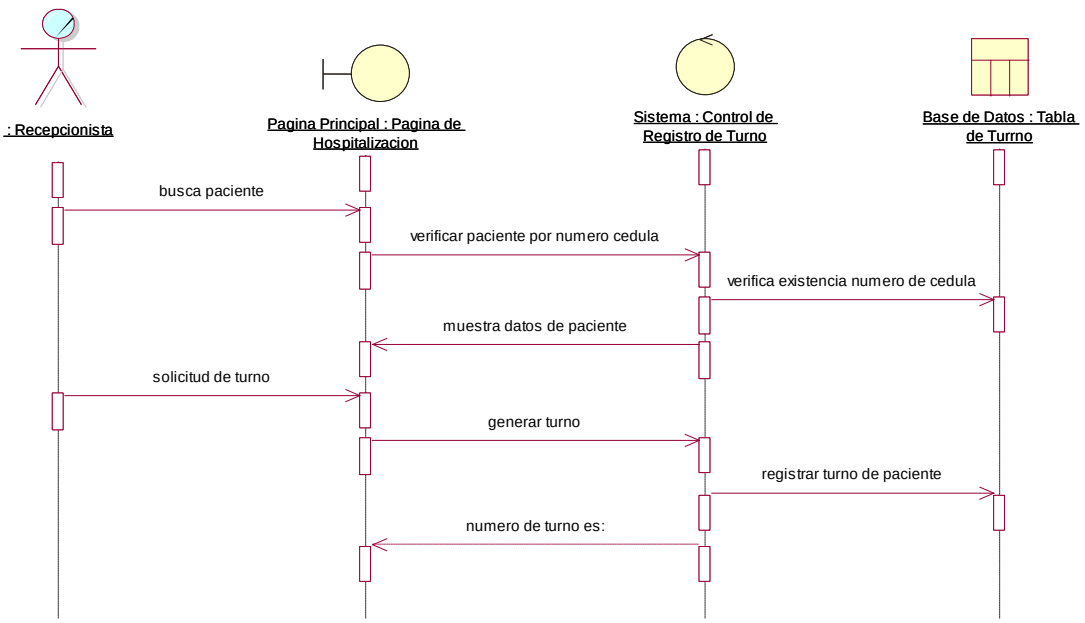


Figura 24. Para que el paciente pueda ser atendido necesita ingresar los datos del paciente de acuerdo, este proceso lo ejecutará el usuario Recepcionista, el cual deberá buscar el número de historial del paciente, en el caso de no encontrar la información deberá crear uno nuevo eso se especifica en el diagrama de Ingresar Nuevo Paciente, deberá ingresar el estado actual del paciente y guardar los mismos, se asignará un Turno para que pueda ser atendido.

Diagrama de Secuencia: **Historial Clínico**

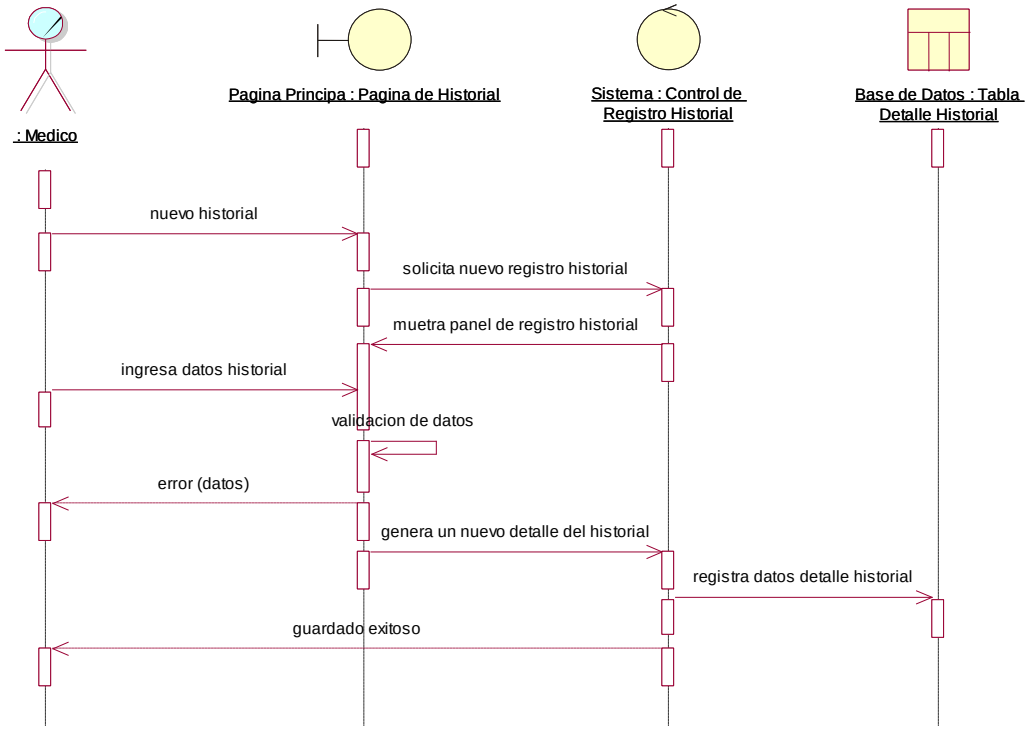


Figura 25. El usuario Doctor realiza el diagnóstico del paciente, ingresara información de algún examen médico sugerido por el doctor y lo registra una receta la cual debe ser registrada para una posterior verificación y consulta

Diagrama de Secuencia: **Hospitalización**

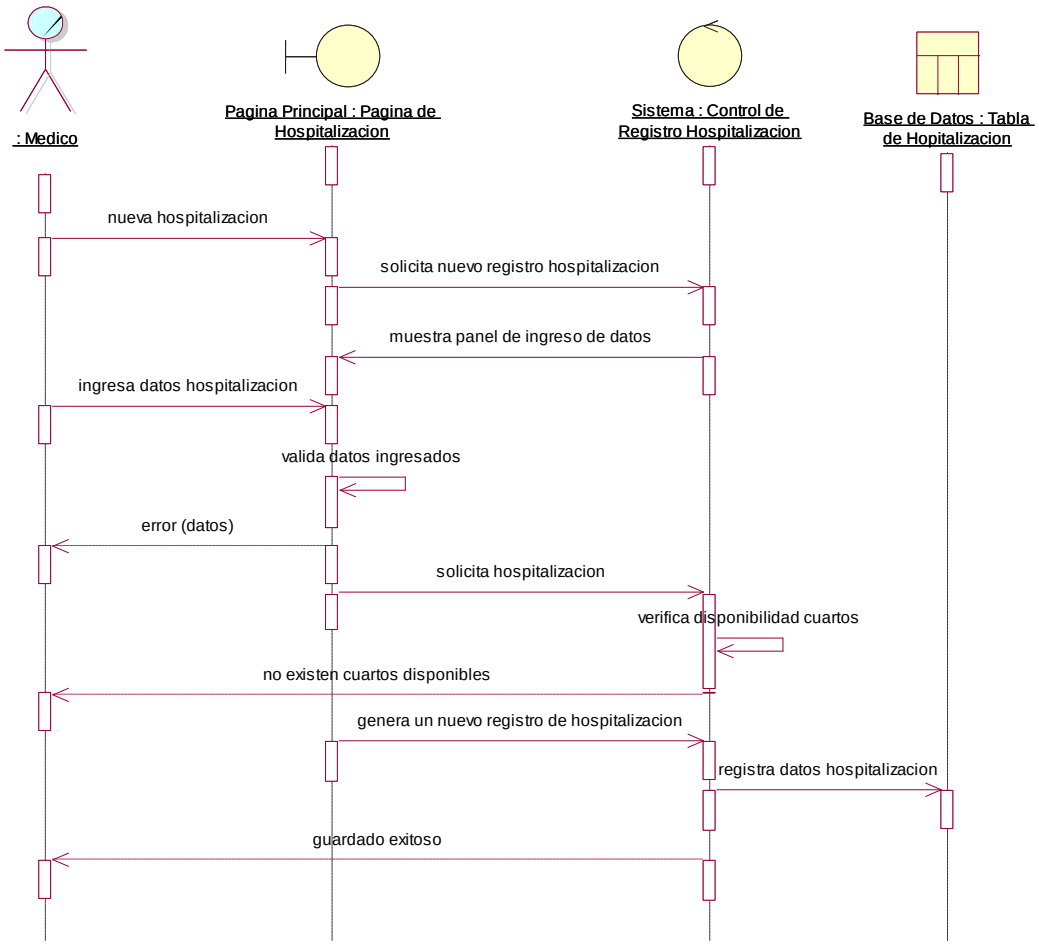


Figura 26. Permite registrar una nueva hospitalización por paciente donde se muestra el proceso a realizar para el registro.

Diagrama de Secuencia: **Registro Visita**

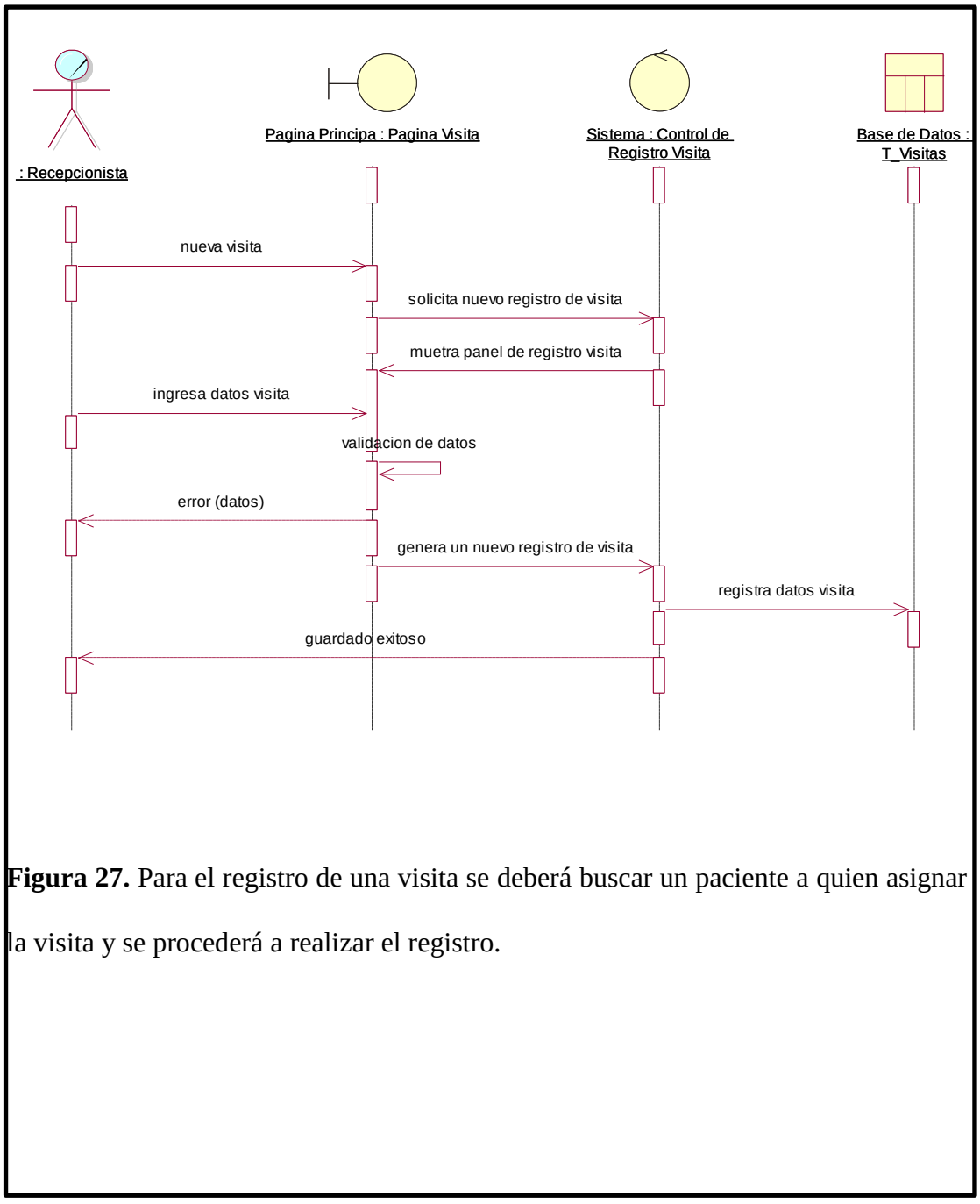


Figura 27. Para el registro de una visita se deberá buscar un paciente a quien asignar la visita y se procederá a realizar el registro.

5.02.04 Diagrama de Colaboración

Diagrama de Colaboración: **Login**

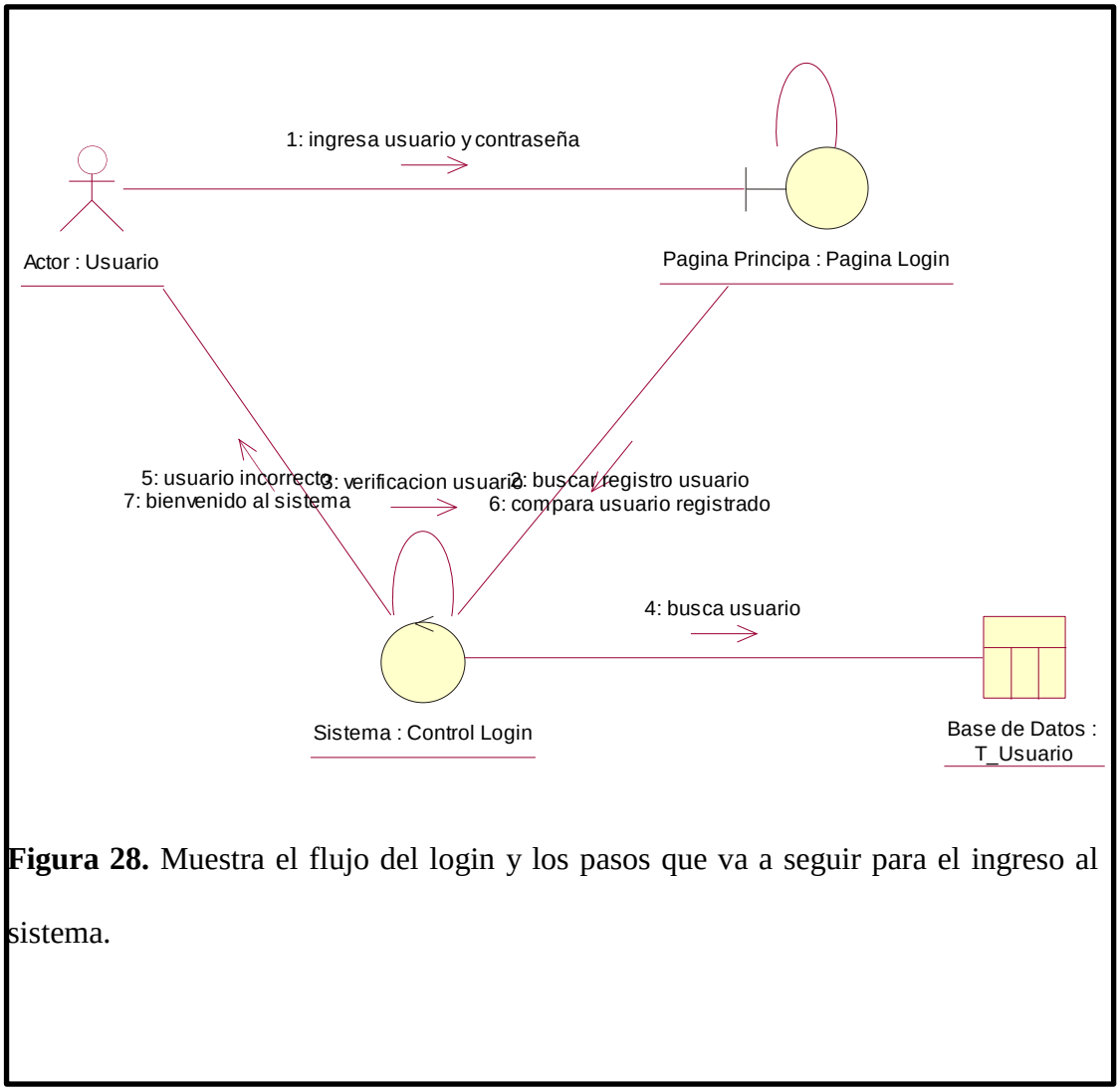


Figura 28. Muestra el flujo del login y los pasos que va a seguir para el ingreso al sistema.

Diagrama de Colaboración: **Nuevo Usuario**

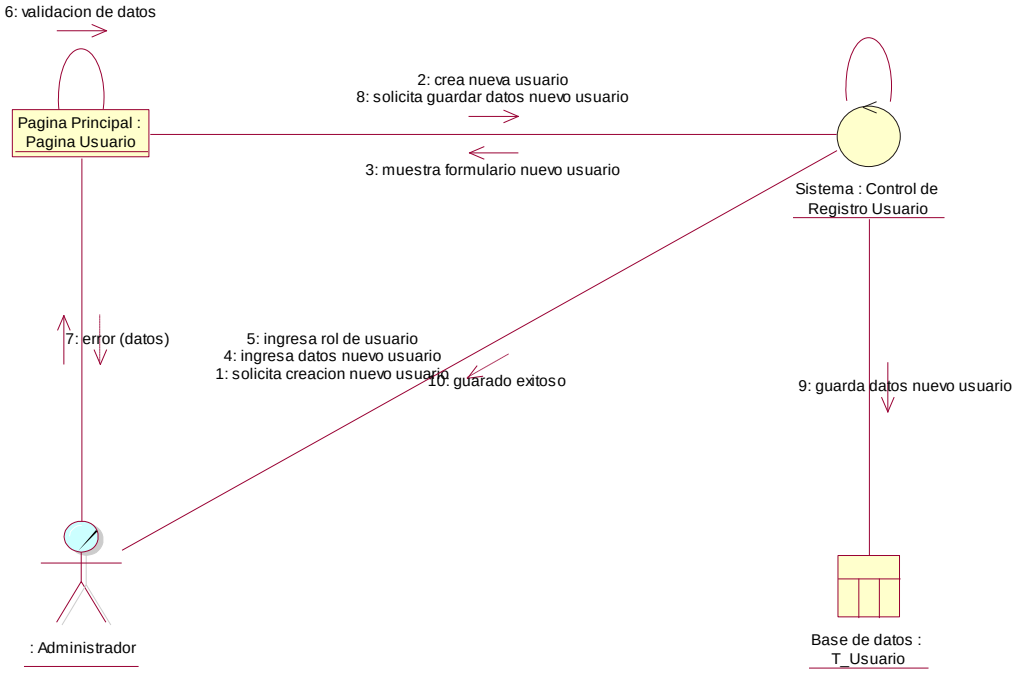


Figura 29. Nos muestra las relaciones que se encuentran entre los procesos que se van a realizar en el registro de un usuario

Diagrama de Colaboración: **Nuevo Médico**

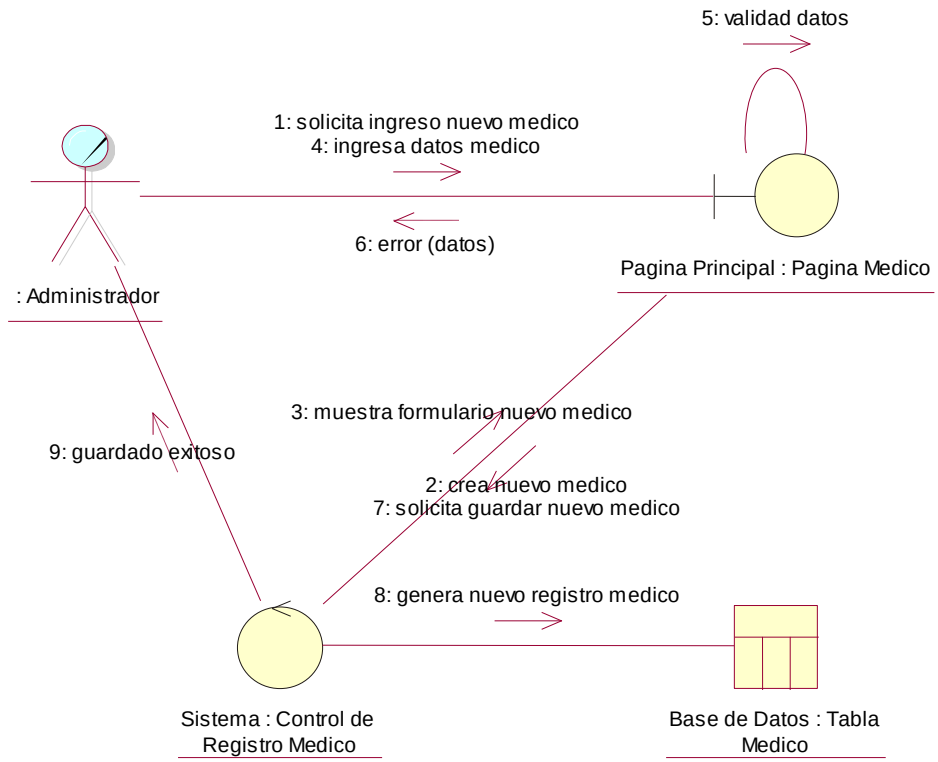


Figura 30. Identificamos paso a paso el proceso de ingreso de un médico al sistema.

Diagrama de Colaboración: **Administración Cuartos**

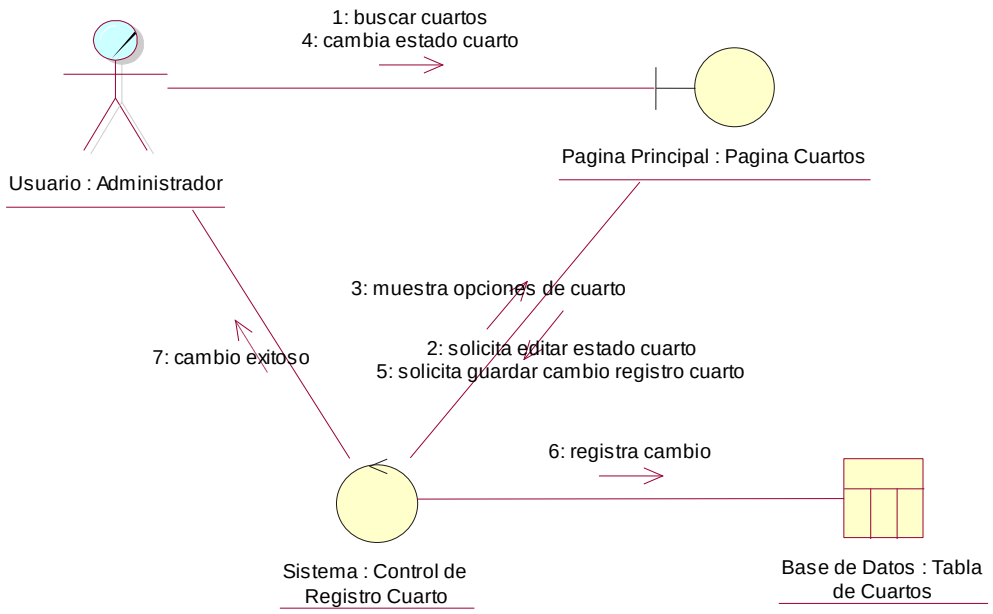


Figura 31. Muestra cómo se realiza la administración de un cuarto para disponerlos a los pacientes.

Diagrama de Colaboración: **Nuevo Recurso**

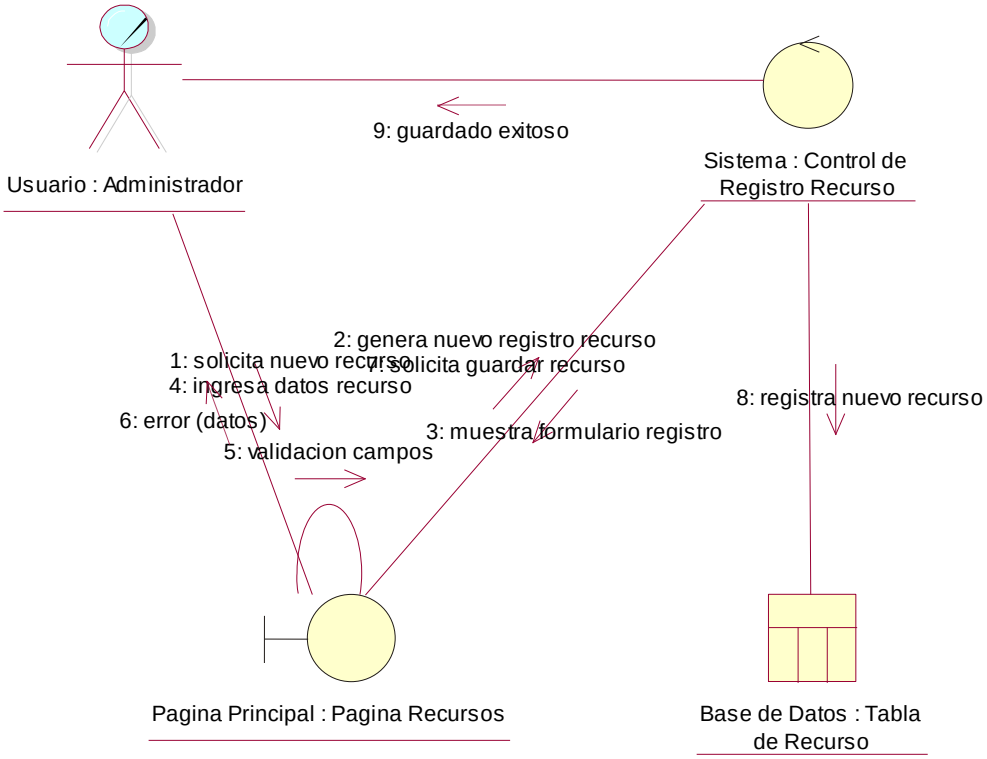


Figura 32. Describe el proceso de registro de un nuevo recurso.

Diagrama de Colaboración: Nuevo Paciente

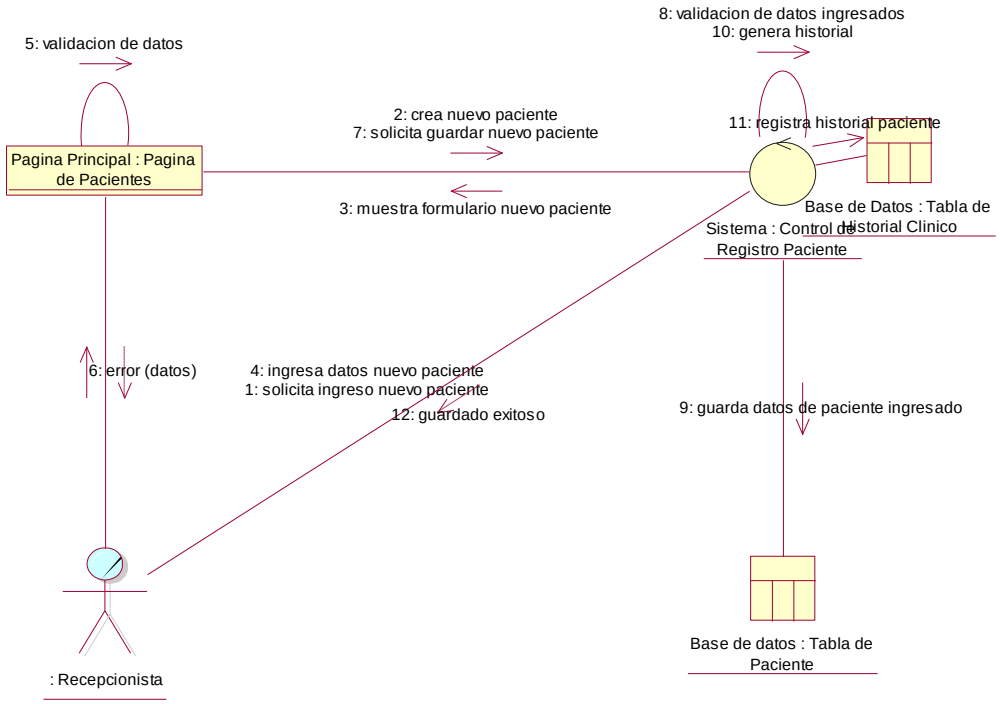


Figura 33. Procesos de Registro de nuevo Paciente y la creación de un historial al mismo.

Diagrama de Colaboración: **Nuevo Turno**

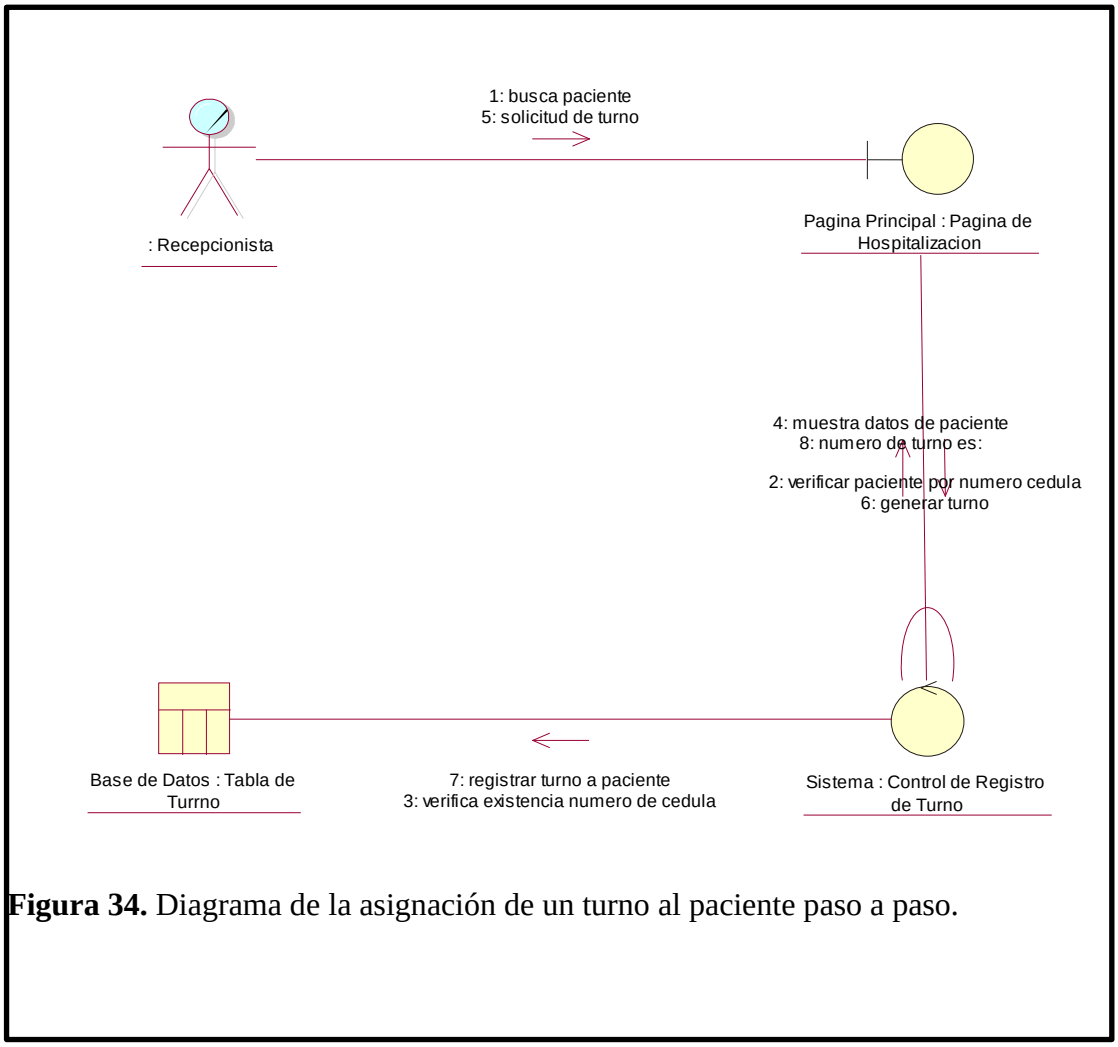


Figura 34. Diagrama de la asignación de un turno al paciente paso a paso.

Diagrama de Colaboración: **Nueva Historia**

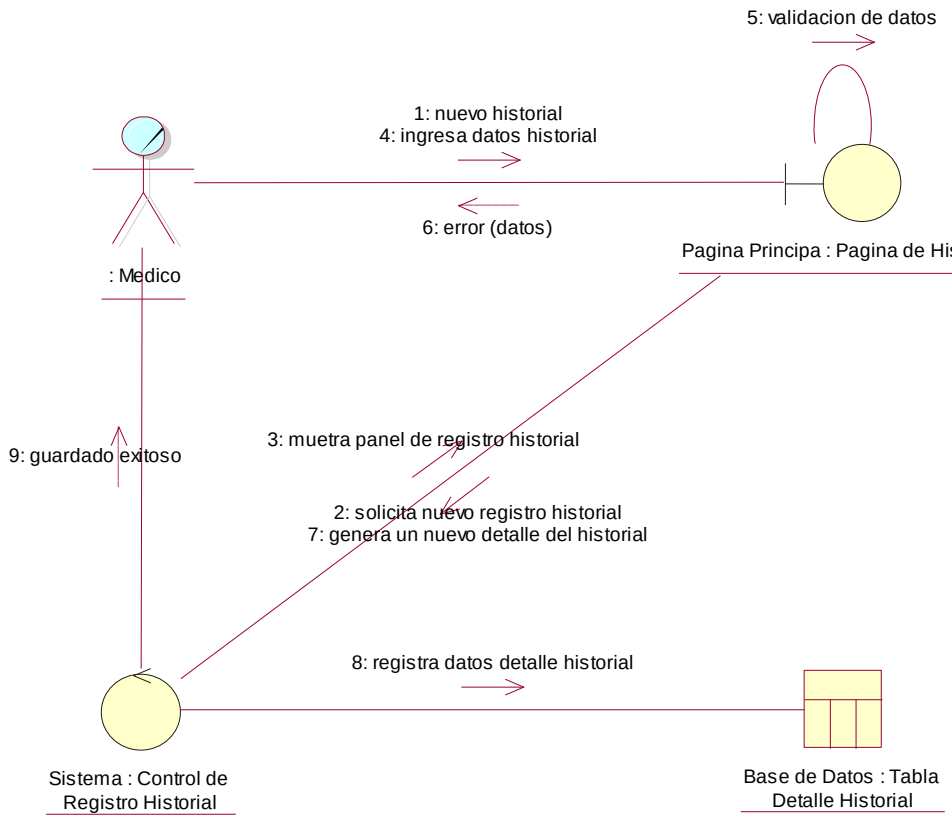


Figura 35. Diagrama que grafica cada paso a seguir para realizar el registro del historial.

Diagrama de Colaboración: **Nueva Hospitalización**

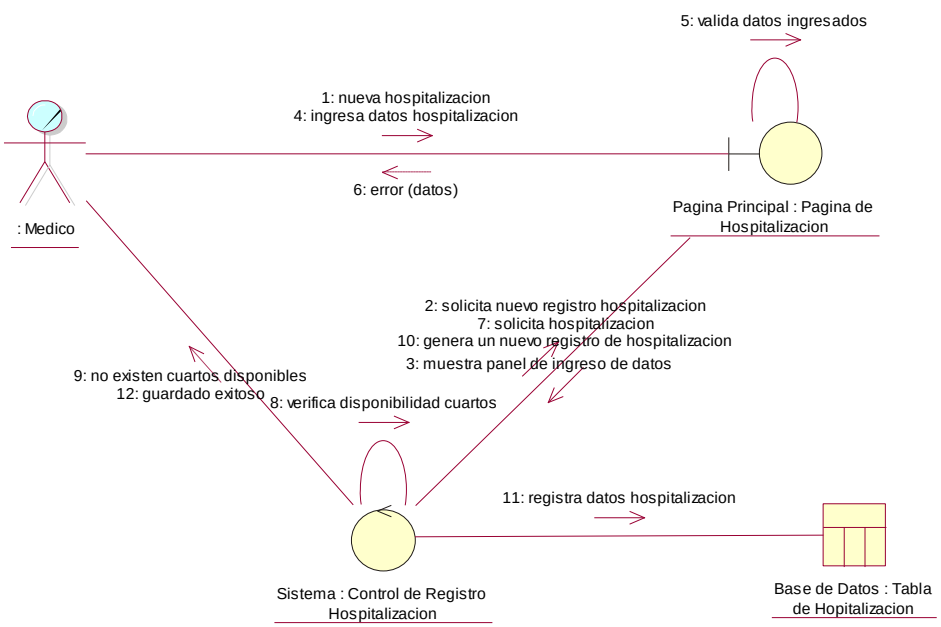


Figura 36. Proceso que conlleva realizar el registro de una hospitalización.

Diagrama de Colaboración: **Nueva Visita**

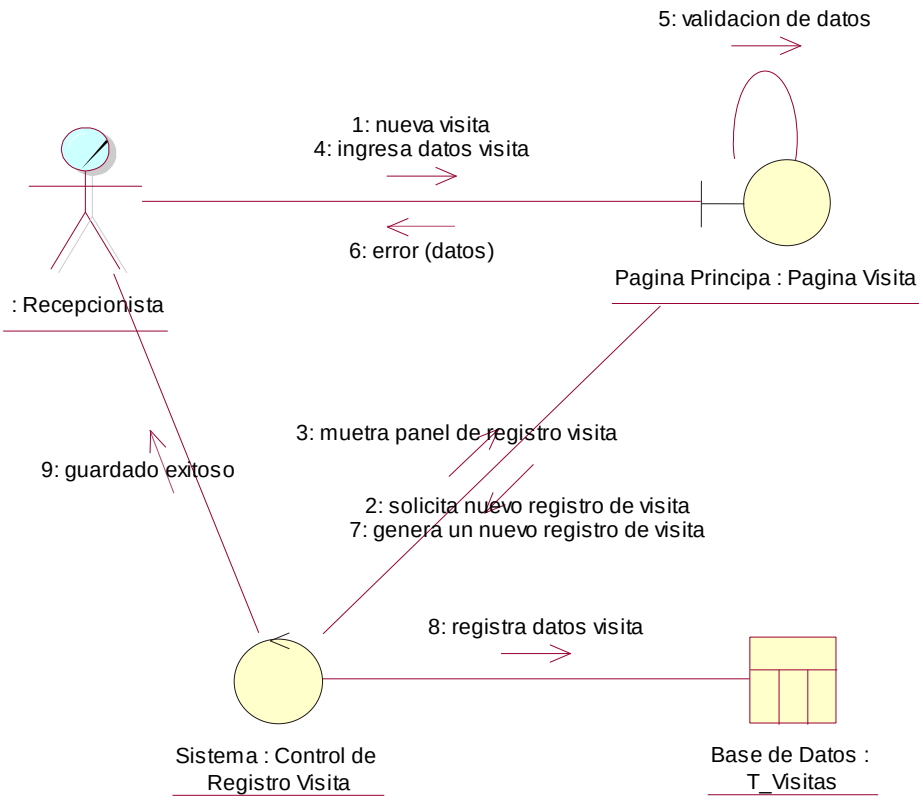


Figura 37. Diagrama de registro de visitas a pacientes que se encuentran hospitalizados.

5.02.05 Diagramas de Clases

El presente literal muestra en forma física y lógica la Base de Datos sobre la cual va a interactuar el sistema

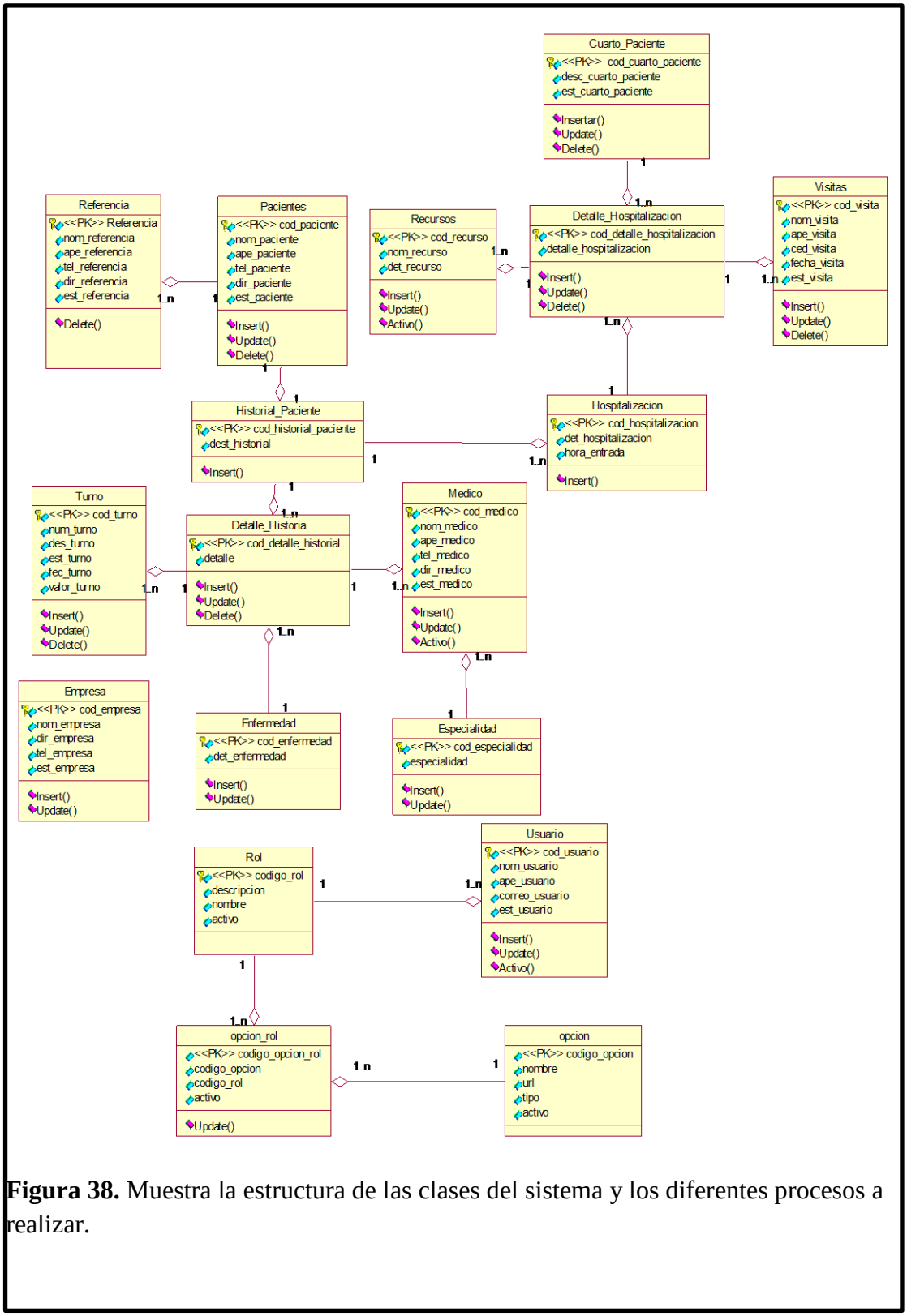


Figura 38. Muestra la estructura de las clases del sistema y los diferentes procesos a realizar.

5.02.05.01 Lógico

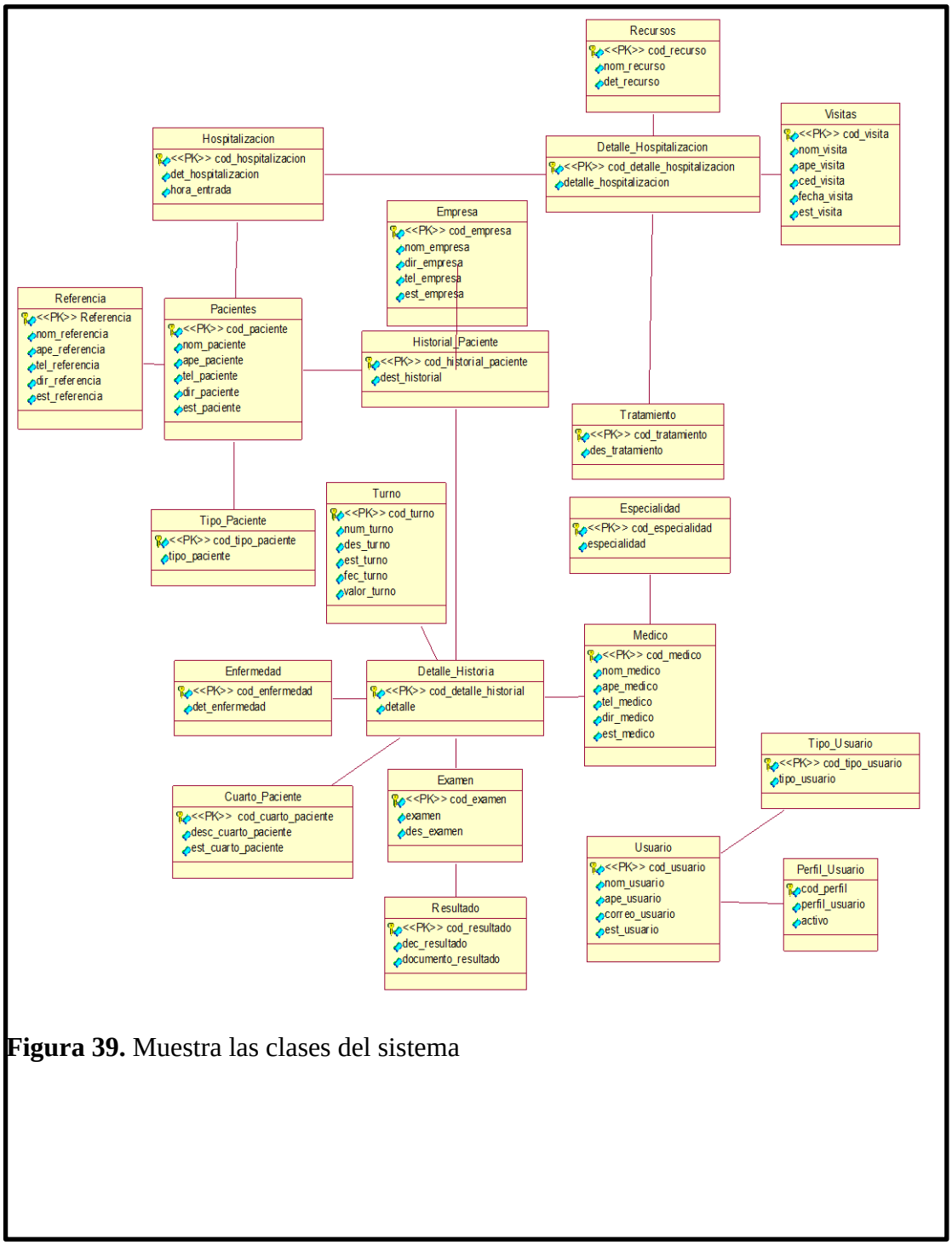


Figura 39. Muestra las clases del sistema

5.02.05.02 Físico

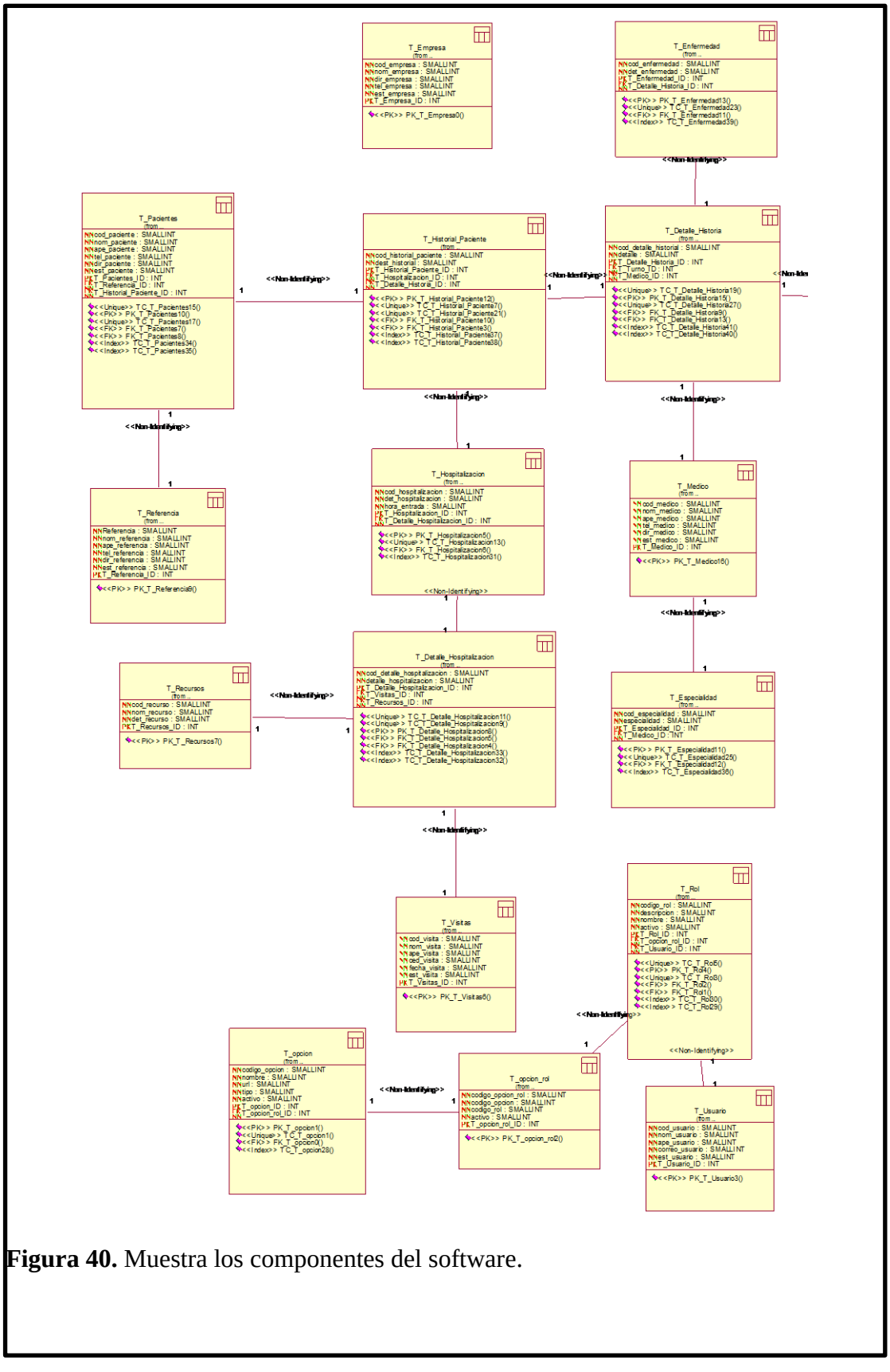


Figura 40. Muestra los componentes del software.

5.03 Desarrollo

El desarrollo del sistema consta de capas bien definidas las cuales ayudan a separar la lógica del negocio, de la persistencia y de la presentación.

5.03.01 Arquitectura de Software

En el desarrollo de la aplicación se realizó en tres capas Controller, Servicio y Dao. Estas últimas utilizan interfaz para sus implementaciones, y el Controller un Data Manager (DM).

CAPAS

Controller

El controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el modelo. Entra en acción cuando se realiza alguna operación, ya sea un cambio en la información del modelo o una interacción sobre la Vista. Se comunica con el modelo y la vista a través de una referencia al propio modelo, controlas las acciones realizadas en las deferentes interfaces.

Servicio

Lógica del negocio

Interface

Las interfaces son casos particulares de las clases de Java, que carecen de implementación y que se espera que otras clases implementen. Usualmente funcionan como una especie de contrato, indicando lo que la clase implementada debería hacer, mientras que los detalles de más bajo nivel corresponderían al

implementador. El procedimiento de creación de interfaces es muy similar al procedimiento de creación de nuevas clases ya detallado. Aunque una interfaz no puede implementar ninguna interfaz, sí que puede extender otra interfaz mediante una relación de herencia.

Las interfaces proporcionan las siguientes ventajas:

- Organizar la programación.
- Obligar a que ciertas clases utilicen los mismos métodos (nombres y parámetros).
- Establecer relaciones entre clases que no estén relacionadas

Implementación

Contienen las implementaciones de los métodos creados en las interfaces. Cuando implementamos una interface, estamos acordando que añadimos el contrato definido en una interface. Esto significa que estamos de acuerdo en proveer una implementación legal para cada método definido en la interface, y cualquiera que sepa lo que parecen hacer los métodos de la interface (No como deben ser implementados) pueden estar seguros de que pueden invocar estos métodos en una instancia de la clase que estemos implementando.

DAO

Objeto de acceso a Datos, implementa el mecanismo de acceso requerido para trabajar con la fuente de datos. Los Objetos de Acceso a Datos son un Patrón de los subordinados de Diseño Core J2EE y considerados una buena práctica. La ventaja de usar objetos de acceso a datos es que cualquier objeto de negocio (aquel que contiene

detalles específicos de operación o aplicación) no requiere conocimiento directo del destino final de la información que manipula.

Tets

Unit Test es la capa que nos ayuda a realizar las pruebas de unidad en donde la aplicación reconocerá a las clases y al momento de levantar la aplicación no las tomara en cuenta para su funcionamiento.

Estos archivos por lo general utilizan la palabra Test en su nombra para identificarlos de esta manera.

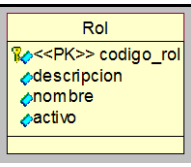
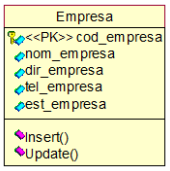
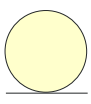
5.03.02 Estándares de Programación

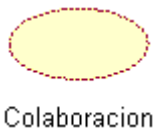
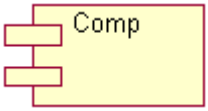
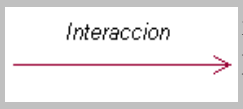
Estándar de Análisis y Diseño

Para realizar los diferentes diagramas y casos de uso se utilizaron los siguientes componentes:

Tabla 15.

Estándares de Diseño

ELEMENTOS	Clase		Describe un conjunto de objetos que comparten los mismos atributos, métodos, relaciones y semántica. Las clases implementan una o más interfaces.
	Clase activa		Se trata de una clase, en la que existen procesos o hilos de ejecución concurrentes con otros elementos. Las líneas del contorno son más gruesas que en la clase "normal"
	Actor		EL grafico de un actor al cual será participe de los principales diagramas y del caso de uso general.
	Control		Se lo define como control de las interfaces o de los diferentes procesos que se realicen en el análisis.
	Interfaz		Agrupación de métodos u operaciones que especifican un servicio de una clase o componente, describiendo su

			comportamiento, completo o parcial, externamente visible.
ESTRUCTURALES	Colaboración		Define una interacción entre elementos que cooperan para proporcionar un comportamiento mayor que la suma de los comportamientos de sus elementos.
	Caso de uso		Describe un conjunto de secuencias de acciones que un sistema ejecuta, para producir un resultado observable de interés. Se emplea para estructurar los aspectos de comportamiento de un modelo.
	Componente		Parte física y por tanto reemplazable de un modelo, que agrupa un conjunto de interfaces, archivos de código fuente, clases, colaboraciones y proporciona la implementación de dichos elementos.
Elementos de comportamiento	Interacción		Comprende un conjunto de mensajes que se intercambian entre un conjunto de objetos, para cumplir un objetivo específico.

Estándares de Base de datos

Normas generales

Los nombres de objetos de base de datos deberán estar escritos en español, ser auto-descriptivos.

No se aceptan espacios en blanco en medio de los identificadores.

Todos los nombres deben estar en idioma español.

Usar solo caracteres alfanuméricos (A-Z) más el guion bajo (_) como separador de palabras.

No se deben incluir letras con tilde ni Ñ, ñ, letras con diéresis Ejemplo: ïÿû.

El primer carácter del nombre debe ser una letra (A-Z).

No usar caracteres especiales como por ejemplo ($, #, *, - +, etc., ´, ¨;).

El nombre del objeto debe tener como longitud máxima 30 caracteres.

No usar palabras reservadas de SQL y PL/SQL. (Ver Anexos A y B)

No emplear acrónimos ni abreviaturas como nombres.

Si un nombre no es suficiente para caracterizar un objeto, utilizar dos o más palabras, separar las palabras con el guion bajo (_).

Tabla 16.

Tipos de Datos

Tipo de datos	Prefijo	Ejemplo
Integer	integer	b_Encontrado
Character Varing	vchar	vchar_descripcion
Numeric	num	num_Datos
Date (Time)	Dt	dt_Inicio
Boolean	Dbl	Activo
Long	L	l_Distancia
Object	Obj	obj_Activo
Char	Chr	Chr_Letter
String	Str	str_NombreF

Tabla 17.

Nombre de Tablas

Nombre	Nomenclatura
Tabla de usuario	usuario
Tabla Detalle Historial	detalle_historial

Tabla 18.

Campos de Tablas

Nombre	Nomenclatura
Identificador de usuario	codigo_usuario
Nombre usuario	usuario_nombre
Apellido usuario	usuario_apellido

Tabla 19.

Formato de Relaciones

Nombre	Nomenclatura
Relación entre tipo de usuario y usuario	FK_OPTIONSID-TBL_USUARIO-TBL_TUSUARIO

Estándares de Programación

Organización de ficheros

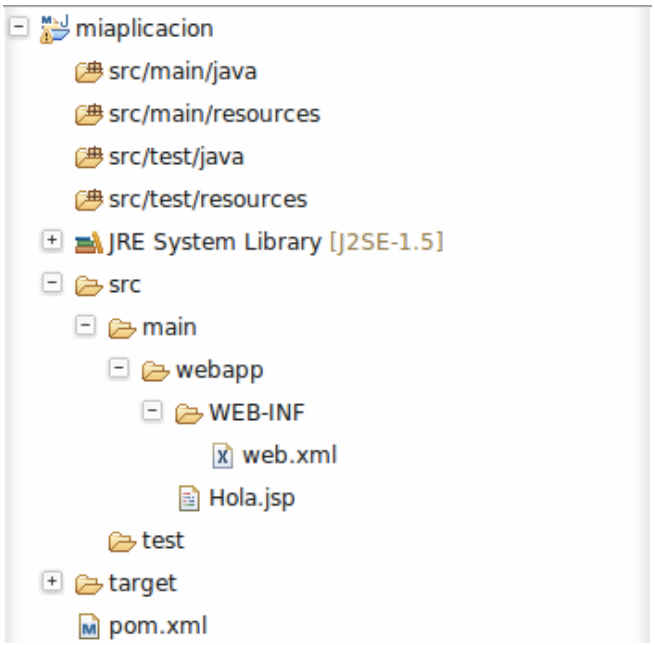


Figura 41. Muestra la estructura de un proyecto maven.

Las clases en Java se agrupan en paquetes. Estos paquetes se deben organizar de manera jerárquica, de forma que todo código desarrollado para el proyecto tendrá que estar incluido dentro del paquete "ec.proyecto".

Dentro del paquete principal las clases se organizarán en subpaquetes en función del área, organismo o sección del Ayuntamiento al que pertenezca el código desarrollado.

Un fichero consta de secciones que deben estar separadas por líneas en blanco y comentarios opcionales que identifiquen cada sección.

Deben evitarse los ficheros de gran tamaño que contengan más de 1000 líneas. En ocasiones, este tamaño excesivo provoca que la clase no encapsule un comportamiento claramente definido, albergando una gran cantidad de métodos que realizan tareas funcional o conceptualmente heterogéneas.

Fichero fuente Java (.java)

Cada fichero fuente Java debe contener una única clase o interfaz pública. El nombre del fichero tiene que coincidir con el nombre de la clase. Cuando existan varias clases privadas asociadas funcionalmente a una clase pública, podrán colocarse en el mismo fichero fuente que la clase pública. La clase pública debe estar situada en primer lugar dentro del fichero fuente.

En todo fichero fuente Java distinguimos las siguientes secciones:

- Comentarios de inicio.
- Sentencia de paquete.
- Sentencias de importación.
- Declaraciones de clases e interfaces.

Sentencias de paquete

La primera línea no comentada de un fichero fuente debe ser la sentencia de paquete, que indica el paquete al que pertenece(n) la(s) clase(s) incluída(s) en el fichero fuente. Por ejemplo,

Tabla 20.

Formato de Paquetes y Librerías

Paquetes	**Librerías**
package javax.crypto	import java.io.Serializable

Sentencias de importación

Los nombres de clases deben ser sustantivos y deben tener la primera letra en mayúsculas. Si el nombre es compuesto, cada palabra componente deberá comenzar con mayúsculas.

Los nombres serán simples y descriptivos. Debe evitarse el uso de acrónimos o abreviaturas, salvo en aquellos casos en los que dicha abreviatura sea más utilizada que la palabra que representa (URL, HTTP, etc.).

Las interfaces se nombrarán siguiendo los mismos criterios que los indicados para las clases. Como norma general toda interfaz se nombrará con el prefijo "I" para diferenciarla de la clase que la implementa (que tendrá el mismo nombre sin el prefijo "I").

Tabla 21.

Clases

Clases	Nombre
Clase Usuario	public class Paciente
Clase Paciente	public class Usuario

Métodos

Un método en Java es un conjunto de instrucciones definidas dentro de una clase, que realizan una determinada tarea y a las que podemos invocar mediante un nombre.

Tabla 22.

Métodos

Métodos	Estructura
Guardar	public void guardar(Hospitalizacion hospitalizacion)
Buscar	public List<Hospitalizacion> buscar(String stringBusqueda)

Variables

Las variables se escribirán siempre en minúsculas. Las variables compuestas tendrán la primera letra de cada palabra componente en mayúsculas.

Las variables nunca podrán comenzar con el carácter "_" o "$". Los nombres de variables deben ser cortos y sus significados tienen que expresar con suficiente claridad la función que desempeñan en el código. Debe evitarse el uso de nombres de variables con un sólo carácter, excepto para variables temporales.

•Unidad unidad;

Agenda agenda;

Constantes

Todos los nombres de constantes tendrán que escribirse en mayúsculas. Cuando los nombres de constantes sean compuestos las palabras se separarán entre sí mediante el carácter de subrayado "_".

private Pacientes paciente;

private String stringBusqueda;

5.03.03 Diseño de Interfaz

Cuadro de Objetos utilizados en el diseño de las interfaces

Tabla 23.

Cuadro de Objetos

Objeto	Nombre
OuputText	A
Button	B
CheckBox	C
Dialog	D
CommandButton	E
Image	F
InputText	G
Panel	H
Message	I
Calendar	J
SelectOneMenu	K
DataTable	L
Menu	M

Interfaces

Home

Home será la página principal que se mostrará al usuario después de iniciar sesión en la aplicación, donde se mostrará el nombre del usuario del sistema y la opción **salir** en la parte superior izquierda, en el costado derecho se mostrará el menú de opciones del sistema, este está divido por tres secciones Administración, Paciente y Reportes cada uno permitirá realizar diferentes funciones como su nombre lo indica.

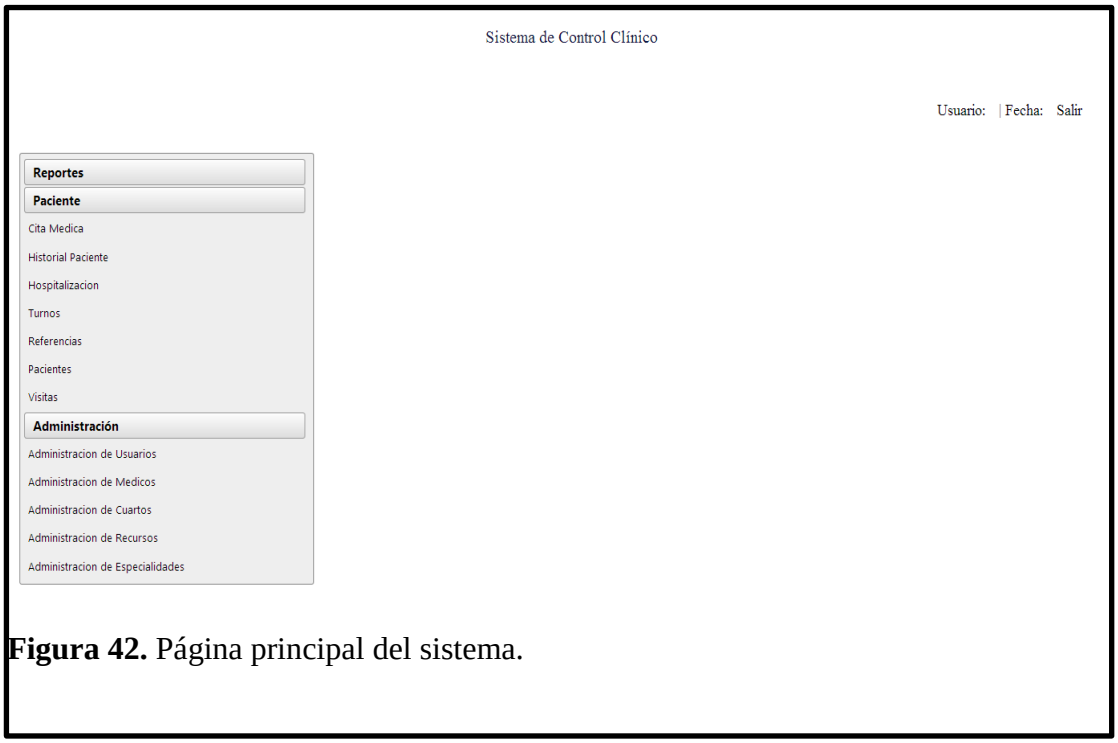


Figura 42. Página principal del sistema.

Administración

El menú administración permitirá a los usuarios de tipo administrador a realizar los diferentes registros que de las tablas generales, tales como usuario donde permitirá crear, modificar y eliminar registros del sistema de acuerdo a su uso y rol dentro del mismo.

Administración de Usuarios

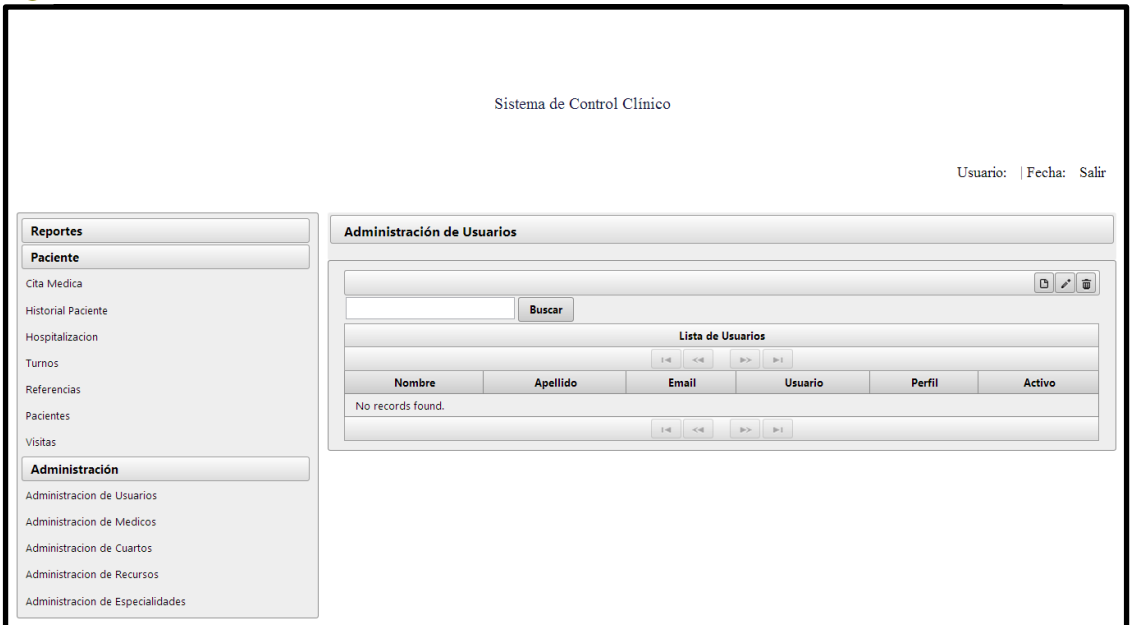


Figura 43. Página de administración de usuarios, permite ingresar editar y desactivar un usuario.

Administración de Recursos

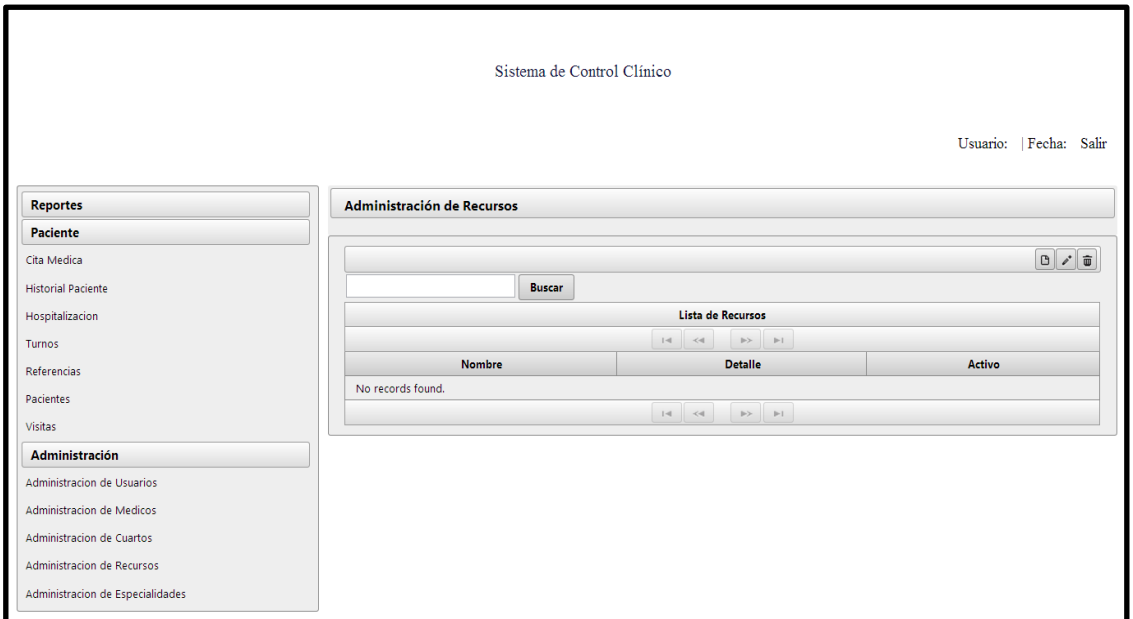


Figura 44. Ventana de administración de recursos médicos permitirá ingresar, modificar y eliminar registros de recursos y se mostrará en la Lista de Recursos.

Administración de Cuartos

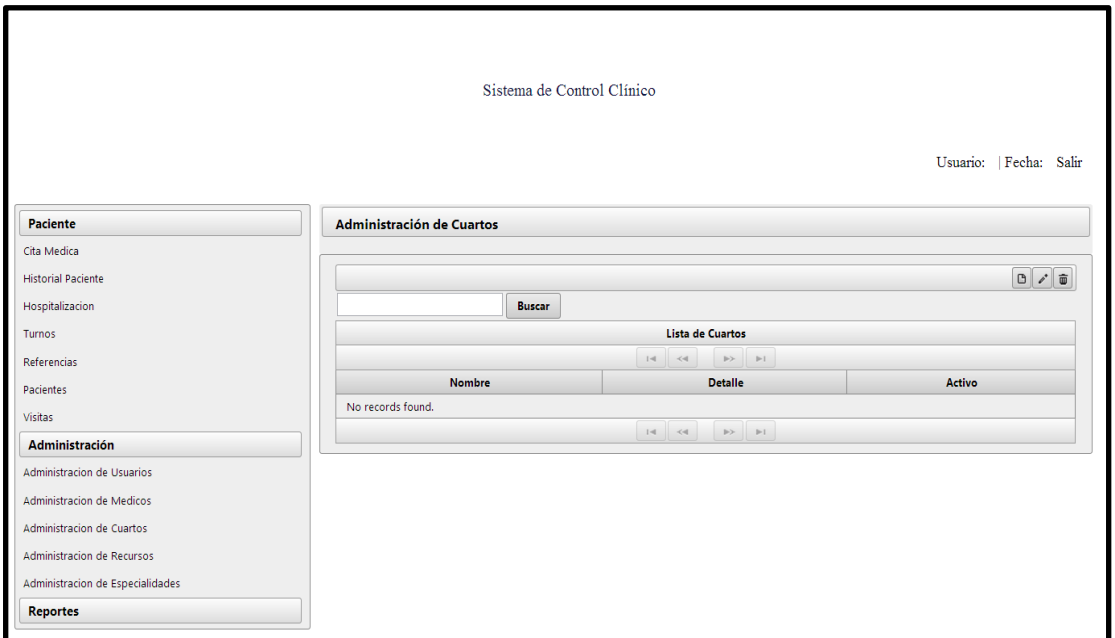


Figura 45. Ventana de administración de cuartos de pacientes donde se podrá administrar un cuarto y determinar su estado para que pueda ser utilizado.

Administración de Especialidades

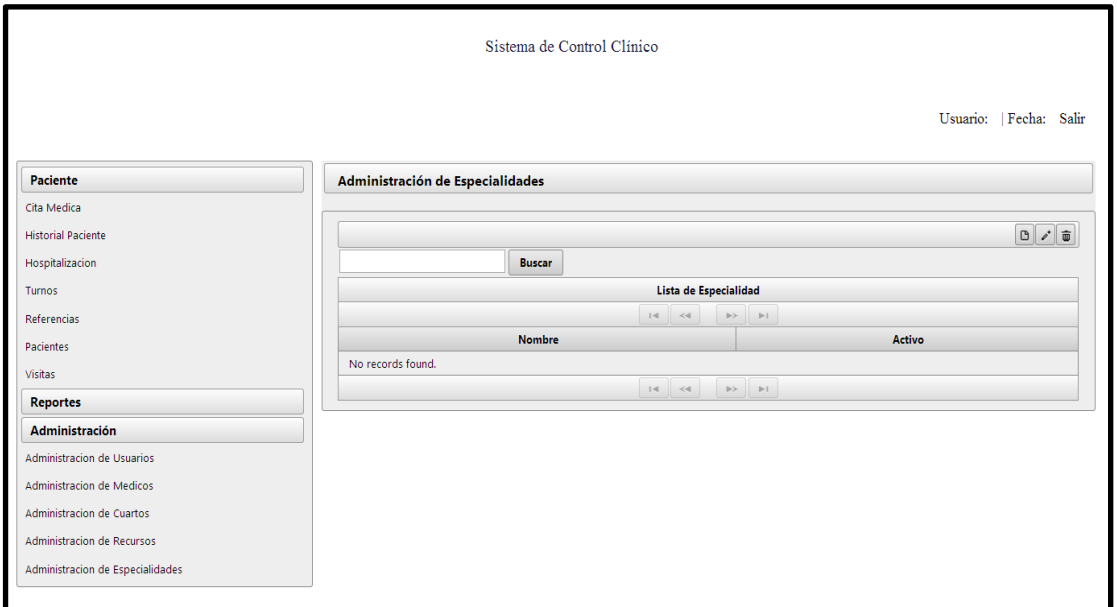


Figura 46. Interfaz de administración de Especialidades, según se vayan implementando nuevas áreas se podrá añadir al sistema.

Administración de Médicos

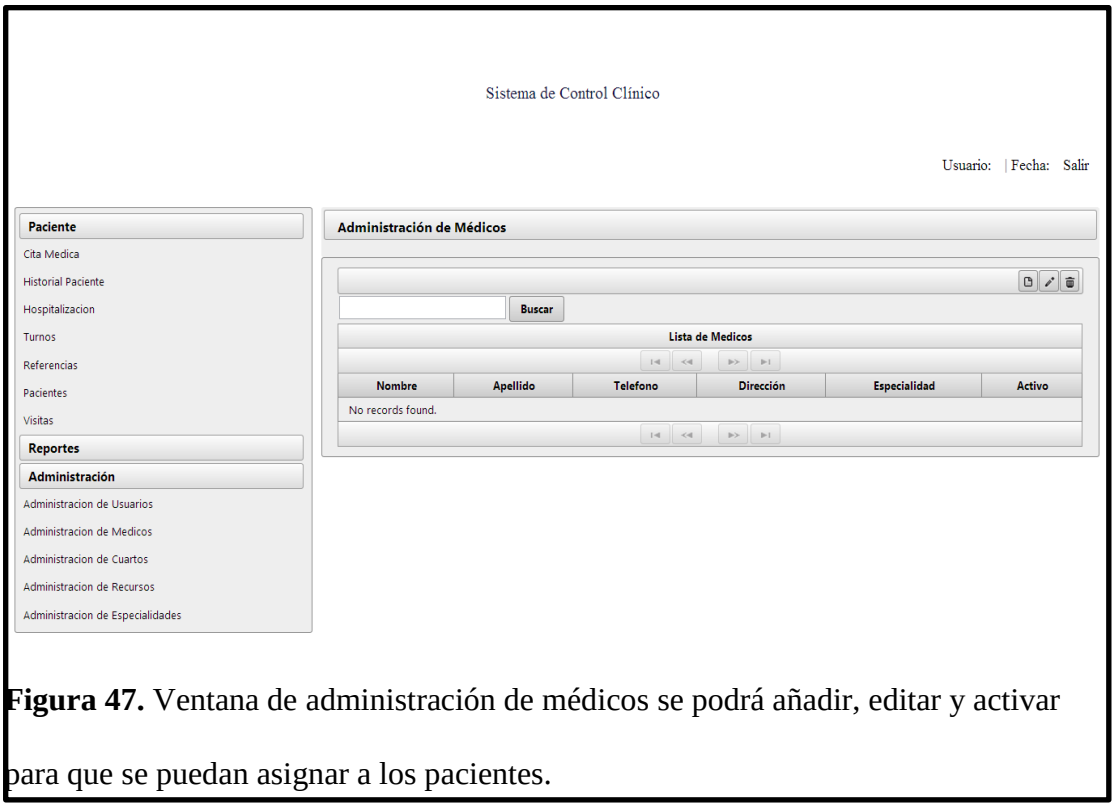


Figura 47. Ventana de administración de médicos se podrá añadir, editar y activar para que se puedan asignar a los pacientes.

Registro de Pacientes

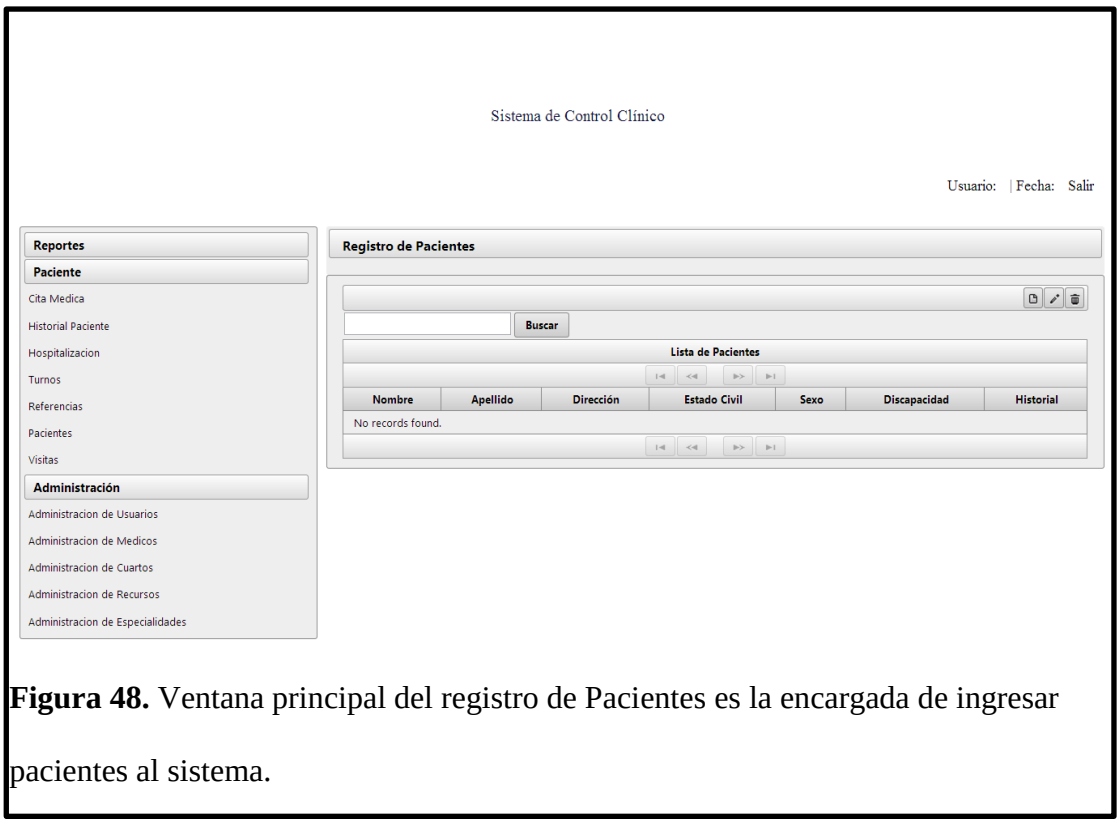


Figura 48. Ventana principal del registro de Pacientes es la encargada de ingresar pacientes al sistema.

Ventanas Emergentes

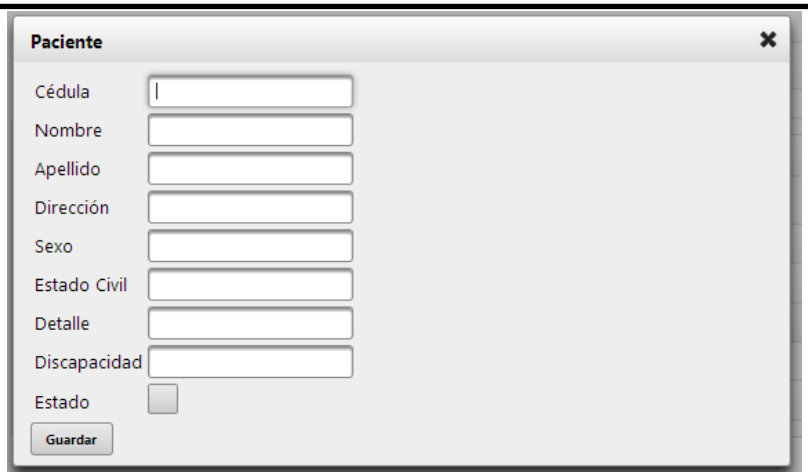


Figura 49. Ventana emergente del ingreso de pacientes permitirá ingresar datos en todos los campos.

Historial Clínico

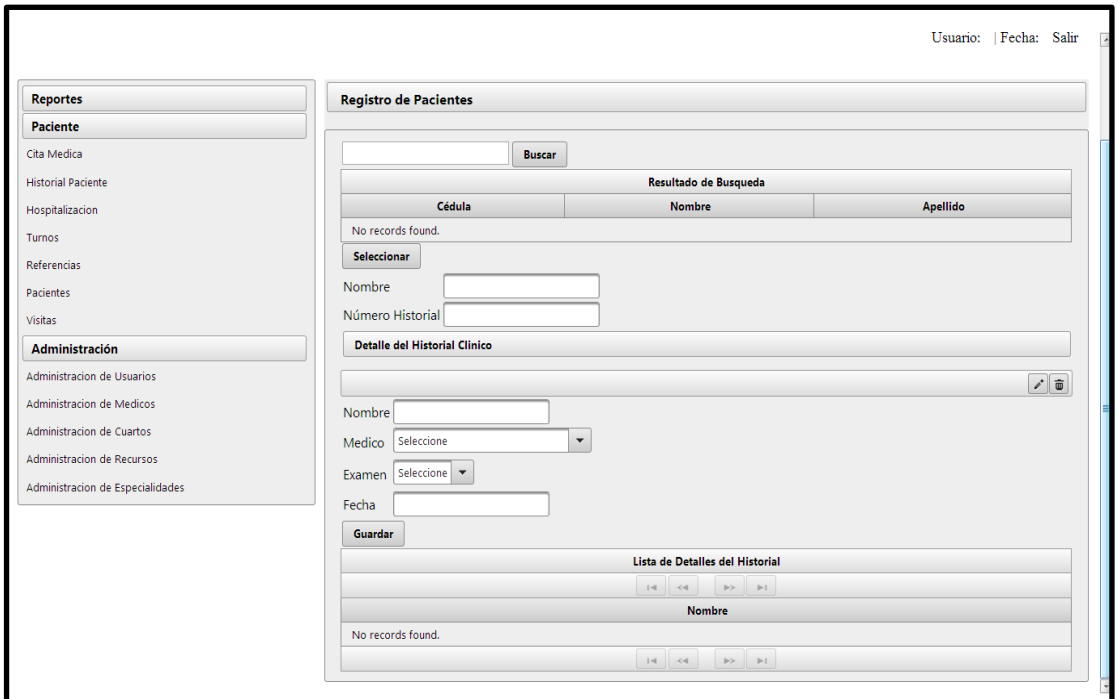


Figura 50. Ventana de registro del historial de pacientes, registra todas las atenciones que ha tenido el paciente donde se detallas las atenciones recibidas en la clínica.

Asignación de Turnos

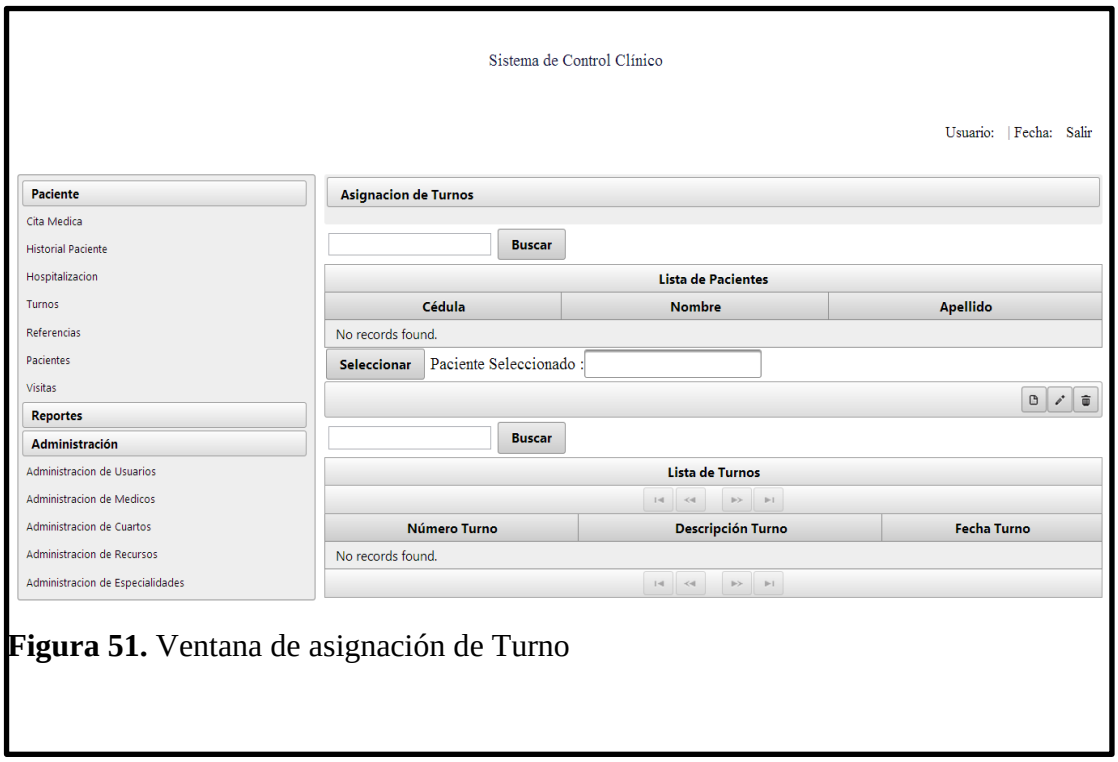


Figura 51. Ventana de asignación de Turno

Registro de Hospitalización

El registro consta de dos partes la cabecera donde constaran los nombres del paciente y los datos del ingreso del mismo.

Figura 52. Ventana para el registro de las hospitalizaciones.

Registro de Visitas

Figura 53. Ventana para el registro de las visitas.

5.04 Plan de Pruebas Maestro

Como tal nos ayuda a seguir y controlar el correcto funcionamiento del sistema, donde se tomaran algunos puntos de validación, verificación y persistencia.

5.04.01 Propósito

Describir el plan para probar las funcionalidades y características según los siguientes objetivos:

- Plan de integridad de Base de Datos
- Módulo de Mantenimiento
- Pruebas de Interfaz de Usuarios.
- Pruebas de Seguridad

- Pruebas del Sistema
- Pruebas de Desempeño
- Pruebas de Unidad

5.04.01.01 Plan de Integridad de Datos y Base de Datos

- Integridad de los datos
- Correcta obtención de datos
- Actualización
- Secuencia de códigos automáticos

5.04.01.02 Módulo de Mantenimiento

- Acceso idóneo a la base de datos.
- Validación de campos de acuerdo al tipo de datos.
- Generación de mensajes de error al persistir la base de datos

5.04.01.03 Pruebas de Interfaz de Usuario

Estas pruebas se las realiza inicialmente verificando como el usuario se desempeña en realizar las diferentes operaciones y funcionalidades en el sistema clínico, el tiempo de respuesta en la navegación en los diferentes browser, el tiempo de carga debe ser óptimo.

En los formularios verificamos que el salto de las cajas de texto se las realice con la tecla tabulación, igualmente se cumple con la verificación de los estándares GUI que implica color de fondo de las cajas de texto, color de texto de la recuperación de

datos simetría en la distribución de cajas recuperadoras de información, diseño de ubicación de objetos de interfaz de usuario (text box, check box etc).

Por otro lado se determinó una prueba de ejecución del manejo de los iconos inicialmente que estén acorde con la información solicitada; en este punto se verifico ventanas y mensajes de alerta, ventanas y mensajes de información de la acción que se ha ejecutado, ventana y mensajes de captura de errores con el manejo de excepciones.

5.04.01.04 Pruebas de Seguridad

El ingreso erróneo en cualquiera de estas dos cajas de captura de datos (usuario y clave) abortara el acceso al sistema y enviara a la página principal.

El usuario podrá ejecutar cualquier proceso dependiendo del rol que tenga dentro del sistema que le será asignado por el administrador del sistema al momento de crearlo.

Las validaciones de los campos se mostrara en caso de que sean campos obligatorios y en el caso de que necesiten otra validación ya sea de solo mayúsculas o solo números.

5.04.01.05 Pruebas del Sistema

- Verificar CU Asignación de Turno
- Verificar CU Registro de Datos
- Verificar CU Crear Cita
- Verificar CU Ficha Médica
- Verificar CU Asignar Ubicación del Paciente

- Verificar CU Atención al Paciente
- Verificar CU Asignación de Médico

Registro de Pacientes

Los pacientes deberán ser registrados en el sistema antes de realizarles el chequeo médico, el operador del sistema deberá ingresar todos los campos que sean obligatorios, una vez creado el registro del paciente se deberá generar un historial al mismo se mostrarán mensajes de confirmación una vez terminado cada proceso.

Asignación de Turnos

Para la asignación de turnos el paciente deberá estar registrado en el sistema y con esto se le realizara la asignación, en este módulo no existirán muchas validaciones ya que el operador deberá generar un número y otorgarlo al paciente.

Historial Clínico

En este proceso el operador deberá buscar al paciente mediante su número de cedula, se mostrará registros anteriores de las atenciones recibidas y se procederá a ingresar uno nuevo, deberá ingresar los campos obligatorios para su correcto registro y se actualizara la tabla de historias clínicas.

Hospitalización

El operador deberá buscar al paciente al cual se le va a realizar la hospitalización dentro del sistema, se generara una nueva hospitalización y se procederá a ingresar los las diferentes asignaciones que necesite para su atención.

Tabla 24.

Caso de Prueba Asignación de Turnos

CASOS DE PRUEBA				
Sistema Clínico				
Caso de Uso	**CU_01 Asignación de Turnos**			
Caso de Prueba	**CU_01 Asignación de Turnos**			
Actor	Recepcionista			
Pre Condiciones:	Ingresar en el sistema.			
Propósito:	Asignar un turno al paciente			
Escenario	**CP_01_E01: Entregar un turno para que sea chequeado el paciente**			
	Sec.	**Actividad**	**Clase de equivalencia**	**Resultado Esperado**
	1	Buscar al paciente por su número de cédula	Consulta	Visualiza los datos del paciente encontrado.
	2	Asigna un turno al paciente	Genera número	Muestra un numero de turno

Tabla 25.

Caso de Prueba Registrar Datos

CASOS DE PRUEBA				
Sistema Clínico				
Caso de Uso	**CU_02 Registrar Datos**			
Caso de Prueba	**CU_02 Registrar Datos**			
Actor	Recepcionista			
Pre Condiciones:	Ingresar en el sistema.			
Propósito:	Registrar los datos del paciente			
Escenario	**CP_02_E02: Ingresar los datos del paciente**			
	Sec.	**Actividad**	**Clase de equivalencia**	**Resultado Esperado**
	1	Ingresar los datos del paciente	Válida	Visualiza los datos del paciente encontrado.
	CP_02_E02: Generar un Numero de Historia Clínica			
	Sec.	**Actividad**	**Clase de equivalencia**	**Resultado Esperado**
	1	Generar un número de historial	Válida	Mensaje de generación

Tabla 26.

Casos de Prueba Crear Cita

CASOS DE PRUEBA				
Sistema Clínico				
Caso de Uso	**CU_03 Crear Cita**			
Caso de Prueba	**CU_03 Crear Cita**			
Actor	Recepcionista			
Pre Condiciones:	Ingresar al sistema, Tener registrado al paciente			
Propósito:	Ingresar al sistema			
Escenario	**CP_03_E03: Crear Cita**			
	Sec.	**Actividad**	**Clase de equivalencia**	**Resultado Esperado**
	1	Buscar al paciente	Consulta	Datos del paciente encontrados
	2	Ingresar los datos de la cita nueva	Válida	Ingresar los datos de la cita nueva

Tabla 27.

Casos de Prueba Ficha Médica

CASOS DE PRUEBA				
Sistema Clínico				
Caso de Uso	**CU_04 Ficha Médica**			
Caso de Prueba	**CU_04 Ficha Médica**			
Actor	Doctor			
Pre Condiciones:	Ingresar al sistema			
Propósito:	Registrar la una atención médica			
Escenario	**CP_04_E04: Comprobar el correcto Ingreso de Productos**			
	Sec.	**Actividad**	**Clase de equivalencia**	**Resultado Esperado**
	1	Buscar al paciente en el sistema	Consulta	Visualiza los datos del paciente
	2	Registrar el nuevo detalle	Valida	Guardado exitoso
Escenario	**CP_04_E04: Modificar el registro de la ficha medica**			
	Sec.	**Actividad**	**Clase de equivalencia**	**Resultado Esperado**
	1	Buscar el registro	Consulta	Datos a modificar

Tabla 28.

Asignar Ubicación del Paciente

CASOS DE PRUEBA				
Sistema Clínico				
Caso de Uso	**CU_05 Asignar Ubicación del Paciente**			
Caso de Prueba	**CU_05 Asignar Ubicación del Paciente**			
Actor	Director Clínico			
Pre Condiciones:	Ingresar en el sistema			
Propósito:	Ubicar al paciente en un cuarto y asignarle los recursos necesarios			
Escenario	**CP_05_E05: Asignar cuarto al paciente**			
	Sec.	**Actividad**	**Clase de equivalencia**	**Resultado Esperado**
	1	Buscar al paciente que se va asignar un cuarto	Valida	Visualiza los datos del paciente
	2	Verificar estado de cuartos	Consulta	Muestra lista de cuartos
Escenario	3	Registra asignación de cuarto al paciente	Registro	Guardado exitoso
	CP_05_E05: Asignar recursos			
	Sec.	**Actividad**	**Clase de equivalencia**	**Resultado Esperado**
	1	Nuevo recurso a asignar	Valida	Muestra lista de cuartos
	2	Registra asignación de recurso	Registro	Guardado exitoso

5.04.01.06 Pruebas de Desempeño

- Verificar el Tiempo en generar los registros del paciente
- Verificar el tiempo en generar un turno.
- Verificar cada mantenimiento realizado.
- Verificar el tiempo en ingresar al sistema.

5.04.01.07 Pruebas de Unidad

Realizar pruebas de unidad antes de realizar las funcionalidades del sistema, estas deberán ejecutarse de acuerdo a la técnica del Desarrollo Guiado por Pruebas TDD, que consiste en pruebas de unidad y refactorizar. En primer lugar, se escribe una prueba y esta debe fallar, se procede implementando el código que origina que la prueba pase satisfactoriamente y se culmina refactorizando el código creado.

Capítulo VI: Aspectos Administrativos

El siguiente capítulo, detalla los diferentes recursos utilizados para generar el proyecto final de grado, los materiales empleados, el costo que conllevo a utilizar los mismos, he indica la planificación del tiempo que se utilizó para desarrollar los diferentes capítulos e implementar el sistema.

6.01 Recursos

La descripción de todos los materiales utilizados durante la ejecución del proyecto, así como el personal que participo durante el mismo.

Se debe tener en cuenta cinco tipos de recursos:

- Los insumos (materiales, energía y servicios);
- Las instalaciones y equipo (inversiones de capital);
- El personal de la Clínica;
- La información;
- El dinero.

En la implementación se utilizó los siguientes recursos.

6.1.1. Personales

- **Tutor**

Ing. Richard Mafla

- Médicos y personal que labora en la clínica.

6.1.2. Bienes

Son todos los bienes utilizados para el desarrollo del sistema los cuales son valorados de la siguiente manera:

Tabla 29.

Bienes

DESCRIPCION	CANTIDAD	P. UNIT ($/.)	TOTAL
Portátil Toshiba	1	650.00	650.00
Millar de Papel A4 de 80 gr.	1 Millar	25.00	25.00
Cartucho N°23 para impresora	1 Cartucho de Tinta Negra	65.00	65.00
Útiles de Escritorio: Folders, etc.	Global	15.00	15.00
Otros	Global	50.00	50.00
	TOTAL	805.00	805.00

6.1.3. Servicios

Son todos los servicios forman parte de los servicios utilizados indirectamente para la ejecución del proyecto.

Tabla 30.

Servicios

DESCRIPCION	CANTIDAD	P. UNIT ($/.)	TOTAL
Internet	1024kb	20.00	2000.00
Luz	150 kxh	0.30	45.00
Transporte	10 viajes	5.00	50.00
Fotocopias	400 hojas	0.05	10.00
Teléfono	Global	35.00	35.00
Otros	Global	50.00	50.00
	TOTAL	110.35	2300.00

Los siguientes recursos son los que forman parte directa del proyecto ya que constan de los diferentes software's que se utilizaron.

Tabla 31.

Recursos 2

DESCRIPCION	CANTIDAD	P. UNIT ($/.)	TOTAL
Ubuntu 12.04	1 Unidad	00.00	00.00
Software	6	00.00	00.00
	TOTAL	00.00	00.00

6.02 Presupuesto

Es el capital previsto para la ejecución del proyecto.

Tabla 32.

Presupuesto

DESCRIPCION	DISPONIBLES	NO DISPONIBLES	IMPORTE ($/.)
Bienes	1010.00	--	1010.00
Servicios	115.00	170.00	285.00
TOTAL	1125.00	170.00	1295.00

6.03 Cronograma

El diseño fue realizado en Microsoft Vicio 2010 en el cual se utilizó el Diagrama de Gantt como plantilla para la elaboración de la programación de la implementación del proyecto indicando los tiempos y la actividad realizada en un periodo de tiempo determinado.

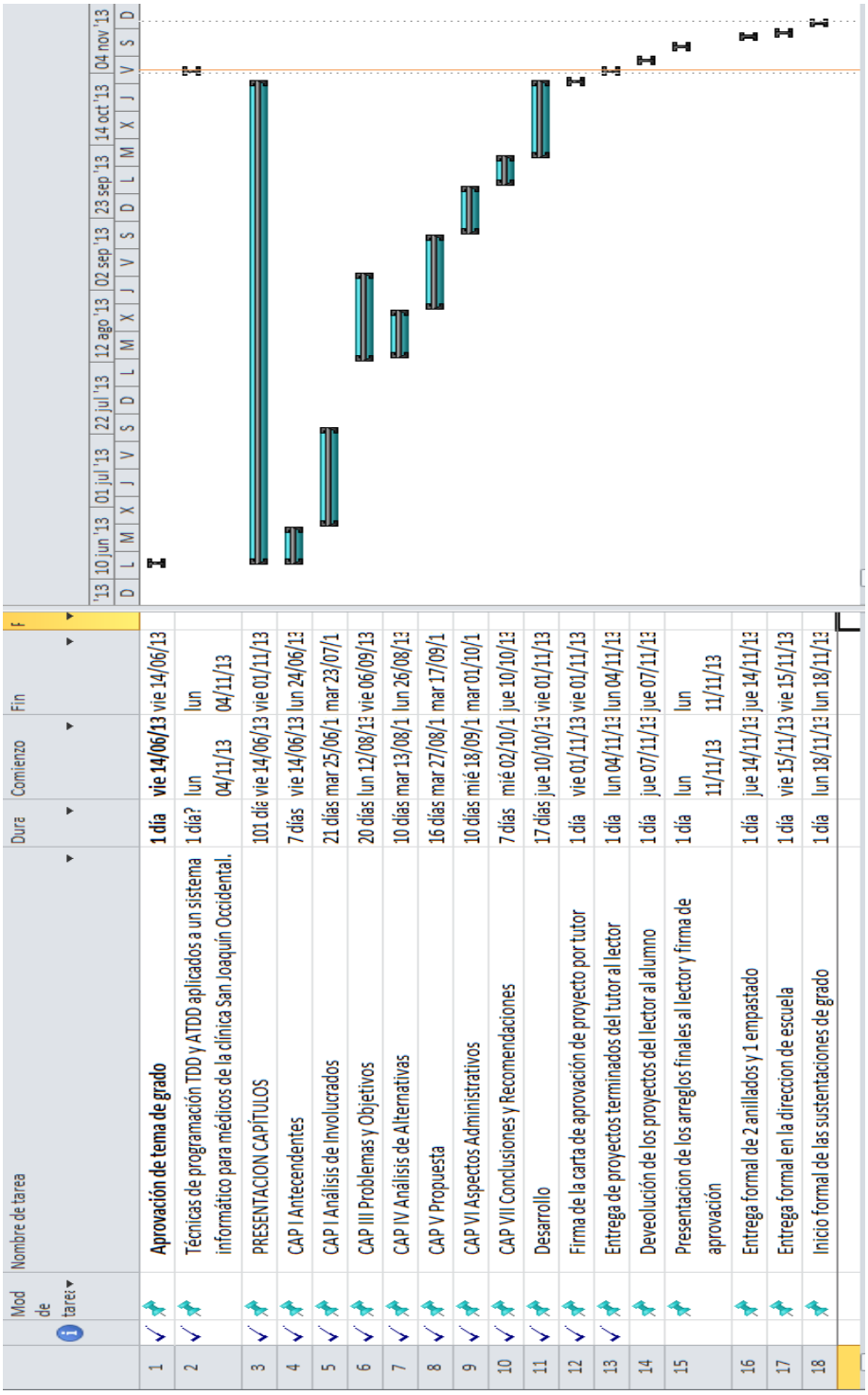


Figura 54. Cronograma para el seguimiento del desarrollo y estado del proyecto.

Capítulo VII: Conclusiones Y Recomendaciones

7.01 Conclusiones

Para concluir este trabajo de tesis, este capítulo se dedicará a mostrar las conclusiones y recomendaciones obtenidas a lo largo del trabajo en este proyecto. Lo anterior será con el fin de que se le pueda dar continuidad al proyecto, así como mostrar los beneficios obtenidos.

El Sistema Clínico se terminó con éxito y se hacen las siguientes conclusiones:

- Con la futura implementación del sistema se obtendrá un mayor control sobre los pacientes.
- La aplicación de Registro de Historias Clínicas agilizara y permitirá un mejor control de sus procesos.
- La asignación de turnos automatizado permitirá un mejor control sobre la atención y será oportuna.
- Controlar los recursos por medio del sistema permitirá asignar los materiales necesarios a cada paciente.
- Realizar las Pruebas de Unidad ayuda a que la funcionalidad del código creado sea eficiente.
- El reautorizar ayuda a que se realice menos código y con la misma funcionalidad.
- Mediante la técnica de programación TDD se centra en el hecho de que se realizan las pruebas antes que programar funcionalidades, siguiendo el falla, pasa y refactoriza.

7.02 Recomendaciones

Mediante la investigación para la realización del proyecto se realizan las siguientes recomendaciones:

- Se deberá realizar una capacitación al personal para que pueda realizar los registros en el sistema,
- Para tener un mayor control sobre los pacientes y sus enfermedades realizar los registros por cada atención.
- Para llevar a cabo una atención oportuna se recomienda realizar la asignación de turnos acorde a los síntomas del paciente.
- Verificar el estado de los recursos ayudara a que estos sean utilizados con orden y cuidado.
- Para una mejor creación de código fuente es necesario realizar pruebas de unidad.
- Refactorizar el código ayudándose de las pruebas de unidad.
- Se recomienda realizar las pruebas de unidad antes de las funcionalidades para que se vaya creando lo que se va necesitando mediante las mismas.

Anexos (Apéndices)

Bibliografía

-Carlos Blé Jurado y colaboradores (2010). Prólogo de José Manuel Beas Diseño Ágil con TDD

-Lic. Salvador Olivares José, Desarrollo de aplicaciones con PrimeFaces.

-Jendrock Eric, Evans Ian, Gollapudi Devika, Haase Kim, The Java EE 6 Tutorial.

-Flower Martin , Beck Kent Brant John Opdike Willian (2002), Refactoring: Improving the Desing of Existing Code.

Web grafía

http://www.ubuntu.com/

http://www.postgresql.org.es/

http://hk9888.wordpress.com/category/jboss-developer-studio/

http://www.primefaces.org/

http://es.wikipedia.org/wiki/Maven

http://es.wikipedia.org/wiki/JUnit

http://www.suarezdefigueroa.es/manuel/IAW/Java/teoriajsf.html

http://www.comscigate.com/cs/valle/cjava.htm Enrique Serrano Valle, Creative Commons Attribution 2.0.

http://es.wikipedia.org/wiki/Data_Access_Object

MANUAL TÉCNICO

1. Introducción

La finalidad del manual técnico es la de proporcionar al lector la lógica con la que se ha desarrollado una aplicación, la cual se sabe que es propia de cada programador, por lo considero ser documentada.

El presente no pretende ser un manual de aprendizaje, sino documentar su aplicación en el desarrollo del sistema clínico.

2. Objetivo

Proporcionar una guía para el lector, del desarrollo de la aplicación de Sistema de control…

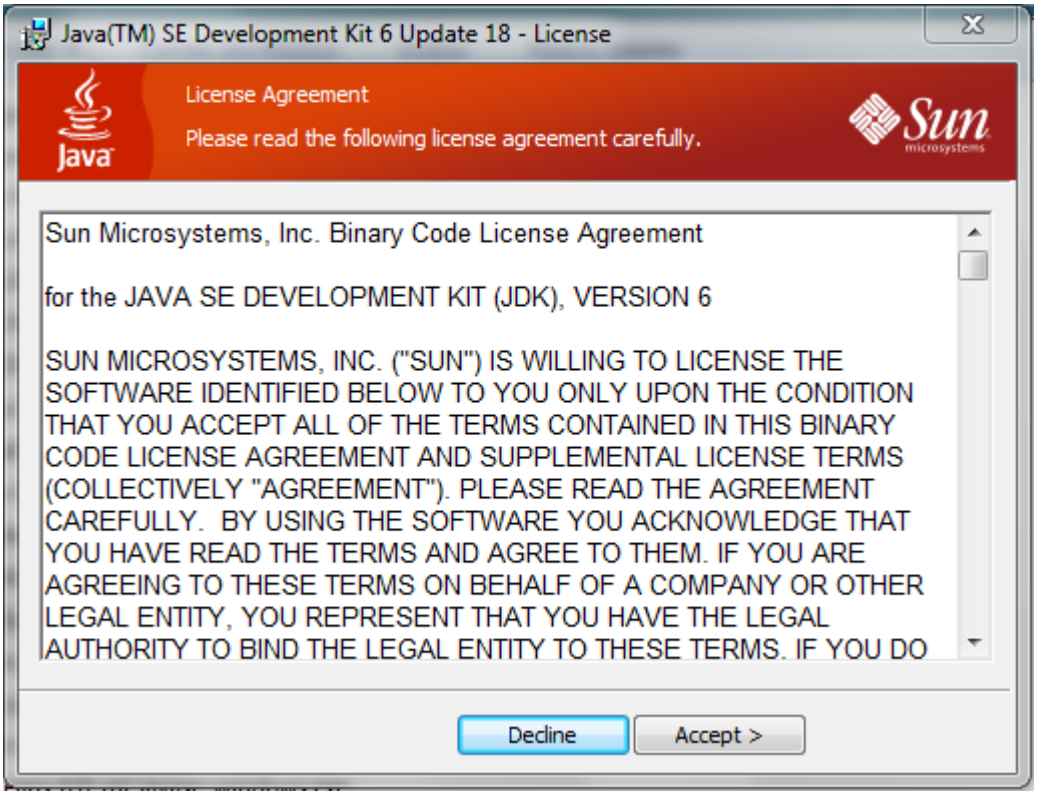
3. Contenido

El proyecto fue realizado en java utilizando el IDE de java Jboss Developer Studio 7.0.0.GA, el servidor de aplicaciones jboss-as-7.1.1.Final, el repositorio apache-maven-3.0.5, el Java Development Kit o (JDK) jdk1.6.0_23 y el motor de base de datos PostgreSQL 9.3.0, la configuración para poder utilizar todas aplicaciones y servicios se las va a especificar en el manual.

3.1. Instalación de JDK y Configuración de JAVA

Instalación de jdk.

En la instalación del jdk que vamos a utilizar la versión 6.1.



Paso Nº 1

Descargar jdk cualquier versión (en este caso se utilizó la versión 6.1.0)

Paso Nº 2

Comenzar la instalación, ubicar la carpeta en /urs/lib/jvm/jdk/, en el caso de que no exista crear esta dirección.

Configuración del java

Tenemos que registrar la siguiente variable:

JAVA_HOME

Vamos a utilizar el archivo .bash que se encuentra ubicado en el directorio /home/usuario/ en el caso de no encontrar el archivo utilizar el atajo ctrl+h, que nos permite ver los archivos ocultos.

Ubicar en la parte final de todo el testo lo siguiente:

```
JAVA_HOME=/usr/lib/jvm/jdk1.6.0_26/
CATALINA_HOME=/usr/share/tomcat6; export CATALINA_HOME
export JAVA_HOME
export PATH=$PATH:$JAVA_HOME/bin
```

Abrir una consola y escribir lo siguiente

java -version

Configuración del MAVEN

Descargar el apache-maven cualquier versión en este caso se utilizó la versión apache-maven-3.0.5

Descomprimir apache-maven.gzip y ubicar la carpeta de descomprimida en la siguiente dirección /home/usuario/java/maven.

Añadir lo siguiente al archivo .bash

```
M2_HOME=/home/usuario/java/maven/apache-maven-3.0.5
export M2_HOME
export PATH=$PATH:$M2_HOME/bin
```

Instalación de PostgrSQL

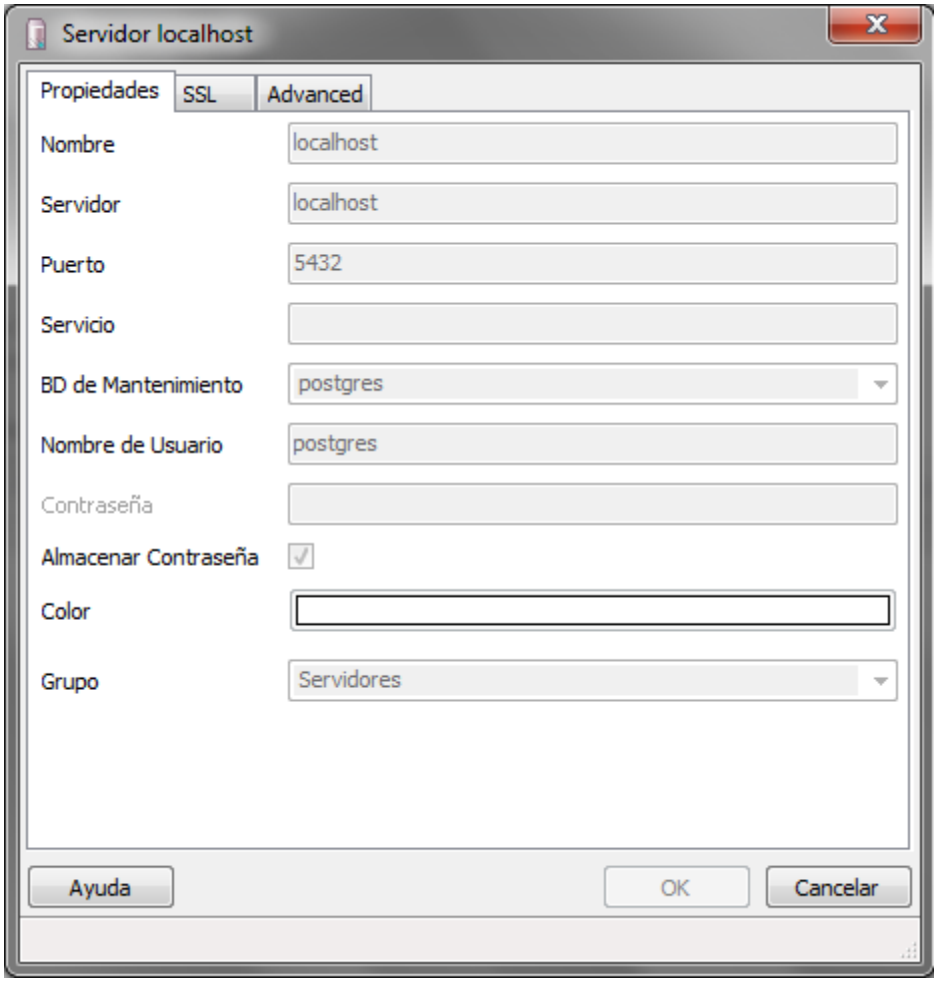
Para iniciar la instalación del PostgreSQL, abrimos una terminal y ejecutamos el siguiente comando:

```
sudo apt-get install postgresql
```

La versión a instalar se instalara de acuerdo a la que se encuentre disponible. En este caso se utilizó la 9.4.

Una vez terminado la instalación, se debe configurar el servidor PostgreSQL en base a nuestras necesidades, en este caso se usó la que se realiza por defecto.

Procedemos abrir el PostgreSQL y vamos a crear un servidor local para nuestra aplicación en la cual vamos a crear nuestra base de datos.



Para este caso utilizaremos un localhost y el puerto 5432 y de usuario postgres.

Script de Base de datos

```
CREATE SCHEMA clinica;

CREATE SEQUENCE seq_cuarto_paciente
   START WITH 1
   INCREMENT BY 1
   NO MINVALUE
   NO MAXVALUE
   CACHE 1;
```

```
ALTER TABLE public.seq_cuarto_paciente OWNER TO postgres;

SET default_tablespace = '';

SET default_with_oids = false;

CREATE TABLE cuarto_paciente (
    codigo_cuarto_paciente integer DEFAULT
nextval('seq_cuarto_paciente'::regclass) NOT NULL,
    descripcion_cuarto_paciente character varying(500),
    estado_cuarto_paciente character varying(255),
    activo boolean
);


ALTER TABLE public.cuarto_paciente OWNER TO postgres;

CREATE SEQUENCE seq_detalle_historia
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public.seq_detalle_historia OWNER TO postgres;

CREATE TABLE detalle_historia (
    codigo_detalle_historial integer DEFAULT
nextval('seq_detalle_historia'::regclass) NOT NULL,
    codigo_examen integer,
    codigo_enfermedad integer,
    codigo_historial_paciente integer,
    codigo_medico integer,
    detalle character varying(300),
    fecha_detalle_historia date,
    codigo_hospitalizacion integer
);

ALTER TABLE public.detalle_historia OWNER TO postgres;

CREATE SEQUENCE seq_detalle_hospitalizacion
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_detalle_hospitalizacion OWNER TO postgres;
```

```
CREATE TABLE detalle_hospitalizacion (
    codigo_detalle_hospitalizacion integer DEFAULT
nextval('seq_detalle_hospitalizacion'::regclass) NOT NULL,
    codigo_recurso integer,
    codigo_hospitalizacion integer,
    codigo_visita integer,
    fecha_entrada_hospitalizacion date,
    fecha_salida_hospitalizacion date
);


ALTER TABLE public.detalle_hospitalizacion OWNER TO postgres;

CREATE SEQUENCE seq_empresa
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_empresa OWNER TO postgres;

CREATE TABLE empresa (
    codigo_empresa integer DEFAULT nextval('seq_empresa'::regclass) NOT NULL,
    nombre_empresa character varying(300),
    direccion_empresa character varying(300),
    telefono_empresa numeric(10,0),
    activo boolean
);


ALTER TABLE public.empresa OWNER TO postgres;

CREATE SEQUENCE seq_enfermedad
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_enfermedad OWNER TO postgres;

CREATE TABLE enfermedad (
    codigo_enfermedad integer DEFAULT nextval('seq_enfermedad'::regclass) NOT
NULL,
    nombre_enfermedad character varying(500),
```

```
    detalle_enfermedad character varying(500),
    codigo_tratamiento integer
);


ALTER TABLE public.enfermedad OWNER TO postgres;

CREATE SEQUENCE seq_especialidad
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_especialidad OWNER TO postgres;

CREATE TABLE especialidad (
    codigo_especialidad integer DEFAULT nextval('seq_especialidad'::regclass) NOT
NULL,
    nombre_especialidad character varying(500),
    activo boolean
);


ALTER TABLE public.especialidad OWNER TO postgres;

CREATE SEQUENCE seq_examen
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_examen OWNER TO postgres;

CREATE TABLE examen (
    codigo_examen integer DEFAULT nextval('seq_examen'::regclass) NOT NULL,
    nombre_examen character varying(500),
    descripcion_examen character varying(500),
    resultado_examen character varying,
    documento_examen character varying(500)
);


ALTER TABLE public.examen OWNER TO postgres;

COMMENT ON TABLE examen IS 'Examen que se realice el paciente';
```

```
CREATE SEQUENCE seq_historial_paciente
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_historial_paciente OWNER TO postgres;

CREATE TABLE historial_paciente (
    codigo_historial_paciente integer DEFAULT
nextval('seq_historial_paciente'::regclass) NOT NULL,
    codigo_paciente integer
);


ALTER TABLE public.historial_paciente OWNER TO postgres;

CREATE SEQUENCE seq_hospitalizacion
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_hospitalizacion OWNER TO postgres;

CREATE TABLE hospitalizacion (
    codigo_hospitalizacion integer DEFAULT nextval('seq_hospitalizacion'::regclass)
NOT NULL,
    codigo_detalle_historial integer,
    detalle_hospitalizacion character varying(500),
    hora_entrada date,
    codigo_cuarto_paciente integer
);


ALTER TABLE public.hospitalizacion OWNER TO postgres;

CREATE SEQUENCE seq_medico
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
```

```
ALTER TABLE public.seq_medico OWNER TO postgres;

CREATE TABLE medico (
    codigo_medico integer DEFAULT nextval('seq_medico'::regclass) NOT NULL,
    codigo_especialidad integer,
    nombre_medico character varying(500),
    apellido_medico character varying(500),
    direccion_medico character varying(500),
    activo boolean,
    telefono_medico character varying(10)
);


ALTER TABLE public.medico OWNER TO postgres;

CREATE SEQUENCE seq_opcion
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_opcion OWNER TO postgres;

CREATE TABLE opcion (
    codigo_opcion integer DEFAULT nextval('seq_opcion'::regclass) NOT NULL,
    nombre character varying(255),
    url character varying(255),
    tipo character varying(255),
    activo boolean
);


ALTER TABLE public.opcion OWNER TO postgres;

CREATE SEQUENCE seq_opcion_rol
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_opcion_rol OWNER TO postgres;

CREATE TABLE opcion_rol (
    codigo_opcion_rol integer DEFAULT nextval('seq_opcion_rol'::regclass) NOT
NULL,
    codigo_opcion integer,
```

```
    codigo_rol integer,
    activo boolean
);


ALTER TABLE public.opcion_rol OWNER TO postgres;

CREATE SEQUENCE seq_paciente
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_paciente OWNER TO postgres;

CREATE TABLE paciente (
    codigo_paciente integer DEFAULT nextval('seq_paciente'::regclass) NOT NULL,
    codigo_historial_paciente integer DEFAULT
nextval('seq_historial_paciente'::regclass),
    codigo_referencia integer,
    nombre_paciente character varying(500),
    apellido_paciente character varying(500),
    telefono_paciente numeric(10,0),
    direccion_paciente character varying(500),
    est_paciente character(1),
    cedula_paciente character varying(10),
    discapacidad_paciente character varying(25),
    estado_civil character varying(50),
    residencia_paciente character varying(255),
    sexo_paciente character varying(2),
    activo_paciente boolean
);


ALTER TABLE public.paciente OWNER TO postgres;

CREATE SEQUENCE seq_recursos
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_recursos OWNER TO postgres;

CREATE TABLE recursos (
    codigo_recurso integer DEFAULT nextval('seq_recursos'::regclass) NOT NULL,
```

```
    nombre_recurso character varying(300),
    detalle_recurso character varying(300),
    activo boolean
);


ALTER TABLE public.recursos OWNER TO postgres;

CREATE SEQUENCE seq_referencia
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_referencia OWNER TO postgres;

CREATE TABLE referencia (
    codigo_referencia integer DEFAULT nextval('seq_referencia'::regclass) NOT
NULL,
    nombre_referencia character varying(500),
    apellido_referencia character varying(500),
    telefono_referencia numeric(10,0),
    direccion_referencia character varying(500)
);


ALTER TABLE public.referencia OWNER TO postgres;

CREATE SEQUENCE seq_rol
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_rol OWNER TO postgres;

CREATE TABLE rol (
    codigo_rol integer DEFAULT nextval('seq_rol'::regclass) NOT NULL,
    descripcion character varying(255),
    nombre character varying(255),
    activo boolean
);


ALTER TABLE public.rol OWNER TO postgres;
```

```
CREATE SEQUENCE seq_cita_medica
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_cita_medica OWNER TO postgres;

--
-- TOC entry 201 (class 1259 OID 77194)
-- Name: seq_resultado; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE seq_resultado
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_resultado OWNER TO postgres;

CREATE SEQUENCE seq_tipo_tratamiento
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_tipo_tratamiento OWNER TO postgres;

CREATE SEQUENCE seq_tratamiento
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_tratamiento OWNER TO postgres;

CREATE SEQUENCE seq_turno
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
```

```
    CACHE 1;


ALTER TABLE public.seq_turno OWNER TO postgres;

CREATE SEQUENCE seq_usuario
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_usuario OWNER TO postgres;

CREATE SEQUENCE seq_visita
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.seq_visita OWNER TO postgres;

CREATE TABLE tratamiento (
    codigo_tratamiento integer DEFAULT nextval('seq_tratamiento'::regclass) NOT
NULL,
    descripcion_tratamiento character varying(500)
);


ALTER TABLE public.tratamiento OWNER TO postgres;

CREATE TABLE turno (
    codigo_turno integer DEFAULT nextval('seq_visita'::regclass) NOT NULL,
    codigo_paciente integer,
    numero_turno numeric(8,0),
    descripcion_turno character varying(500),
    fecha_turno date
);


ALTER TABLE public.turno OWNER TO postgres;

CREATE TABLE usuario (
    codigo_usuario integer DEFAULT nextval('seq_usuario'::regclass) NOT NULL,
    codigo_rol character(10),
    nombre_usuario character varying(500),
    apellido_usuario character varying(500),
```

```
    correo_usuario character varying(500),
    user_usuario character varying(500),
    pass_usuario character varying(500),
    activo boolean
);

ALTER TABLE public.usuario OWNER TO postgres;

COMMENT ON TABLE usuario IS 'Usuarios del sistema';

CREATE TABLE visitas (
    codigo_visita integer DEFAULT nextval('seq_visita'::regclass) NOT NULL,
    fecha_visita date,
    hora_visita time without time zone,
    ced_visita numeric(10,0),
    nombre_visita character varying(500),
    detalle_visita character varying(255)
);
```

Instalación del servidor jboss-as-7.1.1.final

Debemos descargar el servidor jboss cualquier versión (en este caso se utilizó la versión jboss-as-7.1.1.final).

Se debe ubicar la carpeta que descargad y ubicar en la dirección /home/usuario/java/SERVERS/

Para poder iniciar el servidor de aplicaciones debemos abrir una terminar y escribimos /home/usuario/java/SERVERS/jboss-as-7.1.1.final/bin/ y escribimos ./standalone.sh, con esto el servidor se levanta en el caso de terminar el proceso es ctrl+c.

Para poder conectarnos a la base de datos debemos configurar el archivo standalone.xml que se encuentra dentro de la carpeta /jboss-as-7.1.1.final/standalone/configuración/

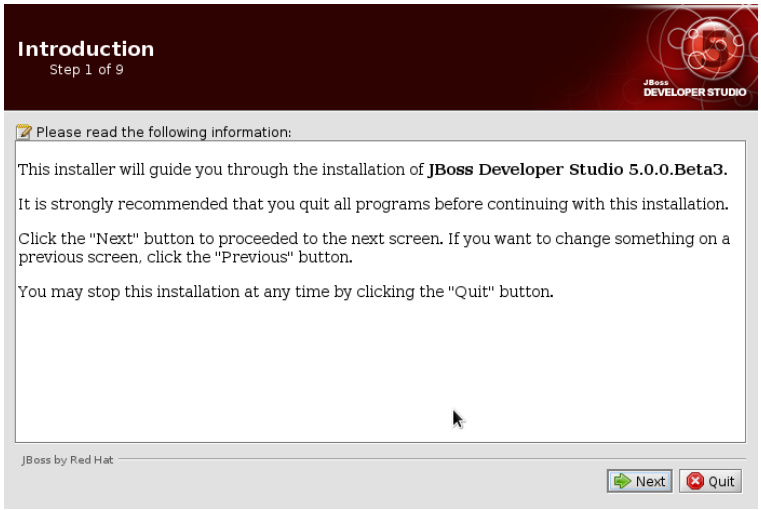
Debemos ubicar lo siguiente en la parte del datasourse:

```
        <datasource jta="false" jndi-name="java:jboss/datasources/ClinicaDS"
pool-name="ClinicaDS" enabled="true" use-ccm="false">
            <connection-url>jdbc:postgresql://localhost:5432/clinica</connection-
url>
            <driver-class>org.postgresql.Driver</driver-class>
            <driver>org.postgres</driver>
            <security>
               <user-name>postgres</user-name>
               <password>postgres</password>
            </security>
            <validation>
               <validate-on-match>false</validate-on-match>
               <background-validation>false</background-validation>
            </validation>
            <statement>
               <share-prepared-statements>false</share-prepared-statements>
            </statement>
         </datasource>
         <drivers>
            <driver name="org.postgres" module="org.postgres"/>
         </drivers>
```

Con esto le indicamos la dirección del driver del postgres y le indicamos la dirección de la base de datos, el nombre y las credenciales con los que este cuenta levantamos nuevamente el servidor con esto nos indica que el servidor se conectó a la base de datos.

```
13:43:48,961 INFO  [org.jboss.as.remoting] (MSC service thread 1-1) JBAS017100: Listening on /127.0.0.1:9999
13:43:48,961 INFO  [org.jboss.as.remoting] (MSC service thread 1-5) JBAS017100: Listening on /127.0.0.1:4447
13:43:50,017 INFO  [org.jboss.as.connector.subsystems.datasources] (MSC service thread 1-5) JBAS010400: Bound data source [java:jboss/d
atasources/ClinicaDS]
13:43:50,364 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015951: Admin console listening on http://127.0.0.1:9990
13:43:50,365 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015874: JBoss AS 7.1.1.Final "Brontes" started in 34567ms - Started 133
of 208 services (74 services are passive or on-demand)
```

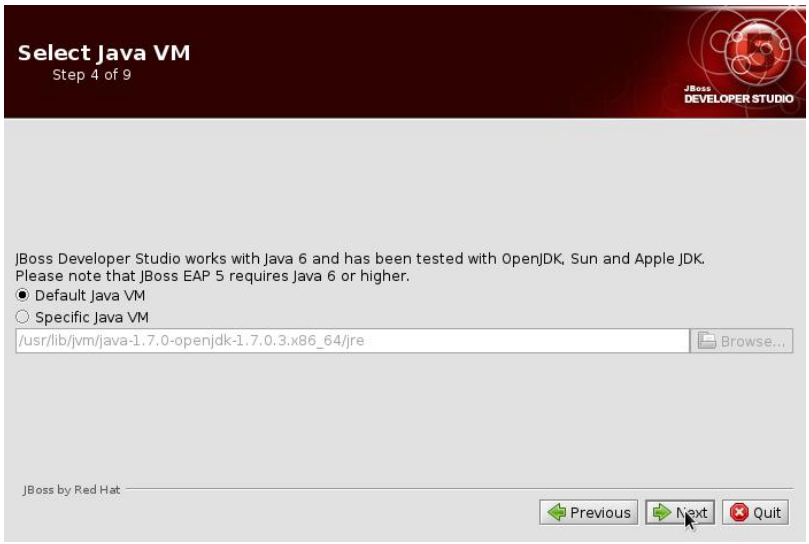
Instalación del Jboss Developer Studio



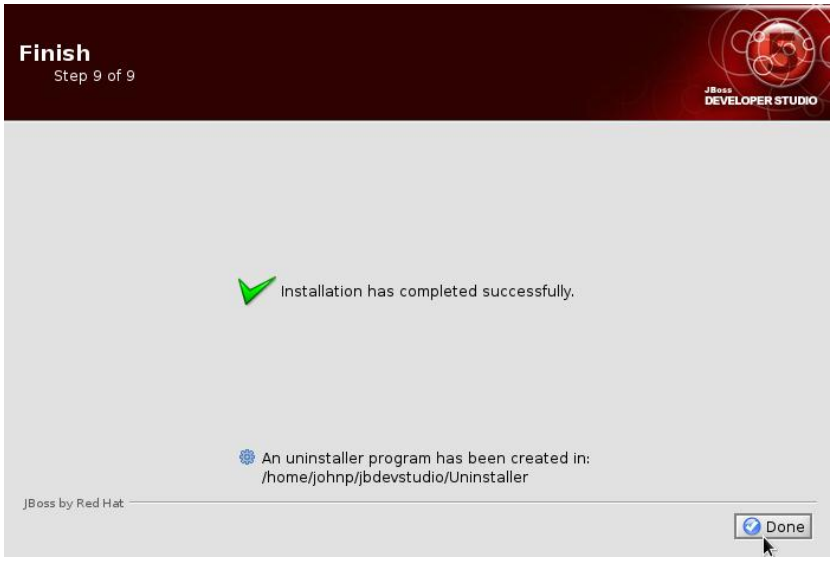
Debemos escoger la dirección donde vamos a realizar la instalación.

En nuestro caso lo vamos a relizar en el directorio /home/usuario/java/

La instalación del jboss se la debe realizar escogiendo la versión del jdk está la realizamos ubicando el directorio /urs/lib/jvm/jdk, la demás configuración de la instalación la realizamos por defecto.



Fin de la instalación



3.2. Proyecto MAVEN Historial

Para poder realizar el sistema se utiliza una pantilla del repositorio maven, ya que nos ayuda, mediante una plantilla a tener la estructura correcta del proyecto.

Abrimos una terminal y nos ubicamos en la siguiente dirección /home/usuario/workspace/ (en el caso de no existir lo debemos crear, ya que el jboss developer utilizara esta carpeta para poder ubicar los proyectos que se desarrollen de esta forma)

En la consola ingresamos lo siguiente:

mvn archetype:generate -Dfilter=jee

en este paso el maven descarga las pantillas de los proyectos que se pueden utilizar

damos las configuraciones para el nombre y mas del mismo

```
[INFO] Scanning for projects...
Downloading: http://repository.jboss.org/nexus/content/groups/public/com/oracle/weblogic/maven-metadata.xml
Downloading: http://repository.jboss.org/nexus/content/groups/public/org/apache/maven/plugins/maven-metadata.xml
Downloading: http://repository.jboss.org/nexus/content/groups/public/org/codehaus/mojo/maven-metadata.xml
Downloaded: http://repository.jboss.org/nexus/content/groups/public/org/apache/maven/plugins/maven-metadata.xml (9 KB at 3.9 KB/sec)
Downloaded: http://repository.jboss.org/nexus/content/groups/public/org/codehaus/mojo/maven-metadata.xml (21 KB at 9.3 KB/sec)
Downloading: http://repository.jboss.org/nexus/content/groups/public/org/apache/maven/plugins/maven-archetype-plugin/maven-metadata.xml

Downloaded: http://repository.jboss.org/nexus/content/groups/public/org/apache/maven/plugins/maven-archetype-plugin/maven-metadata.xml
(384 B at 0.4 KB/sec)
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom <<<
[INFO]
[INFO] --- maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Interactive mode
[INFO] No archetype defined. Using maven-archetype-quickstart (org.apache.maven.archetypes:maven-archetype-quickstart:1.0)
Choose archetype:
1: remote -> org.codehaus.mojo.archetypes:appclient-jee5 (-)
2: remote -> org.codehaus.mojo.archetypes:ear-jee5 (-)
3: remote -> org.codehaus.mojo.archetypes:ejb-jee5 (-)
4: remote -> org.codehaus.mojo.archetypes:webapp-jee5 (-)
5: remote -> org.fluttercode.knappsack:jee6-basic-archetype (-)
6: remote -> org.fluttercode.knappsack:jee6-minimal-archetype (-)
7: remote -> org.fluttercode.knappsack:jee6-sandbox-archetype (-)
8: remote -> org.fluttercode.knappsack:jee6-sandbox-demo-archetype (-)
9: remote -> org.fluttercode.knappsack:jee6-servlet-basic-archetype (-)
10: remote -> org.fluttercode.knappsack:jee6-servlet-demo-archetype (-)
11: remote -> org.fluttercode.knappsack:jee6-servlet-minimal-archetype (-)
12: remote -> org.fluttercode.knappsack:jee6-servlet-sandbox-archetype (-)
13: remote -> org.imixs.application:imixs-workflow-jee-archetype (Imixs Workflow JEE Archetype provides a JEE Sample Application)
14: remote -> org.jboss.weld.archetypes:weld-jsf-jee (Weld archetype for creating a Java EE 6 application using JSF 2.0, CDI 1.0, EJB 3
.1 and JPA 2.0 (persistence unit included))
15: remote -> org.jboss.weld.archetypes:weld-jsf-jee-minimal (Weld archetype for creating a minimal Java EE 6 application using JSF 2.0
, CDI 1.0 and EJB 3.1 (persistence unit not included))
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): :
```

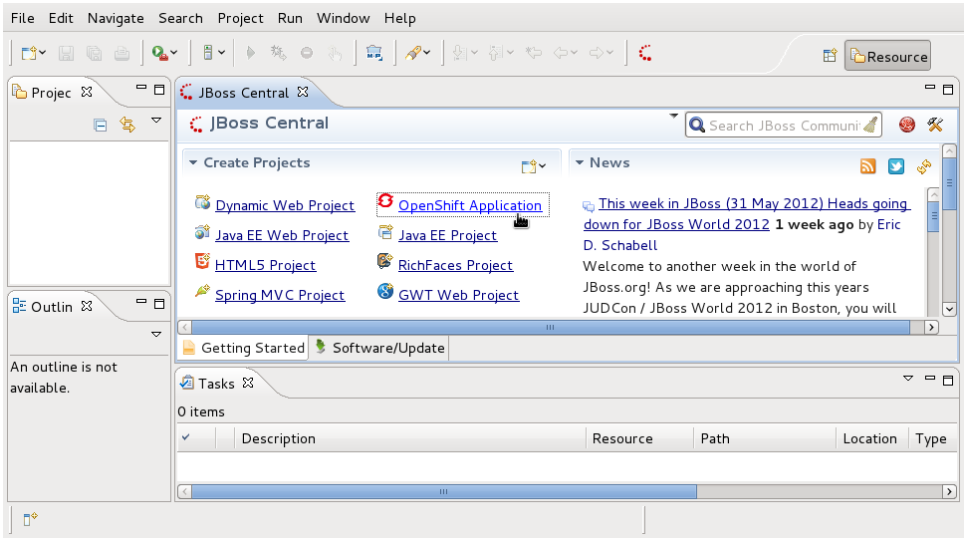
Elegimos el proyecto que vamos a utilizar y le damos le damos un nombre.

mvn clean

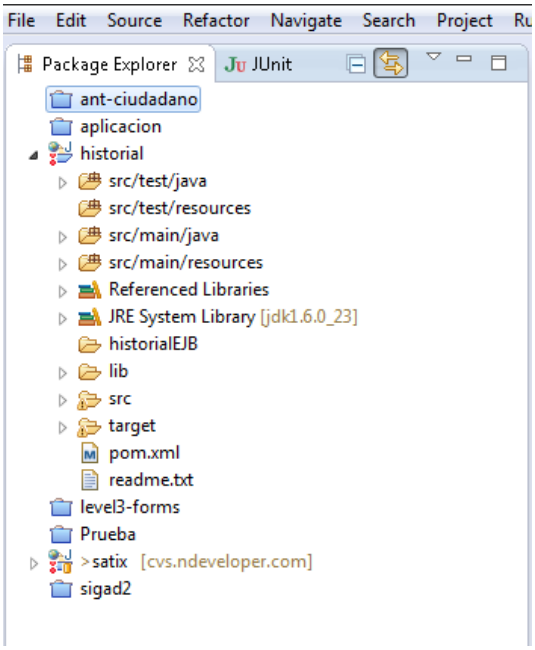
Con esto el proyecto se va a limpiar

Le damos mvn eclipse:eclipse para que se descargue las librerías que este proyecto utilizara dentro del mismo

Luego procedemos abrir el Jboss Developer Studio.



Cargamos el proyecto dentro del mismo y podremos visualizarlo y dentro de este se mostrara la estructura y como se van a ubicar las diferentes clases y librerías



Una vez realizado las diferentes configuraciones continuamos con la programación del proyecto.

3.3. Diseño y Código fuente

El proyecto como tal debe tener los siguientes archivos de configuración, donde se podrá definir la base de datos con la que va a trabajar, las librerías que va utilizar, asi como las clases paquetes y las paginas xhtml.

POM.xml

Project Object Model es un fichero xml, es la unidad principal de un proyecto maven, cuenta con toda la información del proyecto, paquetes, test, dependencias, plugins, la versión, dirección del servidor, etc. Al ejecutarse un proyecto lo primero que se lee es este archivo.

Contenido.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>ec.proyecto</groupId>
        <artifactId>historial</artifactId>
        <packaging>war</packaging>
        <version>1.0-SNAPSHOT</version>
        <name>historial</name>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <jboss.home>C:\Users\DIEGOIVA\Java\SERVERS\jboss-as-
7.1.1.Final</jboss.home>
                <jboss.domain>default</jboss.domain>
                <powermock.version>1.5.1</powermock.version>
                <junit.version>4.10-redhat-2</junit.version>
        </properties>
        <dependencies>
                <dependency>
                        <groupId>org.jboss.spec</groupId>
                        <artifactId>jboss-javaee-6.0</artifactId>
                        <version>1.0.0.Final</version>
                        <scope>provided</scope>
                        <type>pom</type>
                </dependency>
                <dependency>
                        <groupId>org.hibernate</groupId>
```

```
            <artifactId>hibernate-validator</artifactId>
            <version>4.0.0.GA</version>
            <scope>provided</scope>
        </dependency>
        <!-- primefaces -->
        <dependency>
            <groupId>org.primefaces</groupId>
            <artifactId>primefaces</artifactId>
            <version>3.5</version>
        </dependency>
        <!-- LOG -->
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.16</version>
            <exclusions>
                <exclusion>
                    <groupId>log4j</groupId>
                    <artifactId>log4j</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-api</artifactId>
            <version>6.0</version>
            <scope>provided</scope>
        </dependency>
        <!-- Test dependencies -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>${junit.version}</version>
            <type>jar</type>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.powermock</groupId>
            <artifactId>powermock-module-junit4</artifactId>
            <version>${powermock.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.powermock</groupId>
            <artifactId>powermock-api-mockito</artifactId>
            <version>${powermock.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

```
<repositories>
<!-- JBoss Repository used for hibernate-validator 4.0.0.GA and Java-ee
        spec -->
    <repository>
        <id>repository.jboss.org</id>
        <name>JBoss Repository</name>
<url>http://repository.jboss.org/nexus/content/groups/public-jboss/</url>
    </repository>
</repositories>
<build>
    <finalName>historial</finalName>
    <plugins>
        <!-- Facilitates downloading source and javadoc in Eclipse -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-eclipse-plugin</artifactId>
            <version>2.8</version>
            <configuration>
                <wtpversion>2.0</wtpversion>
                <downloadSources>true</downloadSources>
                <downloadJavadocs>true</downloadJavadocs>
            </configuration>
        </plugin>
        <!-- Ensures we are compiling at 1.6 level -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>2.3.2</version>
            <configuration>
                <source>1.6</source>
                <target>1.6</target>
            </configuration>
        </plugin>
    <!-- JBoss AS plugin for command line deployment -->
<plugin>
<artifactId>maven-antrun-plugin</artifactId>
<executions>
<execution>
<phase>install</phase>
<configuration>
<tasks>
<delete file="${jboss.home}/standalone/deployments/${project.artifactId}.war" />
<delete
file="${jboss.home}/standalone/deployments/${project.artifactId}.war.dodeploy" />
<delete
file="${jboss.home}/standalone/deployments/${project.artifactId}.war.deployed" />
<copy                                    file="target/${project.artifactId}.war"
tofile="${jboss.home}/standalone/deployments/${project.artifactId}.war" />
<touch
file="${jboss.home}/standalone/deployments/${project.artifactId}.war.dodeploy" />
```

```
        </tasks>
        </configuration>
        <goals>
        <goal>run</goal>
        </goals>
        </execution
        </executions>
        </plugin>
        </plugins>
        </build>
</project>
```

Persistence.xml

Este contiene la información de la conexión a la base de datos, describe las unidades de persistencia. Es el archivo del descriptor de despliegue para la persistencia mediante la JPA (API de persistencia Java). Se utiliza para declarar lo siguiente:

- Clases de persistencia gestionada

  Las clases gestionadas en este proyectos son @Entity

- Especificar correlación de objeto/relacional
- Información de configuración para gestores de unidad y clases de fábrica de gestores de entidad.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0"
        xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
    http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
        <persistence-unit name="puClinica">
        <jta-data-source>java:jboss/datasources/ClinicaDS</jta-data-source>
                <properties>
<property                                       name="javax.persistence.jdbc.url"
value="jdbc:postgresql://localhost:5432/clinica" />
<property name="javax.persistence.jdbc.user" value="postgres" />
<property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver" />
<property name="javax.persistence.jdbc.password" value="postgres" />
                </properties>
        </persistence-unit>
</persistence>
```

Web.xml

Las aplicaciones web Java utilizan un archivo descriptor de implementación para determinar la forma en que las URL se asignan a los servlets y las direcciones URL que necesitan autentificación, entre otra información. Forma parte del estándar de servlet para las aplicaciones web.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation=" http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
        <welcome-file-list>
                <welcome-file>index.html</welcome-file>
        </welcome-file-list>
        <context-param>
                <param-name>facelets.DEVELOPMENT</param-name>
                <param-value>true</param-value>
        </context-param>
        <servlet>
                <servlet-name>Faces Servlet</servlet-name>
                <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
                <load-on-startup>1</load-on-startup>
        </servlet>
        <servlet-mapping>
                <servlet-name>Faces Servlet</servlet-name>
                <url-pattern>*.jsf</url-pattern>
        </servlet-mapping>
        <context-param>
      <param-name>primefaces.THEME</param-name>
      <param-value>afterwork</param-value>
</context-param>
</web-app>
```

Index.xhtml

Permite identificar al proyecto cual es la página principal la cual se mostrará al ingresar al sitio Web.

```
<html>
<head>
  <meta http-equiv="Refresh" content="0; URL=home.jsf">
</head>
</html>
```

Home.xhtml

Esta es la página principal del proyecto que está compuesto en este caso por solo pro el template.xhtml.

```
<?xml version="1.0" encoding="UTF-8"?>
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
        xmlns:ui="http://java.sun.com/jsf/facelets"
        xmlns:f="http://java.sun.com/jsf/core"
        xmlns:h="http://java.sun.com/jsf/html"
        xmlns:p="http://primefaces.org/ui"
        template="/WEB-INF/templates/template.xhtml">
        <ui:define name="content">
        </ui:define>
</ui:composition>
```

Template.xhtml

El template consta de tres partes header, contenido y footer, el header y footer son heredados, y el contenido está definido x dos partes el menú p:menu y Main Content, a este último se tendrá acceso al heredar el template.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="http://java.sun.com/jsf/html"
        xmlns:ui="http://java.sun.com/jsf/facelets"
        xmlns:p="http://primefaces.org/ui">
<h:head>
        <title>SISTEMA CLINICO</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <h:outputScript name="jsf.js" library="javax.faces" />
        <h:outputScript name="jquery.js" />
        <h:outputScript library="js" name="util.js" />
        <h:outputStylesheet library="css" name="screen.css" />
</h:head>
<h:body>
        <div>
                <ui:insert name="header">
                        <ui:include src="/WEB-INF/templates/header.xhtml" />
                </ui:insert>
        </div>
        <div style="clear: both" />
```

```
        <div>
        <h:panelGroup id="page" layout="block">
        <h:panelGrid columns="2" width="100%" style="vertical-align:top;"
                        columnClasses="menupanel,contentpanel">
                                <h:form>
        <p:menu model="#{menuController.menuBar}" style="width:95%" />
                                </h:form>
        <h:panelGroup id="content" layout="block" style="width:100%">
                                <ui:insert name="content">Main Content</ui:insert>
                                </h:panelGroup>
                        </h:panelGrid>
                </h:panelGroup>
        </div>
        <div style="clear: both" />
        <div>
                <ui:insert name="header">
                        <ui:include src="/WEB-INF/templates/footer.xhtml" />
                </ui:insert>
        </div>
</h:body>
</html>
```

Header.xhtml

La cabecera de la página en esta se encuentra el título, además del usuario que inicio sección en el sistema así como la opción de salir del sistema.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="http://java.sun.com/jsf/html"
        xmlns:f="http://java.sun.com/jsf/core"
        xmlns:ui="http://java.sun.com/jsf/facelets">
        <div id="logo-container"></div>
        <p style="text-align: center; color: #111542; font-style: arial;">
<!--            CLINICA SAN JOAQUIN OCCIDENTAL  -->
                <br />Sistema de Control Clínico <br />
        </p>
        <span id="header-text-container"> <h:form id="headerForm"
                        style="margin:0px;padding:0px;">
        <h:outputText value="#{messages['lbl.usuario']}:"
                        styleClass="headerTextItem" />
        <b><h:outputText value="#{generalController.user}"
                                styleClass="headerTextItem" /></b>
        <h:outputText value=" | #{messages['lbl.fecha']}: "
```

```
                        styleClass="headerTextItem" />
                    <b><h:outputText value="#{generalController.date}"
                            styleClass="headerTextItem">
                            <f:convertDateTime dateStyle="default" />
                        </h:outputText></b>
                    <h:outputLink
        value="#{facesContext.externalContext.request.contextPath}/logout.jsp"
                        styleClass="headerTextItem">
                        <h:outputText value="#{messages['lbl.salir']}" />
                    </h:outputLink>
                </h:form>
        </span>
</ui:composition>
```

Footer.xhtml

Es el pie de la página web y puede contener la dirección y teléfono de la empresa asi como el autor del sistema.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="http://java.sun.com/jsf/html"
        xmlns:f="http://java.sun.com/jsf/core"
        xmlns:ui="http://java.sun.com/jsf/facelets">
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="http://java.sun.com/jsf/html"
        xmlns:f="http://java.sun.com/jsf/core"
        xmlns:ui="http://java.sun.com/jsf/facelets">
        <div id="footer" class="footer">
        <p style="text-align: center; color: #111542; font-style: arial;">
<!--            texto pie de pagina -->Quito, Ecuador
        </p>
        </div>
</ui:composition>
</html>
```

Páginas de diseño

Contiene:

La herencia del template los componentes que

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
        xmlns:ui="http://java.sun.com/jsf/facelets"
        xmlns:f="http://java.sun.com/jsf/core"
        xmlns:h="http://java.sun.com/jsf/html"
        xmlns:p="http://primefaces.org/ui"
        template="/WEB-INF/templates/template.xhtml">
        <ui:define name="content">
                <h:form id="formUsuarios">
                <p:panel header="#{messages['titulo.administracion_usuarios']}"
                                style="border:none;" />
                        <p:messages id="generalMessages" />
                        <p:panel>
                <p:dialog id="usuariosModal" header="#{messages['lbl.usuario']}"
                modal="true" resizable="false" width="450" showEffect="clip"
                hideEffect="fold" visible="#{usuariosDM.mostrarModal}">
                                <h:panelGrid columns="3" id="formularioUsr">
                <h:outputLabel value="#{messages['lbl.perfil']}" for="usrPerfil" />
                <p:selectOneMenu value="#{usuariosDM.usuario.rol.codigoRol}"
                                                id="usrPerfil">
                        <f:selectItem itemLabel="Seleccione" itemValue="" />
                        <f:selectItems value="#{selectItemsController.rolesItems}"
                                var="item" itemValue="#{item.codigoRol}"
                                itemLabel="#{item.nombre}" />
                                <p:ajax event="change" update="formularioUsr" />
                                        </p:selectOneMenu>
                                        <p:message for="usrPerfil" />
        <h:outputLabel value="#{messages['lbl.nombre']}" for="usuNombre" />
        <p:inputText value="#{usuariosDM.usuario.nombreUsuario}"
                                id="usuNombre" required="True" />
                                        <p:message for="usuNombre" />
        <h:outputLabel value="#{messages['lbl.apellido']}"
                                                for="usuApellido" />
        <p:inputText value="#{usuariosDM.usuario.apellidoUsuario}"
                                                id="usuApellido" />
                                        <p:message for="usuApellido" />
        <h:outputLabel value="#{messages['lbl.email']}" for="usuCorreo" />
        <p:inputText value="#{usuariosDM.usuario.emailUsuario}"
                                                id="usuCorreo" />
                                        <p:message for="usuCorreo" />
        <h:outputLabel value="#{messages['lbl.usuario']}" for="usuUser" />
        <p:inputText value="#{usuariosDM.usuario.userUsuario}"
                                                id="usuUser" />
                                        <p:message for="usuUser" />
        <h:outputLabel value="#{messages['lbl.contrasena']}" for="usuPass" />
        <p:password value="#{usuariosDM.usuario.passUsuario}" id="usuPass"
        feedback="true" promptLabel="Use Varios Caracteres"
        weakLabel="Corta" goodLabel="Mediana" strongLabel="Segura" />
                                        <p:message for="usuPass" />
        <h:outputText value="#{messages['lbl.estado']}" />
        <p:selectBooleanCheckbox value="#{usuariosDM.usuario.activo}" />
```

```
                                    </h:panelGrid>
<p:commandButton value="#{messages['ttl.guardar']}"
update="generalMessages,tablaEspecialidades, usuariosModal "
id="guardarBtn" action="#{usuariosCtrlr.guardar}"
styleClass="ui-priority-primary" />
                                </p:dialog>
                                <p:toolbar>
                                    <p:toolbarGroup align="right">
<p:commandButton title="#{messages['ttl.editar']}" id="nuevoBtn"
icon="ui-icon-document" action="#{usuariosCtrlr.nuevo}"
process="@this, tablaEspecialidades"
update="generalMessages, usuariosModal">
                                    </p:commandButton>
<p:commandButton title="#{messages['ttl.editar']}" id="editarBtn"
icon="ui-icon-pencil" action="#{usuariosCtrlr.editar}"
process="@this, tablaEspecialidades"
update="generalMessages, usuariosModal">
                                    </p:commandButton>
<p:commandButton title="#{messages['ttl.eliminar']}"
id="eliminarBtn" icon="ui-icon-trash"
action="#{usuariosCtrlr.eliminar}"
process="@this,tablaEspecialidades"
update="generalMessages, tablaEspecialidades">
                                    </p:commandButton>
                                </p:toolbarGroup>
                            </p:toolbar>
<h:inputText value="#{usuariosDM.stringBusqueda}"
        id="stringBusqueda" />
<p:commandButton value="#{messages['ttl.buscar']}"
update="generalMessages, tablaEspecialidades" id="buscarBtn"
action="#{usuariosCtrlr.buscar}" styleClass="ui-priority-primary" />
<p:dataTable id="tablaEspecialidades" var="esp"
value="#{usuariosDM.usuarios}" paginator="true" rows="10"
rowKey="#{esp.codigoUsuario}"
selection="#{usuariosDM.seleccionado}" selectionMode="single">
                                    <f:facet name="header">
                    Lista de Usuarios
                    </f:facet>
            <p:column>
                                    <f:facet name="header">
<h:outputText value="#{messages['lbl.nombre']}" />
                                    </f:facet>
<h:outputText value="#{esp.nombreUsuario}" />
                                    </p:column>
                                    <p:column>
                                    <f:facet name="header">
<h:outputText value="#{messages['lbl.apellido']}" />
                                    </f:facet>
<h:outputText value="#{esp.apellidoUsuario}" />
                                    </p:column>
```

```
                                        <p:column>
                    <f:facet name="header">
        <h:outputText value="#{messages['lbl.email']}" />
                                        </f:facet>
        <h:outputText value="#{esp.emailUsuario}" />
                                        </p:column>
                                        <p:column>
        <f:facet name="header">
        <h:outputText value="#{messages['lbl.usuario']}" />
                                        </f:facet>
        <h:outputText value="#{esp.userUsuario}" />
                                        </p:column>
                                        <p:column>
                        <f:facet name="header">
        <h:outputText value="#{messages['lbl.perfil']}" />
                                        </f:facet>
        <h:outputText value="#{esp.rol.nombre}" />
                                        </p:column>
        <p:column headerText="#{messages['lbl.activo']}">
        <p:selectBooleanCheckbox disabled="true" rendered="#{esp.activo}"
        styleClass="ui-icon ui-icon-check" style="border: none;">
                                        </p:selectBooleanCheckbox>
        <p:selectBooleanCheckbox disabled="true" rendered="#{!esp.activo}"
        styleClass="ui-icon ui-icon-cancel" style="border: none;">
                                        </p:selectBooleanCheckbox>
                                        </p:column>
                                </p:dataTable>
                        </p:panel>
        <p:remoteCommand name="initUsr" process="@this"
        actionListener="#{usuariosCtrlr.init}" update="formUsuarios" />
                </h:form>
                <script type="text/javascript">
                        initUsr();
                </script>
        </ui:define>
</ui:composition>
```

UsuarioCtrlr.java

Esta clase es quien controlara las acciones que se realicen en la página de Administración de usuarios en el cual contiene botón de buscar, nuevo, editar y eliminar así como la ventana emergente, el datatable, cajas de texto y las checbok.

```
package ec.proyecto.controller;
import javax.ejb.EJB;
```

```
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.ViewScoped;
import ec.proyecto.data.UsuariosDM;
import ec.proyecto.model.Usuario;
import ec.proyecto.servicio.UsuariosServicio;
@ManagedBean(name = "usuariosCtrlr")
@ViewScoped

public class UsuariosCtrlr extends BaseController {
	@EJB
	private UsuariosServicio servicio;
	@ManagedProperty(value = "#{usuariosDM}")
	private UsuariosDM usuariosDM;
	public void guardar() {
		try {
			servicio.guardar(usuariosDM.getUsuario());
			usuariosDM.setMostrarModal(Boolean.FALSE);
			addErrorMessage(getMessageProperty("msg.guardar_error"));
		} catch (Exception e) {
		}
	}
	public void nuevo() {
		usuariosDM.setUsuario(new Usuario());
		usuariosDM.setMostrarModal(Boolean.TRUE);
	}
	public void eliminar() {
		usuariosDM.setUsuario(usuariosDM.getSeleccionado());
		try {
			Boolean activo = null;
			usuariosDM.getUsuario().setActivo(activo);
			servicio.guardar(usuariosDM.getUsuario());
		} catch (Exception e) {
		}
	}
	public void buscar() {
	usuariosDM.setUsuarios(servicio.buscar(usuariosDM.getStringBusqueda()));
	}
	public void editar() {
		if (usuariosDM.getSeleccionado().getCodigoUsuario() == null) {

	addErrorMessage(getMessageProperty("msg.debe_seleccionar"));
		} else {
			usuariosDM.setUsuario(usuariosDM.getSeleccionado());
			usuariosDM.setMostrarModal(Boolean.TRUE);
		}
	}
	public void init() {
	}
	public UsuariosDM getUsuariosDM() {
```

```
		return usuariosDM;
	}
	public void setUsuariosDM(UsuariosDM usuariosDM) {
		this.usuariosDM = usuariosDM;
	}
}
```

UsuariosDM.java

Se utilizan principalmente para manejar las variables que se van a utilizar en la página web, se deben generar los Getters and Setters.

```
package ec.proyecto.data;
import java.util.List;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import ec.proyecto.model.Rol;
import ec.proyecto.model.Usuario;

@ManagedBean(name = "usuariosDM")
@SessionScoped
public class UsuariosDM {

	private Usuario usuario = new Usuario();
	private String stringBusqueda;
	private List<Usuario> usuarios;
	private Usuario seleccionado;
	private Boolean mostrarModal = Boolean.FALSE;

	public Boolean getMostrarModal() {
		return mostrarModal;
	}

	public void setMostrarModal(Boolean mostrarModal) {
		this.mostrarModal = mostrarModal;
	}

	public String getStringBusqueda() {
		return stringBusqueda;
	}

	public void setStringBusqueda(String stringBusqueda) {
		this.stringBusqueda = stringBusqueda;
	}
	public Usuario getUsuario() {
		if(usuario.getRol()==null){
			usuario.setRol(new Rol());
		}
```

```
		return usuario;
	}

	public void setUsuario(Usuario usuario) {
		this.usuario = usuario;
	}
	public List<Usuario> getUsuarios() {
		return usuarios;
	}

	public void setUsuarios(List<Usuario> usuarios) {
		this.usuarios = usuarios;
	}

	public Usuario getSeleccionado() {
		return seleccionado;
	}

	public void setSeleccionado(Usuario seleccionado) {
		this.seleccionado = seleccionado;
	}
}
```

UsuarioServicio.java

La interfaces del servicio es el a que va a contener la los métodos de la lógica del negocio.

```
package ec.proyecto.servicio;
import java.util.List;
import javax.ejb.Local;
import ec.proyecto.model.Rol;
import ec.proyecto.model.Usuario;

@Local
public interface UsuariosServicio {

	void guardar(Usuario usuario) ;

	List<Usuario> buscar(String stringBusqueda);

	void eliminar(Usuario usuario);

	List<Rol> buscarRoles(Rol rol);

	Usuario buscarUsuario(String remoteUser);
}
```

UsuarioServicioImpl.java

Contiene la implementación de los métodos de la interface de servicio.

```
package ec.proyecto.servicio.impl;
import java.util.List;
import javax.ejb.EJB;
import javax.ejb.Stateful;
import ec.proyecto.dao.UsuariosDao;
import ec.proyecto.exceptions.EntidadNoBorradaException;
import ec.proyecto.exceptions.EntidadNoGrabadaException;
import ec.proyecto.model.Rol;
import ec.proyecto.model.Usuario;
import ec.proyecto.servicio.UsuariosServicio;

@Stateful
public class UsuariosServicioImpl implements UsuariosServicio {

	@EJB
	private UsuariosDao dao;

	@Override
	public void guardar(Usuario usuario) {
		try {
			if (usuario.getCodigoUsuario() == null) {
				dao.crear(usuario);
			} else {
				dao.actualizar(usuario);
			}
		} catch (EntidadNoGrabadaException e) {
			e.printStackTrace();
		}
	}

	@Override
	public void eliminar(Usuario usuario) {
		try {
			dao.eliminar(usuario);

		} catch (EntidadNoBorradaException e) {
			e.printStackTrace();
		}
	}

	@Override
	public List<Usuario> buscar(String stringBusqueda) {
		return dao.buscar(stringBusqueda);
```

```
	}

		@Override
		public List<Rol> buscarRoles(final Rol rol) {
			List<Rol> roles = dao.buscarRol(rol);
			for (Rol rol2 : roles) {
				rol2.getListaOpcionRol().size();
			}
			return roles;
		}

		@Override
		public Usuario buscarUsuario(String remoteUser) {
			// TODO Auto-generated method stub
			return null;
		}

}
```

UsuarioDao.java

```
package ec.proyecto.dao;
import java.util.List;
import javax.ejb.Local;
import ec.proyecto.model.Rol;
import ec.proyecto.model.Usuario;

@Local
public interface UsuariosDao extends GenericDAO<Usuario, Long> {

	List<Usuario> buscar(String stringBusqueda);

	List<Usuario> buscarTodo(String stringUb);

	List<Rol> buscarRol(Rol rol);

}
```

UsuarioDaoImpl.java

```
package ec.proyecto.dao.impl;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.ejb.Stateless;
import javax.persistence.Query;
import ec.proyecto.dao.UsuariosDao;
import ec.proyecto.model.Rol;
```

```
import ec.proyecto.model.Usuario;

@Stateless
public class UsuariosDaoImpl extends GenericDAOImpl<Usuario, Long>
		implements UsuariosDao {

	private static final long serialVersionUID = 1L;

	public UsuariosDaoImpl() {
		super(Usuario.class);
	}

	@Override
	@SuppressWarnings("unchecked")
	public List<Usuario> buscar(String nombreUsuario) {
	StringBuffer sql = new StringBuffer();
	Map<String, Object> mapaParametros = new HashMap<String, Object>();
	sql.append("Select a from Usuario a");
	if (nombreUsuario != null && !nombreUsuario.isEmpty()) {
	sql.append(" where Upper(a.nombreUsuario) LIKE :nombreUsuario");
	mapaParametros.put("nombreUsuario", nombreUsuario.toUpperCase());
		}
		sql.append(" order by a.nombreUsuario");

		Query q = em.createQuery(sql.toString());

		for (String key : mapaParametros.keySet()) {
			q.setParameter(key, "%" + mapaParametros.get(key) + "%");
		}
		return q.getResultList();
	}

	@Override
	@SuppressWarnings("unchecked")
	public List<Usuario> buscarTodo(String stringUb) {
		StringBuffer sql = new StringBuffer();
		sql.append("SELECT e FROM Usuario e");
		Query q = em.createQuery(sql.toString());
		return q.getResultList();

	}

	@Override
	@SuppressWarnings("unchecked")
	public List<Rol> buscarRol(Rol rol) {
	StringBuffer sql = new StringBuffer();
	HashMap<String, Object> parametros = new HashMap<String, Object>();
		sql.append("Select r from Rol r where 1=1 ");
		if (rol.getCodigoRol() != null) {
			sql.append(" and r.codigoRol= :codigoRol");
```

```
                parametros.put("codigoRol", rol.getCodigoRol());
            }
            if (rol.getNombre() != null && !rol.getNombre().isEmpty()) {
                sql.append(" and lower(r.nombre) like :nombre");
        parametros.put("nombre", "%" + rol.getNombre().toLowerCase() + "%");
            }
            if (rol.getActivo() != null) {
                sql.append(" and r.activo= :activo");
                parametros.put("activo", rol.getActivo());
            }
            StringBuffer consulta = new StringBuffer();
            consulta.append("");

            final Query q = em.createQuery(sql.toString());
            for (Map.Entry<String, Object> entry : parametros.entrySet()) {
                q.setParameter(entry.getKey(), entry.getValue());
            }
            return q.getResultList();
        }

}
```

BaseController.java

Controla los mensajes desplegados en las pantallas.

```
package ec.proyecto.controller;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.ResourceBundle;
import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.bean.ManagedProperty;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.log4j.Logger;
import ec.proyecto.data.SelectItemsDM;

public abstract class BaseController {

        protected static final Logger log = Logger.getLogger(BaseController.class);
```

```
@ManagedProperty(value = "#{selectItemsDM}")
protected SelectItemsDM selectItemsDM;

public SelectItemsDM getSelectItemsDM() {
	return selectItemsDM;
}

public void setSelectItemsDM(final SelectItemsDM selectItemsDM) {
	this.selectItemsDM = selectItemsDM;
}

protected void addErrorMessages(final List<String> errors) {
	for (String error : errors) {
		addErrorMessage(error);
	}
}

protected void addErrorMessage(final String resumen) {
	FacesMessage message = new FacesMessage(resumen);
	message.setSeverity(FacesMessage.SEVERITY_ERROR);
	FacesContext.getCurrentInstance().addMessage(null, message);
}

protected void addErrorMessage(final String resumen, final String detalle) {
	FacesMessage message = new FacesMessage(resumen, detalle);
	message.setSeverity(FacesMessage.SEVERITY_ERROR);
	FacesContext.getCurrentInstance().addMessage(null, message);
}

protected void addInfoMessage(final String resumen) {
	FacesMessage message = new FacesMessage(resumen);
	message.setSeverity(FacesMessage.SEVERITY_INFO);
	FacesContext.getCurrentInstance().addMessage(null, message);
}

protected void addInfoMessage(final String resumen, final String detalle) {
	FacesMessage message = new FacesMessage(resumen, detalle);
	message.setSeverity(FacesMessage.SEVERITY_INFO);
	FacesContext.getCurrentInstance().addMessage(null, message);
}

protected void addWarnMessage(final String resumen) {
	FacesMessage message = new FacesMessage(resumen);
	message.setSeverity(FacesMessage.SEVERITY_WARN);
	FacesContext.getCurrentInstance().addMessage(null, message);
}

protected void addWarnMessage(final String resumen, final String detalle) {
	FacesMessage message = new FacesMessage(resumen, detalle);
	message.setSeverity(FacesMessage.SEVERITY_WARN);
```

```
                FacesContext.getCurrentInstance().addMessage(null, message);
        }

        public Boolean getExistErrors() {
                Severity maximunSeverity =
getFacesContext().getMaximumSeverity();
                if (FacesMessage.SEVERITY_ERROR.equals(maximunSeverity)) {
                        return true;
                }
                return false;
        }
}
```

SelectItemsDM.java

Controla las variables utilizadas por BaseController.java

```
package ec.proyecto.data;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;

import ec.proyecto.model.Enfermedad;
import ec.proyecto.model.Especialidad;
import ec.proyecto.model.Examen;
import ec.proyecto.model.Medico;
import ec.proyecto.model.Recurso;
import ec.proyecto.model.Rol;
import ec.proyecto.model.Usuario;

@ManagedBean(name = "selectItemsDM")
@ViewScoped
public class SelectItemsDM implements Serializable {

        private static final long serialVersionUID = 1L;

        private List<Rol> roles = new ArrayList<Rol>();

        private List<Usuario> usuarios = new ArrayList<Usuario>();

        private List<Especialidad> especialidad = new ArrayList<Especialidad>();

        private List<Medico> medicos = new ArrayList<Medico>();

        private List<Enfermedad> enfermedades = new ArrayList<Enfermedad>();
```

```
private List<Recurso> recursos = new ArrayList<Recurso>();

private List<Examen> examenes= new ArrayList<Examen>();

public List<Rol> getRoles() {
        return roles;
}

public void setRoles(final List<Rol> roles) {
        this.roles = roles;
}

public List<Usuario> getUsuarios() {
        return usuarios;
}

public void setUsuarios(final List<Usuario> usuarios) {
        this.usuarios = usuarios;
}

public List<Especialidad> getEspecialidad() {
        return especialidad;
}

public void setEspecialidad(final List<Especialidad> especialidad) {
        this.especialidad = especialidad;
}

public List<Medico> getMedicos() {
        return medicos;
}

public void setMedicos(List<Medico> medicos) {
        this.medicos = medicos;
}

public List<Enfermedad> getEnfermedades() {
        return enfermedades;
}

public void setEnfermedades(List<Enfermedad> enfermedades) {
        this.enfermedades = enfermedades;
}

public List<Recurso> getRecursos() {
        return recursos;
}

public void setRecursos(List<Recurso> recursos) {
        this.recursos = recursos;
```

```
	}

	public List<Examen> getExamenes() {
		return examenes;
	}

	public void setExamenes(List<Examen> examenes) {
		this.examenes = examenes;
	}

}
```

MessaggeResourses.properties

Contiene todas las etuquetas de los label, button y titulos que se muestran en la pantalla.

```
#A
lbl.apellido=Apellido
lbl.activo=Activo
#B
ttl.buscar=Buscar
lbl.buscar=Buscar
#C
ttl.cancelar=Cancelar
lbl.cedula=C\u00E9dula
lbl.cedula.visita=C\u00E9dula Visita
lbl.contrasena=Contrase\u00F1a
lbl.cuarto=Cuarto
lbl.cuarto.registro=Registro Cuarto
#D
lbl.descripcion=Descripci\u00F3n
lbl.descripcion.turno=Descripci\u00F3n Turno
lbl.detalle=Detalle
lbl.detalle.historial=Detalle Historial
lbl.detalle.visita=Detalle Visita
lbl.detalle.hospitalizacion=Detalle Hospitalizacion
lbl.direccion=Direcci\u00F3n
lbl.discapacidad=Discapacidad
#E
lbl.eliminar=Eliminar
lbl.email=Email
lbl.estado=Estado
lbl.estado.civil=Estado Civil
lbl.especialidad=Especialidad
lbl.enfermedad=Enfermedad
lbl.examen=Examen
```

ttl.editar=Editar
ttl.eliminar=Eliminar
#F
lbl.fecha=Fecha
lbl.fecha.visita=Fecha Visita
lbl.fecha.turno=Fecha Turno
#G
ttl.guardar=Guardar
lbl.guardar=Guardar
#H
lbl.hora=Hora
lbl.hora.hospitalizacion=Hora Hospitalizacion
lbl.hora.visita=Hora Visita
lbl.Hospitalizacion=Hospitalizacion
lbl.historial=Historial
#M
lbl.medico=Medico
#N
lbl.nombre=Nombre
lbl.nombre.visita=Nombre Visita
ttl.nuevo=Nuevo
lbl.numero.turno=N\u00FAmero Turno
lbl.numero.historia=N\u00FAmero Historial
#O
#P
lbl.paciente=Paciente
lbl.paciente.seleccionado=Paciente Seleccionado :
lbl.perfil=Perfil
#R
lbl.rol=Rol
lbl.recurso=Recurso
#S
lbl.salir=Salir
lbl.seleccionar=Seleccionar
lbl.seleccione_perfil=Seleccione un perfil
lbl.sexo=Sexo
#T
lbl.telefono=Telefono
lbl.turno=Turno
#U
lbl.usuario=Usuario
lbl.url=Url
#V
lbl.valor=Valor
lbl.visita=Visita
lbl.valor.turno=Hora Turno
#messages
msg.activado_exito=Se ha activado con \u00E9xito
msg.borrado_error=No se ha podido desactivar
msg.borrado_exito=Se ha eliminado con \u00E9xito

msg.debe_seleccionar=Debe seleccionar
msg.guardar_error=No se ha podido guardar
msg.guardar_exito=Guardado Exitoso
msg.editar_error=Error al Editar
#titulos
titulo.administracion_cuarto=Administraci\u00F3n de Cuartos
titulo.administracion_cuartos=Administraci\u00F3n de Cuartos
tituto.administracion_detalle.historial.clinico=Detalle del Historial Clinico
titulo.administracion_especialidades=Administraci\u00F3n de Especialidades
tituto.administracion_historial.clinico=Historial Clinico
titulo.administracion_medicos=Administraci\u00F3n de M\u00E9dicos
titulo.administracion_recursos=Administraci\u00F3n de Recursos
titulo.administracion_usuarios=Administraci\u00F3n de Usuarios
titulo.historial_pacientes=Registro de Pacientes
titulo.historial_hospitalizacion=Registro de Hospitalizaci\u00F3n
titulo.historial_visitas=Registro de Vistas
titulo.asigancion_turnos=Asignacion de Turnos
titulo.asigancion_medica=Asignacion de Cita M\u00E9dica
#tildes
a \u00E1 e \u00E9 i \u00ED o \u00F3 u \u00FA A \u00C1 E \u00C9 I \u00CD O \u00D3 U \u00DA enie \u00F1 ENIE \u00D1

Diccionario de Datos

Cuarto Paciente			
Nombre	**DataType**	**Length**	**Precision**
codigo_cuarto_paciente	integer		
descripcion_cuarto_paciente	character varying	255	
estado_cuarto_paciente	character varying	255	
Activo	Boolean		
detalle_historia	character varying	255	
codigo_examen	Integer		
codigo_enfermedad	Integer		
codigo_historial_paciente	Integer		
codigo_medico	Integer		
detalle character	character varying	255	
fecha_detalle_historia	Date		
codigo_hospitalizacion	Integer		
Detalle Hospitalización			
Nombre	**DataType**	**Length**	**Precision**
codigo_detalle_hospitalizacion	Integer		
codigo_recurso	Integer		
codigo_hospitalizacion	Integer		
codigo_visita	Integer		
fecha_entrada_hospitalizacion	Date		
fecha_salida_hospitalizacion	Date		
Empresa			
Nombre	**DataType**	**Length**	**Precision**
codigo_empresa	Integer		
nombre_empresa	character varying	255	
direccion_empresa	character varying	255	
telefono_empresa	Numeric	10	
Activo	Boolean		
Enfermedad			
Nombre	**DataType**	**Length**	**Precision**

codigo_enfermedad	Integer		
nombre_enfermedad	character varying	255	
detalle_enfermedad	character varying	255	
codigo_tratamiento	Integer		
Especialidad			
codigo_especialidad	Integer		
nombre_especialidad	character varying	255	
Activo	Boolean		
Examen			
Nombre	**DataType**	**Length**	**Precision**
codigo_examen	Integer		
nombre_examen	character varying	255	
descripcion_examen	character varying	255	
resultado_examen	character varying	255	
documento_examen	character varying	255	
Historial Paciente			
Nombre	**DataType**	**Length**	**Precision**
codigo_historial_paciente	Integer		
codigo_paciente	Integer		
Hospitalización			
Nombre	**DataType**	**Length**	**Precision**
codigo_hospitalizacion	Integer		
codigo_detalle_historial	Integer		
detalle_hospitalizacion	character varying	255	
hora_entrada	Time		
codigo_cuarto_paciente	Integer		
Médico			
Nombre	**DataType**	**Length**	**Precision**

codigo_medico	Integer		
codigo_especialidad	Integer		
nombre_medico	character varying	255	
apellido_medico	character varying	255	
direccion_medico	character varying	255	
Activo	Boolean		
telefono_medico	Numeric	10	
Opción			
Nombre	**DataType**	**Length**	**Precision**
codigo_opcion	Integer		
Nombre	character varying	255	
url	character varying°	255	
Tipo	character varying	255	
Activo	Boolean		
Opción Rol			
Nombre	**DataType**	**Length**	**Precision**
codigo_opcion_rol	Integer		
codigo_opcion	Integer		
codigo_rol	Integer		
Activo	Boolean		
Paciente			
Nombre	**DataType**	**Length**	**Precision**
codigo_paciente	Integer		
codigo_historial_paciente	Integer		
codigo_referencia	Integer		
nombre_paciente	character varying	255	
apellido_paciente	character varying	255	
telefono_paciente	Numeric	10	
direccion_paciente	character varying	255	
est_paciente	character varying	255	

cedula_paciente	Numeric	10	
discapacidad_paciente	character varying	255	
estado_civil	character varying	255	
residencia_paciente	character varying	255	
sexo_paciente	character varying	1	
activo_paciente	Boolean		
Recursos			
Nombre	**DataType**	**Length**	**Precision**
codigo_recurso	Integer		
nombre_recurso	character varying	255	
detalle_recurso	character varying	255	
Activo	Boolean		
Referencia			
Nombre	**DataType**	**Length**	**Precision**
codigo_referencia	Integer		
nombre_referencia	character varying	255	
apellido_referencia	character varying	255	
telefono_referencia	Numeric	10	
direccion_referencia	character varying	255	
Rol			
Nombre	**DataType**	**Length**	**Precision**
codigo_rol	Integer		
Descripción	character varying	255	
Nombre	character varying	255	
Activo	Boolean		
Tratamiento			
Nombre	**DataType**	**Length**	**Precisio**

			n
codigo_tratamiento	Integer		
descripcion_tratamiento	character varying	255	
Turno			
Nombre	**DataType**	**Length**	**Precisio n**
codigo_turno	Integer		
codigo_paciente	Integer		
numero_turno	Numeric	5	
descripcion_turno	character varying	255	
fecha_turno	date		
Usuario			
Nombre	**DataType**	**Length**	**Precisio n**
codigo_usuario	Integer		
codigo_rol	Integer		
nombre_usuario	character varying	255	
apellido_usuario	character varying	255	
correo_usuario	character varying	255	
user_usuario	character varying	255	
pass_usuario	character varying	255	
Activo	Boolean		
Visitas			
Nombre	**DataType**	**Length**	**Precisio n**
codigo_visita	Integer		
fecha_visita	Date		
hora_visita	Time		
ced_visita	Numeric	10	
nombre_visita	character varying	255	
detalle_visita	character varying	255	

MANUAL DE USUARIOS

Introducción

El presente manual describe de forma clara y detallada la utilización del sistema informático para médicos, de una forma amigable para el usuario.

Objetivo

Guiar al usuario al correcto uso del sistema de control del Sistema de Control del Historial Clínico y la Reservación de turnos.

Contenido

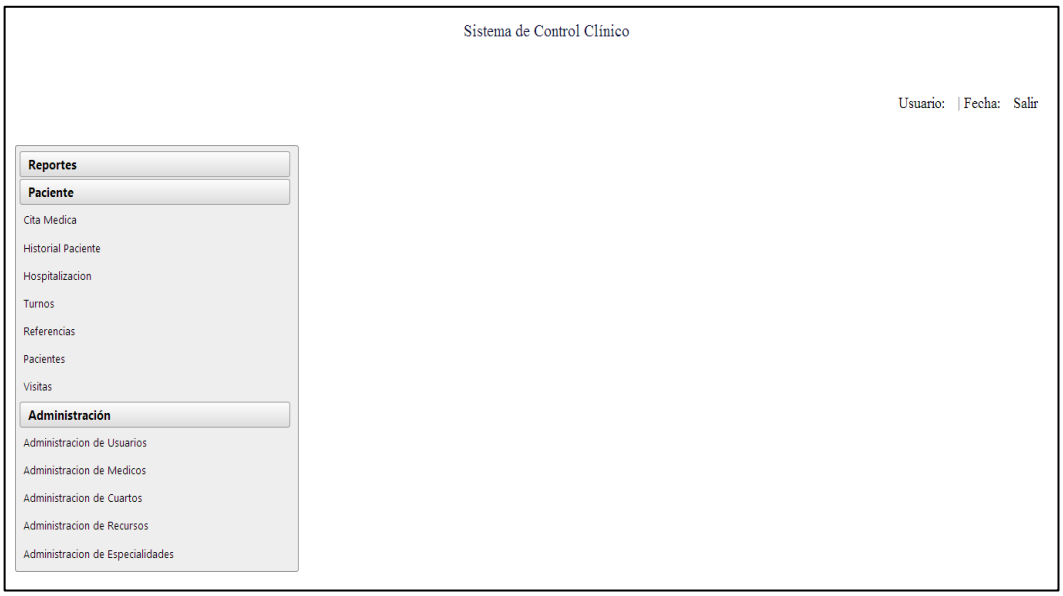
Login

Esta página nos permite ingresar como usuarios del sistema y asi podremos realizar cualquier proceso dependiendo del rol que este haya sido asignado.

Para el efecto deberá ingresar el nombre de Usuario y Contraseña y este será, validado con los registros que se encuentren en la base de datos, al accionar la opción Login el sistema le llevara automáticamente a la página principal del sistema , si la información ingresada no es correcta se mostrara un mensaje de error.

Home

Home será la página principal que se mostrará al usuario después de iniciar sesión en la aplicación, donde se mostrará el nombre del usuario del sistema y la opción **salir** en la parte superior izquierda, en el costado derecho se mostrará el menú de opciones del sistema, este está divido por tres secciones Administración, Paciente y Reportes cada uno permitirá realizar diferentes funciones como su nombre lo indica.

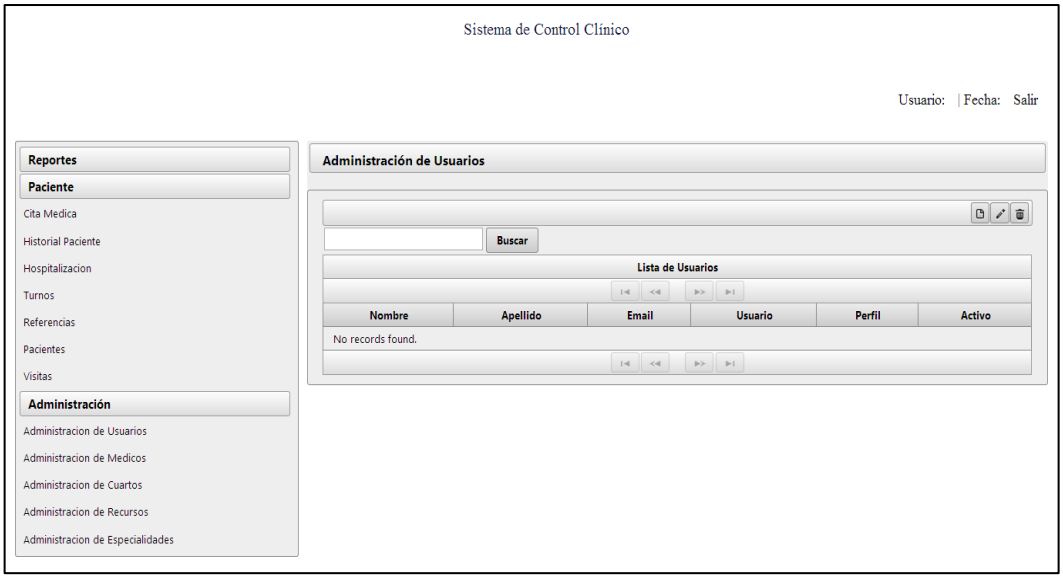


Administración

El menú administración permitirá a los usuarios de tipo administrador a realizar los diferentes registros que de las tablas generales, tales como usuario donde permitirá crear, modificar y eliminar registros del sistema de acuerdo a su uso y rol dentro del mismo.

-Usuarios

La pantalla permite ingresar, modificar y eliminar usuarios al sistema, a estos se les asignará un rol con el cual podrán ingresar al sistema y realizar los diferentes procesos de acuerdo a sus permisos.



Para poder ingresar un nuevo usuario se deberá seleccionar el icono  que se encuentra en la parte derecha de la pantalla y se tendrá que ingresar los siguientes datos:

Perfil: Rol al cual se le va asignar al usuario

Nombre: Nombres del Usuario

Apellido: Apellidos del Usuario

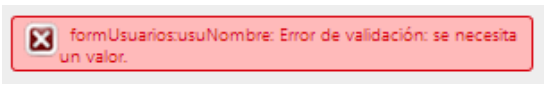
Email: Correo electrónico del Usuario

Usuario: Nombre de Usuario dentro del Sistema

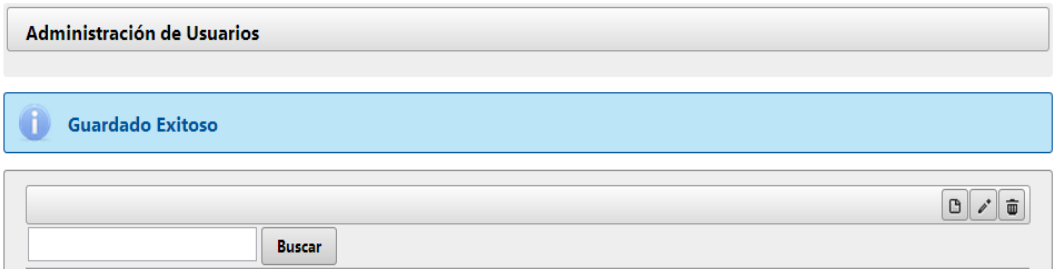
Contraseña: Credencial para poder ingresar al sistema

Activo: Identifica si un usuario está disponible dentro del sistema

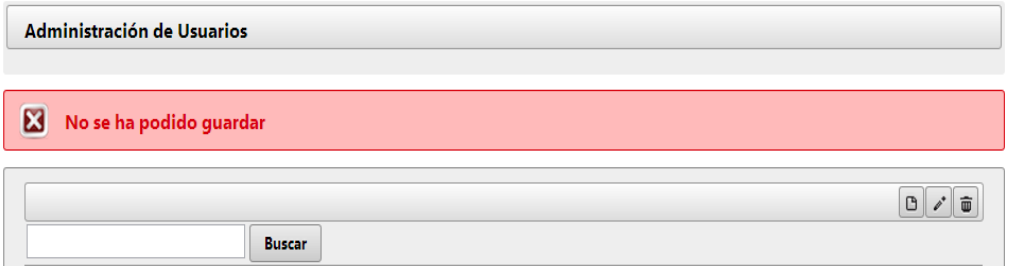
Cada campo que se encuentre con el símbolo de Campo Obligatorio *, en el caso de no ingresar ningún dato correspondiente se mostrara un mensaje de validación.



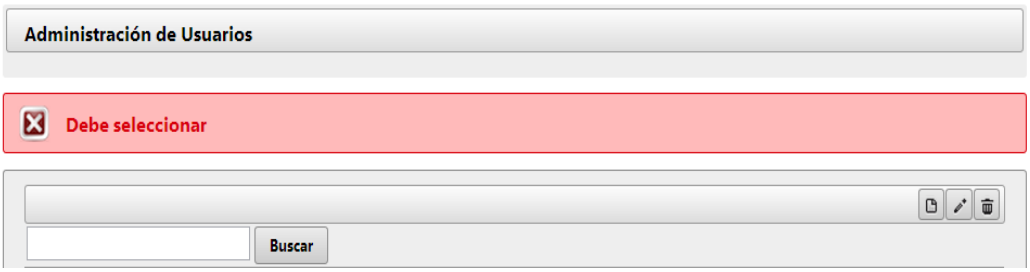
Para poder guardar el Usuario deberá elegir la opción **Guardar** y el nuevo usuario registrado aparecerá en la tabla de Lista de Usuarios y se mostrará un mensaje de confirmación.



En el caso de que el registro no se pudo guardar se mostrará el siguiente mensaje:



También se podrá realizar modificaciones  en los registros y eliminar  uno de ellos para ello deberá seleccionar un registro de la lista, en el caso de no elegir uno se mostrara el siguiente error.

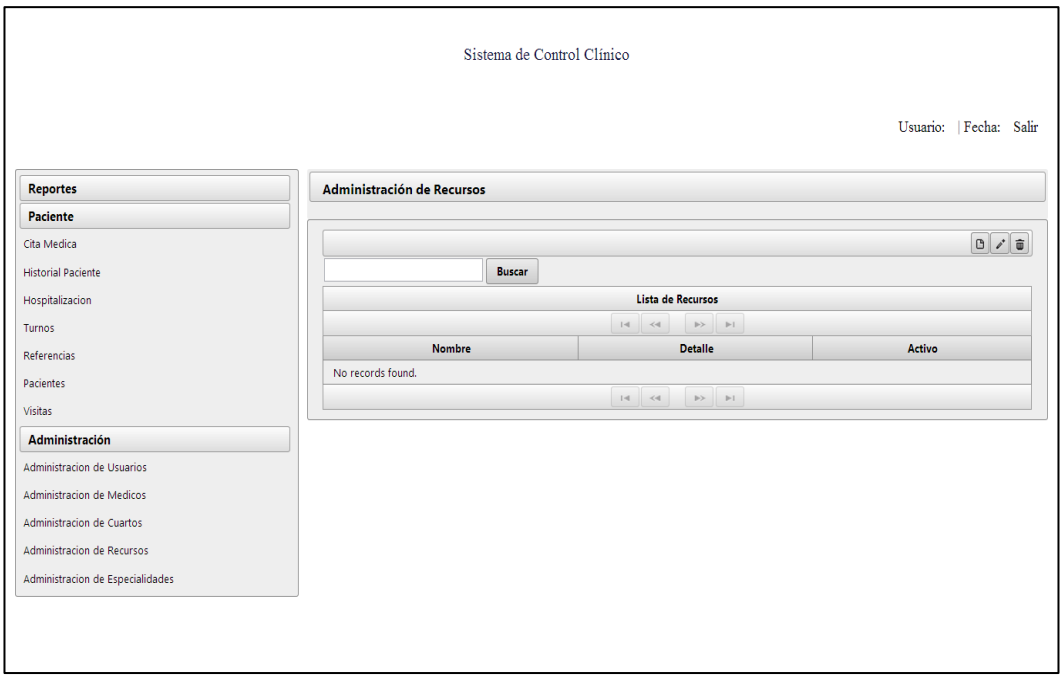


Además la pantalla cuenta con la opción de buscar usuarios donde se deberá ingresar un parámetro de búsqueda.

Los resultados obtenidos de la búsqueda se mostraran en la tabla Lista de Usuarios.

-Recursos

La pantalla Administración de Recursos, permitirá ingresar, modificar y eliminar registros de recursos y se mostrará en la Lista de Recursos.



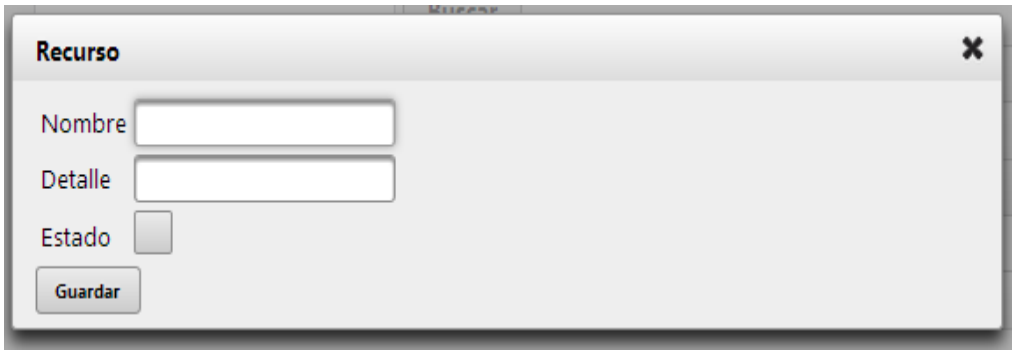
Se podrá registrar nuevos recursos en el sistema ingresando los datos en los campos disponibles y con la opción guardar se registrará el nuevo recurso.

Los campos son:

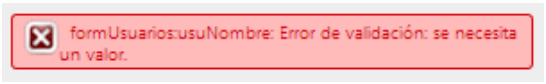
Nombre: Nombre de identificación del recurso.

Detalle: Permite verificar en qué estado se encuentra el recurso

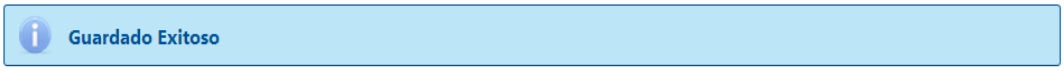
Estado: Identifica si el recurso está activo.



En el caso de no ingresar un campo obligatorio se mostrará el mensaje de validación, esto se lo hará por cada campo.



Se mostrará el mensaje si el registro se guardó correctamente.



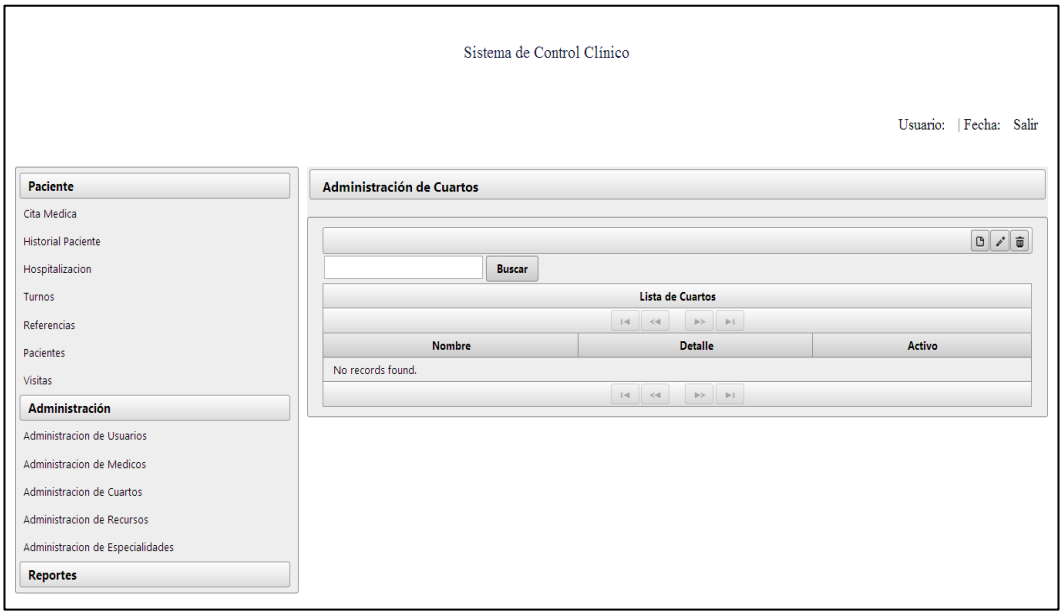
En caso de no guardarse se mostrará

Para realizar modificaciones se deberá escoger un registro de la lista se y elegir la opción  y en el caso de desear eliminar un recurso se selecciona la opción  y la tabla Lista de Recursos se actualizará automáticamente.

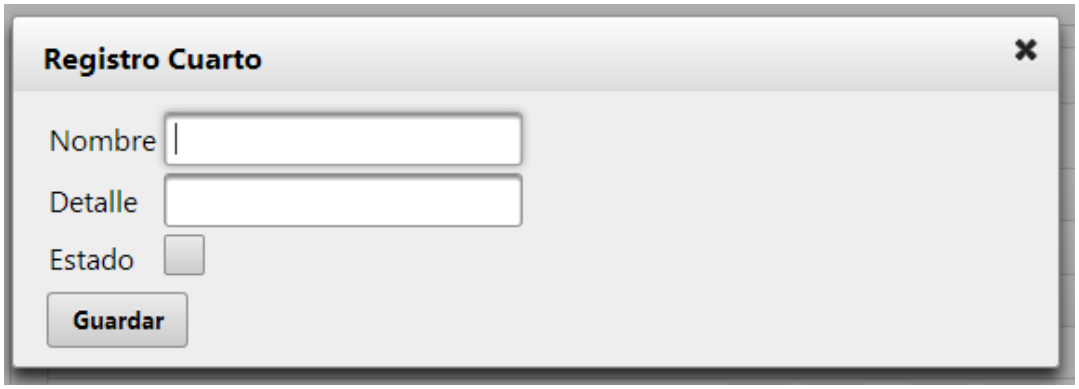
Existe la opción de buscar un registro de recurso mediante la el botón de búsqueda que existe en la página.

-Cuartos Pacientes

La Administración de Cuartos tendrá las opciones de crear y editar un registro de la lista de cuartos.



Para ingresar un registro de un nuevo cuarto elegimos la opción nuevo  con lo que se mostrará un dialogo.



Cuenta con los siguientes campos:

Nombre: Nombre de identificación del cuarto.

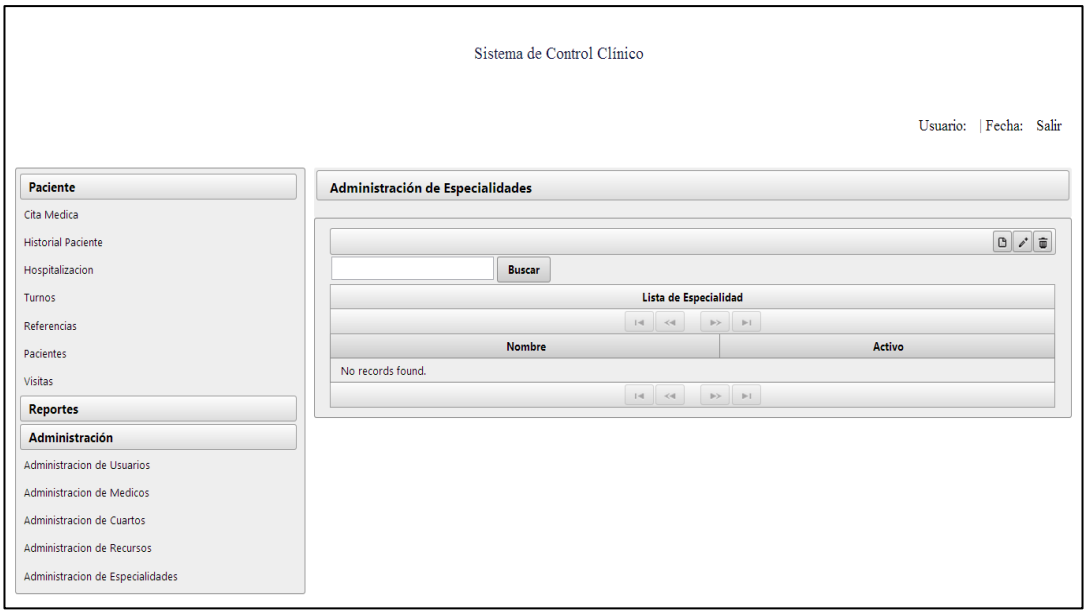
Detalle: Detalla el estado del cuarto.

Activo: Muestra si está activo un cuarto

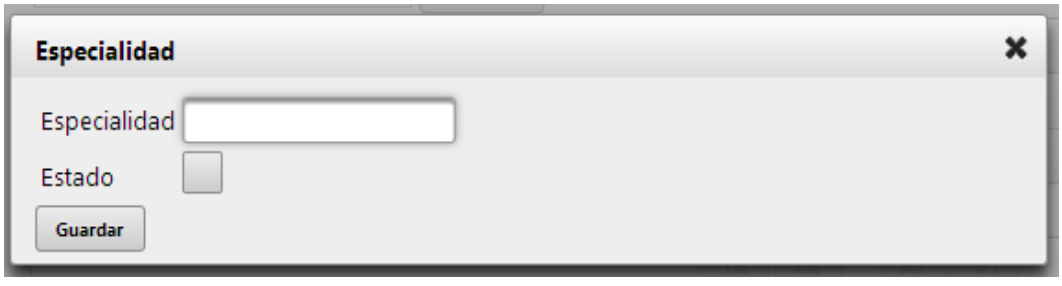
Los mensaje de validación de campos obligatorios, guardado exitoso y de error al guardar se mostraran en los casos correspondientes.

-Administración de Especialidades

La página contara con las opciones de crear, editar y eliminar un registro del catálogo de especialidades.



Para realizar el ingreso de un nuevo registro de especialidad se debe elegir la opción Nuevo  , y se desplegará el siguiente dialogo.

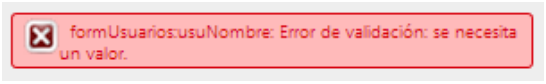


Los campos a ingresar son:

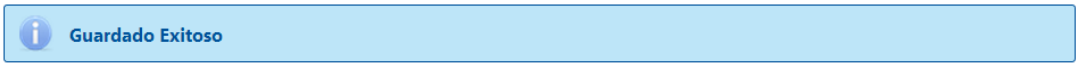
Especialidad: Nombre de identificación.

Activo: Si la especialidad se encuentra disponible.

El campo especialidad es obligatorio y si no se ingresa algún parámetro se mostrara el mensaje de validación.



Una vez ingresado los datos se podrá guardar y se mostraran los siguientes mensajes en caso de guardado correcto.



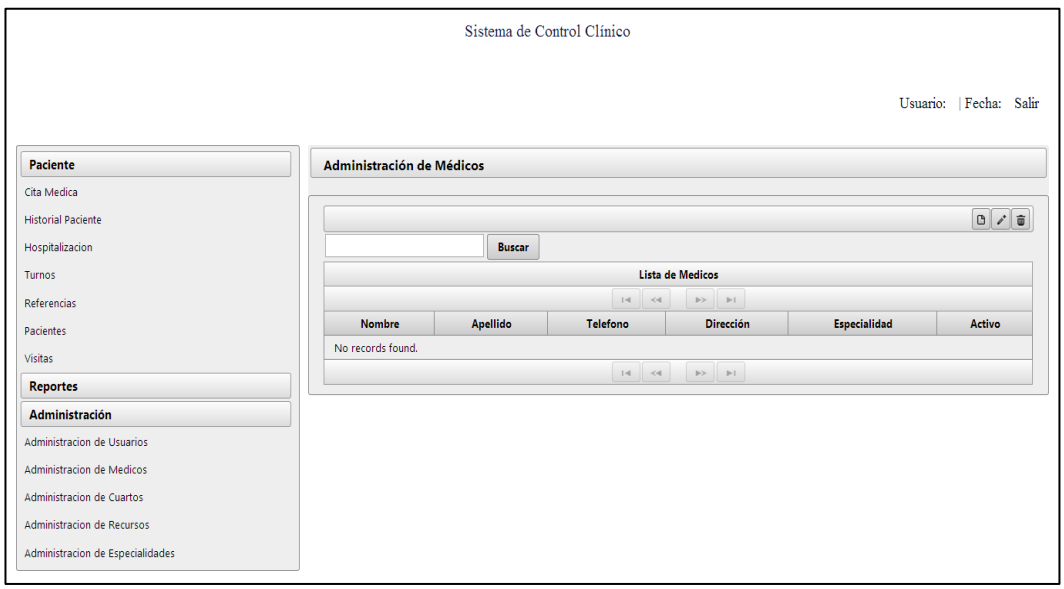
Si no se guarda correctamente se mostrara el error de guardado.

Las opciones de editar  y eliminar  se realizaran eligiendo un registro de la tabla Lista de Especialidades.

La opción de buscar una especialidad se la puede realizar ingresando cualquier texto y los resultados se mostrarán en la lista.

-Administración de Médicos

En la página de Administración de Médicos se permitirá crear, editar y eliminar un registro de un médico.



Para la opción de ingresar un nuevo registro de paciente se elige el botón nuevo .

Y se muestra una ventana de dialogo.

Los campos a ingresar son;

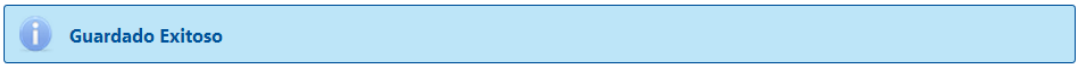
Especialidad: A la que pertenece el médico.

Nombre: Nombre de identificación del médico.

Apellido: Apellido de identificación del médico.

Teléfono: Teléfono del médico.

Para guardar el registro elegimos la opción Guardar, con esto se desplegará el mensaje de conformación en caso de que el registro fue guardado con éxito.



En el caso de que no se pueda guardar se mostrará un mensaje:

Las opciones de editar y eliminar se las realizara eligiendo un registro de la lista.

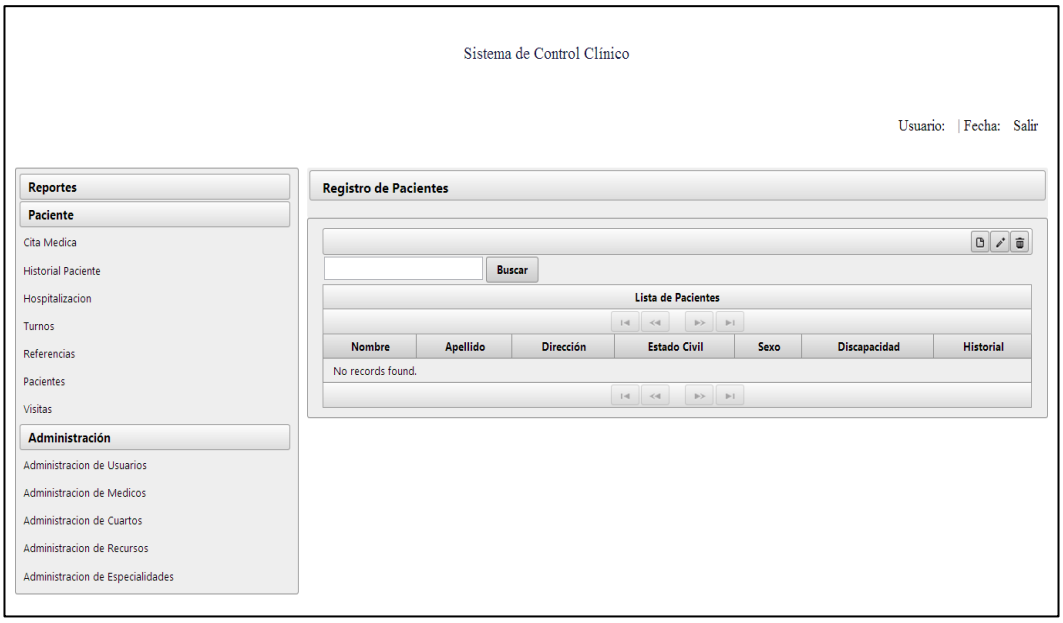
La opción buscar se realizara ingresando un dato en la caja de texto y se mostrará el resultado en la lista de médicos.

PACIENTES

Son las principales funcionalidades que tendrá el sistema donde podrán registrar las historias clínicas, asignación de turnos y registrar una eventual hospitalización.

-Pacientes

En la pantalla Registro de pacientes se tiene la posibilidad de ingresar nuevos pacientes al sistema y este generara automáticamente un codigo que será con el q se cree la nueva historia clínica.



Para el registro de un nuevo paciente debe ingresar los siguientes datos:

Cedula: Número de identidad del paciente.

Nombre: Nombre del paciente.

Apellido: Apellido del paciente

Teléfono: Número telefónico del paciente.

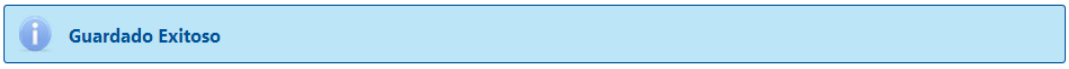
Dirección: Dirección del domicilio del paciente.

Sexo: Genero del paciente.

Estado Civil: Situación sentimental del paciente.

Tipo Sangre: Grupo sanguíneo al cual pertenece.

La opción guardar permite guardar el registro y se mostrará el mensaje.



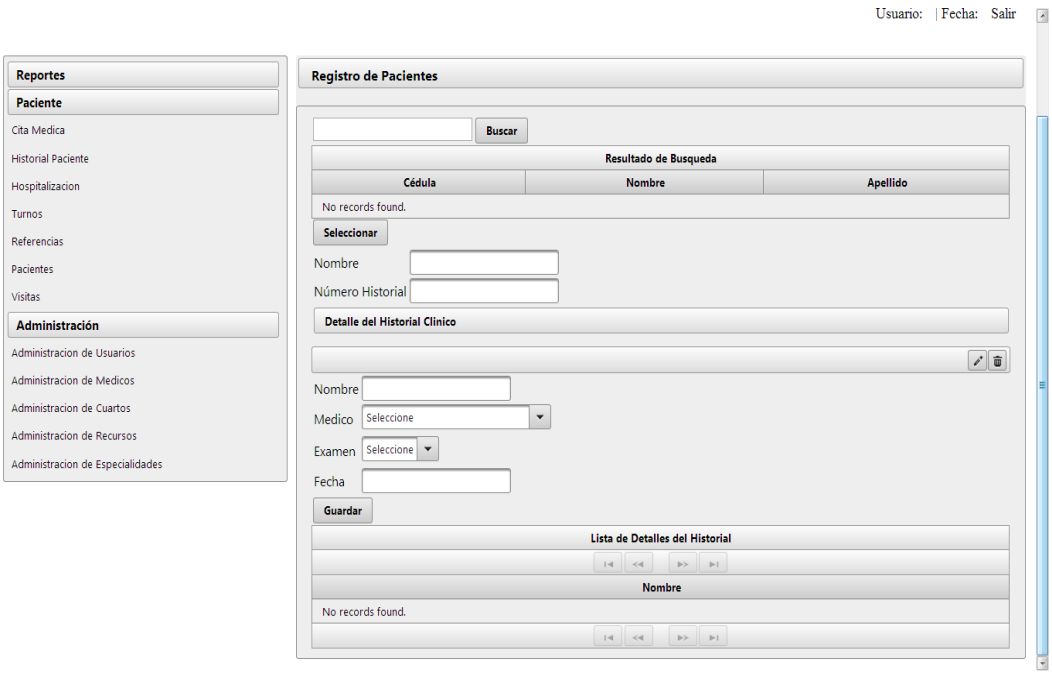
Y en caso de no guardar.

Las opciones de editar  y eliminar  se las realiza al elegir un registro de la lista de pacientes.

Se cuenta con la opción de realizar una búsqueda de pacientes.

-Historial Clínico

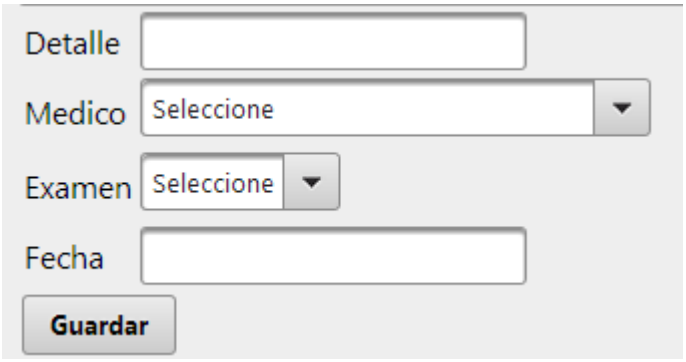
Trata de todos el historial que ha tenido el paciente donde se detallas las atenciones recibidas en la clínica.



El principal proceso del sistema se lo realiza en esta pantalla, donde el usuario podrá registrar todos los datos de los diferentes procesos que realice dentro de la clínica estos son citas médicas, registro de exámenes, tratamientos, médicos asignados, enfermedades y síntomas que presente cada paciente.

Para ingresar los datos deberá ingresar de la detalle de historia clínica deberá realizar la búsqueda del paciente por medio del número de cédula y aparecerá el resultado en la tabla de usuarios, una vez elegido deberá seleccionarla.

Una vez elegido el paciente deberá ingresar los siguientes datos:



Detalle: Detalle de los síntomas que presenta el paciente.

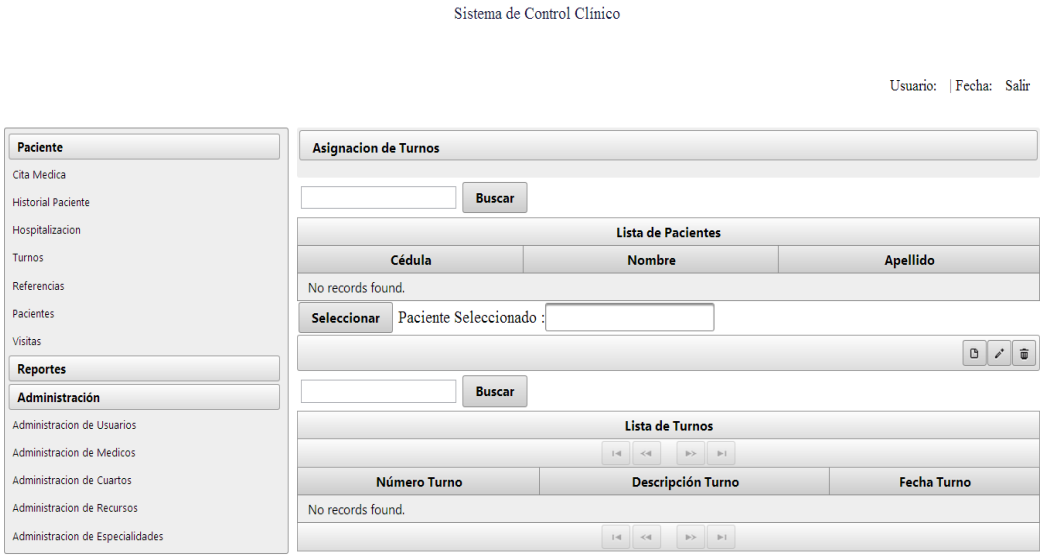
Medico: Médico que atendió al paciente.

Examen: La enfermedad que se le diagnostico el paciente.

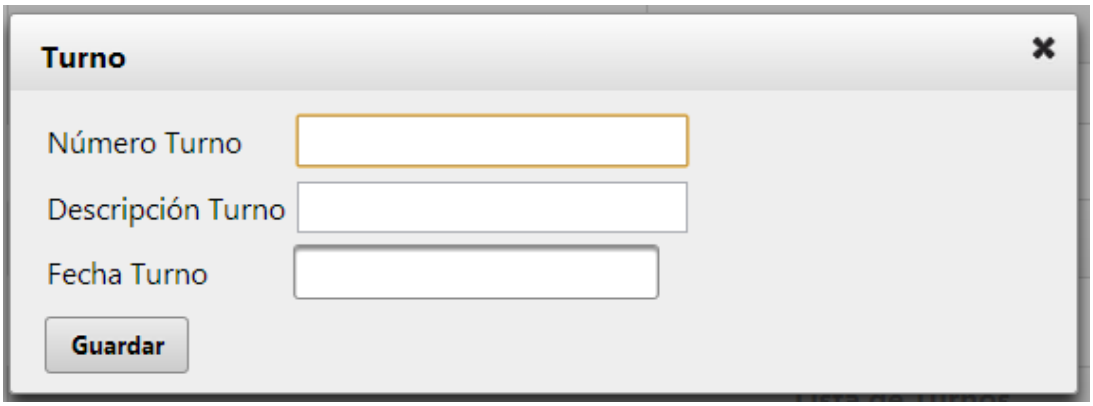
Fecha: La fecha que el paciente recibió la atención.

Estado: En el que se encuentra el paciente.

-Asignación de Turnos



La asignación de turnos lo realiza un usuario operado, tendrá que buscar al paciente el cual tendrá que estar registrado en el sistema, donde se va a realizar un registro del turno que se le va a otorgar para que pueda ser atendido dentro de la clínica.



Los datos necesarios para otorgarle el turno son:

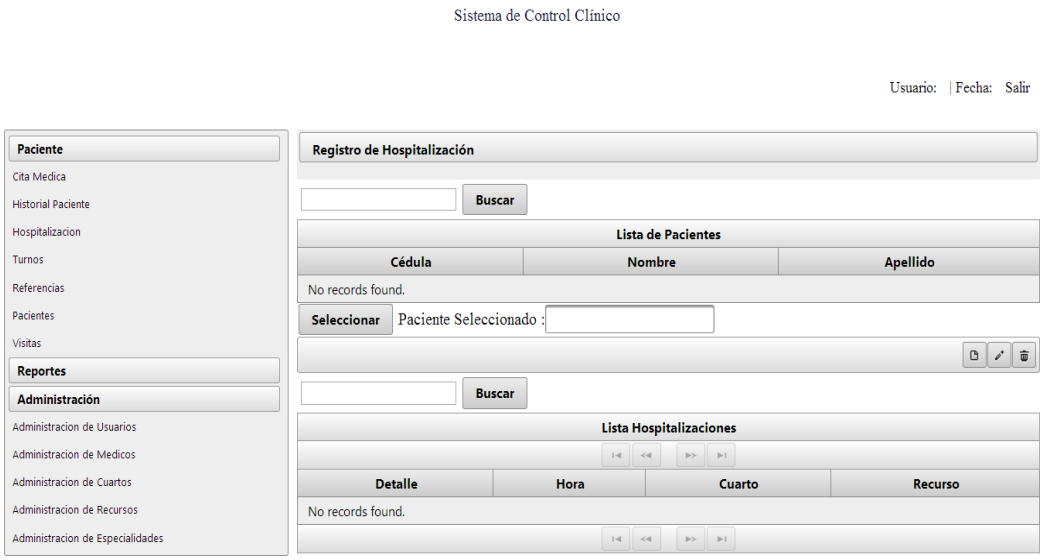
Número Turno: El cual se va generar automáticamente.

Detalle: Donde podrá ingresar alguna descripción acerca del turno.

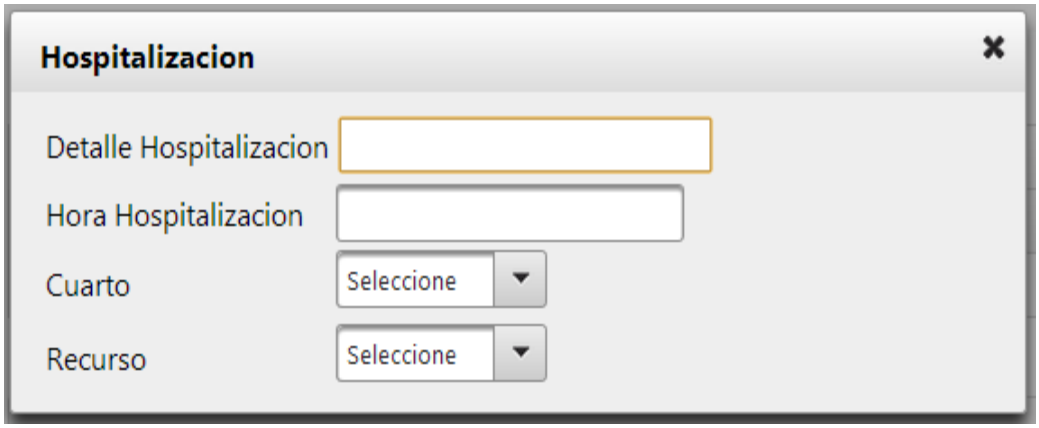
Fecha de Turno: Que se genera automáticamente con la hora actual y fecha.

-Registro de Hospitalización

El registro consta de dos partes la cabecera donde constaran los nombres del paciente y los datos del ingreso del mismo.



En esta pantalla se podrá registrar la hospitalización de un paciente la fecha el cuarto y los diferentes recursos que el paciente necesite para su atención.



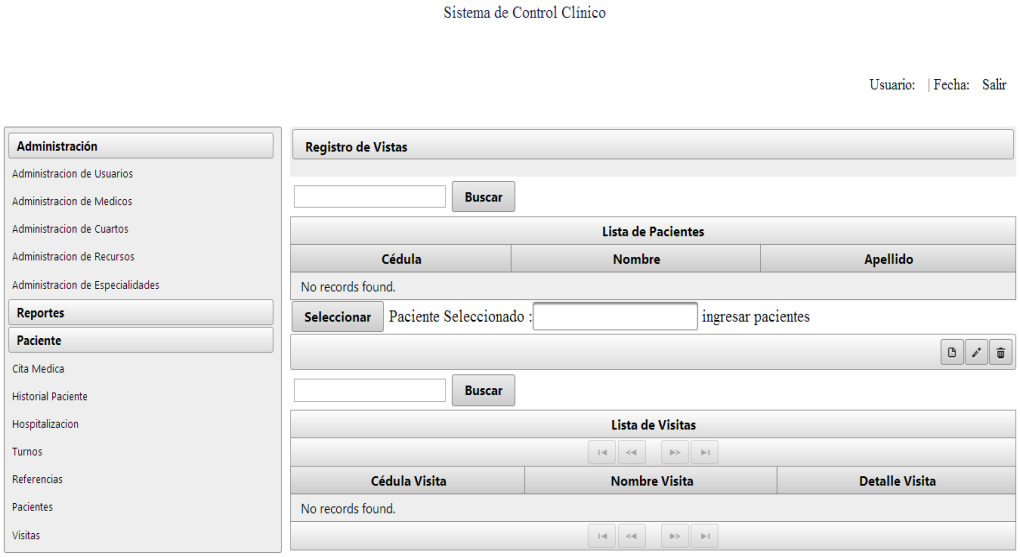
Se deberá buscar un paciente este deberá crear un nuevo registro de hospitalización.

En el detalle de la hospitalización:

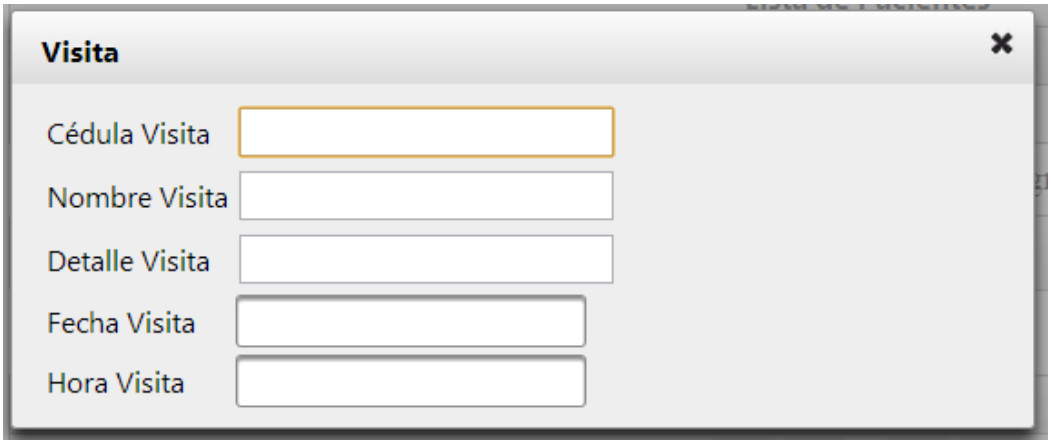
Cuarto paciente: Donde fue asignado el paciente.

Recurso: Detalle de los recursos clínicos que fueron asignados al paciente.

Visitas:



El usuario deberá ingresar las visitas al paciente mediante esta pantalla, donde deberá buscar al paciente que se encuentre internado e ingresar los siguientes datos.



Los campos a ingresar son los siguientes:

Cedula: Cedula de Identidad de la persona que visita al paciente.

Nombre: Nombre de la persona que visita al paciente.

Apellido: Apellido de la visita.

Teléfono: Teléfono de la visita.