PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

ESCUELA DE SISTEMAS

"DISEÑO Y DESARROLLO DE UN SISTEMA PARA LA GESTIÓN DE INFORMACIÓN DE PACIENTES DE UN CONSULTORIO OFTALMOLÓGÍCO"

DAVID ANDRÉS RIVERA JARRÍN

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS Y COMPUTACIÓN

QUITO, OCTUBRE DEL 2017

Tabla de contenidos

DEDICATORIA:

A Dios, por guiarme y darme la fortaleza necesaria en los momentos difíciles.

A mis padres, por la paciencia, los valores que me inculcaron desde pequeño y por apoyarme en la toma de decisiones.

A mi familia, por brindarme soporte cuando lo necesitaba.

A mis amigos y compañeros, por los momentos alegres y esas noches de estudio o de proyectos que parecían eternas.

A mis profesores, no solo por dictar la clase, sino también por compartir su conocimiento y experiencia.

CAPÍTULO 1: INTRODUCCIÓN Y FUNDAMENTOS TEÓRICOS

Este capítulo presenta una breve introducción a la oftalmología y los principales elementos, tanto teóricos como prácticos que se utilizaron para el desarrollo de esta disertación.

1.1 INTRODUCCIÓN

En el campo de la medicina, tener acceso al historial clínico del paciente es algo primordial que permite conocer datos importantes de un paciente, las aflicciones que ha tenido en su salud y así tener la base para determinar cuál es su padecimiento actual o como ha sido el progreso con alguna enfermedad. Este es el motivo por el cual se han implementado, en centros de salud, varios métodos para almacenar dicha información.

En el país, no es mucho tiempo desde que se han comenzado a utilizar, en hospitales, sistemas que permiten agilitar el acceso a los datos de los pacientes al igual que a la información de sus dispensarios de medicinas. El problema radica en que los centros de salud más pequeños no cuentan con estos sistemas, como es el caso de consultorios médicos.

Para el presente proyecto de disertación se realizará el diseño y desarrollo de un programa que permita automatizar el manejo de fichas y consultas de un consultorio oftalmológico.

1.2 ANTECEDENTES

Varios años atrás, se planteó que los registros de los datos de pacientes en fichas médicas se los realizaría en documentos físicos. El problema con estos documentos era tener un lugar adecuado para su almacenamiento, ya que si este sitio era húmedo, con el pasar del tiempo, el papel se deshacía y se perdía parte o toda la información de los pacientes, al igual que debía ser lo suficientemente grande para poder almacenar todos los registros y poder tenerlos ordenados para que sea fácil su búsqueda.

A finales del siglo XX, con la llegada del computador personal y herramientas de software de ofimática, como Excel o Calc, se propuso que el almacenamiento de los registros de pacientes se los debía realizar en estos programas. Si bien fue

y todavía es una de las soluciones que sobrepuso el problema del espacio físico, cuando se tenían muchos documentos sobre diferentes pacientes, la búsqueda de información se volvía algo tediosa, sin mencionar que los documentos se perdían porque se guardaban en diferentes carpetas.

Las Bases de Datos aparecieron para solucionar problemas de almacenamiento, búsqueda y seguridad de la información y será un pilar para el desarrollo de esta disertación.

1.3 JUSTIFICACIÓN

Un consultorio oftalmológico es el encargado de brindar el servicio de atención a pacientes que padecen de problemas visuales. Para llevar a cabo esta actividad es necesario llevar el historial de los pacientes, de tal forma que se refleje de manera completa todo lo que ha sucedido en torno al problema que tiene determinado paciente.

Con el tiempo, esta información se ha ido almacenando manualmente en fichas, lo que ha ocasionado que el almacenamiento físico y la búsqueda de los datos de los pacientes sea un problema.

Por lo expuesto anteriormente se puede concluir que debido a la cantidad de datos de pacientes que un consultorio oftalmológico posee, el análisis de dicha información se ha tornado muy difícil, ya que actualmente, cada ficha médica debe ser buscada de manera manual. La presente disertación de tesis pretende desarrollar un sistema cliente-servidor, utilizando herramientas como: una base de datos robusta de tipo relacional y un lenguaje de programación de alto nivel (PostgreSQL y Java respectivamente), de modo que el sistema permita a los doctores realizar búsquedas más rápidas de las fichas de los pacientes para poder ser analizadas.

1.4 OBJETIVOS

1.4.1 Objetivo General

Diseñar y desarrollar un sistema que permita la automatización de registros de fichas médicas de pacientes, así como también las consultas que se realizan en un consultorio oftalmológico.

1.4.2 Objetivos Específicos

- Llevar un control y registro automatizado de las fichas de los pacientes.
- Dar mayor rapidez y eficiencia al momento de la consulta médica.
- Seleccionar las herramientas de desarrollo adecuadas que serán utilizadas.
- Mantener un mejor orden en cuanto al manejo historial de consultas médicas.
- Conservar la información de los pacientes de manera confidencial.
- Realizar el análisis de requerimientos para diseñar el sistema a través de diagramas UML.

1.5 CONSULTORIOS OFTALMOLÓGICOS

La oftalmología es una rama de la medicina, viene del latín *ofthalmós* que significa ojo y del sufijo *logía*, que significa estudio. Esta especialidad es la encargada del tratado médico y quirúrgico del sistema visual en las personas. Comprende la parte superior del rostro, las cejas, los parpados, los músculos del ojo y sus nervios (Olver & Cassidy, 2005). Se encuentra ligada con otros campos médicos, algunas condiciones oculares son los síntomas de otras enfermedades como la diabetes, condiciones cardiovasculares, neuralgia, etc.

El espacio físico donde los oftalmólogos, personas que ejercen la práctica de la oftalmología, suelen atender a sus pacientes es conocido como consultorio.

Estos pueden encontrarse dentro de hospitales o también en departamentos u oficinas aparte. La complejidad de las funciones que cumple un oftalmólogo en un consultorio dependerá de la adecuación de equipos que posea el mismo, pero generalmente en todos está presente el diagnóstico médico.

Un buen diagnóstico médico depende fuertemente del examinador y de una historia clínica detallada. Esta sirve de soporte para saber cuál es el pasado ocular del paciente (si posee pérdida de visión o algún tipo de enfermedad visual), su pasado médico (enfermedades como la diabetes, alergia a algún medicamento, etc.) e información que ayudará a realizar de manera más exacta un diagnóstico. (James & Bron, 2012)

1.6 SELECCIÓN DE HERRAMIENTAS

1.6.1 Arquitectura:

Esta disertación tiene como objetivo el desarrollo de un software que pueda ser utilizado en el computador de un consultorio oftalmológico. Este computador contendrá tanto la base de datos como el programa, por lo que la arquitectura a utilizar será la de Cliente/Servidor.

En esta arquitectura el denominado ‘Cliente’ es el encargado de presentar al usuario final una interfaz para que pueda realizar solicitudes al ‘Servidor’, este las procesa y devuelve una respuesta que es presentada por el mismo ‘Cliente’ (Sommerville, 2005).

1.6.2 Base de Datos:

Una base de datos es el conjunto de información que puede ser almacenada en distintas formas, una de ellas es utilizando el modelo relacional. Este modelo tiene como fundamento principal guardar información en tablas que poseen varias propiedades y que pueden relacionarse con otras tablas (Piñeiro, 2014), como se indica en la Figura 1.1.

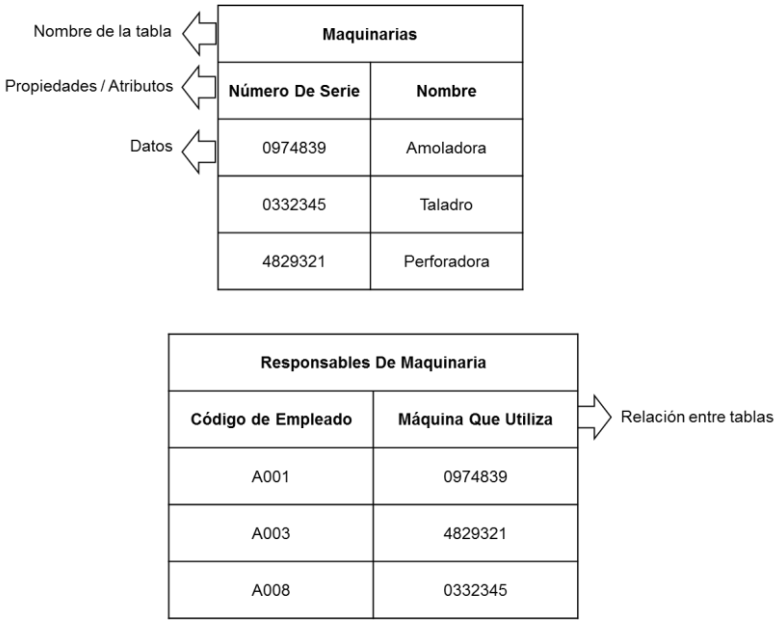

Figura 1.1. Ejemplo de modelo relacional

Para el diseño y desarrollo de este plan de disertación, la base de datos a utilizar será PostgreSQL, creado por la Universidad de Chicago en Berkeley. Este sistema objeto-relacional se caracteriza por ser de código abierto y por su amplia gama de sistemas operativos que lo soportan. A parte de ser una de las principales opciones escogidas por empresas nuevas por sus costos de licenciamiento, también posee una comunidad ordenada en la que se puede hallar soporte para cualquier tipo de problema (Juba, Vannahme, & Volkov, 2015).

1.6.3 Lenguaje de Programación e IDE

Java es un lenguaje de programación que apareció en 1990, cuando un equipo de trabajo en Sun Microsystems comenzó a trabajar en un lenguaje que serviría para funcionar en dispositivos portátiles como PDAs. Una vez completado, se llamó inicialmente Oak y con el paso del tiempo su nombre finalmente cambió a Java (Reilly & Reilly, 2002). Unas de las propiedades más importantes de este lenguaje son: ser simple, portable, seguro y principalmente ser orientado a objetos.

Java será el lenguaje a utilizar en la realización de esta disertación. Para llevar a cabo la programación se utilizará el IDE de NetBeans en su versión 8.0.2. Un IDE, o Entorno de Desarrollo Integrado por sus siglas en inglés (Integrated Development Environment), permite el desarrollo rápido de aplicaciones gracias a poseer el editor de código, compilador y debugger en una misma interfaz gráfica (Dantas, 2011).

Netbeans soporta la programación en varios lenguajes, como Java, C, C++, PHP, y se pueden añadir más mediante plugins. Por ser principalmente un IDE para Java, Netbeans tiene incorporado soporte para el desarrollo de varias aplicaciones, una de ellas es Java SE (Standar Edition), que son las que típicamente suelen utilizarse en computadores de escritorio o portátiles (Heffelfinger, 2015) y que es el tipo de aplicación que se desarrollará en esta disertación.

1.7 METODOLOGÍA BASE (RUP)

1.7.1 Origen y Principios de RUP

El Proceso Unificado de Rational (Rational Unified Process) es un marco de referencia para el desarrollo de software. Fue creado por la corporación Rational Software a finales de los años 90, empresa que fue adquirida por IBM (Sommerville, 2005).

Esta metodología, conocida también como RUP por sus siglas, es utilizada para el desarrollo y entrega incremental de software mediante procesos iterativos implementando la notación UML. El Lenguaje de Modelado Unificado (Unified Modeling Language) es un lenguaje orientado a objetos que provee elementos que ayudan a visualizar de manera clara los requerimientos del cliente, diseño e implementación del software, permitiendo así una simplificada comunicación con el producto a desarrollar (Tsui, Karam, & Bernal, 2016).

1.7.2 Ciclo de Vida

El ciclo de vida de un software describe la forma en la que el mismo será desarrollado, desde su fase de inicio hasta la fase final. Un software elaborado bajo la metodología RUP tiene un ciclo de vida iterativo que puede ser representado en un gráfico de dos dimensiones (Kruchten, 2004), como el que se muestra a continuación:

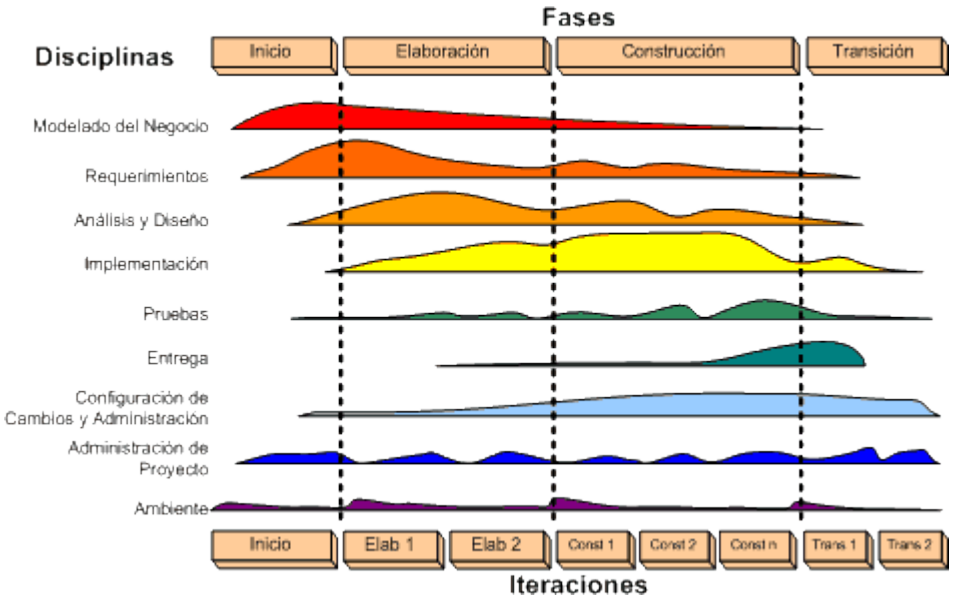

Figura 1.2. Ciclo de Vida RUP (2011). Recuperado de: http://ingenieriaensoftwareivan.blogspot.com/2011/11/software-blog.html

Este gráfico muestra la relación entre los aspectos estáticos y dinámicos que forman parte de RUP. Los dinámicos se encuentran en el eje horizontal, que representa el tiempo y son las fases e iteraciones que se llevarán a cabo. De forma vertical se encuentran representados los aspectos estáticos del proceso que pueden ser actividades, entregables del proyecto que se agrupan en las denominadas disciplinas (Favre, 2003).

1.7.3 Descripción de las Disciplinas

Dentro de la metodología RUP existen 9 disciplinas de manera estándar donde cada una categoriza las tareas a realizar. Estas pueden ser incrementadas o reducidas dependiendo de la necesidad del proyecto. Las disciplinas estándar

pueden ser clasificadas en dos grupos: De Ingeniería y De Soporte (Rossberg, 2014).

1.7.3.1 Disciplinas De Ingeniería

1.7.3.1.1 Modelado del Negocio

Uno de los mayores problemas al desarrollar un software es la falta de entendimiento con la Ingeniería del Negocio, es decir los procesos e involucrados detrás de la organización. El objetivo de esta disciplina es el de describir de manera clara a la organización en donde el sistema a desarrollar será utilizado y para lograrlo se encarga de proveer un lenguaje común como puede ser los diagramas de casos de uso.

1.7.3.1.2 Requerimientos

En esta disciplina se recolectan los requerimientos tanto funcionales (relacionados funcionamiento del sistema) como no funcionales (relacionados con hardware) de la organización. También se identifican los actores y en que procesos participan. De esta forma los desarrolladores y los clientes pueden quedar de acuerdo en lo que se va a realizar.

1.7.3.1.3 Análisis y Diseño

Esta disciplina se encarga de transformar los requerimientos recolectados en diseños del sistema que indican como el software será desarrollado en la fase de construcción, obteniendo así una abstracción del código fuente.

1.7.3.1.4 Implementación

La disciplina de implementación convierte los diseños del sistema en código fuente. Si se desarrollaron componentes por diferentes equipos, esta disciplina también se encarga de integrarlos. En cada iteración de RUP se puede reutilizar componentes ya elaborados previamente o integrarlos con los que se acaban de desarrollar.

1.7.3.1.5 Pruebas

Esta disciplina tiene varios propósitos que son:

- Verificar la correcta interacción entre los componentes desarrollados.
- Verificar que todos los requerimientos hayan sido correctamente implementados.
- Identificar y corregir defectos antes del despliegue del software.

Las pruebas que se realizan se basan en 4 dimensiones:

- Confiabilidad: Probabilidad de que el software no fallé a lo largo de un determinado tiempo.
- Funcionalidad: Que el software realice lo que tiene que hacer.
- Desempeño de la Aplicación: Que el software funcione de manera óptima.
- Desempeño del Sistema: Que el conjunto de aplicaciones funcione de manera óptima.

RUP propone que esta disciplina sea implementada a lo largo de todo el proyecto para así solucionar defectos de la manera más rápida para reducir costos.

1.7.3.1.6 Entrega

La actividad en esta disciplina se centra en entregar de manera exitosa los productos realizados, asegurándose que el software desarrollado sea instalado y puesto en funcionamiento.

1.7.3.2 Disciplinas de Soporte

1.7.3.2.1 Configuración de Cambios y Administración

RUP distingue tres áreas en esta disciplina. La primera es la Administración de Configuración, que se encarga de controlar las versiones de los artefactos producidos. La segunda es la Administración de Solicitud de Cambios, en donde se tiene un registro de las propuestas recibidas por versión de artefacto. La

tercer área es la Administración de Estatus de Cambio, la cual se encarga de registrar cuando un cambio es aprobado y el estado actual de dicho cambio (asignado, completado, etc.).

1.7.3.2.2 Administración del Proyecto

Dentro de las 2 dimensiones de RUP se encuentran cuatro fases y sus respectivas iteraciones. La Administración del Proyecto se encarga de planificar dichas fases, como es el número de iteraciones a realizar o como solucionar posibles riesgos a lo largo de cada fase y el proyecto. Esta disciplina no asegura el éxito del proyecto, pero si incrementa las posibilidades de entregar un software que satisfaga las expectativas del cliente.

1.7.3.2.3 Ambiente

El objetivo de esta disciplina es de indicar a los equipos de desarrollo las herramientas y procesos que van a ser utilizados para elaborar el proyecto. Las herramientas y/o procesos pueden ser: Software y hardware sobre el que se trabajará, estándares de diseño, estándares de programación, etc.

1.7.4 Descripción de las Fases

Las iteraciones que se realizan en RUP están organizadas en 4 fases: inicio, elaboración, construcción y transición; en las cuales se basará el desarrollo de esta disertación.

1.7.4.1 Inicio

El objetivo de esta fase es establecer el alcance del proyecto al igual que los requerimientos del mismo. En esta fase se identifica a todos los involucrados, al igual que posibles riesgos que se puedan ocurrir. Las salidas que generará esta fase serán:

- Diagramas de Casos de Uso.
- Detalle de los Diagramas de Casos de Uso.
- Diagramas de Actividades.

- Diagramas de Estados.
- Descripción de las iteraciones a realizar.

1.7.4.2 Elaboración

Esta fase tiene como propósito el analizar el dominio del problema desarrollando un plan de proyecto. Para esto se necesita de una amplia y profunda visión del sistema. Las salidas que generará esta fase serán:

- Diagramas de Clases.
- Diagramas de Secuencia.
- Diagramas de Colaboración.
- Modelo Entidad – Relación (Base de Datos).
- Diagramas de Paquetes.
- Diagramas de Componentes.
- Diagramas de Despliegue.

1.7.4.3 Construcción y Transición

Durante la fase de construcción se realizará el desarrollo del sistema basándose en los modelos y diagramas de las fases previas. La fase de transición tiene como propósito llevar a cabo la transición del software al usuario. Las salidas que generarán estas fases serán:

- Desarrollo del Software.
- Ejecutable del Software.
- Manuales de Usuario e Instalación.

1.7.5 Descripción de los Entregables

Cada una de las fases a desarrollar generan salidas conocidas también como entregables, las cuales serán descritas a continuación.

1.7.5.1 Diagramas de Casos de Usos

Los actores, procesos de una organización y las asociaciones entre ambos pueden representarse mediante diagramas de casos de uso, brindando así una visión general de la organización (Bittner, 2008). Los elementos utilizados en estos diagramas son:

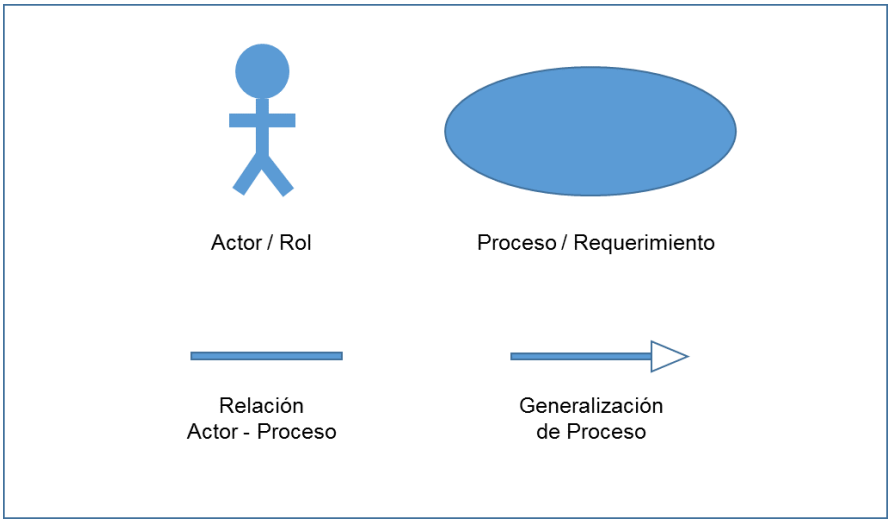

Figura 1.3. Elementos de Diagramas de Casos de Uso

- Actor: Es el que realiza o forma parte del proceso.
- Proceso: Acción o conjunto de acciones.
- Relación: Relaciona a los Actores con Procesos.
- Generalización: Indica que un Proceso es la especialización de otro Proceso Generalizado.

Ejemplo para entender de mejor manera estos diagramas:

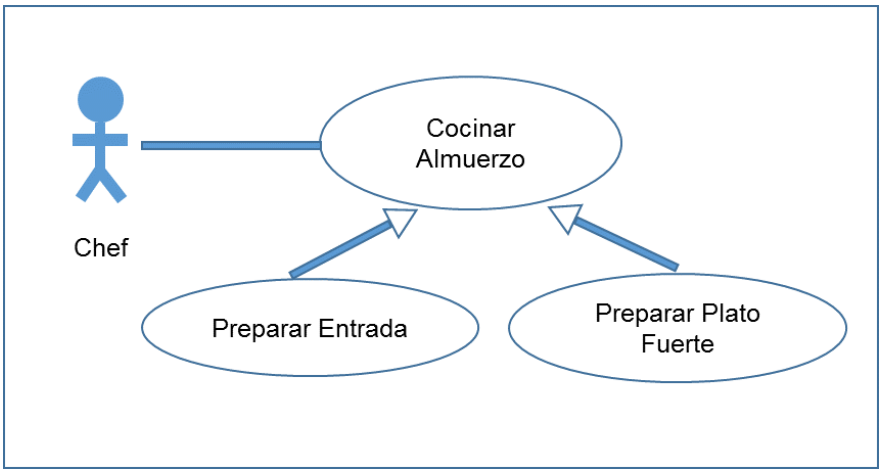

Figura 1.4. Ejemplo de Diagrama de Caso de Uso

1.7.5.2 Detalle de los Diagramas de Caso De Uso

Específica de mejor manera lo que está representado en los Casos de Uso, mostrando cual es el flujo de las acciones y las posibles excepciones que pueden llegar a darse. Ejemplo basado en la Figura 1.4:

Proceso: Preparar Plato Fuerte.

Flujo Principal:

1) Adobar filete de pollo.
2) Cortar papas.
3) Cortar vegetales.
4) Freír filete de pollo (E1).
5) Freír papas cortadas (E2).
6) Unir vegetales y formar la ensalada
7) Servir la comida en plato.

Excepciones:

(E1) El filete de pollo puede sobre-cocinarse y quemarse.

(E2) Las papas pueden sobre-cocinarse y quemarse.

De igual forma se pudiera especificar que se necesita que se utilice cierto tipo de sartén o alguna marca espacial de cuchillos (a manera de ejemplo), siendo estos los requerimientos no funcionales.

1.7.5.3 Diagramas de Actividades

Son muy similares a los flujogramas. El propósito de estos diagramas es clarificar los pasos involucrados y la toma de decisiones que se realizan en un determinado proceso, desde el inicio hasta su final (Schmuller, 2004). En estos diagramas se usan los siguientes elementos:

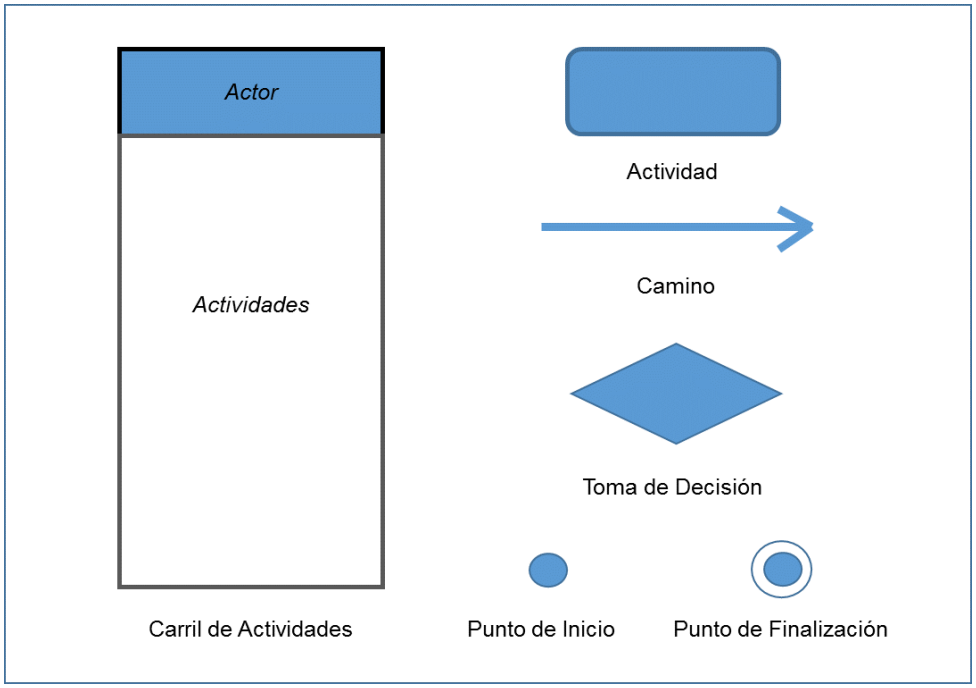

Figura 1.5. Elementos del Diagrama de Actividades

1.7.5.4 Diagramas de Estados

También conocidos como diagramas de transición, se encargan de mostrar los estados que adquieren los objetos a lo largo de un determinado proceso (Favre, 2003). Los elementos que se utilizan en estos diagramas son los siguientes:

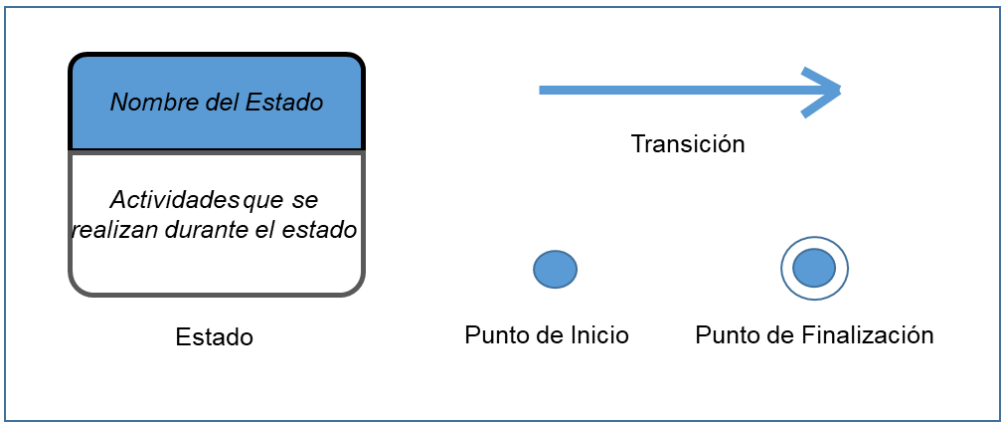

Figura 1.6. Elementos del Diagrama de Estados

1.7.5.5 Descripción de las Iteraciones a Realizar

Se indicará el número de veces que se realizará cada fase.

1.7.5.6 Diagramas de Clases

Los diagramas de clase ayudan a identificar la estructura estática implementada dentro de un sistema. Están conformados principalmente de dos elementos: las Clases, compuestas de atributos y operaciones, y las Relaciones entre clases (Holt, 2007). Los elementos utilizados en este modelo son:

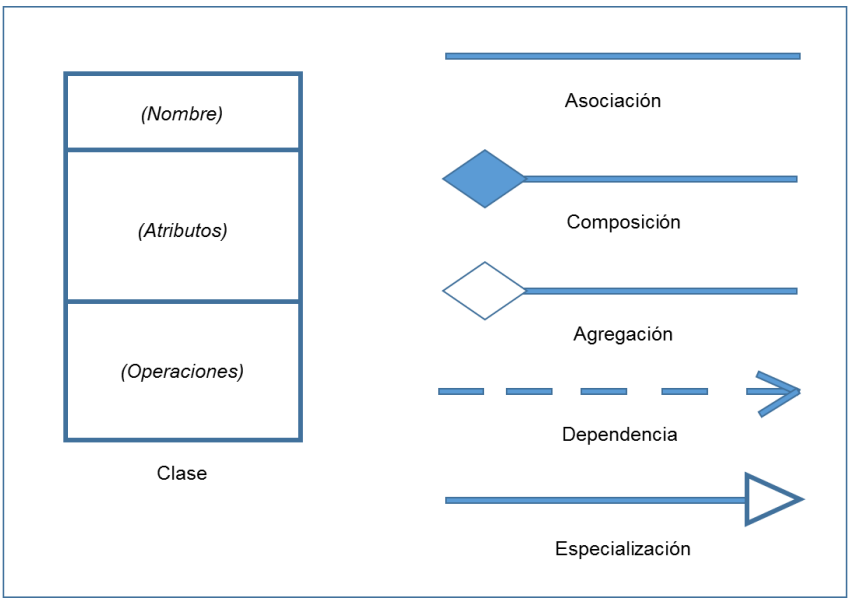

Figura 1.7. Elementos del Diagrama de Clases

- **Asociación:** Establece una simple relación entre clases.
- **Composición y Agregación:** Sirven para establecer la asociación múltiple que puede tener una clase con otra (*Ej. La clase empresa tiene 0 o múltiples empleados, siendo empleados otra clase*). La diferencia entre ambas relaciones radica en la necesidad que tiene para existir una clase de la otra. Un ejemplo de relación de composición es la de una clase Empresa con otra clase Empleados, la clase Empresa posee varios empleados, pero si esta dejase de existir

los empleados de la misma también lo harían. Un ejemplo de relación de agregación puede ser la misma clase Empresa pero esta vez relacionada con otra clase Clientes, la Empresa agrupa varios clientes y si dejase de existir, los Clientes no dejarían de comprar en otras empresas.

- **Dependencia:** Relación utilizada para indicar que una clase (B) utiliza atributos u operaciones de otra clase (A). Cualquier cambio que se haga en la clase no dependiente (A), se verá reflejado en la clase dependiente (B).
- **Especialización:** Relación utilizada para indicar que una clase (B) puede tener los mismos (o más) atributos y/u operaciones de otra clase (A).

1.7.5.7 Diagramas de Secuencia

Estos diagramas sirven para brindar una vista dinámica del sistema a implementar. Para lograrlo, muestran el tiempo y los mensajes que se envían entre entidades al realizar operaciones (Schmuller, 2004). En estos diagramas se utilizan con frecuencia los siguientes elementos:

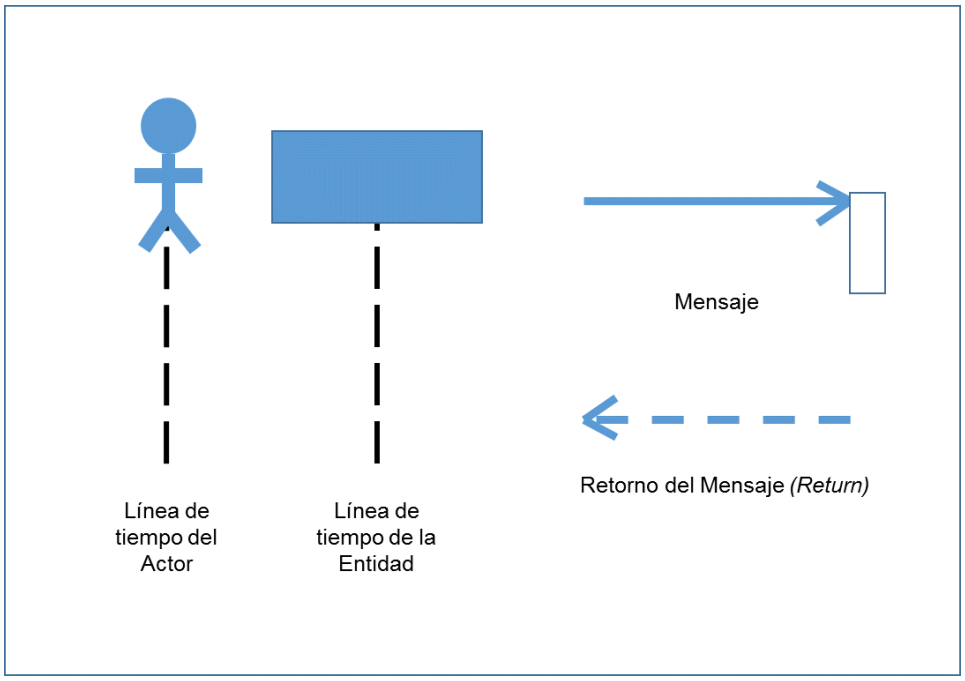

Figura 1.8. Elementos del Diagrama de Secuencia

1.7.5.8 Diagramas de Colaboración

Conocidos también como diagramas de comunicación, estos diagramas son muy similares a los diagramas de secuencia, la diferencia radica en que se enfocan en mostrar las relaciones de las funciones con los objetos (Schmuller, 2004). Sus elementos son:

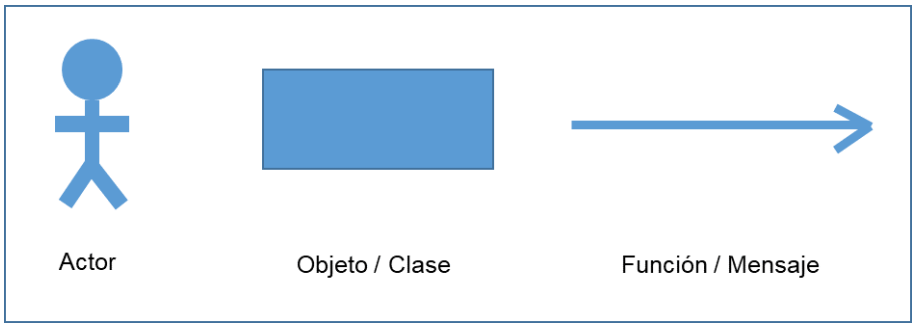

Figura 1.9. Elementos del Diagrama de Colaboración

1.7.5.9 Modelo Entidad –Relación (Base De Datos)

Este modelo ayudará a visualizar cómo está constituida la base de datos en donde se almacenará la información. Existen varias formas de representar un modelo E-R, pero el que se utilizará en está disertación será el conocido como “Patas de Gallo” (Harrington, 2009). Los elementos de este modelo son los siguientes:

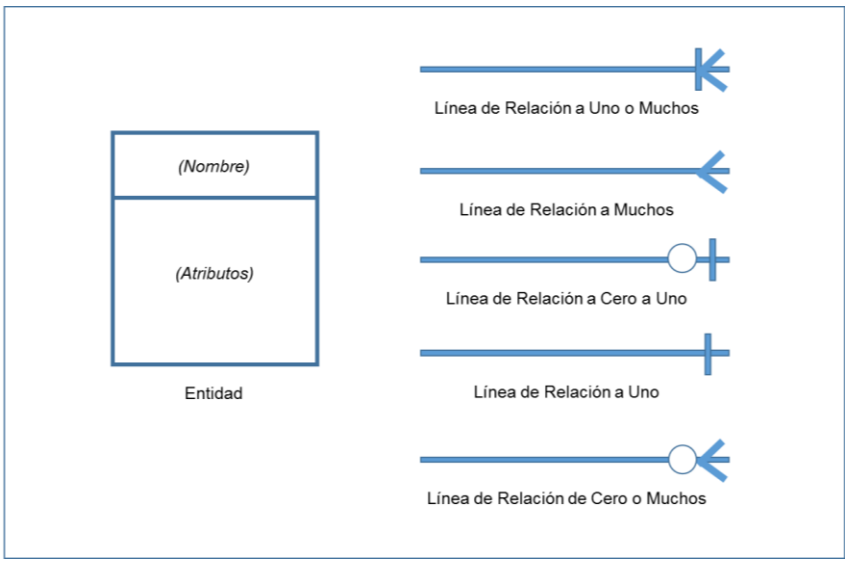

Figura 1.10. Elementos del Diagrama E-R

Para indicar las relaciones que las entidades tienen entre sí, se utilizan las líneas de relación. En los siguientes capítulos se verá la utilización de las mismas.

1.7.5.10 Diagramas de Paquetes

Como el nombre lo indica, los diagramas de paquetes están diseñados para agrupar elementos, como pueden ser clases o casos de uso y uno de sus principales usos es el de representar de mejor manera las capas de un sistema (Schmuller, 2004).

Estos diagramas también pueden utilizar las relaciones de dependencia y generalización presentadas anteriormente y se representan con una carpeta tabulada como lo muestra la Figura 1.8.

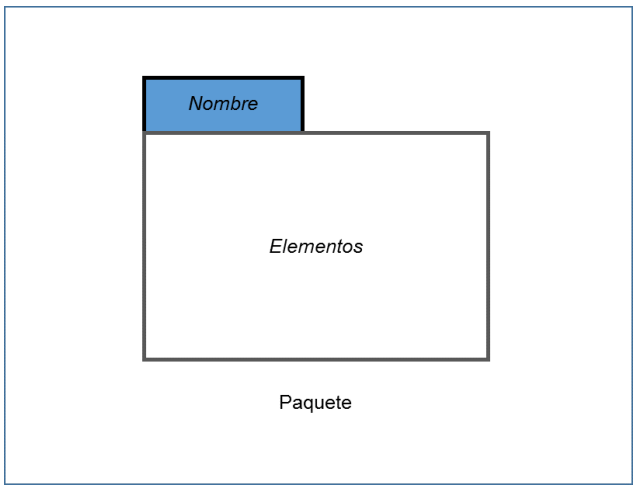

Figura 1.11. Icono de Paquete.

1.7.5.11 Diagramas de Componentes

Estos diagramas se encargan de describir los componentes físicos y de software que se utilizan dentro de un sistema, incluyendo sus relaciones (Holt, 2007). Para realizar estos diagramas se utilizan los siguientes elementos:

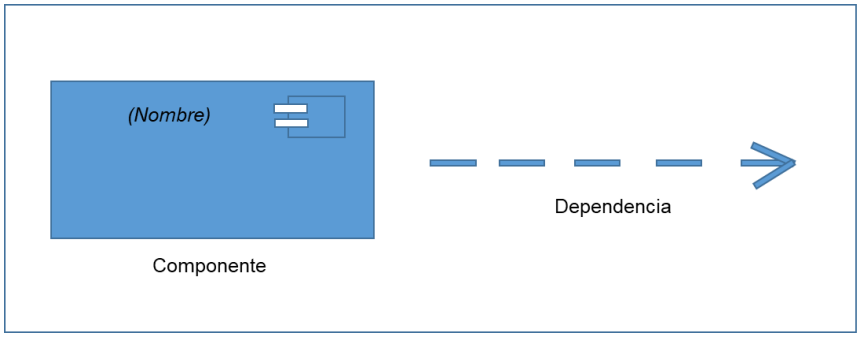

Figura 1.12. Elementos del Diagrama de componentes

1.7.5.12 Diagramas de Despliegue

Los diagramas de despliegue contienen a los diagramas de componentes. Su función es indicar en qué nodo (en que dispositivo real) se van a ejecutar los componentes (Favre, 2003). El diseño de estos diagramas se realiza con los mismos elementos de un diagrama de componentes, incluyendo uno nuevo con forma de cubo, conocido como nodo, de la siguiente manera:

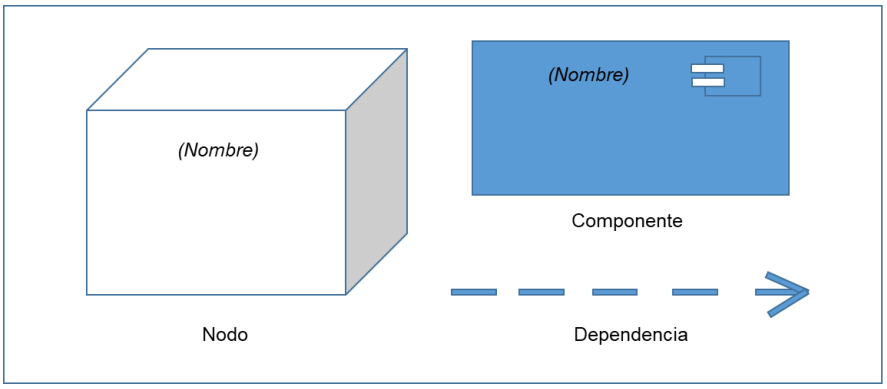

Figura 1.13. Elementos del Diagrama de componentes

1.7.5.13 Ejecutable del Software

Se entregarán todos los elementos necesarios para ejecutar el software.

1.7.5.14 Manuales de Usuario e Instalación.

Contendrán las guías necesarias para poder instalar el software y utilizarlo.

CAPÍTULO 2: FASE DE INICIO

Este capítulo presenta la recolección de requerimientos y una descripción de las iteraciones que se realizarán.

2.1 LEVANTAMIENTO DE REQUERIMIENTOS

En el campo de la medicina, un diagnóstico bien realizado depende fuertemente de una buena historia clínica y esta no es la excepción en la oftalmología. Uno de los elementos principales en la historia clínica es la ficha médica. Existen distintos datos que se deben registrar en estas fichas, mas no todos son registrados necesariamente por un especialista. Una ficha médica es dinámica, es decir que va cambiando según el paciente vaya asistiendo a las consultas, por eso es necesario almacenar las evoluciones que va teniendo el paciente, llevando así un historial de cambios realizados en la ficha (Shaw, 2016).

Podemos tomar estas necesidades y decir que los requerimientos del sistema son los siguientes:

RF0: Ingreso al sistema.

RF1: El sistema permitirá administrar personas.

RF2: El sistema permitirá administrar fichas médicas.

RF3: El sistema permitirá administrar consultas.

2.2 DIAGRAMAS DE CASOS DE USO

2.2.1 Diagrama de Casos de Uso General

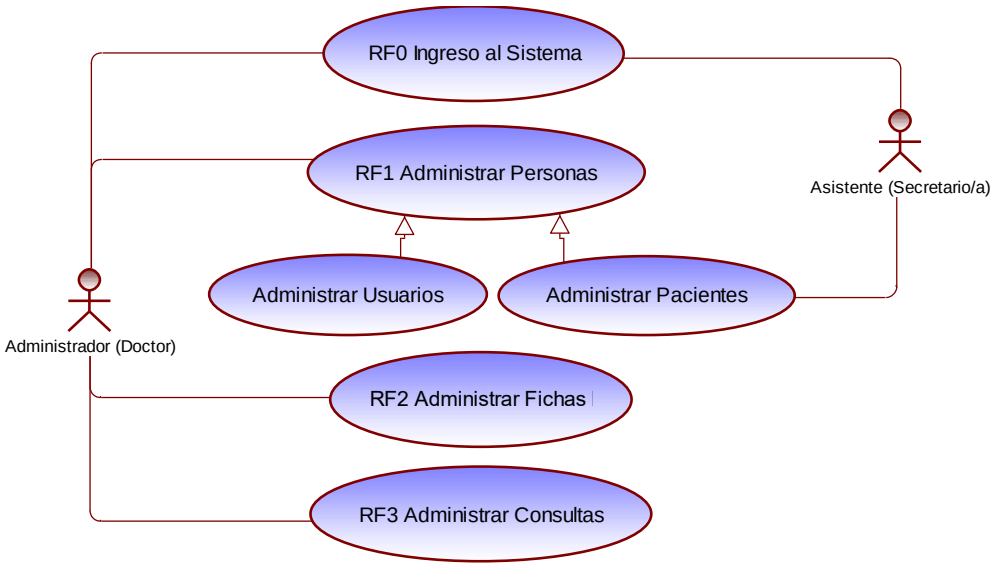

Figura 2.1. Diagrama de Casos de Uso General

2.2.2 Detalle RF1 Administrar Personas

El administrador podrá ingresar, modificar y eliminar las personas que se encuentren en el sistema. El asistente únicamente será capaz de ingresar pacientes y modificar sus datos. Ambos podrán realizar consultas de las personas a nivel general o detallado.

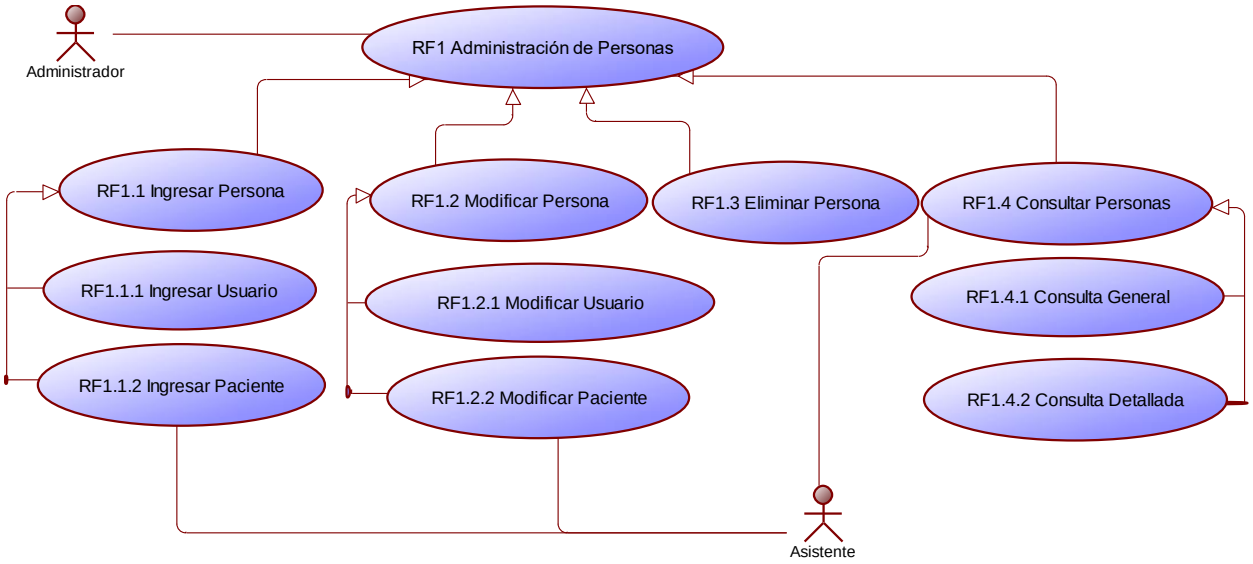

Figura 2.2. Diagrama detallado RF1.

2.2.2.1 Detalle RF1.1 Ingresar Persona

El administrador podrá ingresar cualquier persona, ya sea otro administrador, un asistente o un paciente. El asistente únicamente podrá ingresar pacientes.

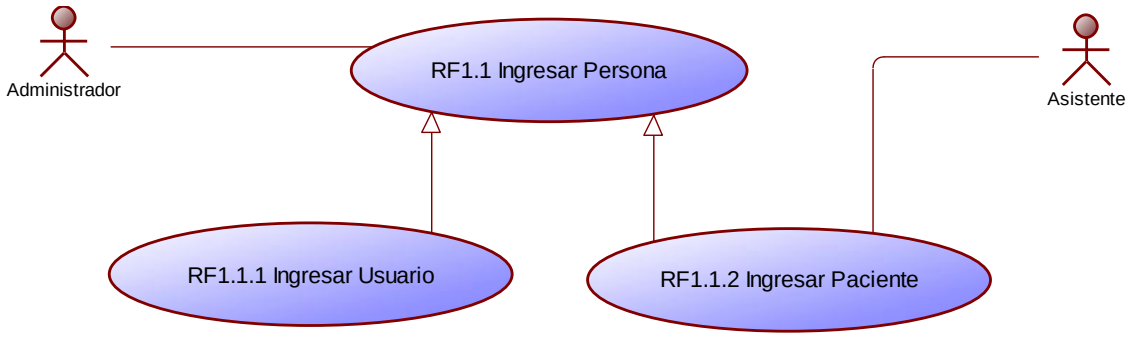

Figura 2.3. Diagrama detallado RF1.1.

Flujo principal:

1. El actor selecciona la opción “Administrar Personas”.
2. Se despliega una ventana con las opciones para administrar personas.
3. El actor selecciona la opción “Ingresar Personas”.
4. Se despliega una ventana para ingresar personas.
5. El actor ingresa el número de identificación de la persona.
6. Validar número de identificación (E1).
7. El actor ingresa fotografía de la persona (en caso de tenerla).
8. El actor ingresa nombres de la persona.
9. El actor ingresa apellidos de la persona.
10. El actor ingresa lugar de nacimiento de la persona.
11. El actor ingresa fecha de nacimiento de la persona.
12. El actor ingresa teléfono fijo de la persona.
13. El actor ingresa teléfono celular de la persona.
14. El actor ingresa dirección de residencia de la persona.
15. El actor selecciona los roles de la persona (F1).
16. El actor pulsa el botón guardar (E2).
17. Sistema almacena datos (E3).

Flujo alterno:

F1: Solo se podrá seleccionar los roles si el actor es “Administrador”, caso contrario la persona a ingresar será un paciente. Si se selecciona el rol de Doctor o Asistente, se mostrará dos cuadros de texto para ingresar el nombre de usuario y su contraseña para acceder al sistema. El nombre de usuario será validado para verificar que no exista uno igual.

Excepciones:

E1: Si ya existe el número de identificación se mostrará un mensaje indicando que no se puede ingresar nuevamente.

E2: No se permitirá el ingreso si no se registraron los campos obligatorios.

E3: Si no se pudo guardar la información en la base de datos se mostrará un mensaje de alerta.

2.2.2.2 Detalle RF1.2 Modificar Persona

El administrador podrá modificar cualquier persona, ya sea otro administrador, un asistente o un paciente. El asistente únicamente podrá modificar pacientes.

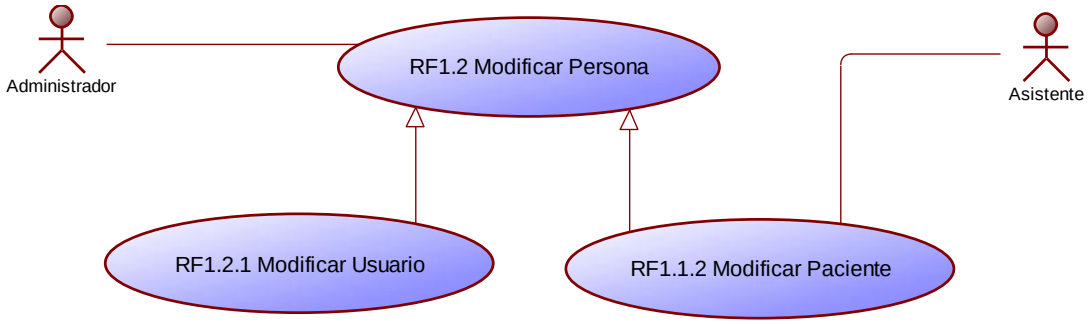

Figura 2.4. Diagrama detallado RF1.2.

Flujo principal:

1. El actor selecciona la opción “Administrar Personas”.
2. Se despliega una ventana con las opciones para administrar personas.
3. El actor selecciona la opción “Modificar Persona”.
4. Se despliega una ventana para modificar personas.
5. El actor ingresa el número de identificación de la persona.
6. Validar número de identificación (E1) (E2).
7. Se despliega los datos de la persona que será modificada.
8. El actor modifica el o los datos de la persona.
9. El actor modifica los roles de la persona (F1).
10. El actor pulsa el botón guardar (E3) (E4).
11. Sistema almacena datos (E5).

Flujo alterno:

F1: Solo se podrá modificar los roles de la persona si el actor es un Administrador.

Excepciones:

E1: Si no existe el número de identificación se mostrará un mensaje indicando que no se encontró la persona.

E2: Si el actor es un “Asistente” únicamente podrá modificar los datos de otros pacientes.

E3: No se permitirá el ingreso si se dejaron vacíos campos obligatorios.

E4: No se permitirá que la primera persona registrada en el sistema se quede si el rol de Administrador.

E5: Si no se pudo guardar la información en la base de datos se mostrará un mensaje de alerta.

2.2.2.3 Detalle RF1.3 Eliminar Persona

El administrador podrá eliminar personas que se encuentren dentro del sistema.

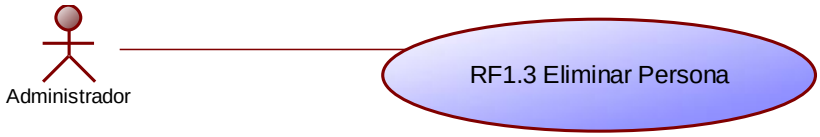

Figura 2.5. Diagrama detallado RF1.3.

Flujo principal:

1. El actor selecciona la opción “Administrar Personas”.
2. Se despliega una ventana con las opciones para administrar personas.
3. El actor selecciona la opción “Eliminar Persona”.
4. Se despliega una ventana para eliminar personas.
5. El actor ingresa el número de identificación de la persona.
6. Validar número de identificación (E1).
7. Se despliega una ventana con la información de la persona a borrar.
8. El actor pulsa el botón borrar (E2) (E3).
9. Sistema almacena datos.

Excepciones:

E1: Si no existe el número de identificación se mostrará un mensaje indicando que no existe la persona.

E2: No se podrá eliminar la primera persona registrada en el sistema como "Administrador".

E3: El actor no podrá eliminarse a sí mismo.

2.2.2.4 Detalle RF1.4.1 Consulta General

Tanto el administrador como el asistente podrán realizar consultas de manera general, es decir, podrán buscar personas por nombres o apellidos. También lo podrán hacer de manera detallada buscando por el número de identificación.

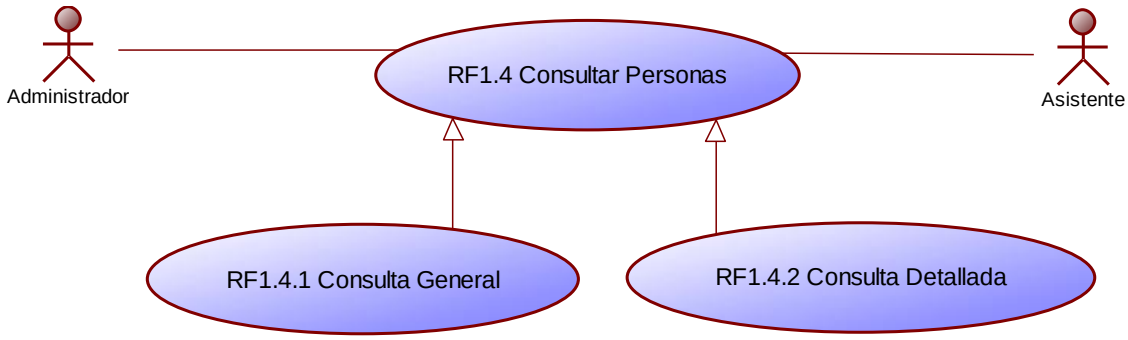

Figura 2.6. Diagrama detallado RF1.4.1.

Flujo principal:

1. El actor selecciona la opción "Administrar Personas".
2. Se despliega una ventana con las opciones para administrar personas.
3. El actor selecciona la opción "Consultar Persona".
4. Se despliega una ventana para buscar personas.
5. El actor selecciona la opción por la que desea buscar personas (por apellido, nombre u identificación).

5.1. En el caso de haber de haber realizado una búsqueda por apellido o nombre se mostrará en una tabla la información general de las personas que cumplen con los parámetros de búsqueda.

5.2. En el caso de haber realizado una búsqueda por número de identificación se mostrará la información de persona (E1).

Excepciones:

E1: Si no existe ninguna persona con el número de identificación ingresado se mostrará un mensaje.

2.2.3 Detalle RF2 Administrar Fichas Médicas

El administrador podrá consultar las fichas médicas del sistema, modificarlas y consultar registros de fichas médicas anteriores.

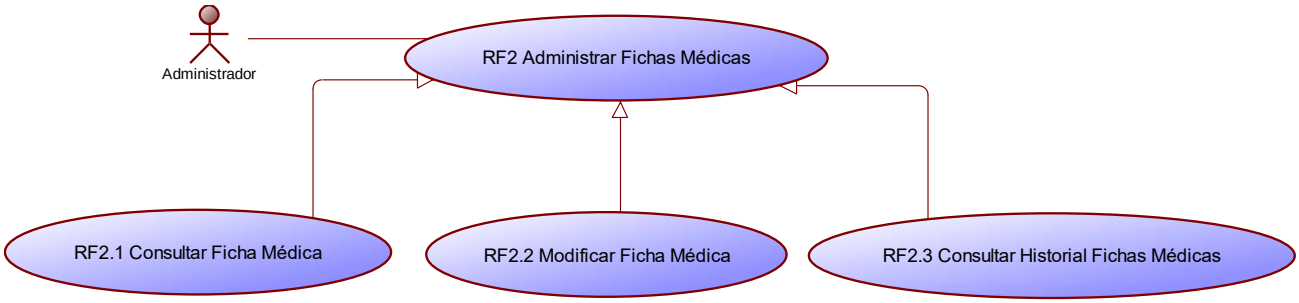

Figura 2.7. Diagrama detallado RF2.

2.2.3.1 Detalle RF2.1 Consultar Ficha Médica

El administrador podrá consultar la ficha médica de un paciente ingresando su número de cédula.

Figura 2.8. Diagrama detallado RF2.1.

Flujo principal:

1. El administrador selecciona la opción “Administrar Fichas Médicas”.
2. Se despliega una ventana para buscar la ficha médica de una persona por su número identificación.
3. El administrador ingresa el número de identificación.
4. Validar número de identificación (E1).
5. Se muestra la ficha médica de la persona.

Excepciones:

E1: Si no existe el número de identificación se mostrará un mensaje indicando que no se encontró la ficha de la persona. Si existe el número de identificación pero no tiene ficha médica se mostrará un mensaje indicando que la persona se encuentra en el sistema pero no es paciente.

2.2.3.2 Detalle RF2.2 Modificar Ficha Médica

El administrador podrá modificar los datos de la ficha médica.

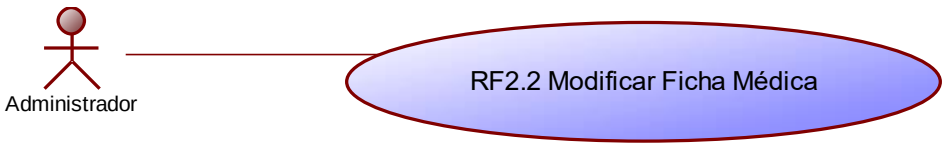

Figura 2.9. Diagrama detallado RF2.2.

Flujo principal:

1. El administrador selecciona la opción “Administrar Fichas Médicas”.
2. Se despliega una ventana para buscar la ficha médica de una persona por su número identificación.
3. El administrador ingresa el número de identificación.
4. Validar número de identificación (E1).
5. Se muestra la ficha médica de la persona.
6. El administrador realiza cambios a los campos de la ficha.
7. El administrador pulsa el botón guardar. (E2).
8. Se despliega un mensaje de confirmación.
9. Sistema almacena datos (E3).

Excepciones:

E1: Si no existe el número de identificación se mostrará un mensaje indicando que no se encontró la ficha de la persona. Si existe el número de identificación pero tiene no ficha médica se mostrará un mensaje indicando que la persona se encuentra en el sistema pero no es paciente.

E2: No se permitirá el ingreso si se dejaron vacíos campos obligatorios.

E3: Si no se pudo guardar la información en la base de datos se mostrará un mensaje de alerta.

2.2.3.3 Detalle RF2.3 Consultar Historial Fichas Médicas

El administrador podrá consultar los cambios realizados en la ficha médica.

Figura 2.10. Diagrama detallado RF2.3.

Flujo principal:

1. El administrador selecciona la opción “Administrar Fichas Médicas”.
2. Se despliega una ventana para buscar la ficha médica de una persona por su número identificación.
3. El administrador ingresa el número de identificación.
4. Validar número de identificación (E1).
5. Se muestra la ficha médica de la persona.
6. El administrador pulsa el botón de historial.
7. Se despliega una ventana con el listado de cambios a la ficha médica por fecha (E2).
8. El administrador selecciona una ficha del listado para cargarla.
9. Se despliega una ventana con la ficha médica de la fecha seleccionada.

Excepciones:

E1: Si no existe el número de identificación se mostrará un mensaje indicando que no se encontró la ficha de la persona. Si existe el número de identificación pero tiene no ficha médica se mostrará un mensaje indicando que la persona se encuentra en el sistema pero no es paciente.

E2: En el caso de no haber cambios en la ficha médica no se muestra nada en el listado.

2.2.4 Detalle RF3 Administrar Consultas

El administrador podrá registrar consultas médicas, modificarlas o eliminarlas del sistema.

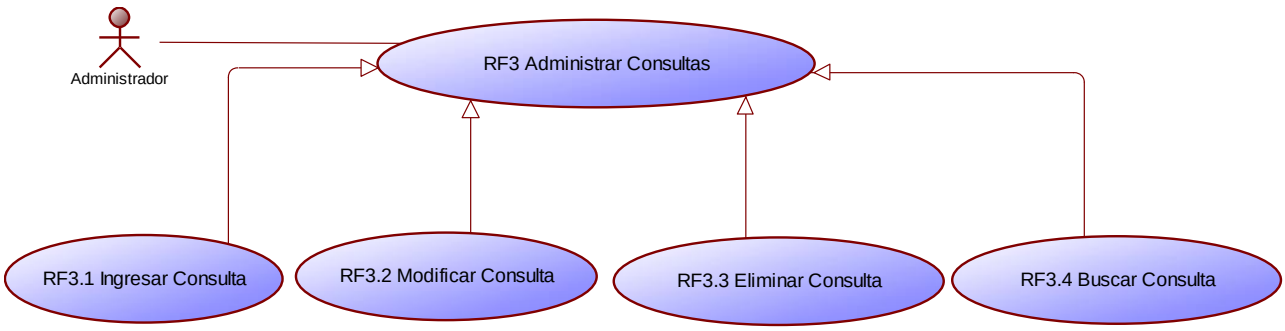

Figura 2.11. Diagrama detallado RF3.

2.2.4.1 Detalle RF3.1 Ingresar Consulta

El administrador podrá ingresar una consulta.

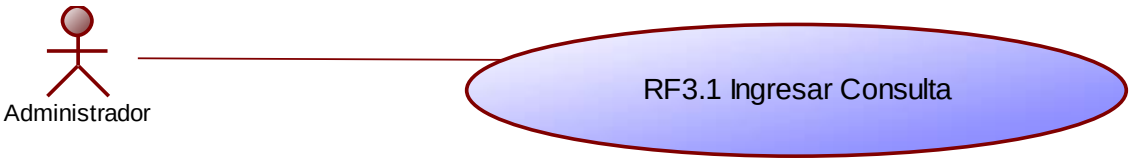

Figura 2.12. Diagrama detallado RF3.1.

Flujo principal:

1. El administrador selecciona la opción “Administrar Fichas Médicas”.
2. Se despliega una ventana para buscar la ficha médica de una persona por su número identificación.
3. El administrador ingresa el número de identificación.
4. Validar número de identificación (E1).
5. Se muestra la ficha médica de la persona.
6. El administrador pulsa el botón de ingresar consulta.
7. Se despliega una ventana para ingresar una consulta
8. El administrador ingresa el motivo de la consulta.
9. El administrador ingresa el diagnostico.
10. El administrador ingresa el tratamiento.
11. El administrador pulsa el botón guardar (E2).
12. Sistema almacena datos (E3).
13. Se muestra un mensaje para imprimir la consulta.
14. El administrador escoge si desea imprimir o no la consulta.

Excepciones:

E1: Si no existe el número de identificación se mostrará un mensaje indicando que no se encontró la ficha de la persona. Si existe el número de identificación pero no tiene ficha médica se mostrará un mensaje indicando que la persona se encuentra en el sistema pero no es paciente.

E2: No se permitirá el ingreso si se dejaron vacíos campos obligatorios.

E3: Si no se pudo guardar la información en la base de datos se mostrará un mensaje de alerta.

2.2.4.2 Detalle RF3.2 Modificar Consulta

El administrador podrá modificar una consulta ingresada.

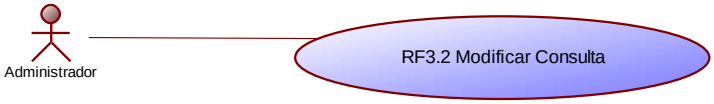

Figura 2.13. Diagrama detallado RF3.2.

Flujo principal:

1. El administrador selecciona la opción “Administrar Fichas Médicas”.
2. Se despliega una ventana para buscar la ficha médica de una persona por su número identificación.
3. El administrador ingresa el número de identificación.
4. Validar número de identificación (E1).
5. Se muestra la ficha médica de la persona.
6. El administrador pulsa el botón de consultas previas.
7. Se despliega una ventana con el listado de consultas previas (E2).
8. El administrador selecciona una consulta del listado para cargarla.
9. Se despliega una ventana con la consulta de la fecha seleccionada.
10. El administrador realiza cambios en la ficha médica.
11. El administrador pulsa el botón guardar (E3).
12. Sistema almacena datos (E4).
13. Se muestra un mensaje para imprimir la consulta.
14. El administrador escoge si desea imprimir o no la consulta.

Excepciones:

E1: Si no existe el número de identificación se mostrará un mensaje indicando que no se encontró la ficha de la persona. Si existe el número de identificación pero no tiene ficha médica se mostrará un mensaje indicando que la persona se encuentra en el sistema pero no es paciente.

E2: En caso de no haber consultas previas no se muestra nada en el listado.

E3: No se permitirá el ingreso si se dejaron vacíos campos obligatorios.

E4: Si no se pudo guardar la información en la base de datos se mostrará un mensaje de alerta.

2.2.4.3 Detalle RF3.3 Eliminar Consulta.

El administrador podrá eliminar consultas.

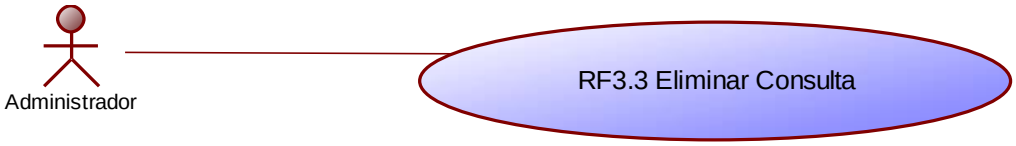

Figura 2.14. Diagrama detallado RF3.3.

Flujo principal:

1. El administrador selecciona la opción “Administrar Fichas Médicas”.
2. Se despliega una ventana para buscar la ficha médica de una persona por su número identificación.
3. El administrador ingresa el número de identificación.
4. Validar número de identificación (E1).
5. Se muestra la ficha médica de la persona.
6. El administrador pulsa el botón de consultas previas.
7. Se despliega una ventana con el listado de consultas previas por fecha (E2).
8. El administrador selecciona una consulta del listado.
9. El administrador pulsa el botón borrar.
10. Se despliega un mensaje de confirmación.
11. Sistema almacena datos (E3).

Excepciones:

E1: Si no existe el número de identificación se mostrará un mensaje indicando que no se encontró la ficha de la persona. Si existe el número de identificación pero no tiene ficha médica se mostrará un mensaje indicando que la persona se encuentra en el sistema pero no es paciente.

E2: En el caso de no haber consultas previas no se muestra nada en el listado.

E3: Si no se pudo guardar la información en la base de datos se mostrará un mensaje de alerta.

2.2.4.4 Detalle RF3.4 Buscar Consulta.

El administrador podrá buscar una consulta y ver su detalle.

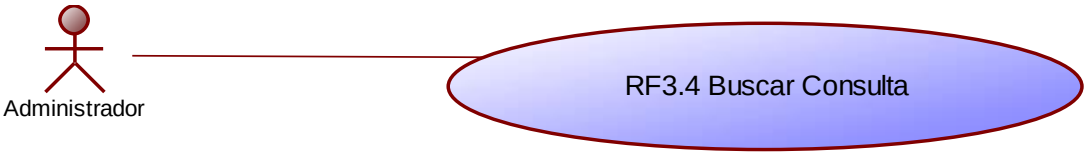

Figura 2.15 Diagrama detallado RF3.4.

Flujo principal:

1. El administrador selecciona la opción “Administrar Fichas Médicas”.
2. Se despliega una ventana para buscar la ficha médica de una persona por su número identificación.
3. El administrador ingresa el número de identificación.
4. Validar número de identificación (E1).
5. Se muestra la ficha médica de la persona.
6. El administrador pulsa el botón de consultas previas.
7. Se despliega una ventana con el listado de consultas previas por fecha (E2).
8. El administrador selecciona una consulta del listado para cargarla.
9. Se despliega una ventana con la consulta de la fecha seleccionada.
10. El administrador puede imprimir la consulta si desea.

Excepciones:

E1: Si no existe el número de identificación se mostrará un mensaje indicando que no se encontró la ficha de la persona. Si existe el número de identificación pero tiene no ficha médica se mostrará un mensaje indicando que la persona se encuentra en el sistema pero no es paciente.

E2: En el caso de no haber consultas previas no se muestra nada en el listado.

2.3 DIAGRAMA DE ACTIVIDADES

De acuerdo a la sección 1.7.5.3 del primer capítulo, el diagrama de actividades de la presente disertación se ilustra en la siguiente figura.

2.3.1. Consulta Médica.

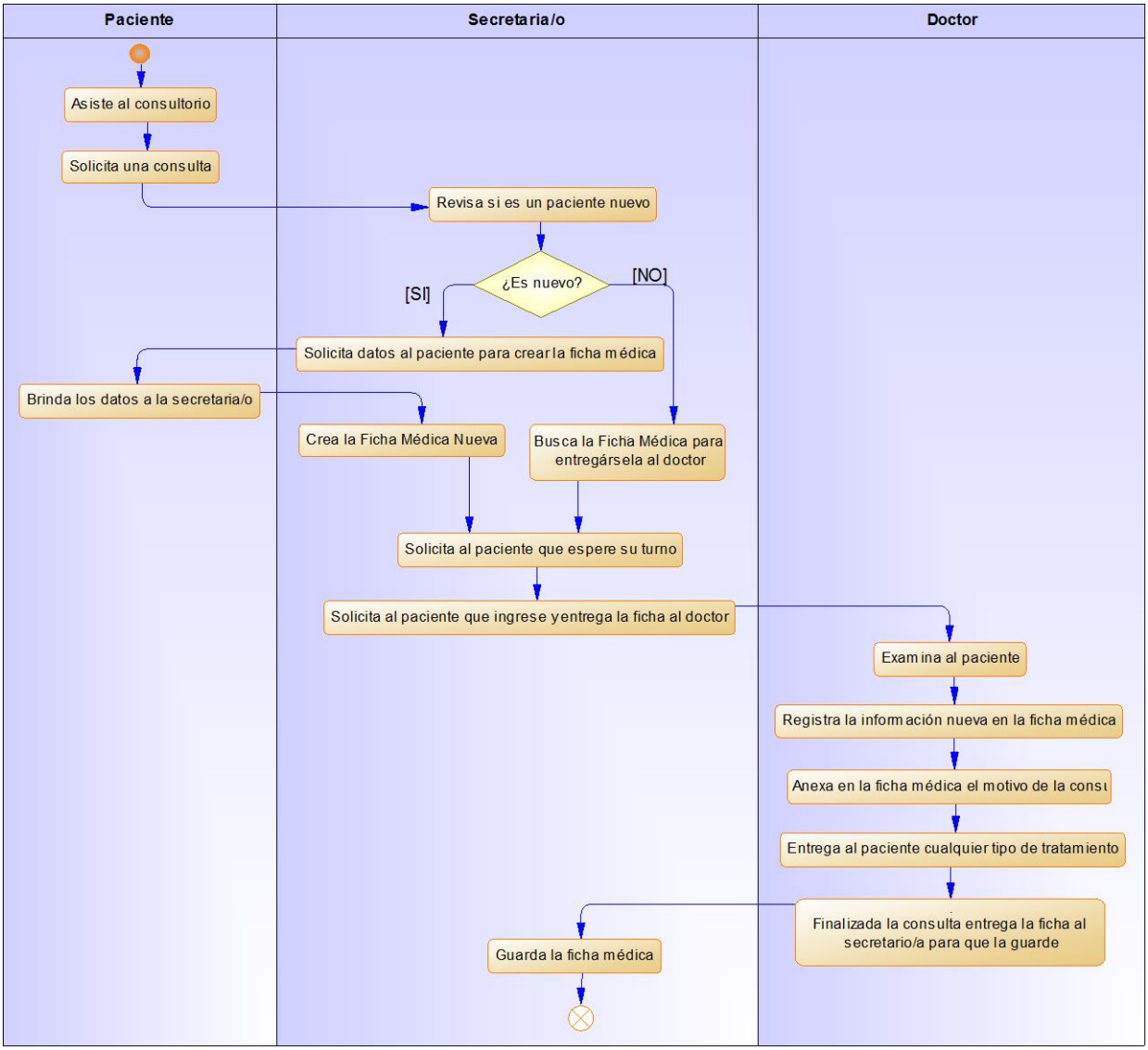

Figura 2.16. Diagrama de Actividad Consulta Médica.

2.4 DIAGRAMA DE ESTADOS

De acuerdo a la sección 1.7.5.4 del primer capítulo, el diagrama de estados de la presente disertación se ilustra en la siguiente figura.

2.4.1. Consulta Médica.

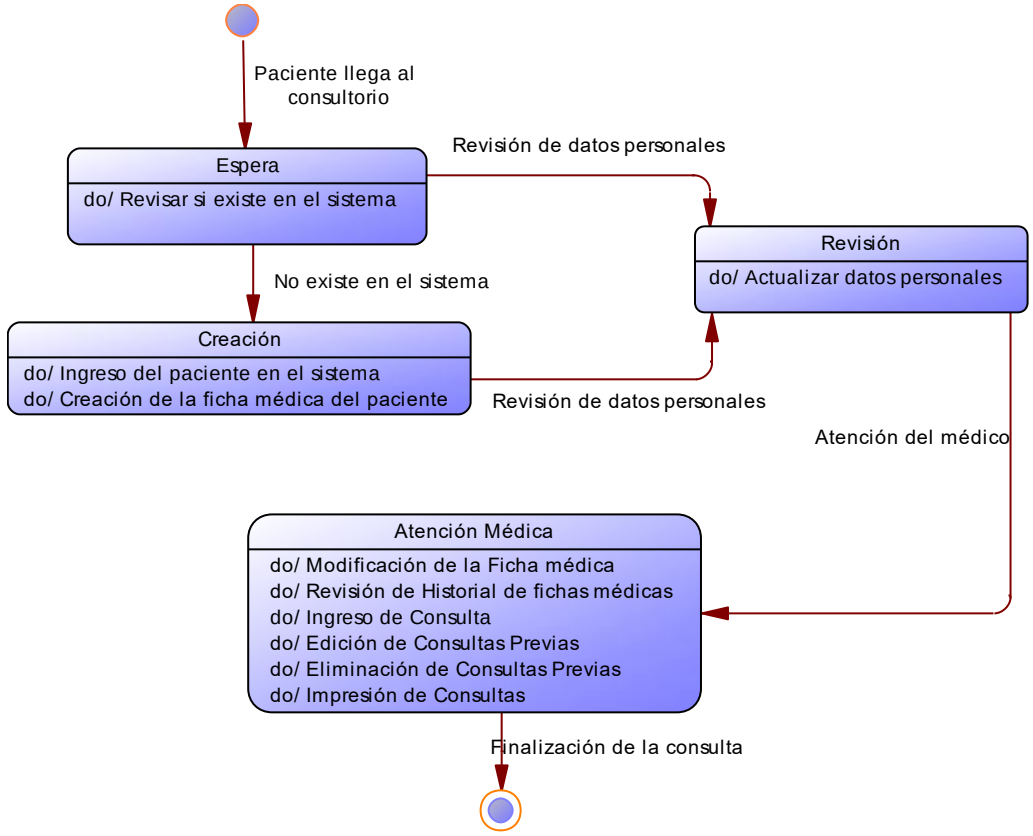

Figura 2.17. Diagrama de Estados Consulta Médica.

2.4 PLAN DE PROYECTO

La metodología RUP es iterativa, es decir que las fases que la conforman se repiten un número determinado de veces para alcanzar un producto de calidad ajustado por completo a los requerimiento del usuario (Rossberg, 2014). En cada fase se trata de elaborar elementos reutilizables para facilitar el trabajo en las siguientes iteraciones.

En esta disertación se realizará una única iteración por cada fase, tratando de generar elementos reutilizables.

CAPÍTULO 3: FASE DE ELABORACIÓN

En este capítulo se presenta los diagramas que servirán para brindar una visión de cómo estará desarrollado el sistema.

3.1 DIAGRAMA DE CLASES

De acuerdo a la sección 1.7.5.6 del primer capítulo, los diagramas de clases para el desarrollo de la presente disertación se ilustran en las figuras 3.1, 3.2, 3.3 y 3.4.

3.1.1 Diagrama General de Clases

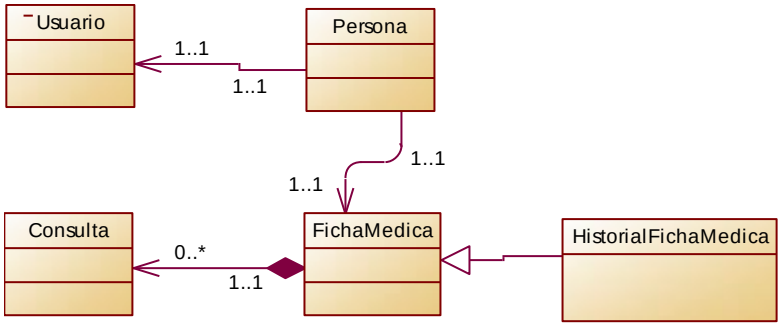

Figura 3.1. Diagrama General de Clases.

3.1.2 Diagrama Detallado de Clases GUI

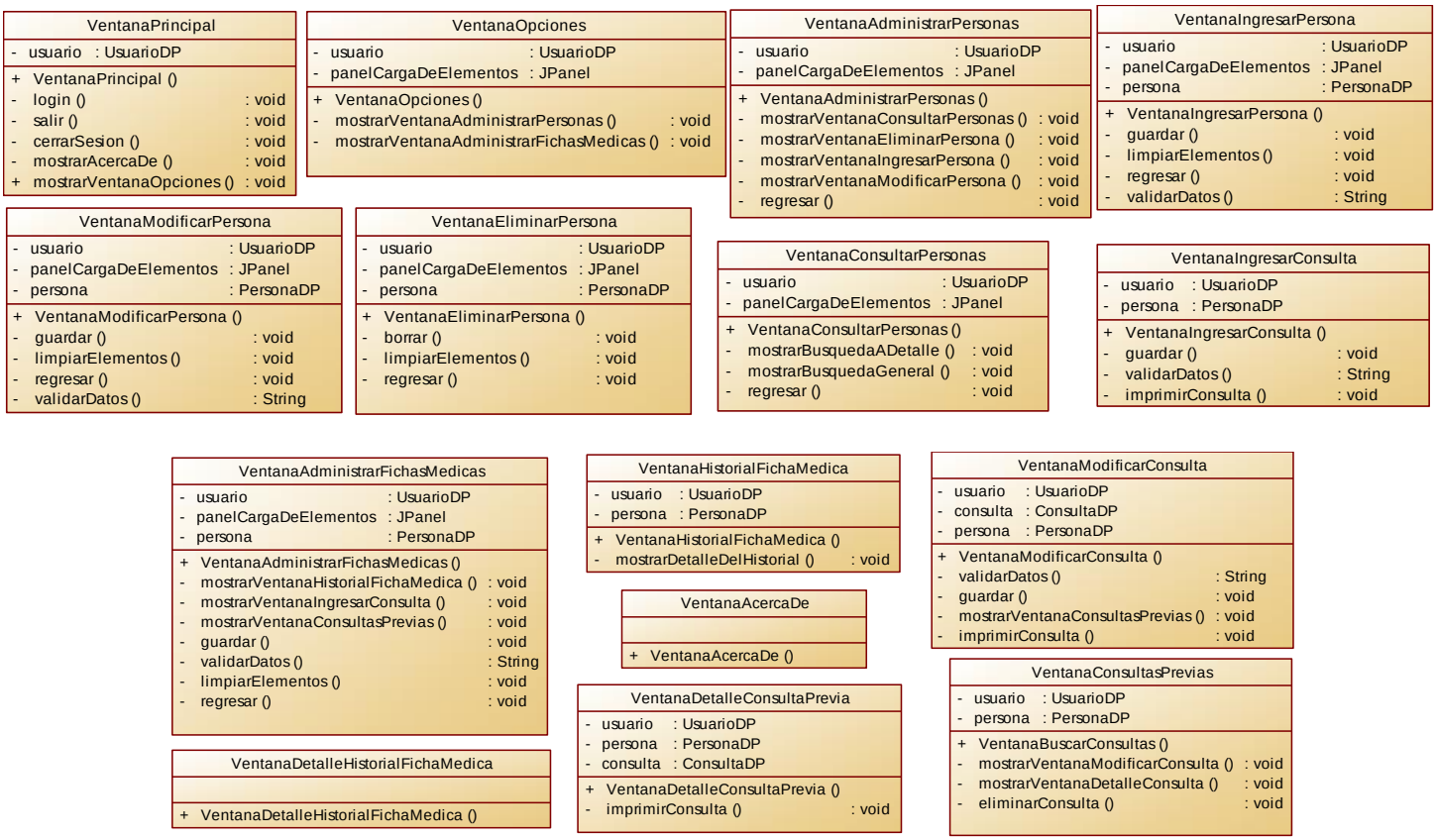

Figura 3.2. Diagrama Detallado de Clases GUI.

3.1.3 Diagrama Detallado de Clases DP

Figura 3.3. Diagrama Detallado de Clases DP.

3.1.4 Diagrama Detallado de Clases MD

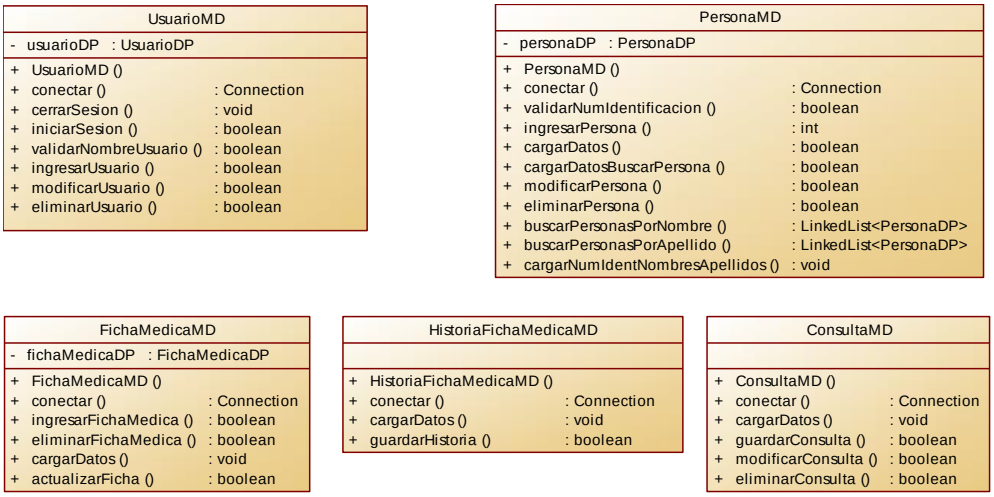

Figura 3.4. Diagrama Detallado de Clases MD.

3.2 DIAGRAMAS DE SECUENCIA

De acuerdo a la sección 1.7.5.7 del primer capítulo, los diagramas de secuencia para el desarrollo de la presente disertación se ilustran en las siguientes figuras.

3.2.1 RF0 Ingreso al Sistema.

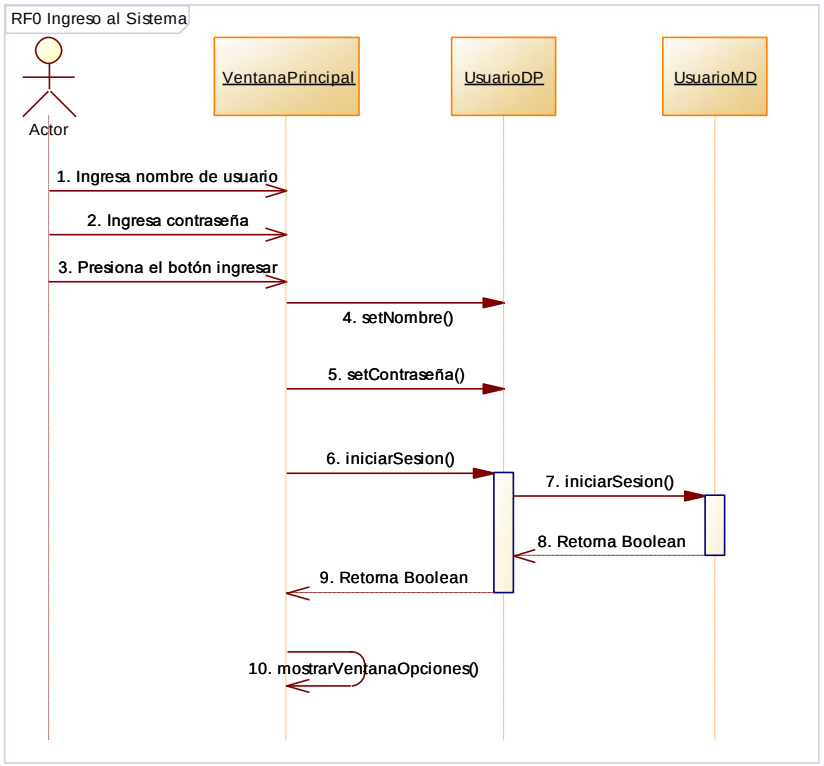

Figura 3.5. Diagrama de Secuencia RF0.

3.2.2 RF1.1 Ingresar Persona.

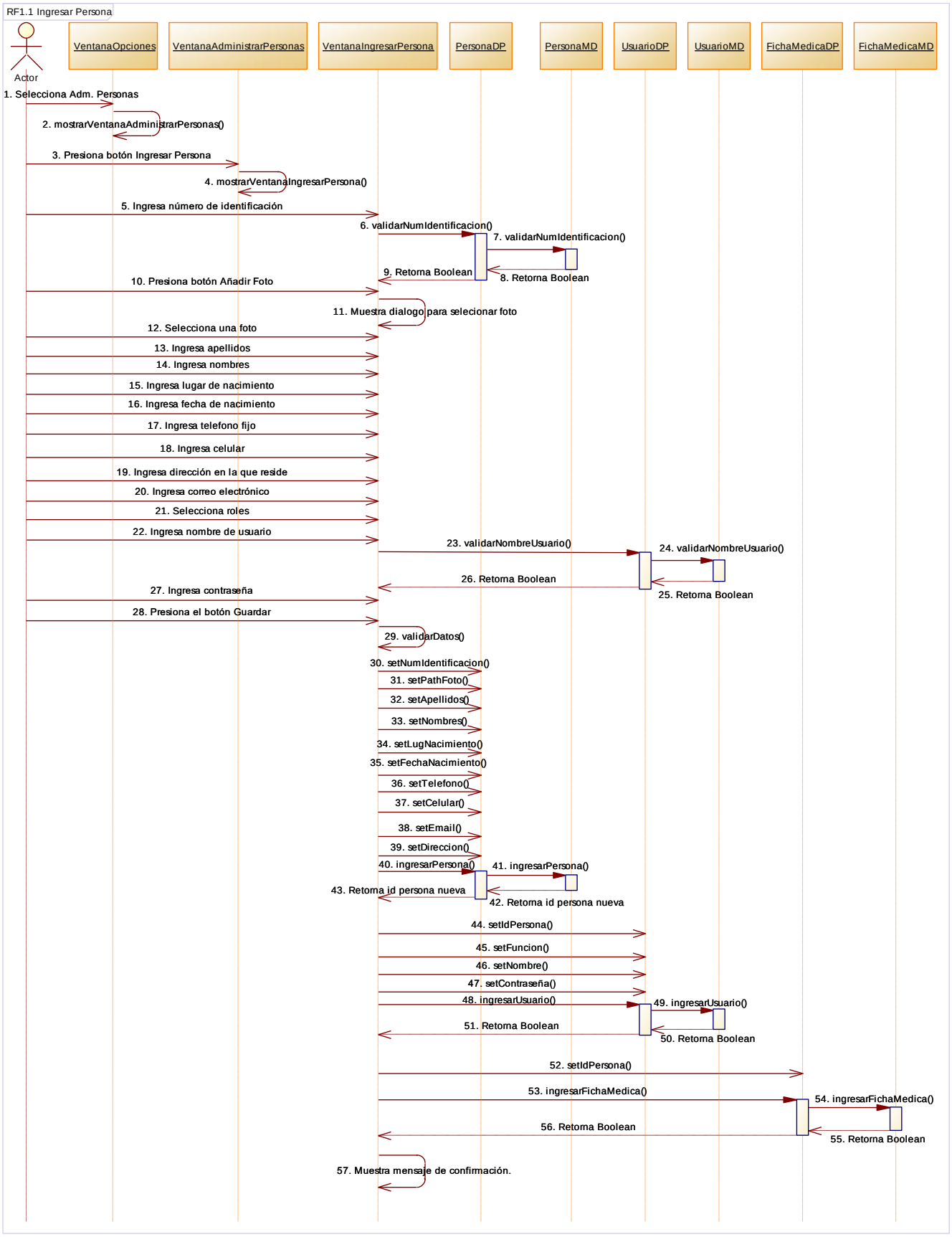

Figura 3.6. Diagrama de Secuencia RF1.1.

3.2.3 RF1.2 Modificar Persona.

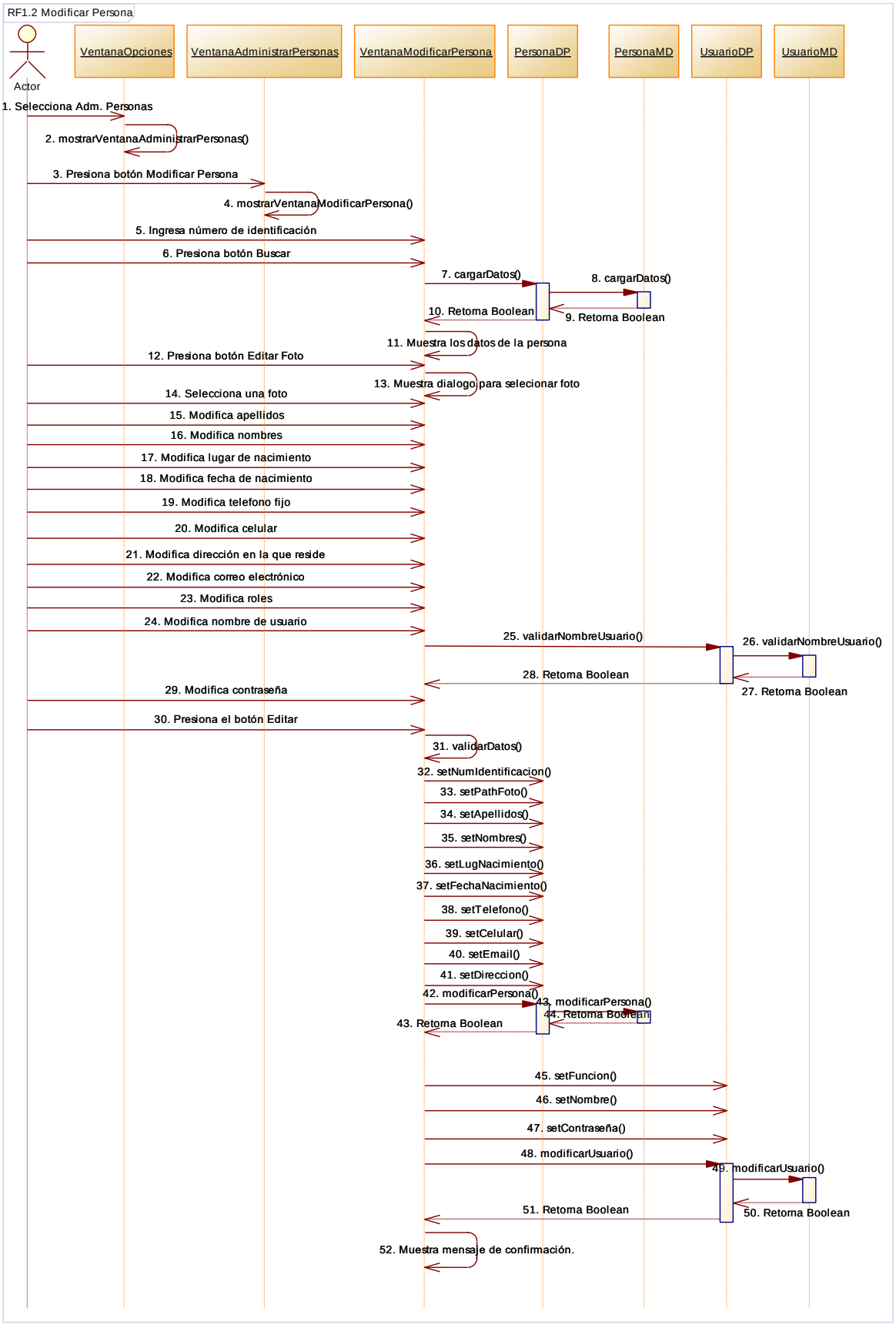

Figura 3.7. Diagrama de Secuencia RF1.2.

3.2.4 RF1.3 Eliminar Persona.

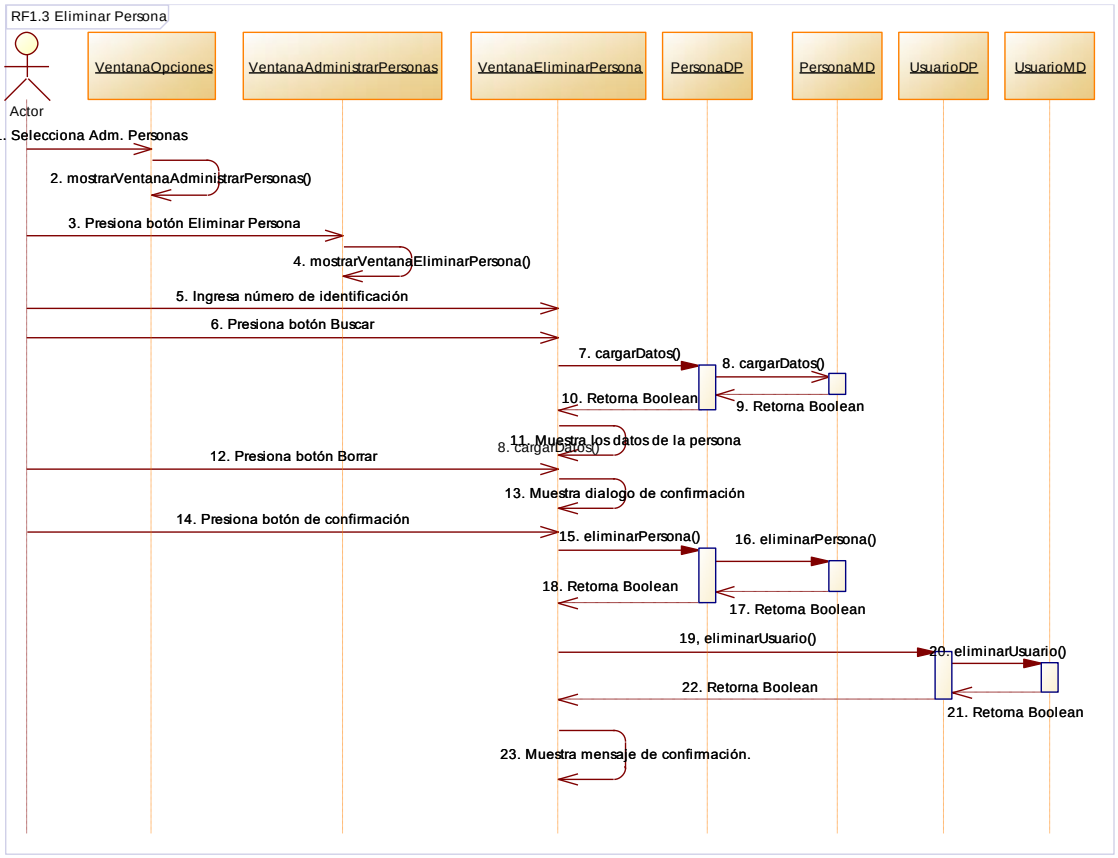

Figura 3.8. Diagrama de Secuencia RF1.3.

3.2.5 RF1.4 Consultar Personas.

3.2.5.1 RF1.4.1 Consulta General.

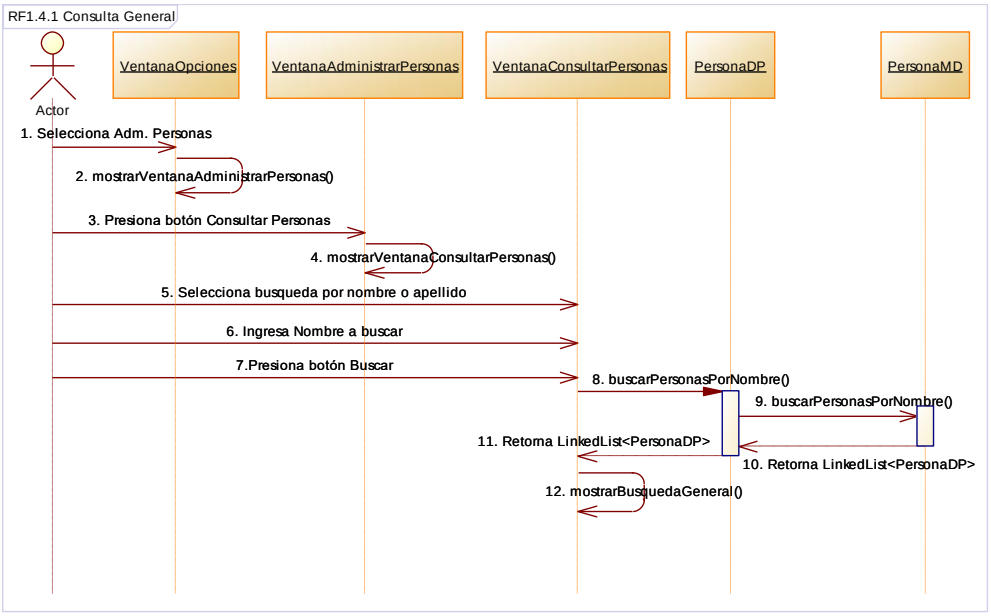

Figura 3.9. Diagrama de Secuencia RF1.4.1.

3.2.5.2 RF1.4.2 Consulta Detallada.

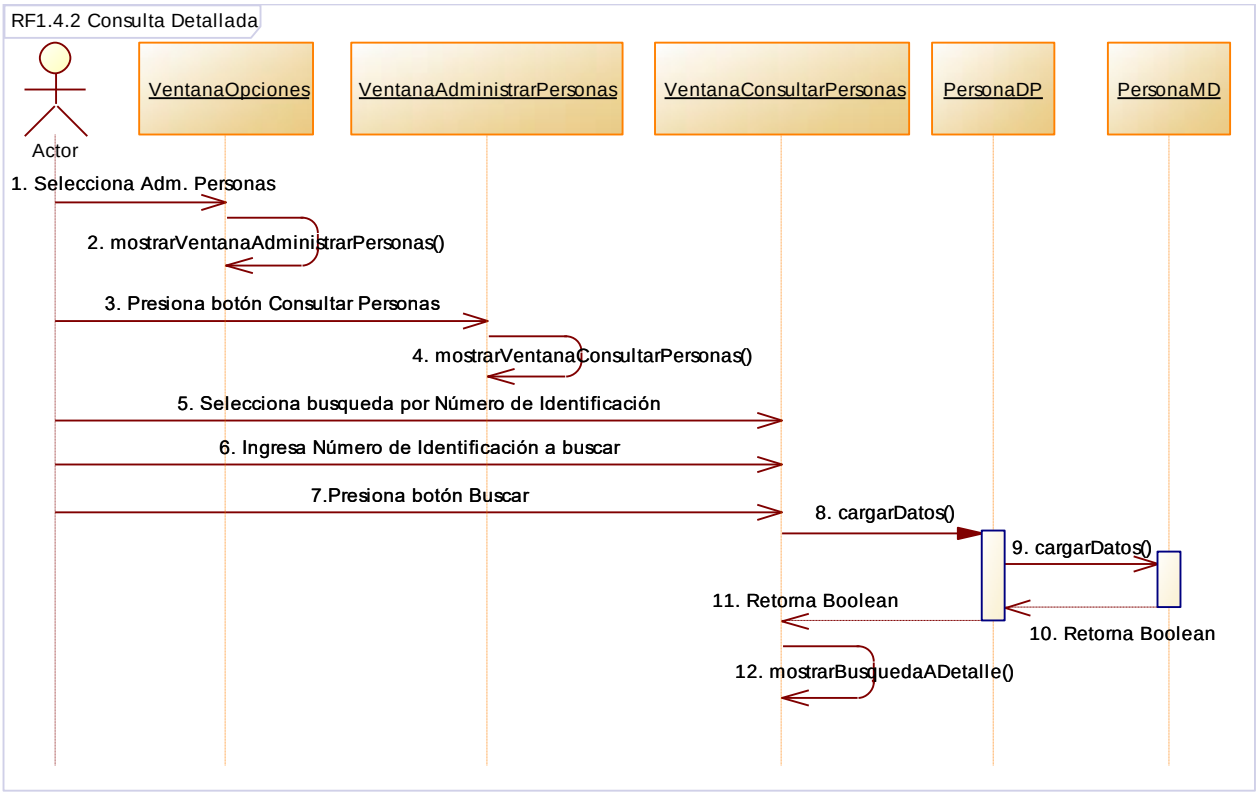

Figura 3.10. Diagrama de Secuencia RF1.4.2.

3.2.6 RF2.1 Consultar Ficha Médica.

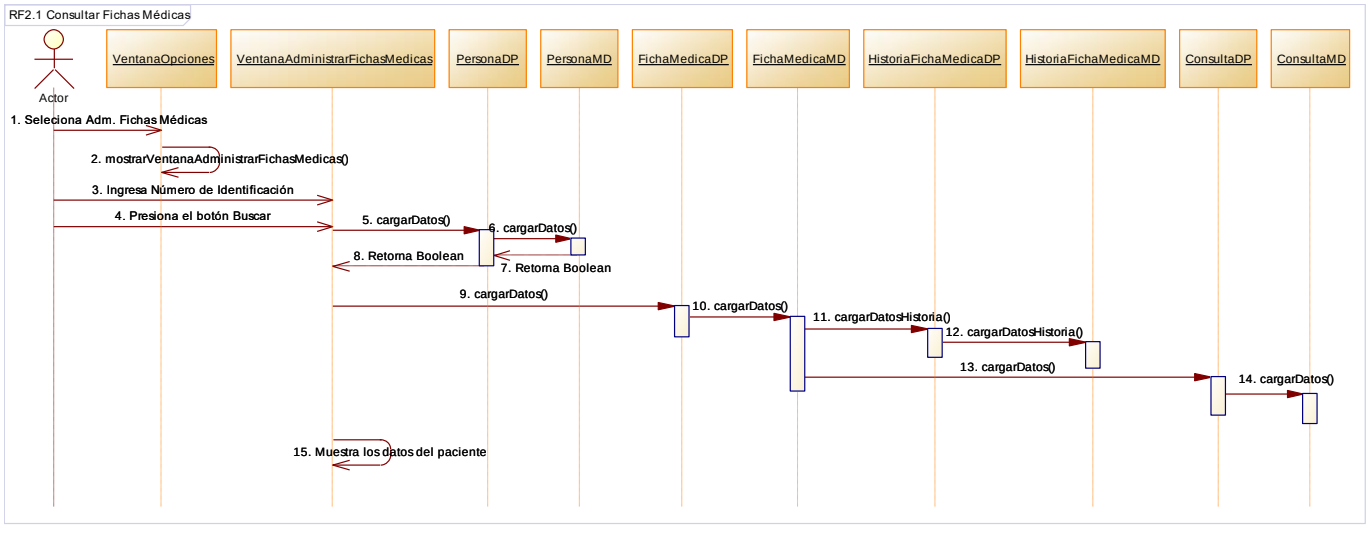

Figura 3.11. Diagrama de Secuencia RF2.1.

3.2.7 RF2.2 Modificar Ficha Médica.

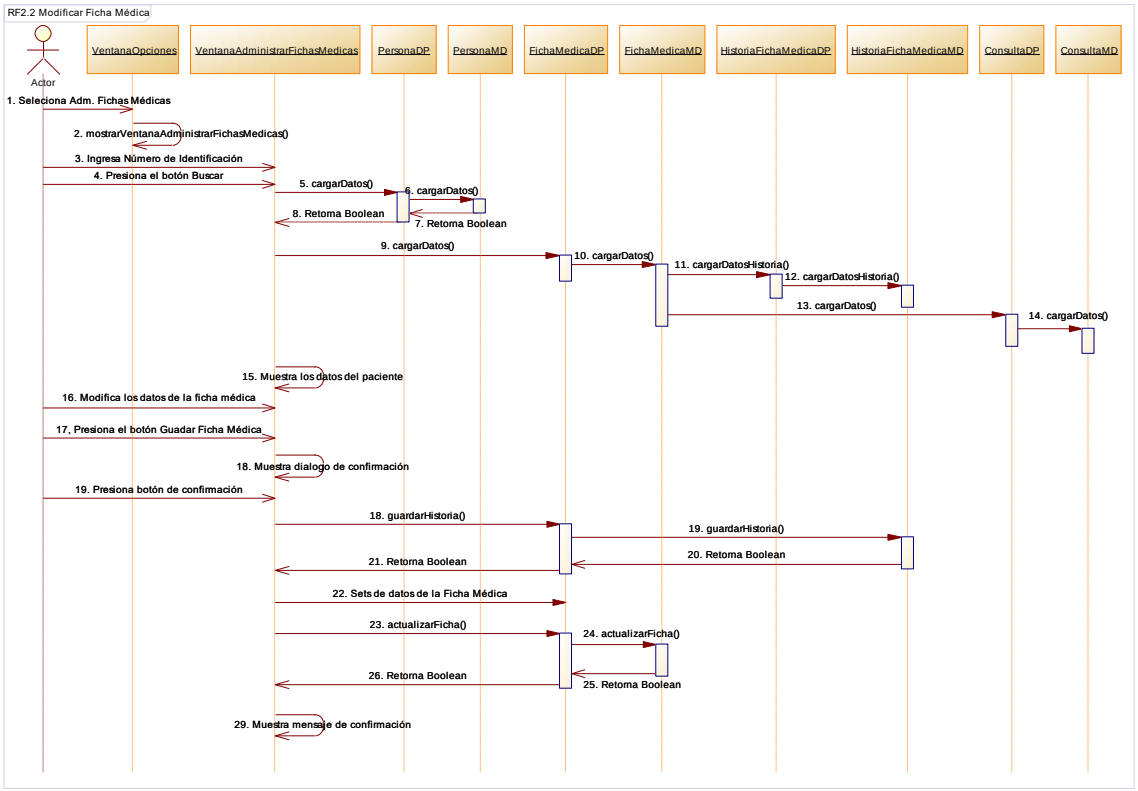

Figura 3.12. Diagrama de Secuencia RF2.2.

3.2.8 RF2.3 Consultar Historial Fichas Médicas.

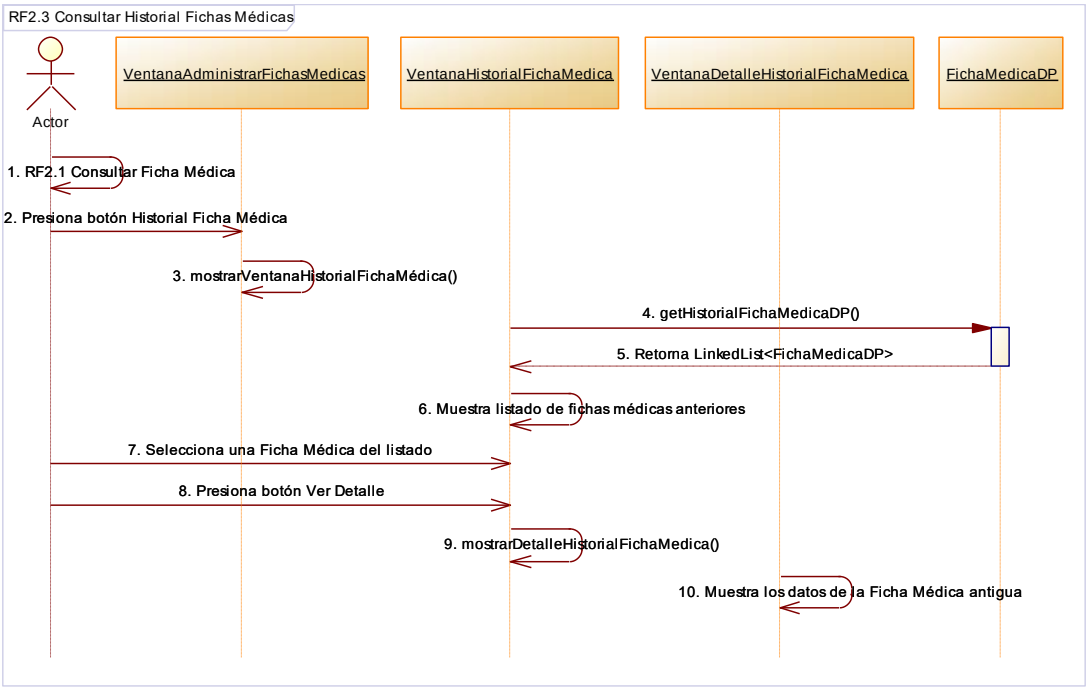

Figura 3.13. Diagrama de Secuencia RF2.3.

3.2.9. RF3.1 Ingresar Consulta.

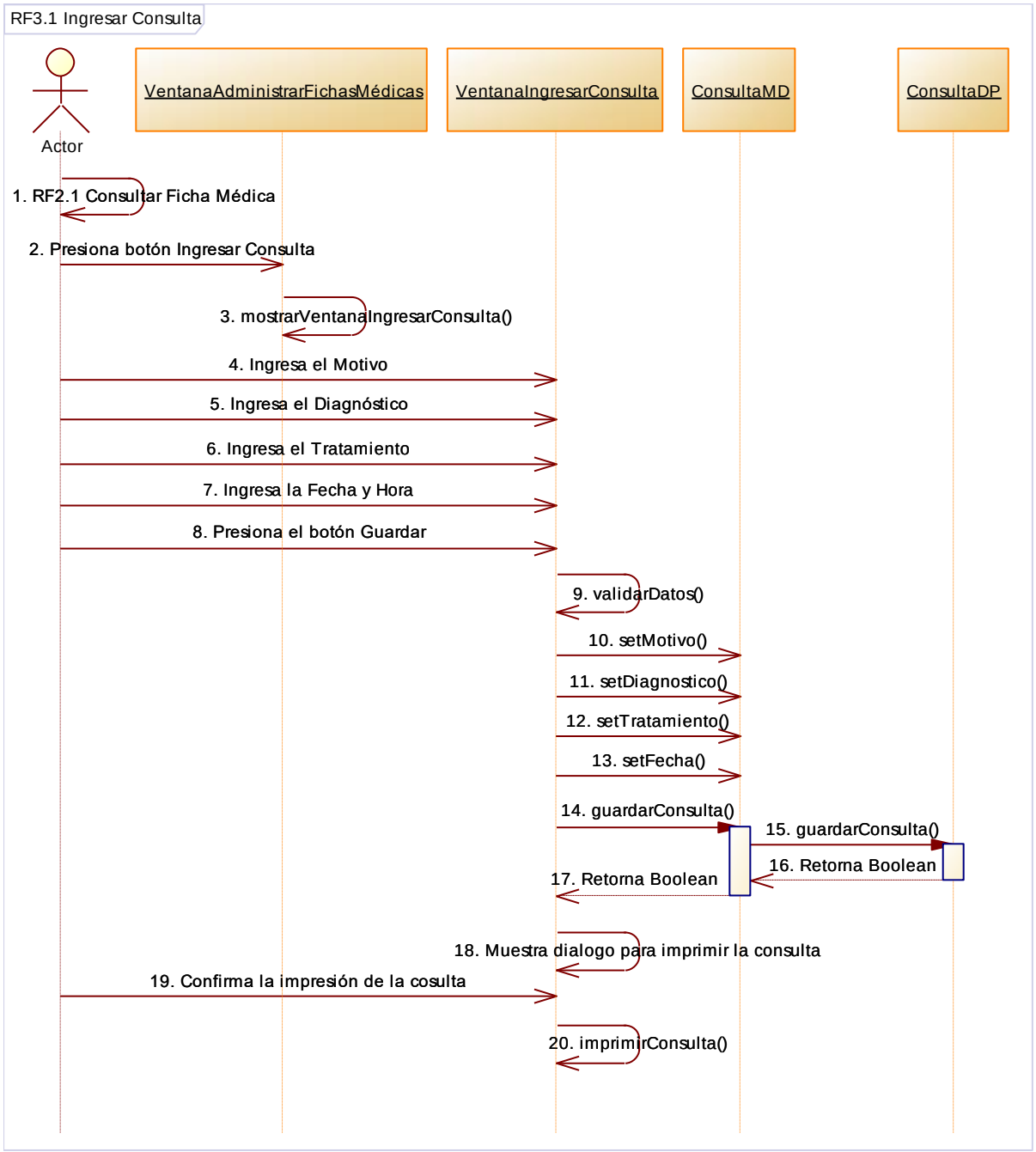

Figura 3.14. Diagrama de Secuencia RF3.1.

3.2.10. RF3.2 Modificar Consulta.

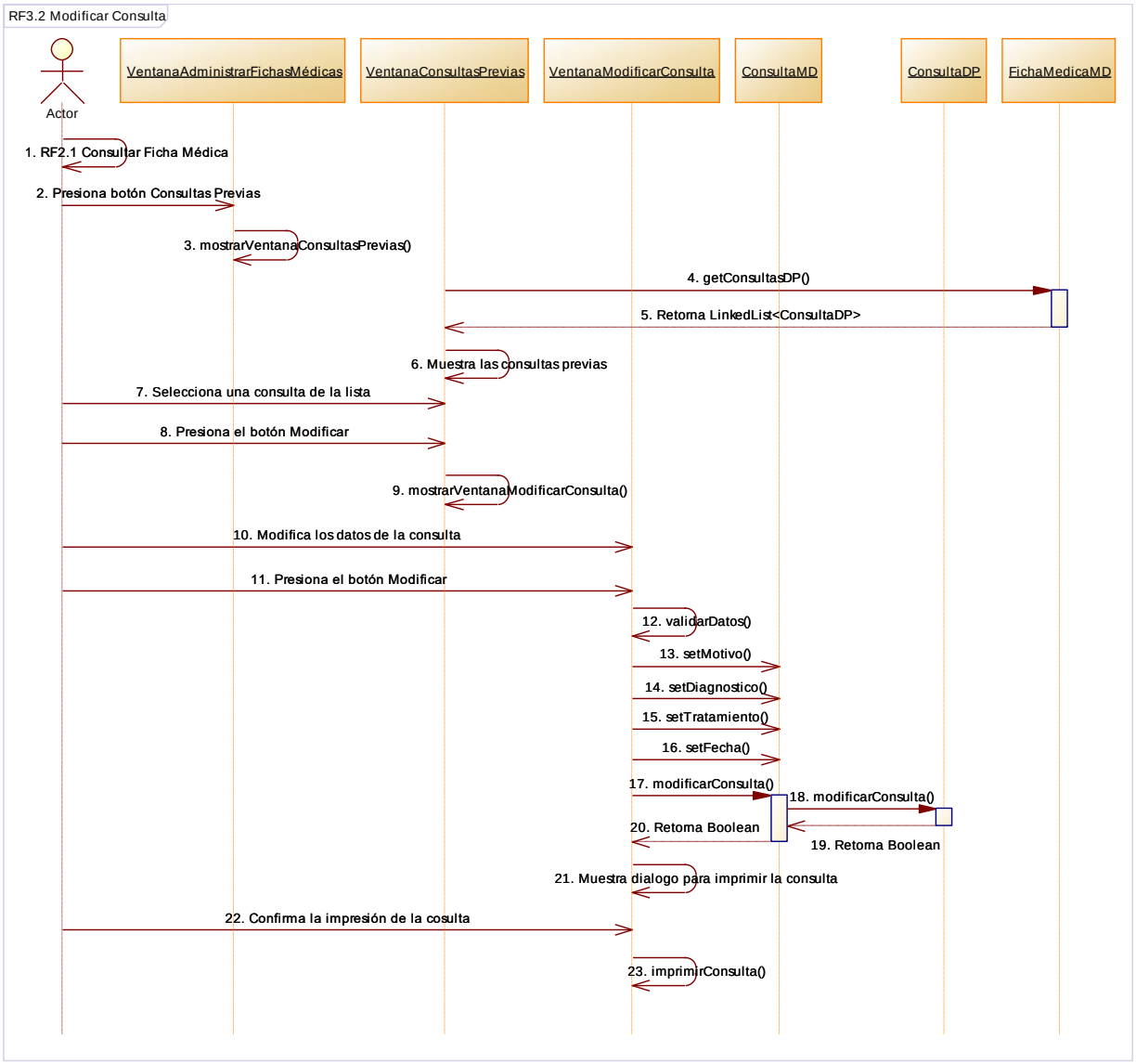

Figura 3.15. Diagrama de Secuencia RF3.2.

3.2.11. RF3.3 Eliminar Consulta.

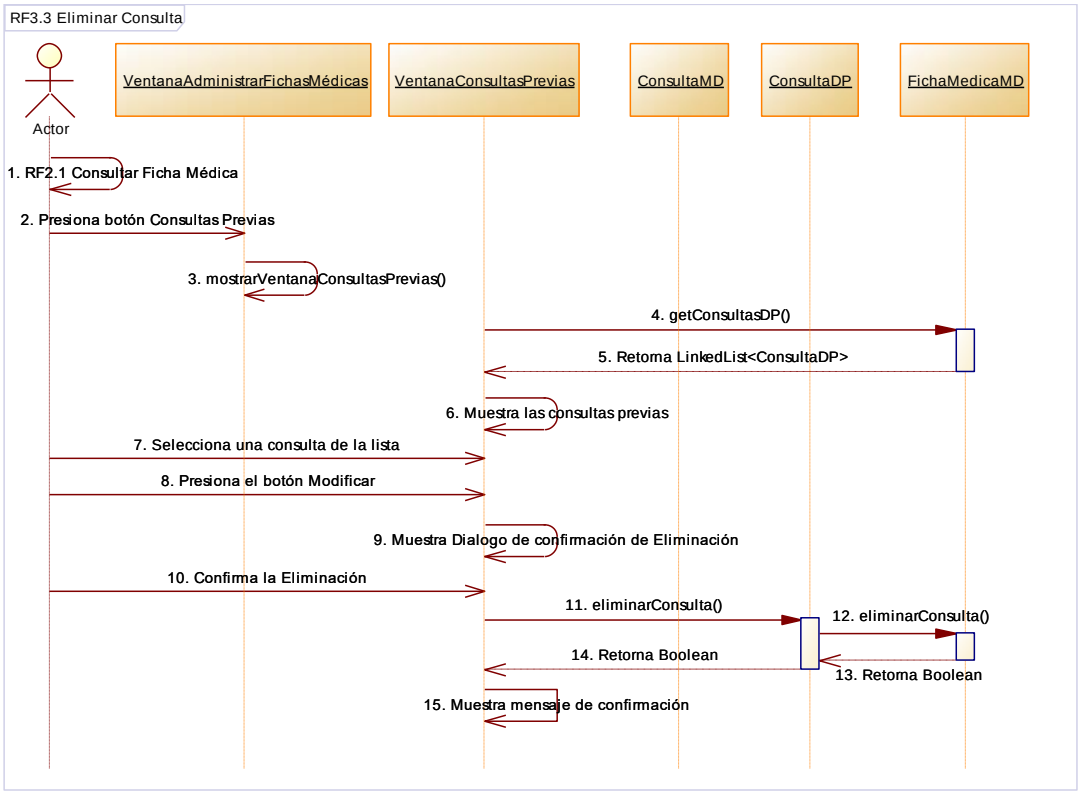

Figura 3.16. Diagrama de Secuencia RF3.3.

3.2.12. RF3.4 Buscar Consulta.

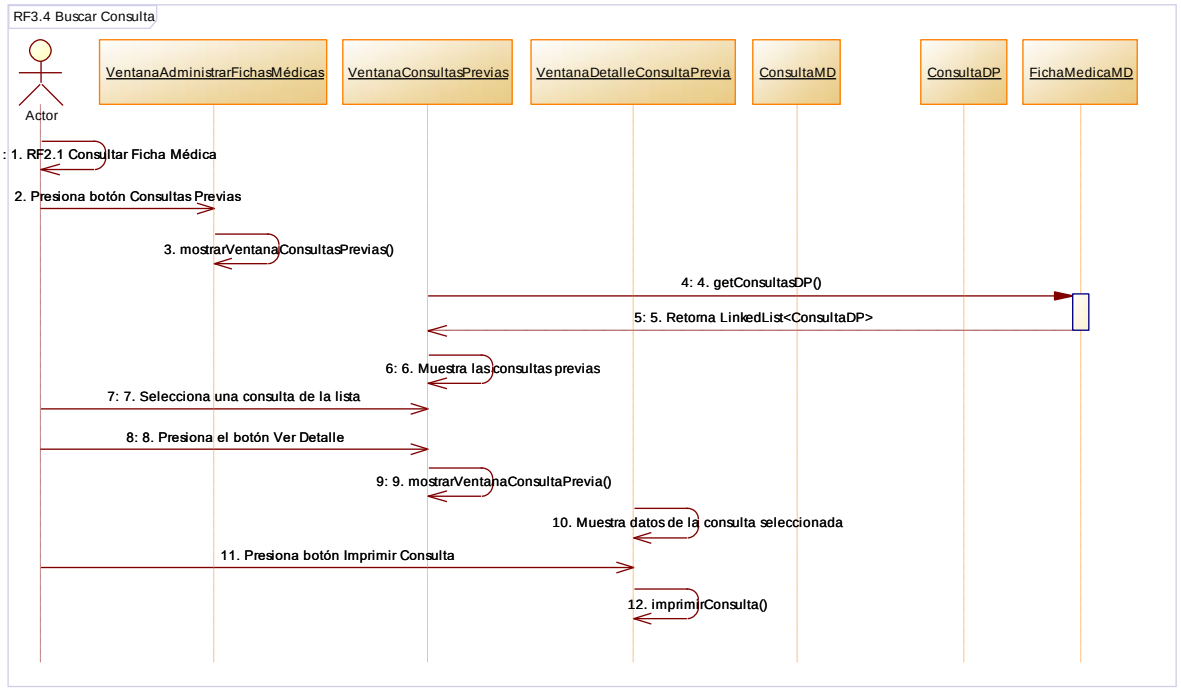

Figura 3.17. Diagrama de Secuencia RF3.4.

3.3 DIAGRAMAS DE COLABORACIÓN

De acuerdo a la sección 1.7.5.8 del primer capítulo, los diagramas de colaboración utilizados en la presente disertación son los siguientes:

3.3.1 RF0 Ingreso al Sistema.

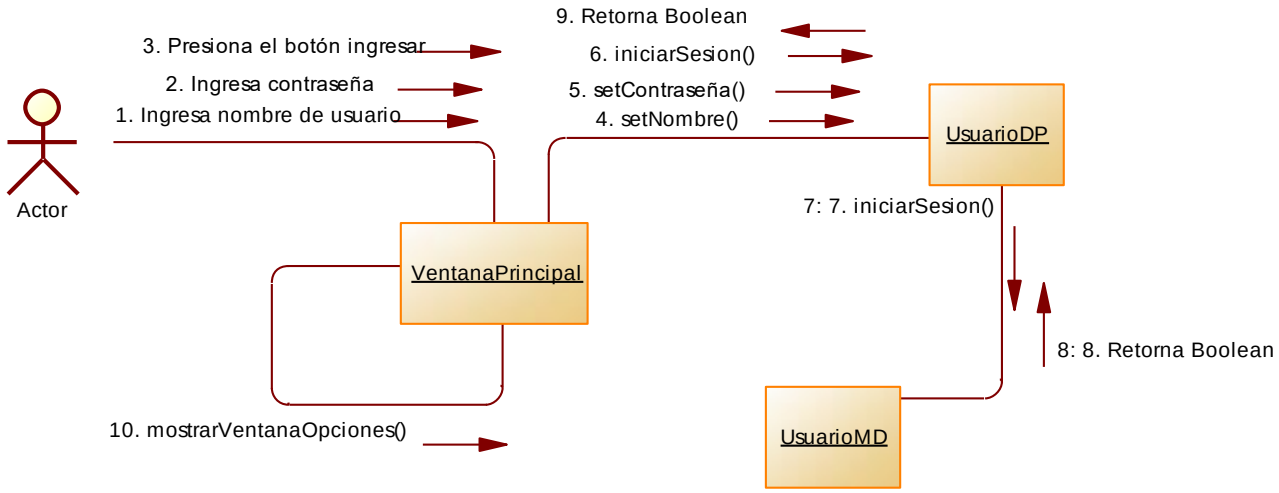

Figura 3.18. Diagrama de Colaboración RF0.

3.3.2 RF1.1 Ingresar Persona.

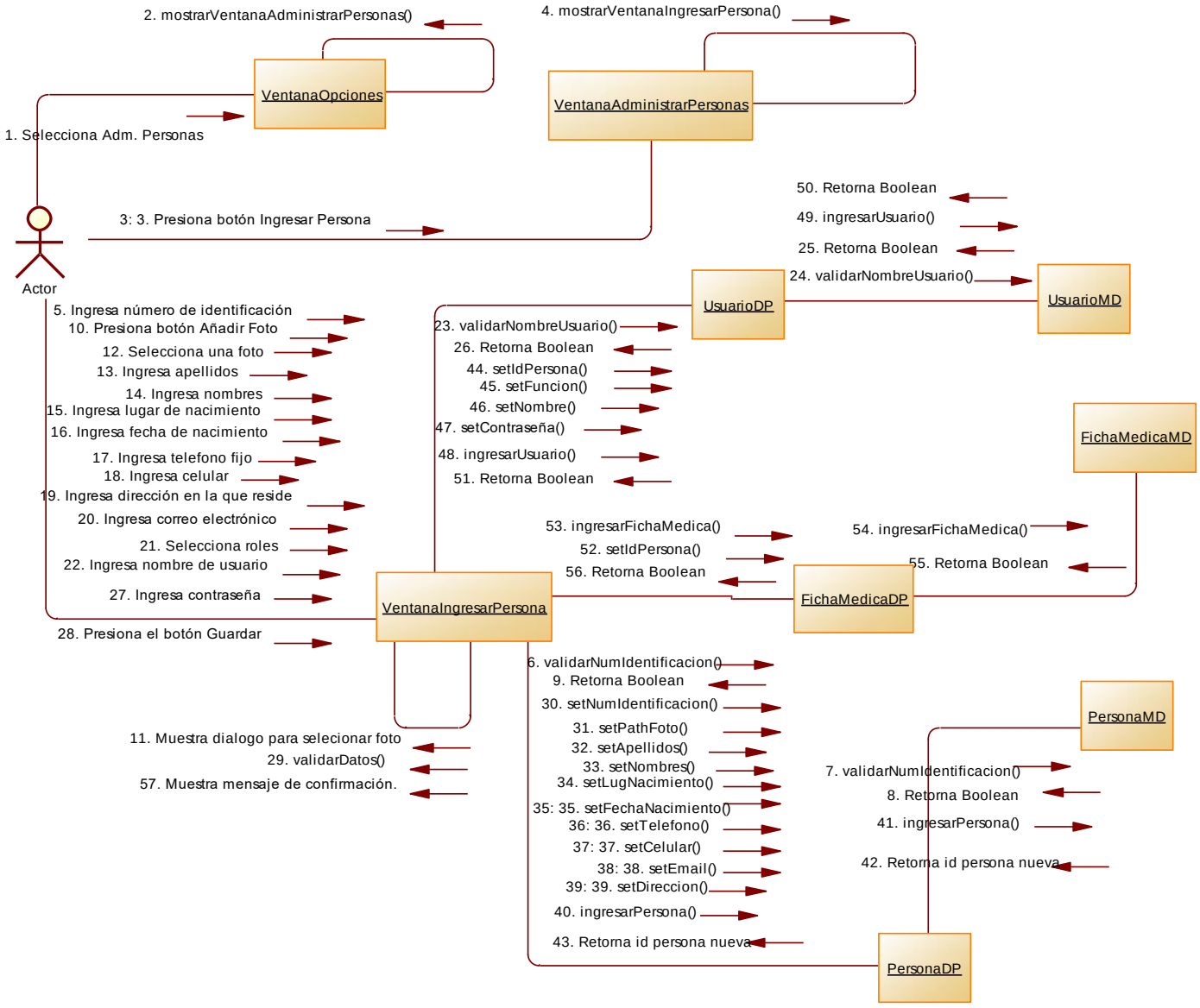

Figura 3.19. Diagrama de Colaboración RF1.1.

3.3.3 RF1.2 Modificar Persona.

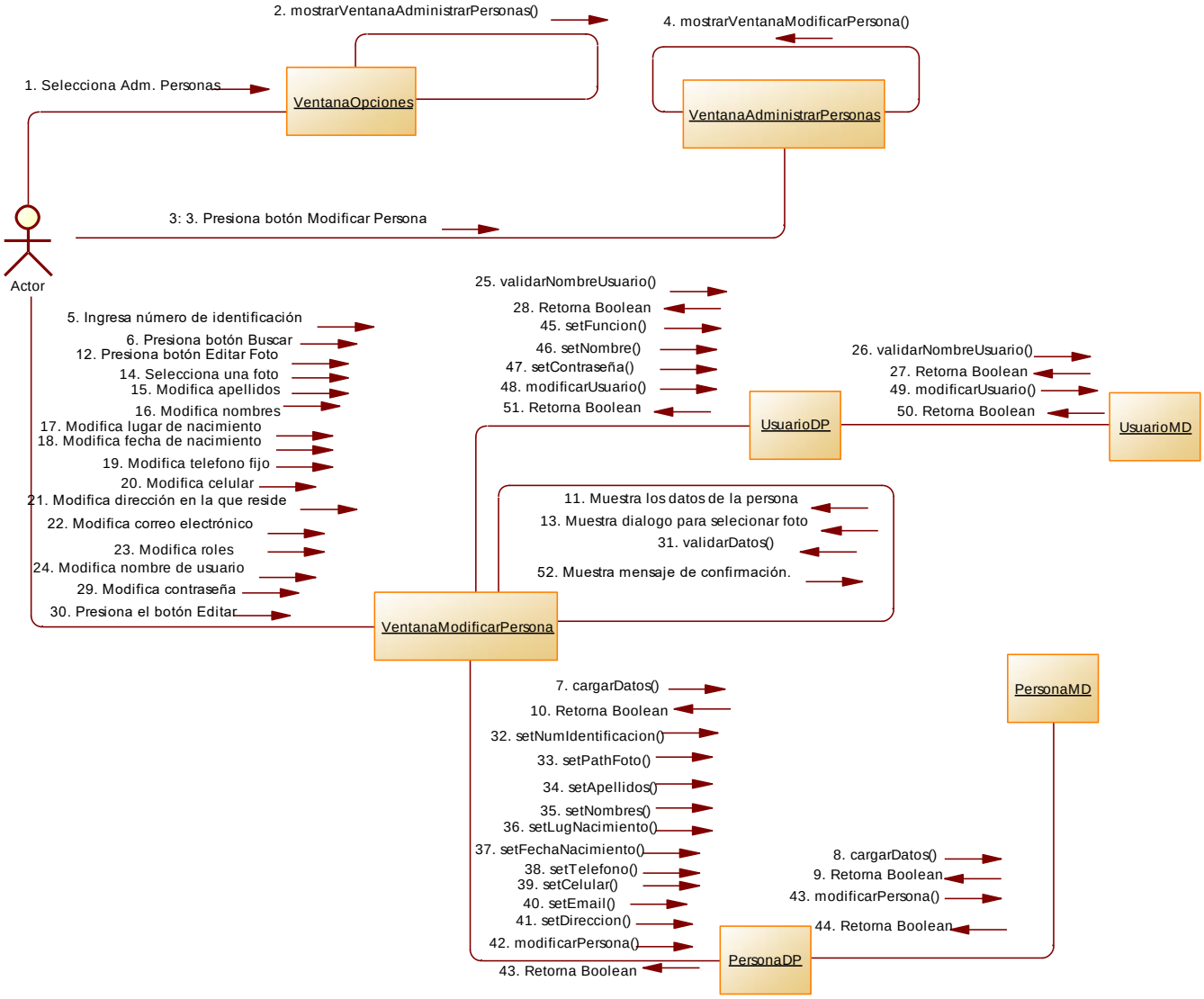

Figura 3.20. Diagrama de Colaboración RF1.2.

3.3.4 RF1.3 Eliminar Persona.

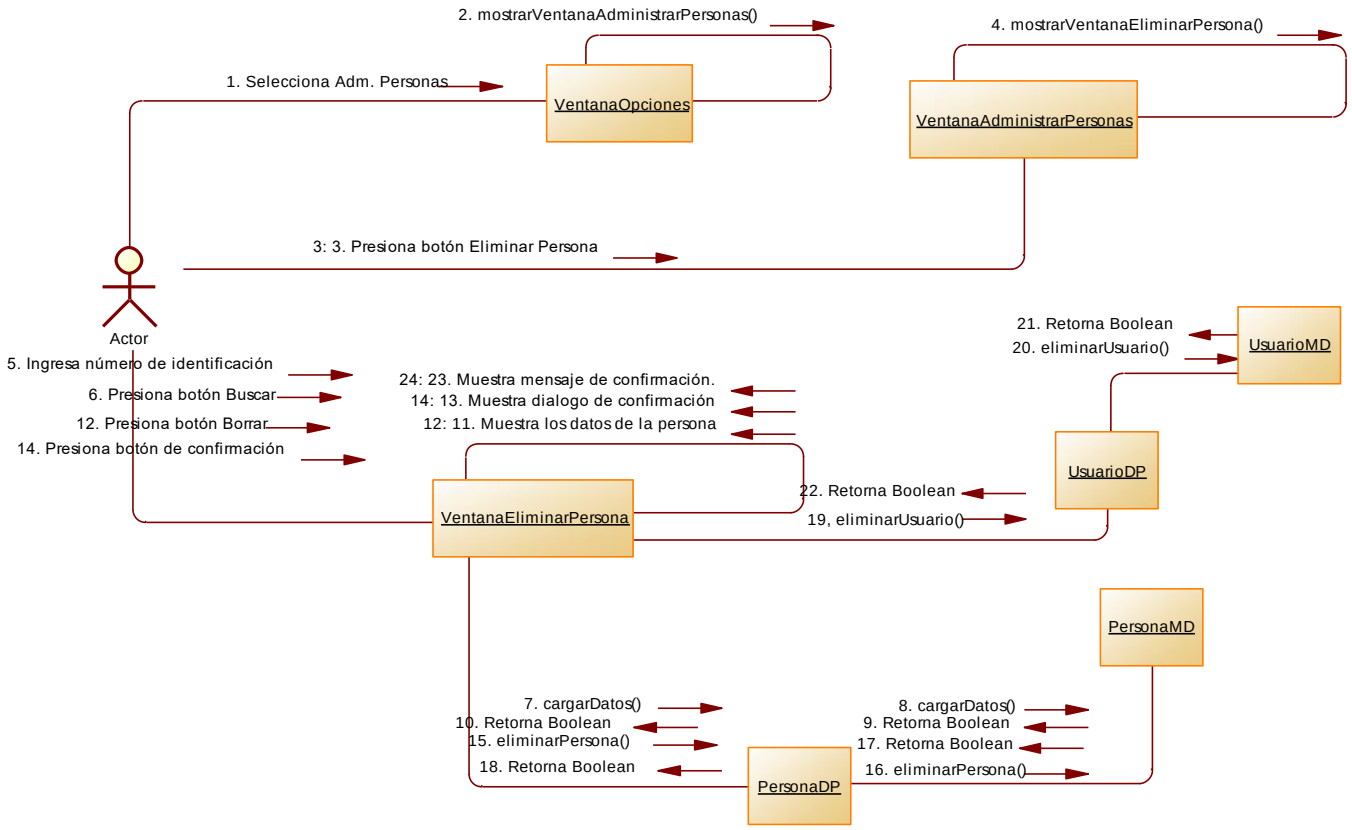

Figura 3.20. Diagrama de Colaboración RF1.3.

3.3.5 RF1.4 Consultar Personas.

3.3.5.1 RF1.4.1 Consulta General.

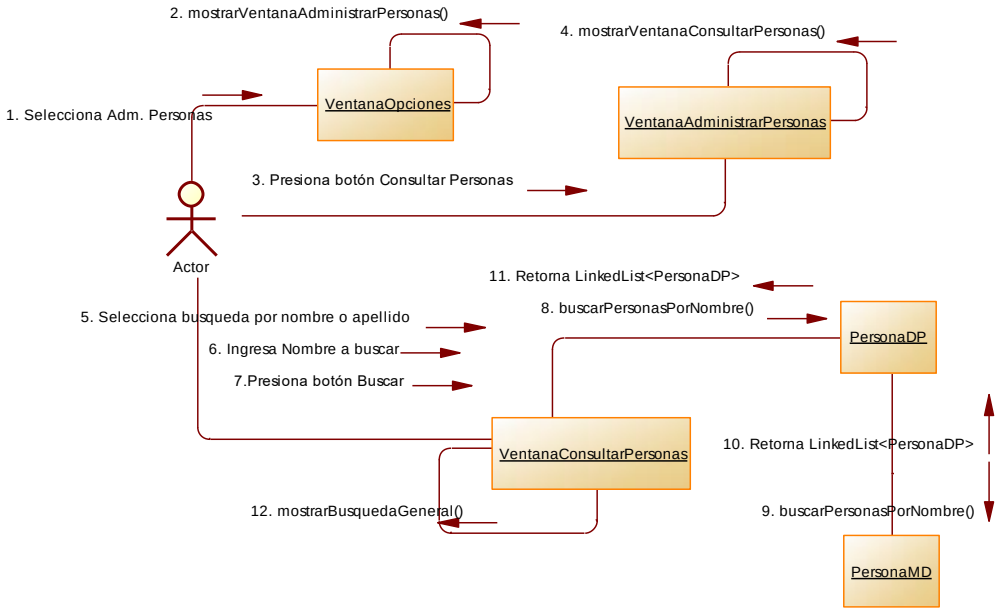

Figura 3.21. Diagrama de Colaboración RF1.4.1.

3.3.5.2 RF1.4.2 Consulta Detallada.

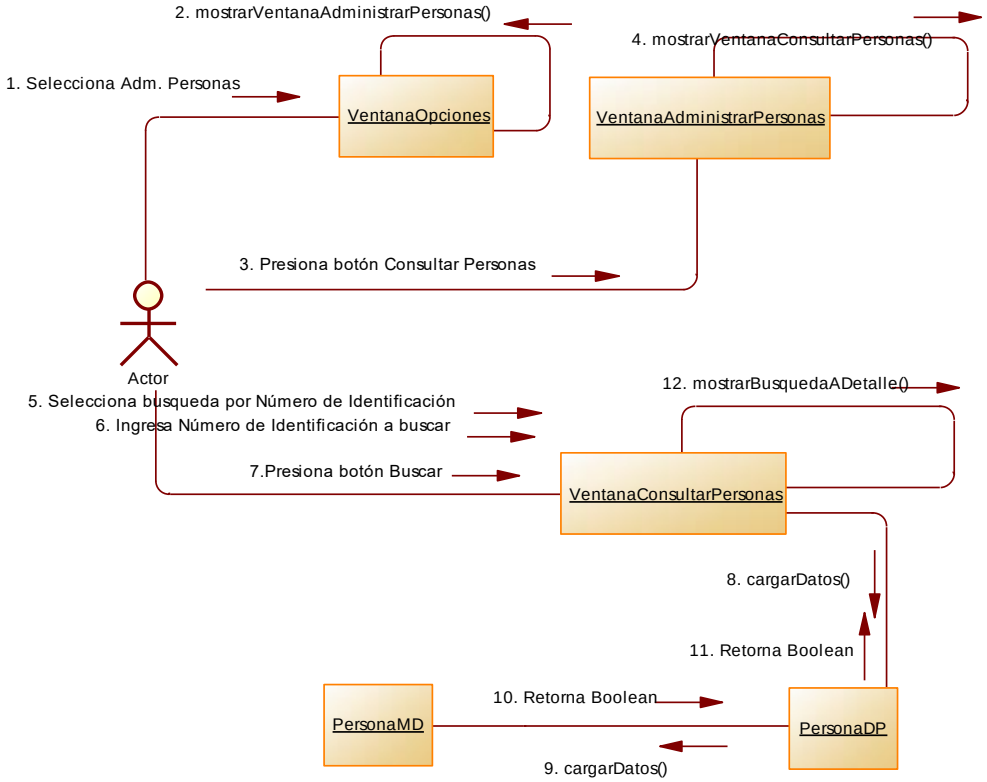

Figura 3.22. Diagrama de Colaboración RF1.4.2.

3.3.6 RF2.1 Consultar Ficha Médica.

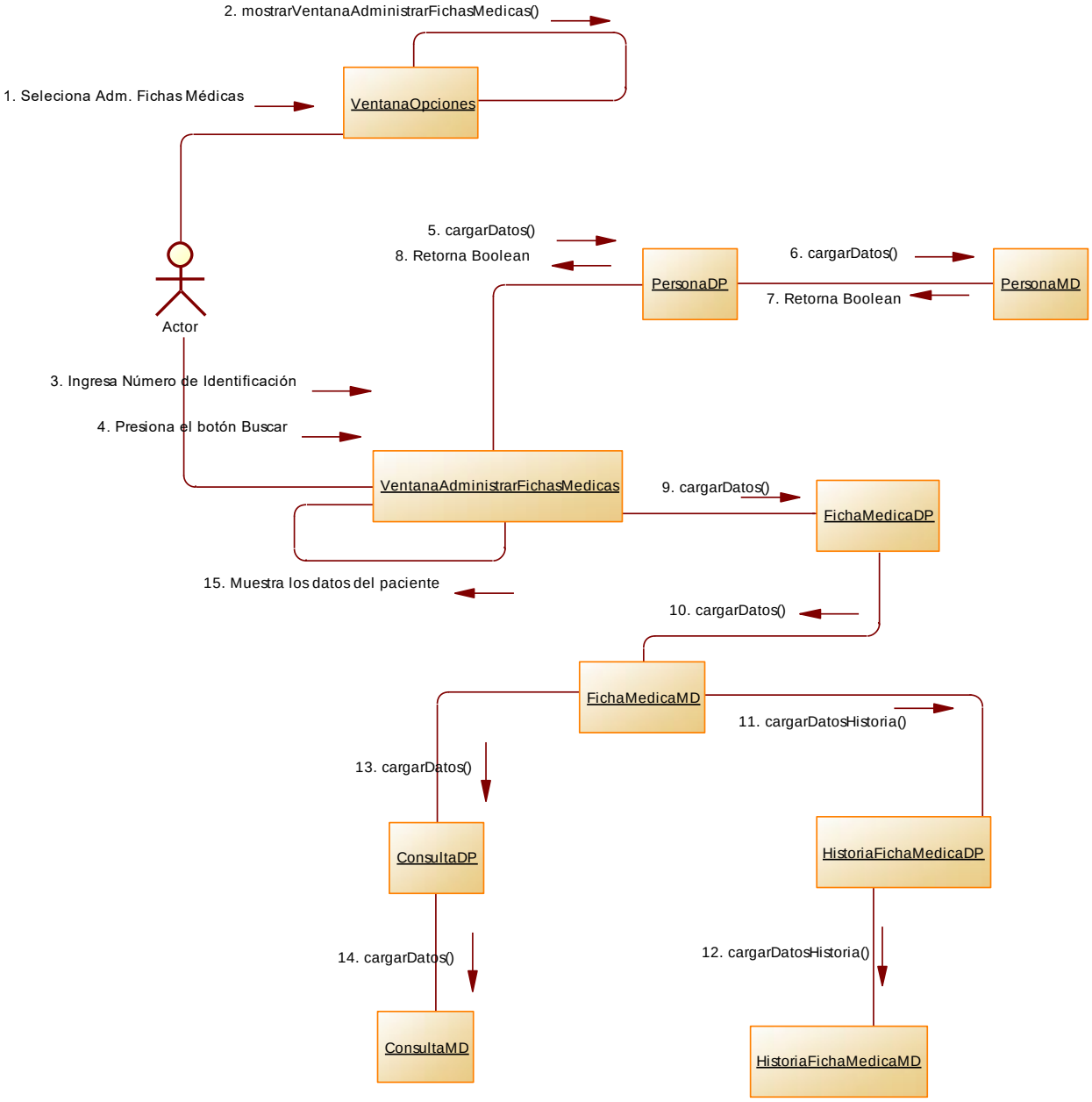

Figura 3.23. Diagrama de Colaboración RF2.1.

3.3.7 RF2.2 Modificar Ficha Médica.

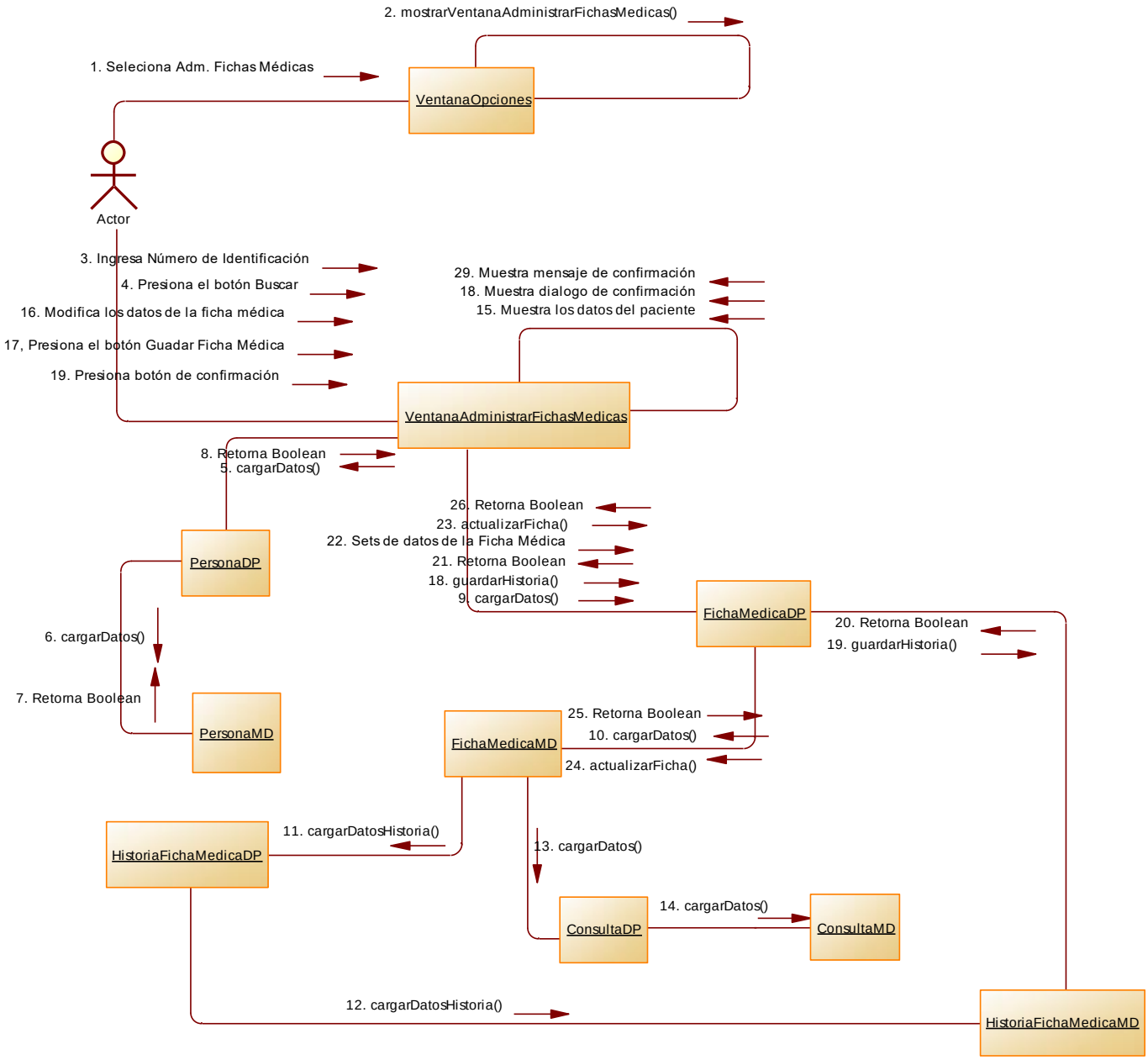

Figura 3.23. Diagrama de Colaboración RF2.2.

3.3.8 RF2.3 Consultar Historial Fichas Médicas.

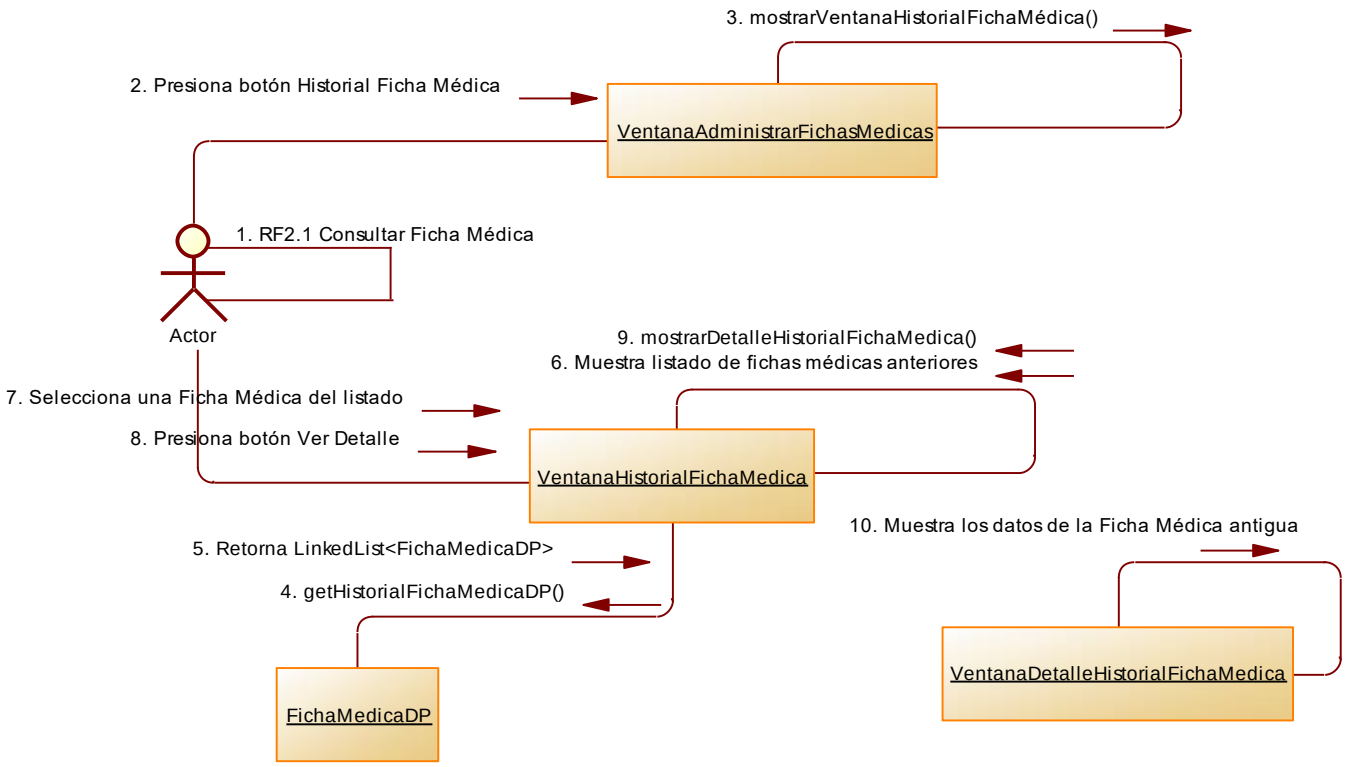

Figura 3.24. Diagrama de Colaboración RF2.3.

3.3.9 RF3.1 Ingresar Consulta.

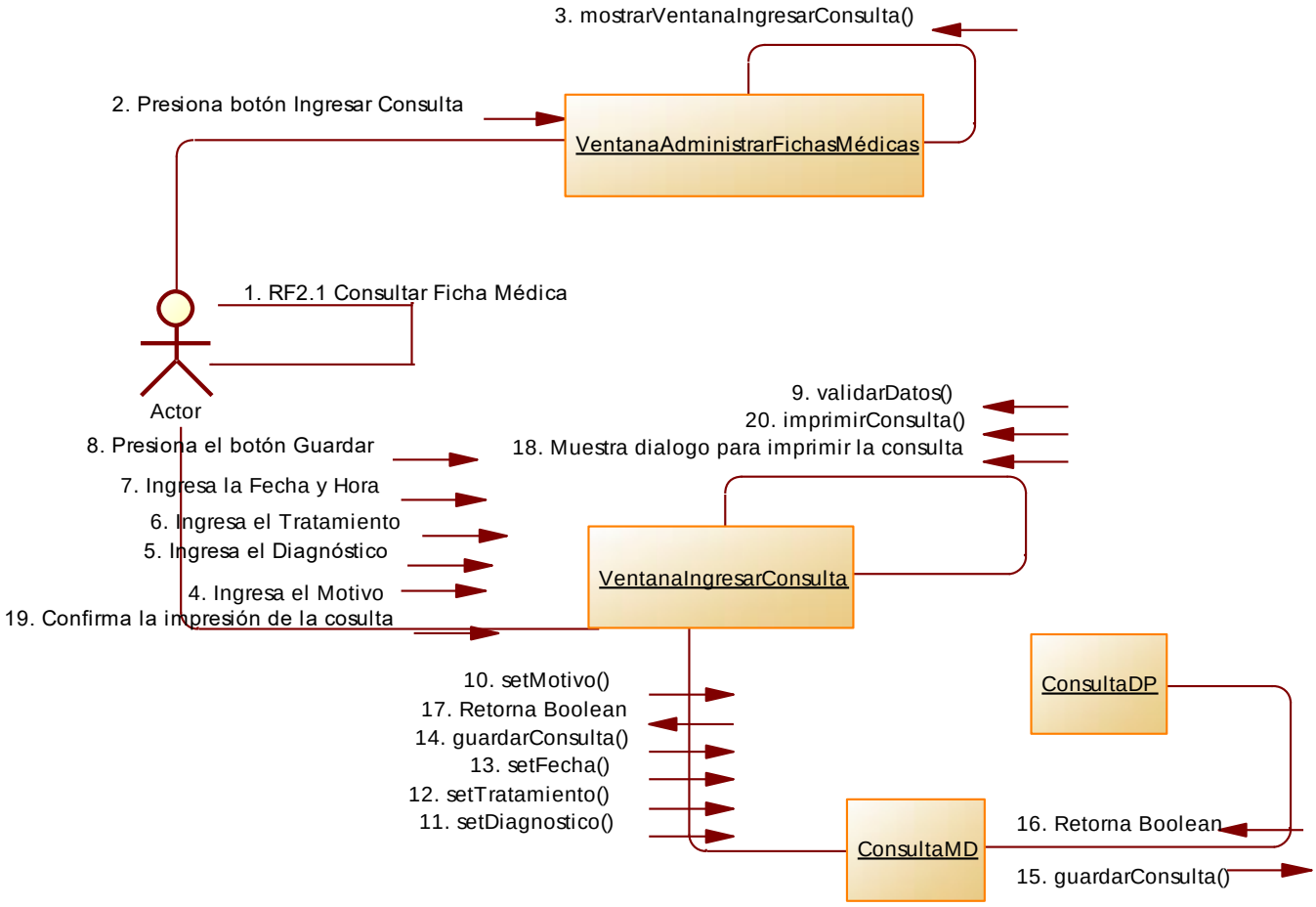

Figura 3.25. Diagrama de Colaboración RF3.1.

3.3.10 RF3.2 Modificar Consulta.

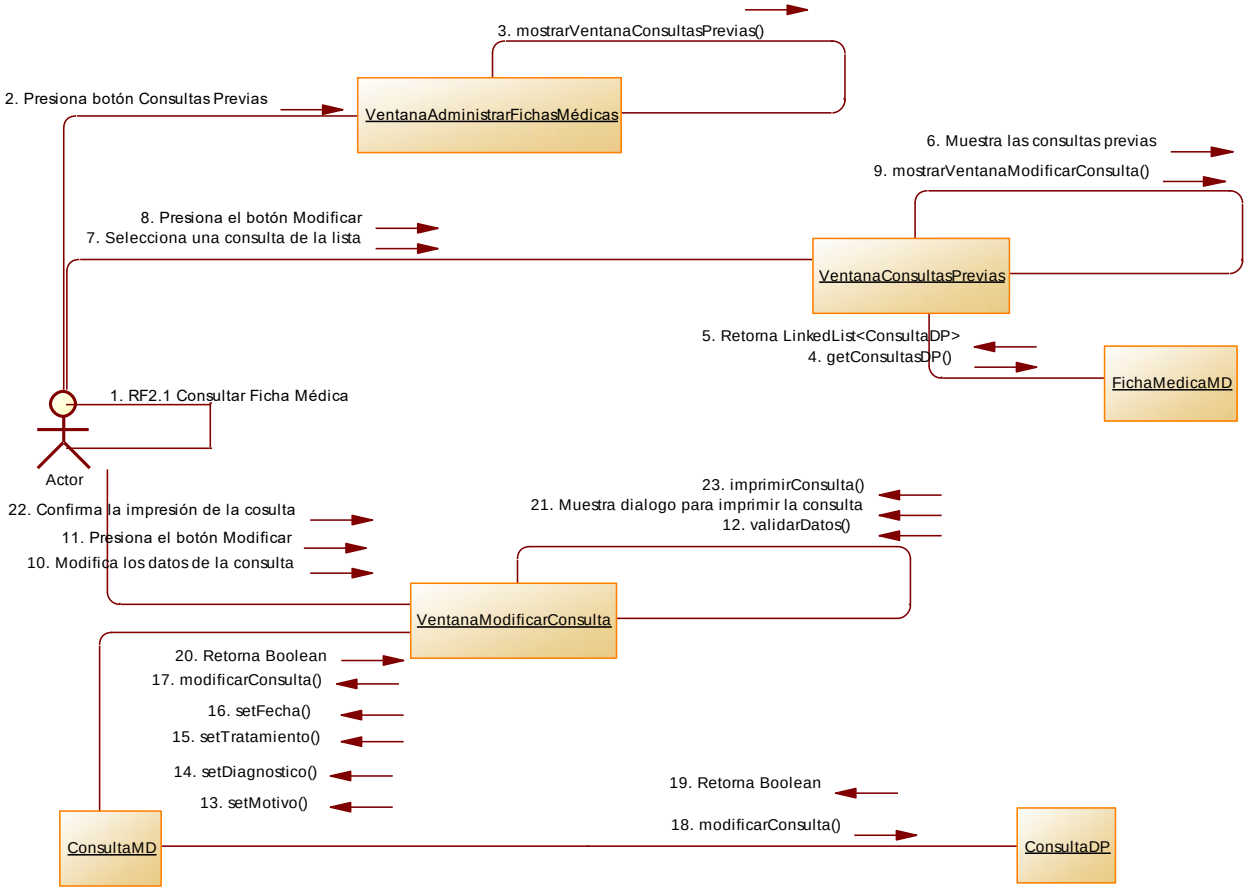

Figura 3.26. Diagrama de Colaboración RF3.2.

3.3.11 RF3.3 Eliminar Consulta.

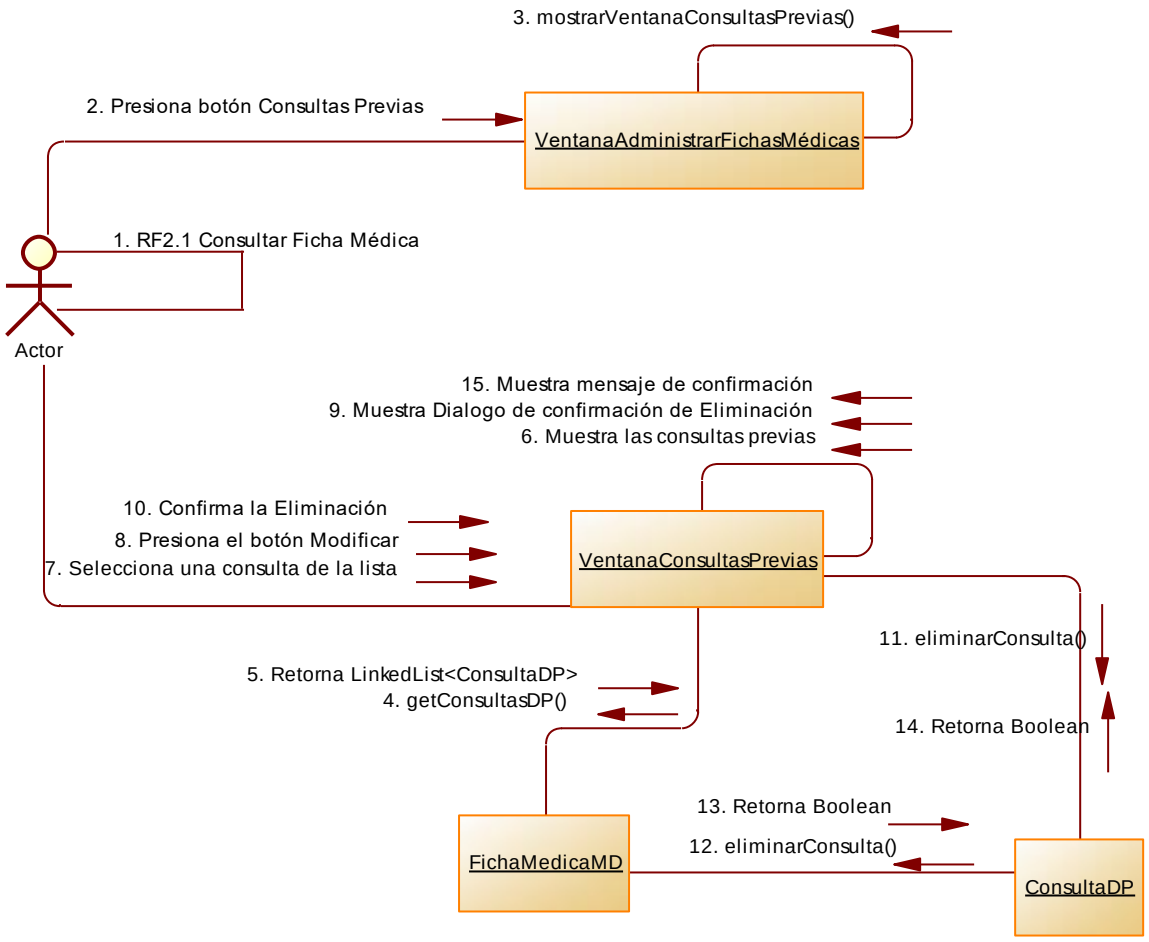

Figura 3.27. Diagrama de Colaboración RF3.3.

3.3.12 RF3.4 Buscar Consulta.

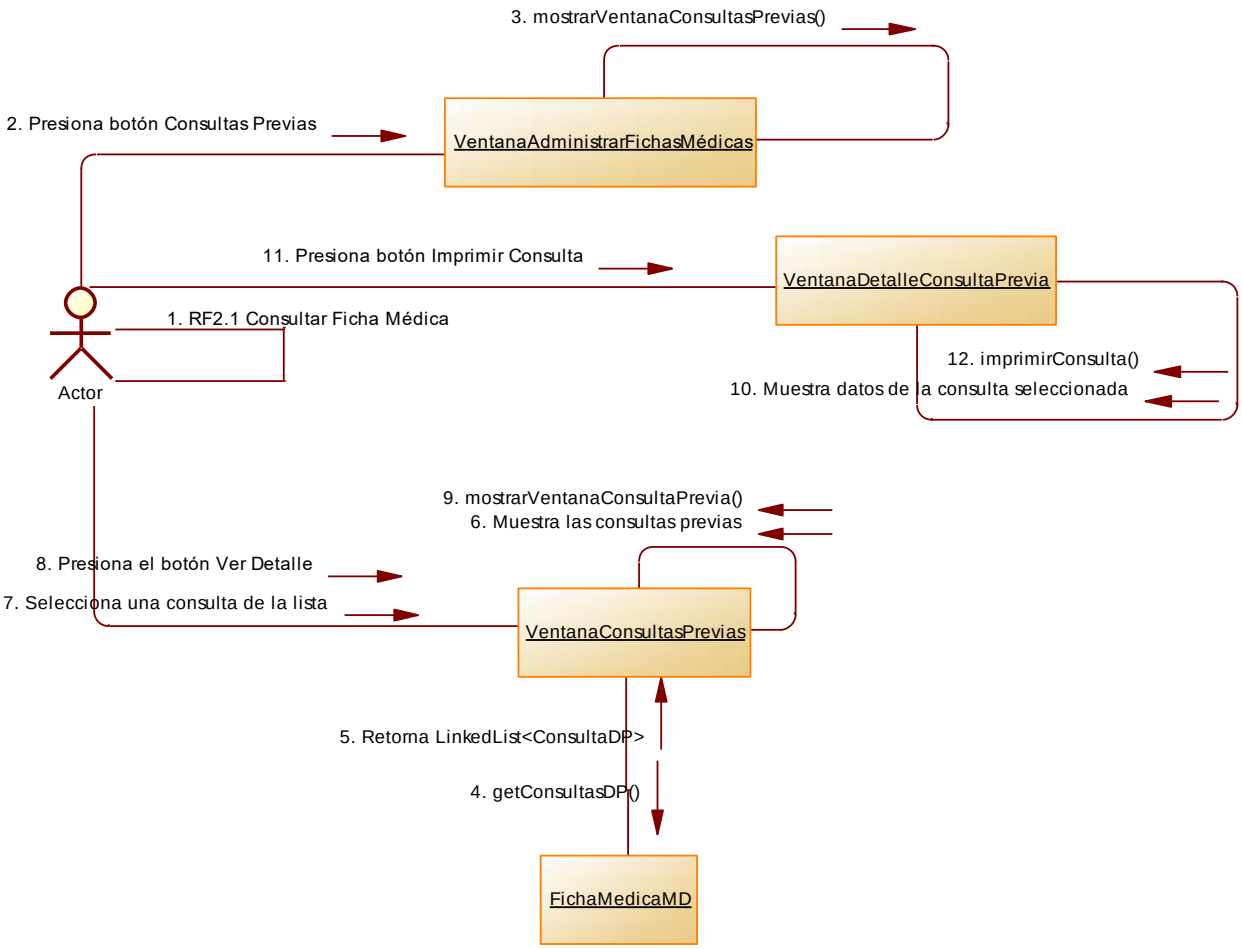

Figura 3.28. Diagrama de Colaboración RF3.4.

3.4 MODELO ENTIDAD – RELACIÓN (BASE DE DATOS)

De acuerdo a la sección 1.7.5.9 del primer capítulo, el modelo de la base de datos utilizado para el desarrollo de la presente disertación se ilustra en las figuras 3.29 y 3.30.

3.3.1 Diagrama Conceptual

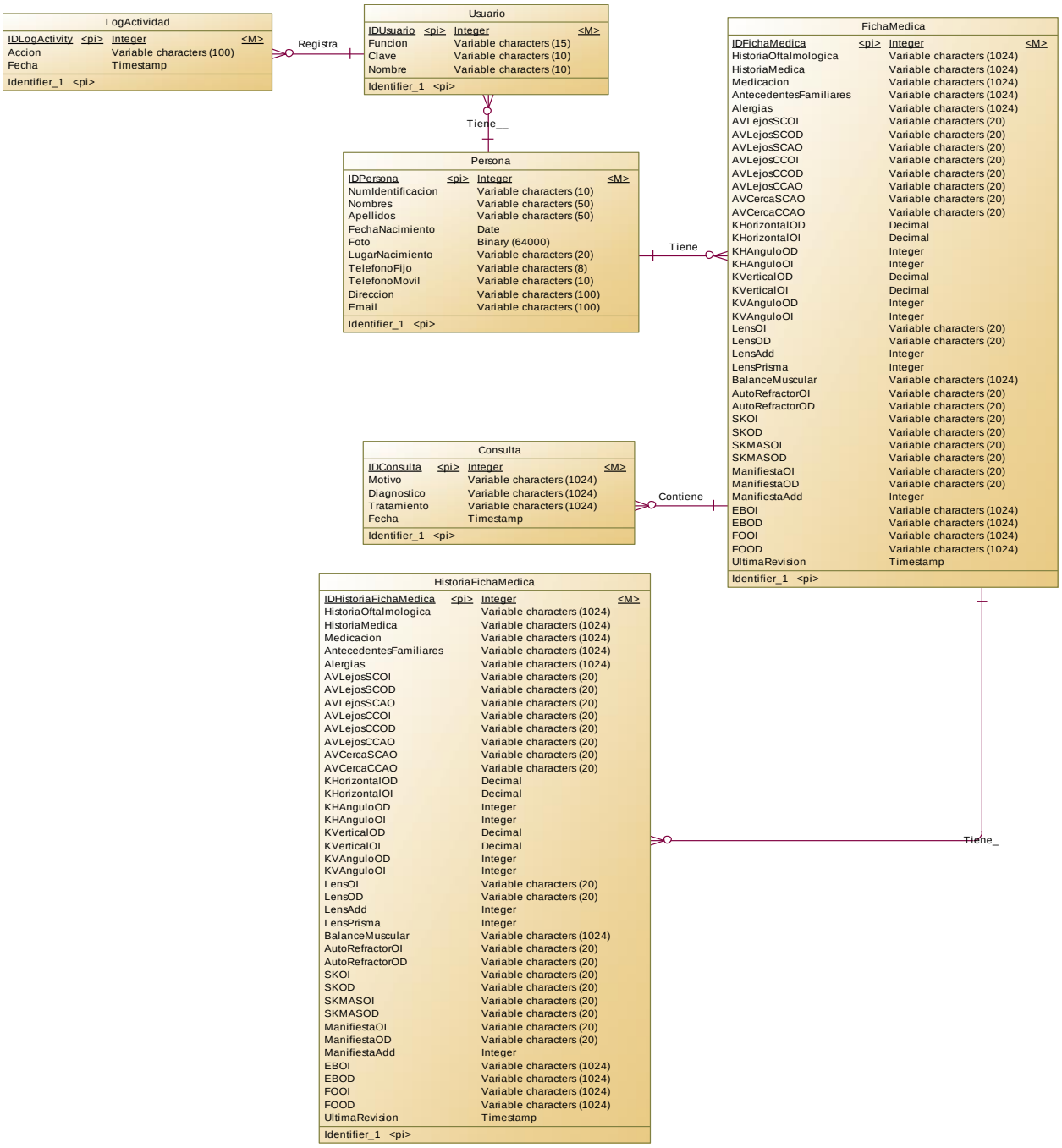

Figura 3.29. Diagrama Conceptual de la Base de Datos.

3.3.2 Diagrama Físico

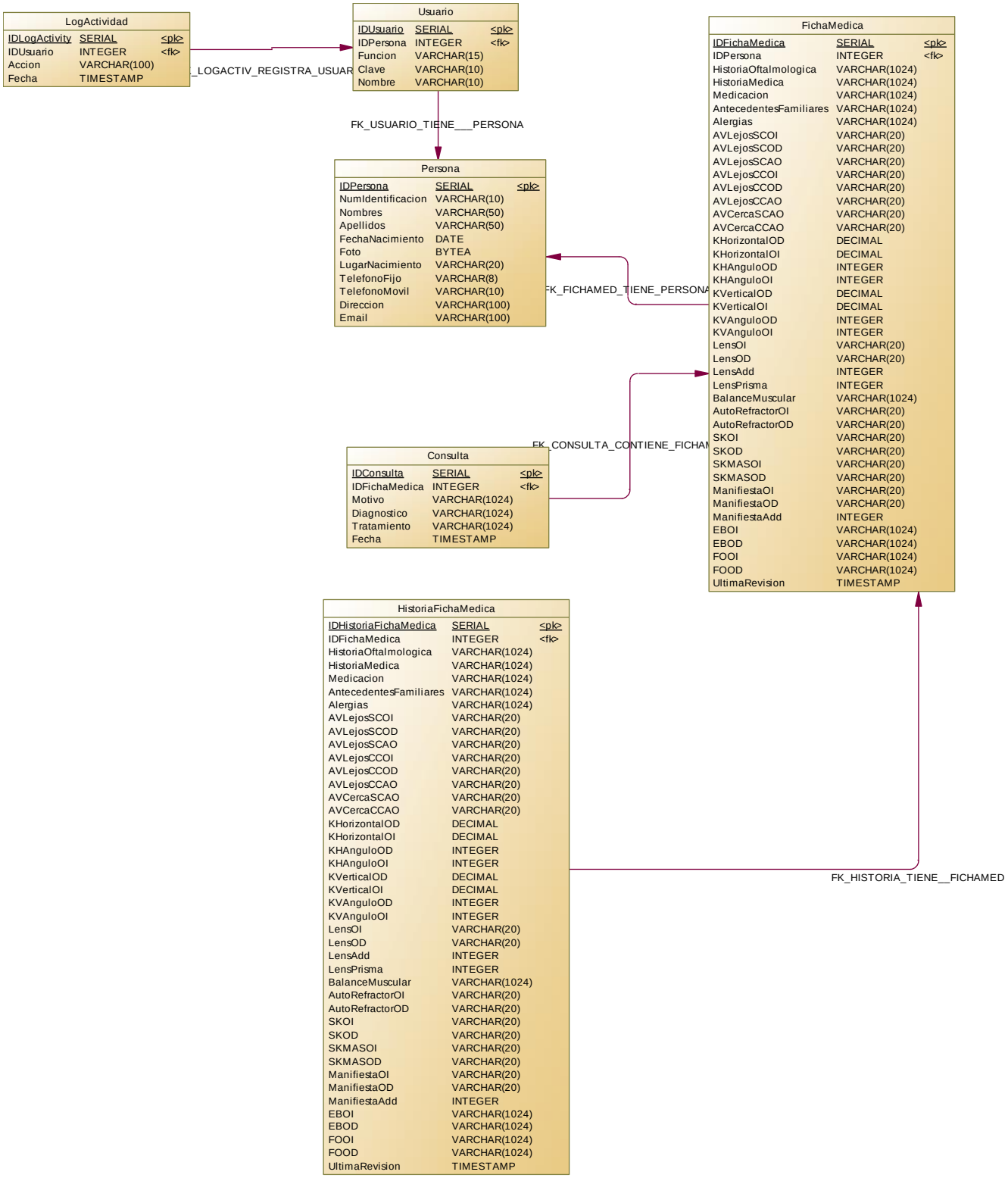

Figura 3.30. Diagrama Físico de la Base de Datos.

3.5 DIAGRAMA DE PAQUETES

De acuerdo a la sección 1.7.5.10 del primer capítulo, el diagrama de paquetes de la presente disertación se ilustra en la siguiente figura.

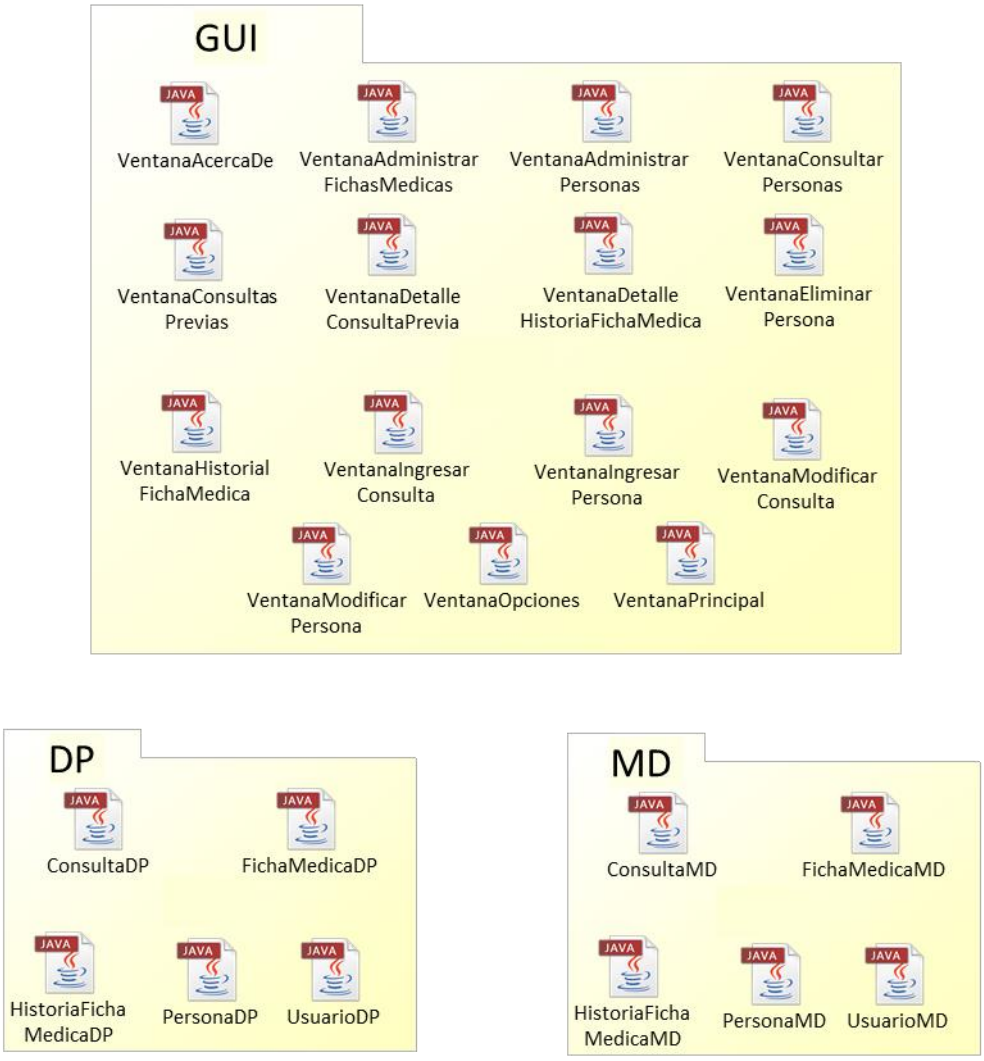

Figura 3.31. Diagrama de Paquetes.

3.6 DIAGRAMA DE COMPONENTES

De acuerdo a la sección 1.7.5.11 del primer capítulo, el diagrama de componentes de la presente disertación se ilustra en la siguiente figura.

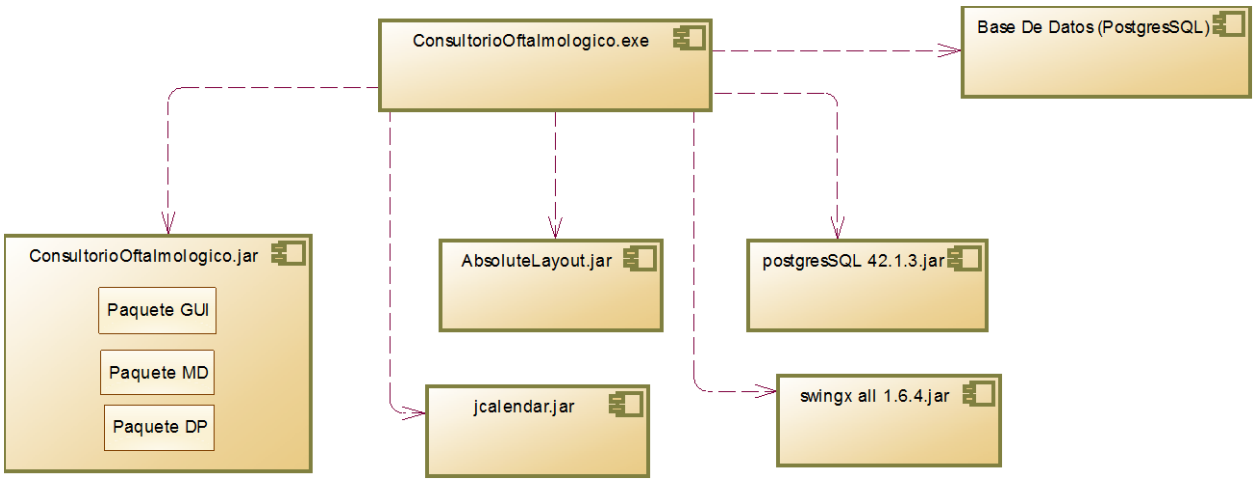

Figura 3.32. Diagrama de Componentes.

3.7 DIAGRAMA DE DESPLIEGUE

De acuerdo a la sección 1.7.5.12 del primer capítulo, el diagrama despliegue de la presente disertación se ilustra en la siguiente figura.

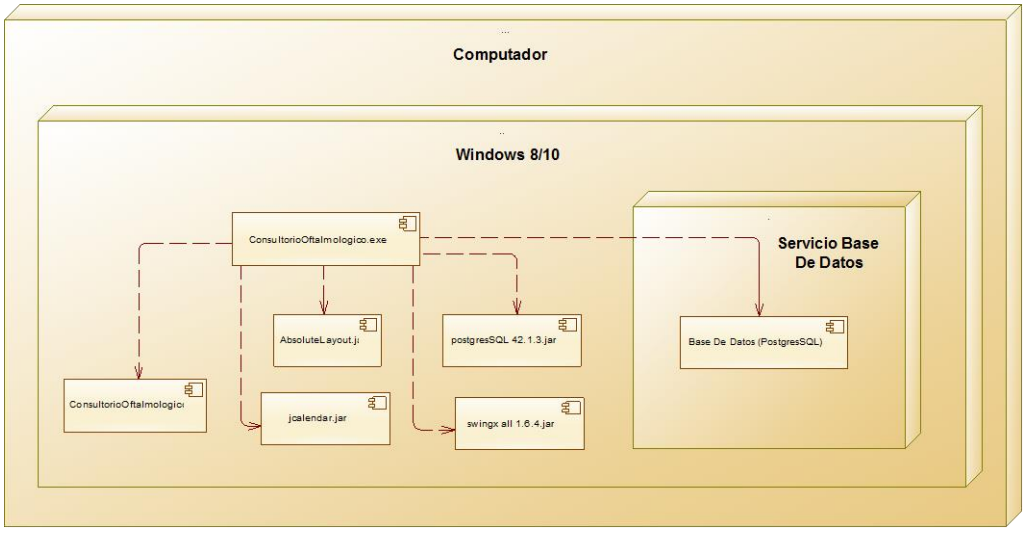

Figura 3.33. Diagrama de Despliegue.

CAPÍTULO 4: FASE DE CONSTRUCCIÓN Y TRANSICIÓN

En este capítulo se desarrolla el programa, se incluyen fragmentos de código y se elaborarán los manuales de instalación y de usuario. La disertación completa se acompaña en el CD adjuntado.

4.1 FRAGMENTOS DE CÓDIGO

4.1.1 Actualización del panel de elementos.

VentanaPrincipal

```
private void mostrarVentanaOpciones(UsuarioDP usuario)
  {
        VentanaOpciones ventanaOpciones  =  new VentanaOpciones(usuario, PanelCargaDeElementos);
        PanelCargaDeElementos.removeAll();
        ventanaOpciones.setVisible(true);
        ventanaOpciones.setSize(PanelCargaDeElementos.getSize());
        PanelCargaDeElementos.add(ventanaOpciones);
        PanelCargaDeElementos.revalidate();
        PanelCargaDeElementos.repaint();
  }
```

El programa fue desarrollado para tener un 'jFrame' principal, el cual contiene un panel (jPanel) de elementos que se actualiza según la opción seleccionada por el usuario. Ese panel de elementos se envía en el constructor de la mayoría de ventanas para que pueda ser redibujado con los elementos de la ventana que se desea visualizar.

4.1.2 Ingreso de una persona.

VentanaIngresarPersona

```
private void guardar()
  {
        String mensajeValidacion = validarDatos();
        if(mensajeValidacion!="")
          {   JOptionPane.showMessageDialog(this, "No se pudo Guardar:\n"+mensajeValidacion , "Error",
              JOptionPane.ERROR_MESSAGE);
          }
        else
        {
          persona.setNumIdentificacion(TXT_NumIdentificacion.getText());
          persona.setNombres(TXT_Nombres.getText());
          persona.setApellidos(TXT_Apellidos.getText());
          persona.setCelular(TXT_Celular.getText());
          persona.setDireccion(TXT_LugResidencia.getText());
          persona.setEmail(TXT_Email.getText());
          persona.setFechaNacimiento(TXT_FechaNacimiento.getDate());
          persona.setLugNacimiento(TXT_LugNacimiento.getText());
          persona.setTelefono(TXT_Telefono.getText());
           if(pathFoto!="")
          {
             persona.setPathFoto(pathFoto);
          }
          else
          {
             persona.setPathFoto(pathFotoDefault);
          }
          int idNuevaPersona = persona.ingresarPersona(usuario.getIdUsuario());
          if(idNuevaPersona!=-1)
          {
            if(RBTN_Administrador.isSelected()||RBTN_Asistente.isSelected())
            {
              UsuarioDP nuevoUsuario = new UsuarioDP(TXT_Usuario.getText(), TXT_Contraseña.getText());
              if(RBTN_Administrador.isSelected())
              {
                 nuevoUsuario.setFuncion("Administrador");
              }
              else
              {
                 nuevoUsuario.setFuncion("Asistente");
              }
              nuevoUsuario.setContraseña(TXT_Contraseña.getText());
              nuevoUsuario.setNombre(TXT_Usuario.getText().toLowerCase());
              nuevoUsuario.setIdPersona(idNuevaPersona);
              nuevoUsuario.ingresarUsuario(usuario.getIdUsuario());
            }
```

```
            if(RBTN_Paciente.isSelected())
            {
                FichaMedicaDP nuevaFichaMedica = new FichaMedicaDP();
                nuevaFichaMedica.ingresarFichaMedica(idNuevaPersona, usuario.getIdUsuario());
            }
            limpiarElementos();
            JOptionPane.showMessageDialog(this, "Se ingreso la persona exitosamente.", "Atencion",
            JOptionPane.INFORMATION_MESSAGE);
        }
        else
        {
            JOptionPane.showMessageDialog(this, "No se pudo ingresar la persona, vuelva a intentarlo", "Error",
            JOptionPane.ERROR_MESSAGE);
        }
    }
```

Para ingresar una persona nueva se validan los datos ingresados, se realizan los respectivos ‘sets’, y se llama al método guardar de la clase PersonaDP. Este método retorna un id entero positivo si la persona fue ingresada correctamente, se registra la actividad en el log y se prosigue a crear el usuario y/o paciente. Si el método retorna menos uno es porque hubo un error al guardar y de ser este el caso, se muestra un mensaje de error. A continuación se presenta el código de las clases que interactúan en este proceso.

PersonaDP

```
public int ingresarPersona(int idUsuarioConectado)
   {
      personaMD = new PersonaMD();
      return personaMD.ingresarPersona(this, idUsuarioConectado);
   }
```

PersonaMD

```
public int ingresarPersona(PersonaDP nuevaPersonaDP, int idUsuarioConectado)
   {   try
       {
        Connection conexion= conectar();

        PreparedStatement query = conexion.prepareStatement("Insert into Persona (numidentificacion, nombres, apellidos, fechanacimiento,
        foto, lugarnacimiento, telefonofijo, telefonomovil, direccion, email) values(?,?,?,?,?,?,?,?,?,?)");
        query.setString(1, nuevaPersonaDP.getNumIdentificacion());
        query.setString(2, nuevaPersonaDP.getNombres());
        query.setString(3, nuevaPersonaDP.getApellidos());
        LocalDate objectDate = nuevaPersonaDP.getFechaNacimiento().toInstant().atZone(ZoneId.systemDefault()).toLocalDate();
        query.setObject(4, objectDate);
        File file = new File(nuevaPersonaDP.getPathFoto());
        FileInputStream fis = new FileInputStream(file);
        query.setBinaryStream(5, fis, (int)file.length());
        query.setString(6, nuevaPersonaDP.getLugNacimiento());
        query.setString(7, nuevaPersonaDP.getTelefono());
        query.setString(8, nuevaPersonaDP.getCelular());
        query.setString(9, nuevaPersonaDP.getDireccion());
        query.setString(10, nuevaPersonaDP.getEmail());
        query.execute();
        query.close();
        fis.close();
        conexion.close();
        conexion= conectar();

        Statement estatuto = conexion.createStatement();
        ResultSet resultado= estatuto.executeQuery("select * from persona where numidentificacion =
       '"+nuevaPersonaDP.getNumIdentificacion() + "'");
```

```
        while(resultado.next())
        {
          int idPersona = resultado.getInt("idpersona");
          estatuto.close();
          conexion.close();
           conexion= conectar();
            query = conexion.prepareStatement("Insert into logactividad (idusuario, accion, fecha) values(?,?,?)");
            query.setInt(1, idUsuarioConectado);
            query.setString(2, "Creacion de persona con id: "+ idPersona);
            DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
            LocalDateTime ahora = LocalDateTime.now();
            Timestamp timestamp = Timestamp.valueOf(ahora);
            query.setTimestamp(3, timestamp);
            query.execute();
            query.close();
            conexion.close();
          return idPersona;
        }
        return -1;
        }
        catch(SQLException e)
        {
             System.out.println(e.getMessage());
             return -1;
        }
        catch(IOException e)
        {

             System.out.println(e.getMessage());
             return -1;

        }
    }
```

4.1.3 Consultar personas por apellido.

VentanaConsultarPersonas

```
        PersonaDP persona = new PersonaDP();
        LinkedList<PersonaDP> personasConNombreoApellidoSimilar = new LinkedList<>();
        personasConNombreoApellidoSimilar =  persona.buscarPersonasPorApellido(TXT_BuscarPor.getText(), usuario.getIdUsuario());
        DefaultTableModel modelo = null;
        int r=0;
        String[] col = {"ID Persona", "Número de Identificación", "Apellidos", "Nombres", "Es Doctor", "Es Asistente", "Es Paciente"};
        String [][] data = {{"","","","","","","",""}};
            modelo = new DefaultTableModel(data, col){
                public boolean isCellEditable(int row, int column)
                {
                    return false;
                }
            };
        TBL_Personas.setModel(modelo);
        for(int i =0; i<personasConNombreoApellidoSimilar.size(); i++)
        {
            String esDoctor = "No";
            String esAsistente = "No";
            String esPaciente = "No";
            if(personasConNombreoApellidoSimilar.get(i).getUsuarioDP()!=null)
            {
                if(personasConNombreoApellidoSimilar.get(i).getUsuarioDP().getFuncion().compareTo("Administrador")==0)
                {
                    esDoctor = "Si";
                }
                else
                {
                    esAsistente = "Si";
                }
            }
            if(personasConNombreoApellidoSimilar.get(i).getFichaMedicaDP()!=null)
            {
                esPaciente = "Si";
            }
            modelo.insertRow(r++, new Object[]{personasConNombreoApellidoSimilar.get(i).getIdPersona(),
            personasConNombreoApellidoSimilar.get(i).getNumIdentificacion(),
            personasConNombreoApellidoSimilar.get(i).getApellidos(), personasConNombreoApellidoSimilar.get(i).getNombres(),
            esDoctor, esAsistente, esPaciente});

        }
```

PersonaDP

```
public LinkedList<PersonaDP> buscarPersonasPorApellido(String apellidoDeBusqueda, int idUsuarioConectado)
   {
       personaMD = new PersonaMD();
       return personaMD.buscarPersonasPorApellido(apellidoDeBusqueda, idUsuarioConectado);
   }
```

PersonaMD

```
public LinkedList<PersonaDP> buscarPersonasPorApellido(String apellidoDeBusqueda, int idUsuarioConectado)
   {
        LinkedList<PersonaDP> personaDPs = new LinkedList<>();
       try
       {
       Connection conexion= conectar();

        PreparedStatement query = conexion.prepareStatement("Select numidentificacion from persona" +
        "  where lower(apellidos) like '%"+apellidoDeBusqueda.toLowerCase()+"%' "
                + "order by apellidos");
        ResultSet rs = query.executeQuery();
           while (rs.next()) {
               String numIdentificacion = rs.getString("numidentificacion");
               PersonaDP personaConNombreIgual = new PersonaDP();
               cargarDatosBuscarPersona(numIdentificacion, personaConNombreIgual);
               personaDPs.add(personaConNombreIgual);
           }
        query.close();
        conexion.close();

        conexion= conectar();
           query = conexion.prepareStatement("Insert into logactividad (idusuario, accion, fecha) values(?,?,?)");
           query.setInt(1, idUsuarioConectado);
           query.setString(2, "Buscar personas con Nombre similar a: "+ apellidoDeBusqueda);
           DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
           LocalDateTime ahora = LocalDateTime.now();
           Timestamp timestamp = Timestamp.valueOf(ahora);
           query.setTimestamp(3, timestamp);

           query.execute();
           query.close();
           conexion.close();


        return personaDPs;
       }
       catch(SQLException e)
       {  System.out.println(e.getMessage());
           return personaDPs;
       }
   }
```

Para la consulta de personas por apellido se realiza una búsqueda SQL con la sentencia ‘Like’. El método que realiza esta consulta, luego de haber registrado la actividad en el log, devuelve una LinkedList de PersonaDP, la misma que es iterada en el GUI para presentar los datos que contiene la lista en un ‘jTable’.

4.1.4 Modificar Ficha Médica

VentanaAdministrarFichasMedicas

```
private void guardar(){

      int resultado = JOptionPane.showConfirmDialog(null, "Se actualizarán los datos de la ficha.\n"
            + "Recuerde que la información antigua será registrada en el historial de la ficha.\n"
            + "¿Desea Continuar?", "Atención", JOptionPane.YES_NO_OPTION);
      if(resultado == JOptionPane.YES_OPTION)
      {
      persona.getFichaMedicaDP().guardarHistoria(usuario.getIdUsuario());
      persona.getFichaMedicaDP().setAlergias(TXT_Alergias.getText());
      persona.getFichaMedicaDP().setAntecedentesFamiliares(TXT_AntFamiliares.getText());
      persona.getFichaMedicaDP().setAutorefractorOD(TXT_AutorrefractorOD.getText());
      persona.getFichaMedicaDP().setAutorefractorOI(TXT_AutorrefractorOI.getText());
      persona.getFichaMedicaDP().setAvCercaCCAO(TXT_AVCercaCCAO.getText());
      persona.getFichaMedicaDP().setAvCercaSCAO(TXT_AVCercaSCAO.getText());
      persona.getFichaMedicaDP().setAvLejosCCAO(TXT_AVLejosCCAO.getText());
      persona.getFichaMedicaDP().setAvLejosSCAO(TXT_AVLejosSCAO.getText());
      persona.getFichaMedicaDP().setAvLejosCCOD(TXT_AVLejosCCOD.getText());
      persona.getFichaMedicaDP().setAvLejosCCOI(TXT_AVLejosCCOI.getText());
      persona.getFichaMedicaDP().setAvLejosSCOD(TXT_AVLejosSCOD.getText());
      persona.getFichaMedicaDP().setAvLejosSCOI(TXT_AVLejosSCOI.getText());
      persona.getFichaMedicaDP().setBalanceMuscular(TXT_BalanceMuscular.getText());
      persona.getFichaMedicaDP().setEbOD(TXT_EBOD.getText());
      persona.getFichaMedicaDP().setEbOI(TXT_EBOI.getText());
      persona.getFichaMedicaDP().setFoOD(TXT_FOOD.getText());
      persona.getFichaMedicaDP().setFoOI(TXT_FOOI.getText());
      persona.getFichaMedicaDP().setHistoriaMedica(TXT_HistoriaMedica.getText());
      persona.getFichaMedicaDP().setHistoriaOftalmologica(TXT_HistoriaOftalmologica.getText());
      persona.getFichaMedicaDP().setKhAnguloOD(Integer.parseInt(TXT_KHAnguloOD.getText().compareTo("")==0?"0":TXT_KHAnguloOD.
      getText()));
      persona.getFichaMedicaDP().setKhAnguloOI(Integer.parseInt(TXT_KHAnguloOD.getText().compareTo("")==0?"0":TXT_KHAnguloOD.g
      etText()));
      persona.getFichaMedicaDP().setKvAnguloOD(Integer.parseInt(TXT_KVAnguloOD.getText().compareTo("")==0?"0":TXT_KVAnguloOD.
      getText()));
      persona.getFichaMedicaDP().setKvAnguloOI(Integer.parseInt(TXT_KVAnguloOI.getText().compareTo("")==0?"0":TXT_KVAnguloOI.get
      Text()));
      persona.getFichaMedicaDP().setkHorizontalOD(Double.parseDouble(TXT_KHorizontalOD.getText().compareTo("")==0?"0":TXT_KHoriz
      ontalOD.getText()));
      persona.getFichaMedicaDP().setkHorizontalOI(Double.parseDouble(TXT_KHorizontalOI.getText().compareTo("")==0?"0":TXT_KHorizo
      ntalOI.getText()));
      persona.getFichaMedicaDP().setkVerticalOD(Double.parseDouble(TXT_KVerticalOD.getText().compareTo("")==0?"0":TXT_KVerticalO
      D.getText()));
      persona.getFichaMedicaDP().setkVerticalOI(Double.parseDouble(TXT_KVerticalOI.getText().compareTo("")==0?"0":TXT_KVerticalOI.g
      etText()));
      persona.getFichaMedicaDP().setLensAdd(TXT_LensAdd.getText());
      persona.getFichaMedicaDP().setLensOD(TXT_LensOD.getText());
      persona.getFichaMedicaDP().setLensOI(TXT_LensOD.getText());
      persona.getFichaMedicaDP().setLensPrisma(TXT_LensPrisma.getText());
      persona.getFichaMedicaDP().setManifiestaAdd(Integer.parseInt(TXT_ManifiestaAdd.getText());
      persona.getFichaMedicaDP().setManifiestaOD(TXT_ManifiestaOD.getText());
      persona.getFichaMedicaDP().setManifiestaOI(TXT_ManifiestaOI.getText());
      persona.getFichaMedicaDP().setMedicacion(TXT_Medicacion.getText());
      persona.getFichaMedicaDP().setSkMasOD(TXT_SKMASOD.getText());
      persona.getFichaMedicaDP().setSkMasOI(TXT_SKMASOI.getText());
      persona.getFichaMedicaDP().setSkOD(TXT_SKOD.getText());
      persona.getFichaMedicaDP().setSkOI(TXT_SKOI.getText());
      persona.getFichaMedicaDP().setUltimaRevision(new java.util.Date());
      persona.getFichaMedicaDP().actualizarFicha(usuario.getIdUsuario());


      JOptionPane.showMessageDialog(this, "Se actualizó la ficha exitosamente.", "Atencion", JOptionPane.INFORMATION_MESSAGE);
      persona.getFichaMedicaDP().setHistorialFichaMedicaDP(new LinkedList<HistoriaFichaMedicaDP>());
      persona.getFichaMedicaDP().setConsultasDP(new LinkedList<ConsultaDP>());
      persona.getFichaMedicaDP().cargarDatos(usuario.getIdUsuario());
      Date ultimaRevision = persona.getFichaMedicaDP().getUltimaRevision();
            String ultimaRevisionString = ultimaRevision!=null? ultimaRevision.toString():"Ninguna";
            TXT_UltimaModificacion.setText(ultimaRevisionString);

     }

   }
```

FichaMedicaDP

```
public void actualizarFicha(int idUsuarioConectado)
  {
     fichaMedicaMD = new FichaMedicaMD();
     fichaMedicaMD.actualizarFicha(this, idUsuarioConectado);
  }

  public void guardarHistoria(int idUsuarioConectado){
     if(ultimaRevision != null)
     {
        historiaFichaMedicaMD = new HistoriaFichaMedicaMD();
        historiaFichaMedicaMD.guardarHistoria(this, idUsuarioConectado);
     }
  }
```

FichaMedicaMD

```
public boolean actualizarFicha(FichaMedicaDP fichaAActualizar, int idUsuarioConectado)
  {
     try
     {
      Connection conexion= conectar();

      PreparedStatement query = conexion.prepareStatement("Update fichamedica set "
            + "historiaoftalmologica = ?, historiamedica = ?, medicacion = ?, antecedentesfamiliares = ?, "
            + "alergias = ?, avlejosscoi = ?, avlejosscod = ?, avlejosscao = ?, avlejosccoi = ?, avlejosccod = ?, "
            + "avlejosccao = ?, avcercascao = ?, avcercaccao = ?, khorizontalod = ?, khorizontaloi = ?, "
            + "khangulood = ?, khangulooi = ?, kverticalod = ?, kverticaloi = ?, kvangulood = ?, kvangulooi = ?, "
            + "lensoi = ?, lensod = ?, lensadd = ?, lensprisma = ?, balancemuscular = ?, autorefractoroi = ?, autorefractorod = ?, "
            + "skoi = ?, skod = ?, skmasoi = ?, skmasod = ?, manifiestaoi = ?, manifiestaod = ?, manifiestaadd = ?, eboi = ?, "
            + "ebod = ?, fooi = ?, food = ?, ultimarevision = ? "
            + "where idfichamedica = "+fichaAActualizar.getIdFichaMedica());

      query.setString(1, fichaAActualizar.getHistoriaOftalmologica());
      query.setString(2, fichaAActualizar.getHistoriaMedica());
      query.setString(3, fichaAActualizar.getMedicacion());
      query.setString(4, fichaAActualizar.getAntecedentesFamiliares());
      query.setString(5, fichaAActualizar.getAlergias());
      query.setString(6, fichaAActualizar.getAvLejosSCOI());
      query.setString(7, fichaAActualizar.getAvLejosSCOD());
      query.setString(8, fichaAActualizar.getAvLejosSCAO());
      query.setString(9, fichaAActualizar.getAvLejosCCOI());
      query.setString(10, fichaAActualizar.getAvLejosCCOD());
      query.setString(11, fichaAActualizar.getAvLejosCCAO());
      query.setString(12, fichaAActualizar.getAvCercaSCAO());
      query.setString(13, fichaAActualizar.getAvCercaCCAO());
      query.setDouble(14, fichaAActualizar.getkHorizontalOD());
      query.setDouble(15, fichaAActualizar.getkHorizontalOI());
      query.setInt(16, fichaAActualizar.getKhAnguloOD());
      query.setInt(17, fichaAActualizar.getKhAnguloOI());
      query.setDouble(18, fichaAActualizar.getkVerticalOD());
      query.setDouble(19, fichaAActualizar.getkVerticalOI());
      query.setInt(20, fichaAActualizar.getKvAnguloOD());
      query.setInt(21, fichaAActualizar.getKvAnguloOI());
      query.setString(22, fichaAActualizar.getLensOI());
      query.setString(23, fichaAActualizar.getLensOD());
      query.setInt(24, fichaAActualizar.getLensAdd());
      query.setInt(25, fichaAActualizar.getLensPrisma());
      query.setString(26, fichaAActualizar.getBalanceMuscular());
      query.setString(27, fichaAActualizar.getAutorefractorOI());
      query.setString(28, fichaAActualizar.getAutorefractorOD());
      query.setString(29, fichaAActualizar.getSkOI());
      query.setString(30, fichaAActualizar.getSkOD());
      query.setString(31, fichaAActualizar.getSkMasOI());
      query.setString(32, fichaAActualizar.getSkMasOD());
      query.setString(33, fichaAActualizar.getManifiestaOI());
      query.setString(34, fichaAActualizar.getManifiestaOD());
      query.setInt(35, fichaAActualizar.getManifiestaAdd());
      query.setString(36, fichaAActualizar.getEbOI());
      query.setString(37, fichaAActualizar.getEbOD());
      query.setString(38, fichaAActualizar.getFoOI());
      query.setString(39, fichaAActualizar.getFoOD());
      Timestamp timestamp = new Timestamp(fichaAActualizar.getUltimaRevision().getTime());
      query.setTimestamp(40, timestamp);

      query.execute();
      query.close();
```

```
            query = conexion.prepareStatement("Insert into logactividad (idusuario, accion, fecha) values(?,?,?)");
            query.setInt(1, idUsuarioConectado);
            query.setString(2, "Actualizar fichamedica con id: "+ fichaAActualizar.getIdFichaMedica());
            LocalDateTime ahora = LocalDateTime.now();
            timestamp = Timestamp.valueOf(ahora);
            query.setTimestamp(3, timestamp);
            query.execute();
            query.close();
            conexion.close();
         return true;
        }
        catch(SQLException e)
        {
             System.out.println(e.getMessage());
             return false;
        }
    }
```

Al actualizar una ficha médica primero se despliega un dialogo para que el usuario confirme si desea guardar los cambios. De ser así, se guardan los datos de la ficha médica no modificada en la tabla HistoriaFichaMedica. Una vez que se registró la ficha antigua se procede a actualizar la ficha médica de la tabla FichaMedica.

4.1.5 Imprimir Consulta

VentanaModificarConsulta

```
private void imprimirConsulta(){

        PrinterJob pj = PrinterJob.getPrinterJob();
        pj.setJobName(" Imprimir Consulta ");
        PageFormat pf = pj.defaultPage();
        pf.setOrientation(PageFormat.LANDSCAPE);
        Printable objectoAImprimir = new Printable() {
          public int print(Graphics pg, PageFormat pf, int pageNum){
            if (pageNum > 0){
            return Printable.NO_SUCH_PAGE;
            }

            Graphics2D g2 = (Graphics2D) pg;

            g2.translate(pf.getImageableX(), pf.getImageableY());
             g2.scale(0.95, 0.95);
             SCRL_Pantalla.paint(g2);

            return Printable.PAGE_EXISTS;
          }
        };
        pj.setPrintable(objectoAImprimir, pf);
        if (pj.printDialog() == false)
        {
            return;
        }

        try {
             pj.print();
        }
        catch (PrinterException ex) {
             SPN_Fecha.setVisible(true);

            TXT_FechaImpresion.setText(diaLabel);
            TXT_FechaImpresion.setVisible(false);

            LBL_FirmaDoctor.setVisible(false);
            LBLI_LineaFirma.setVisible(false);
            LBLB_Guardar.setVisible(true);
        }
}
```

Para imprimir la consulta se utiliza la clase ‘PrinterJob’, que mediante el uso de ‘Graphics2D’, se encarga de mostrar un dialogo para la impresión de elementos en un ‘jPanel’.

4.2 MANUAL DE INSTALACIÓN

Los archivos necesarios para la instalación se encuentran dentro del CD adjuntado a la disertación.

4.2.1 Instalación de la Base de Datos.

a) Dar doble click en PostgreSQL para ejecutar el instalador de la base de datos (Ejecutar con los parámetros que vienen por defecto).

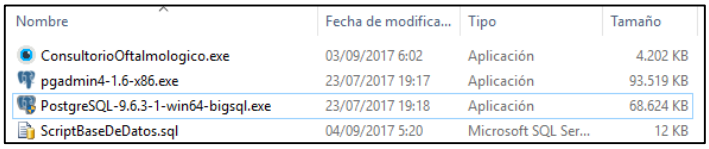

Figura 4.1. Archivos incluidos para instalación del programa.

Figura 4.2. Dialogo para instalar la base de datos.

b) Dar doble click en pgadmin4 para ejecutar el instalador del gestionador visual de la base de datos.

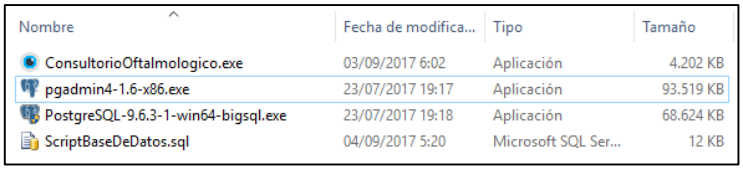

Figura 4.3. Archivos incluidos para instalación del programa.

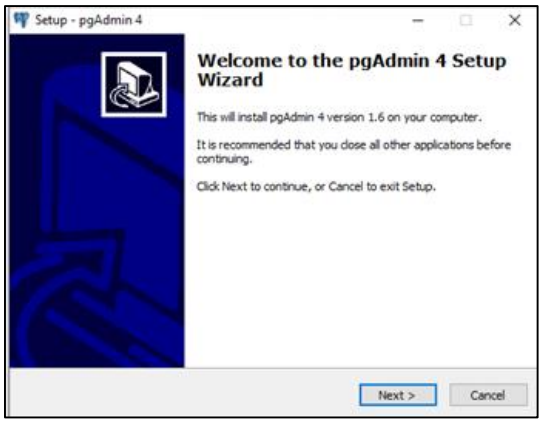

Figura 4.4. Dialogo para instalar pgadmin4.

c) Ejecutar pgadmin4 y crear una base de datos con el nombre de 'ConsultorioOftalmológico'.

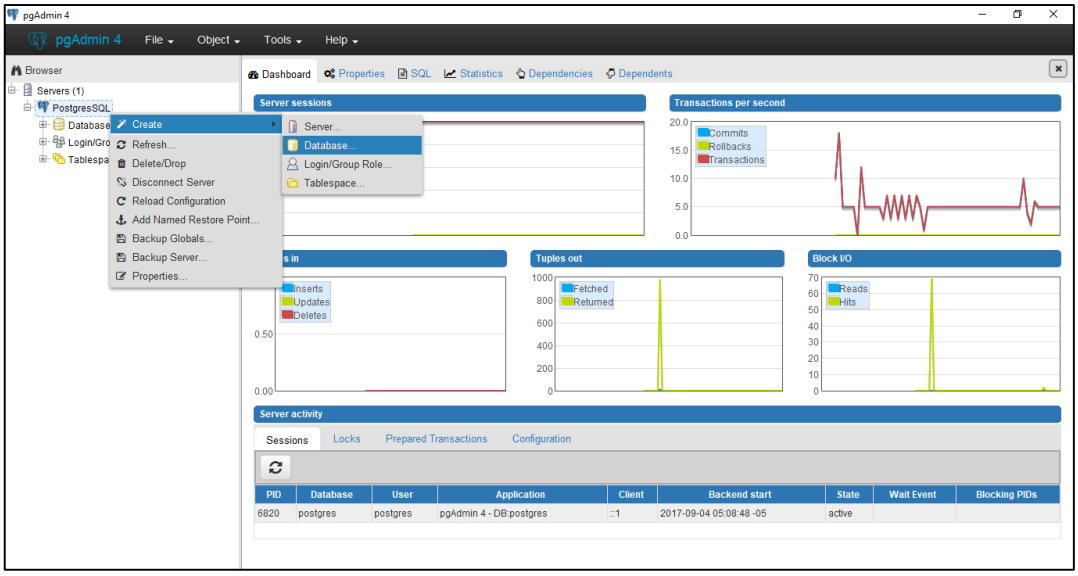

Figura 4.5. pgAdmin4.

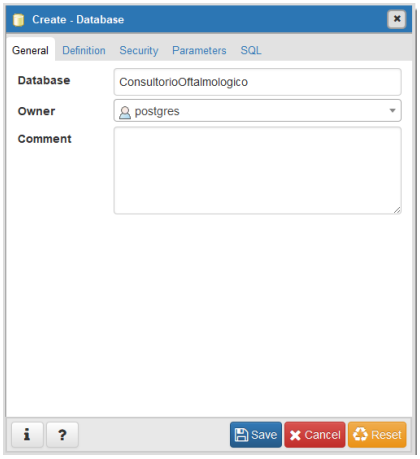

Figura 4.6. Dialogo para crear una base de datos.

d) Ejecutar el Script de la base de datos.

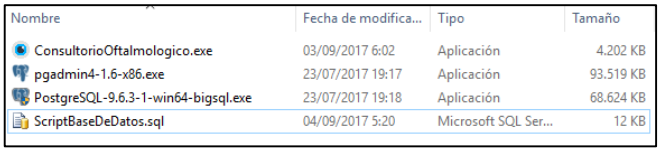

Figura 4.7. Archivos incluidos para instalación del programa.

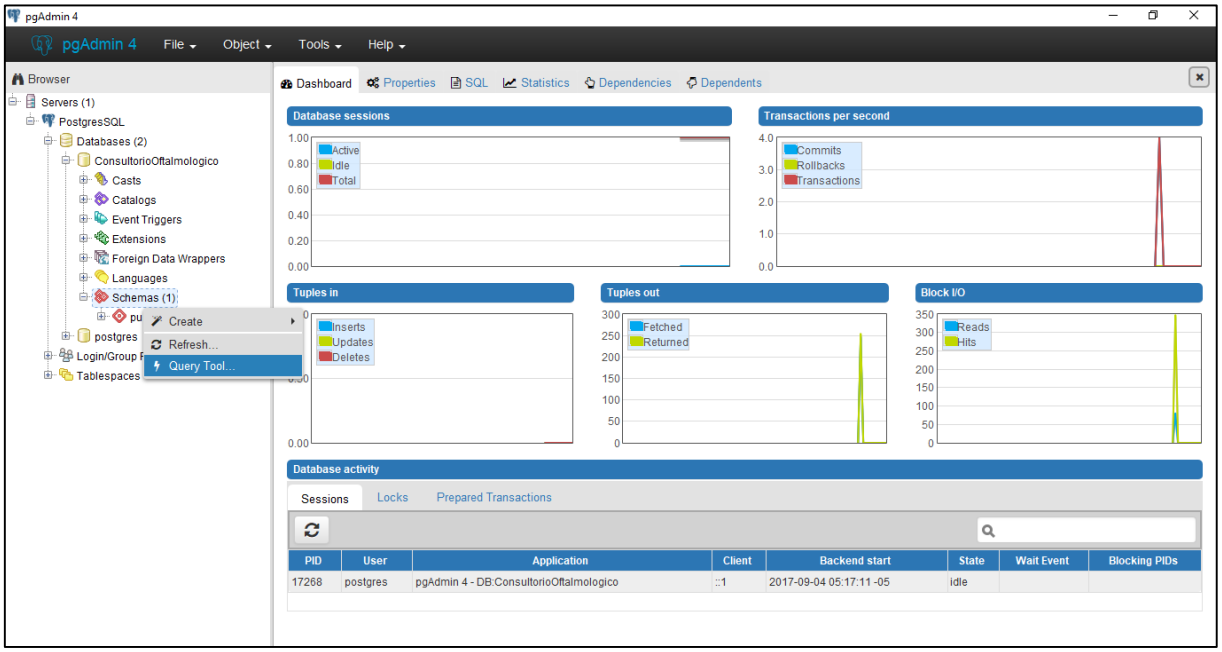

Figura 4.8. Herramienta para querys de pgAdmin.

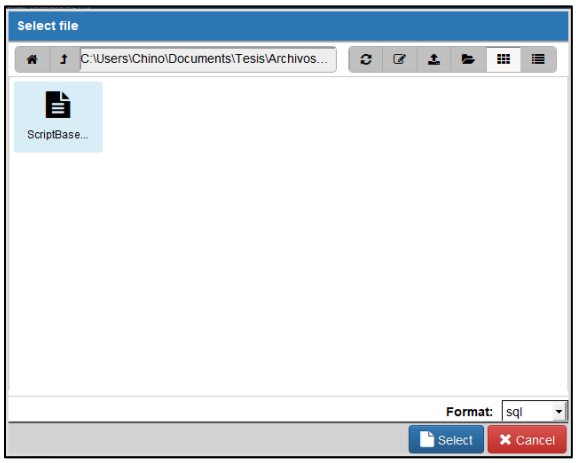

Figura 4.9. Dialogo para seleccionar archivo .sql.

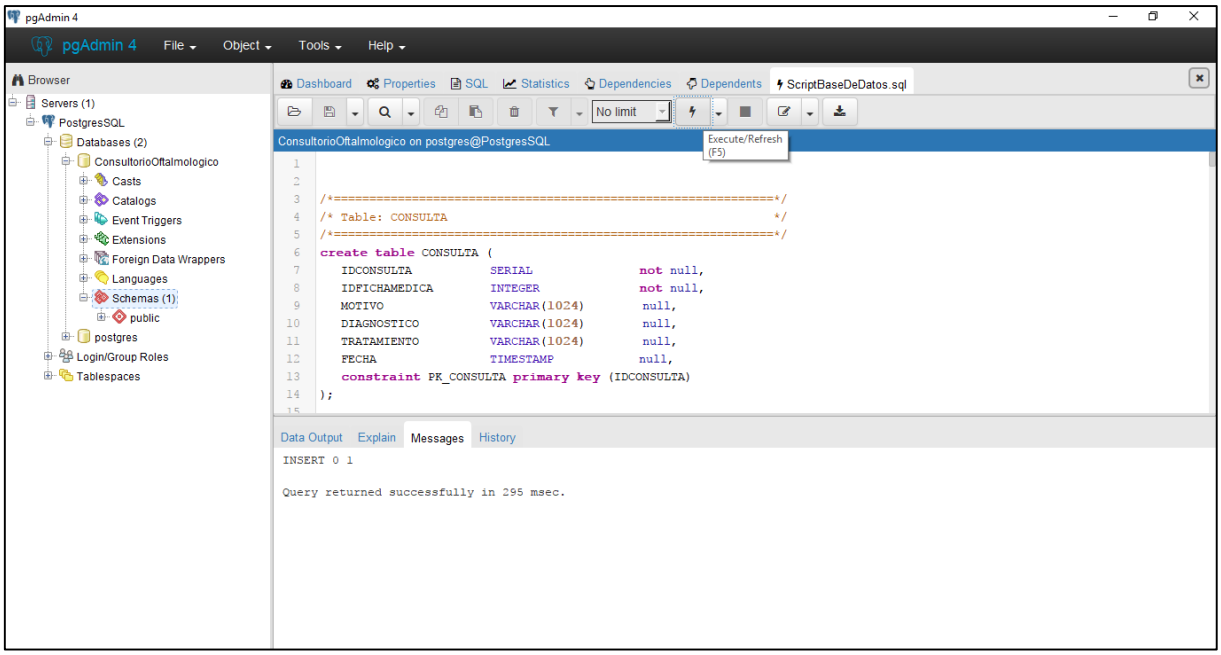

Figura 4.10. Ejecución del script.

e) Ejecutar el programa.

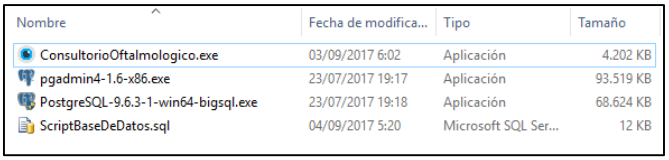

Figura 4.11. Archivos incluidos para instalación del programa.

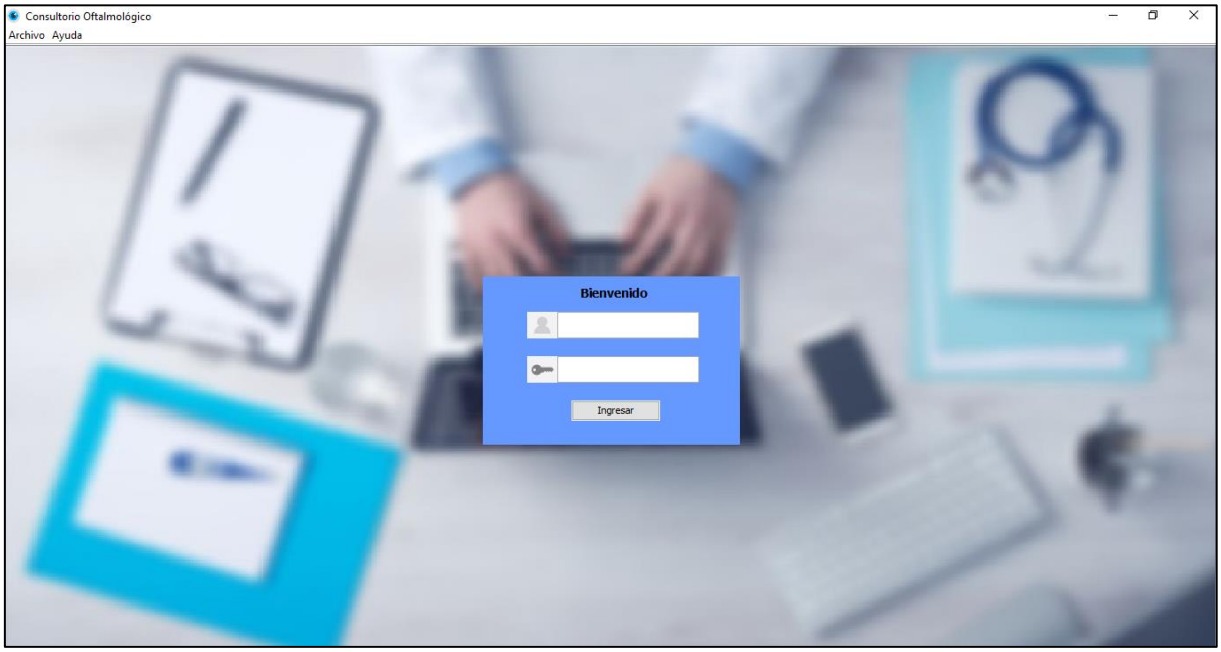

Figura 4.12. Pantalla principal del programa.

4.3 MANUAL DE USUARIO

4.3.1 Manual para administradores/doctores.

Para acceder por primera vez al sistema se ha creado una persona inicial con privilegios de Doctor, es decir que podrá crear otros doctores, asistentes y/o pacientes y a la vez modificar fichas médicas y administrar consultas.

Los datos de esta persona son:

- Usuario: *admin*
- Contraseña: *admin*
- N° de identificación: 123456789
- Nombres: *admin*
- Apellidos: *admin*

4.3.1.1 Administrar Personas.

Para la gestión de personas se debe acceder a la opción de ‘Administrar Personas’. Luego se desplegará una ventana para ingresar, modificar, eliminar y consultar personas.

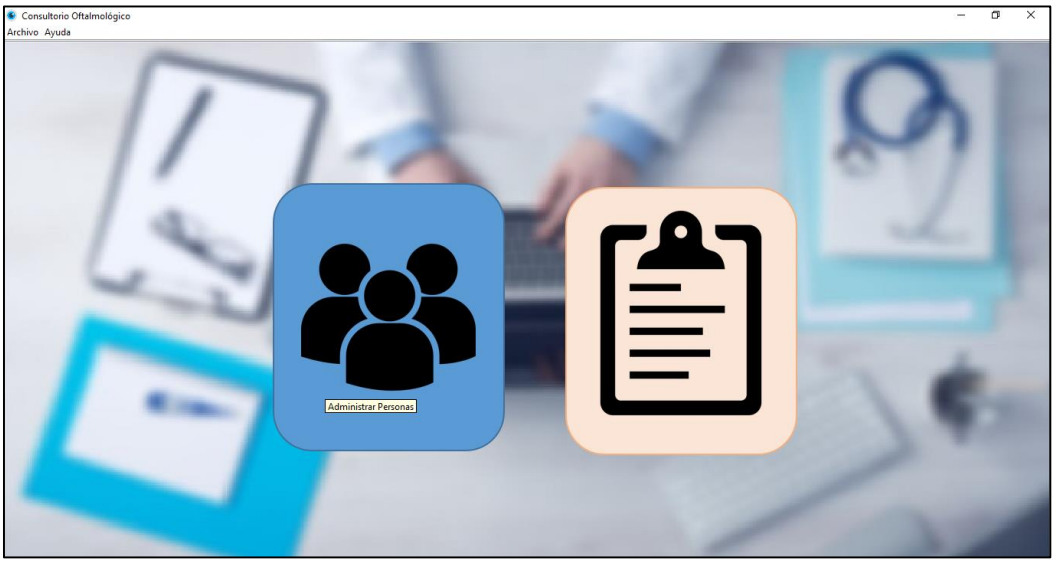

Figura 4.13. Pantalla de ‘Opciones’ de usuarios con rol de ‘Doctor’.

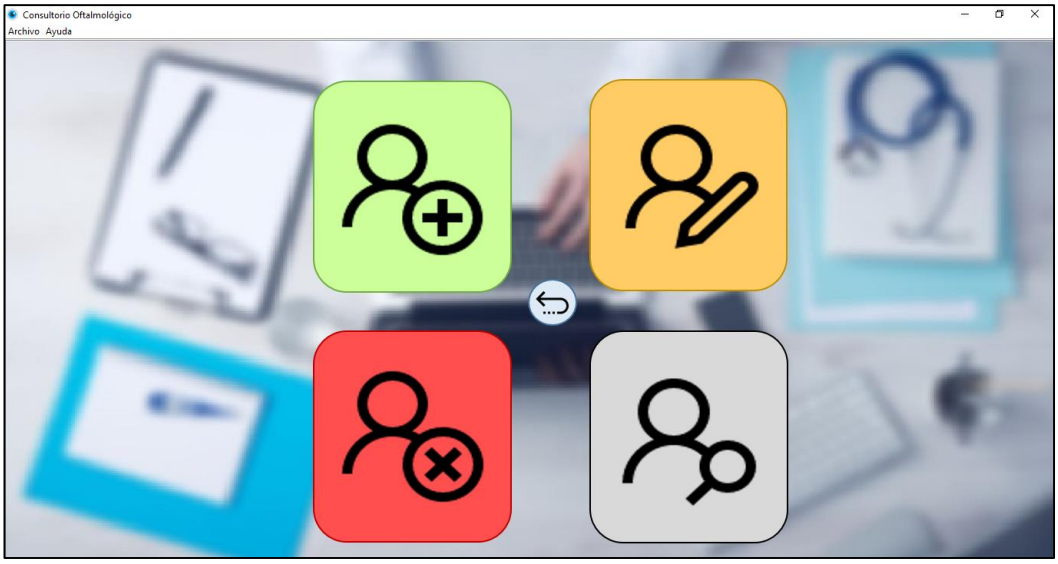

Figura 4.14. Pantalla ‘Administrar Personas’ de usuarios con rol de ‘Doctor’.

a) Para ingresar una persona se debe dar click en el botón de color verde ‘Ingresar Persona’. Se despegará una pantalla para ingresar los datos de una persona. Luego se debe llenar la información y presionar el botón guardar.

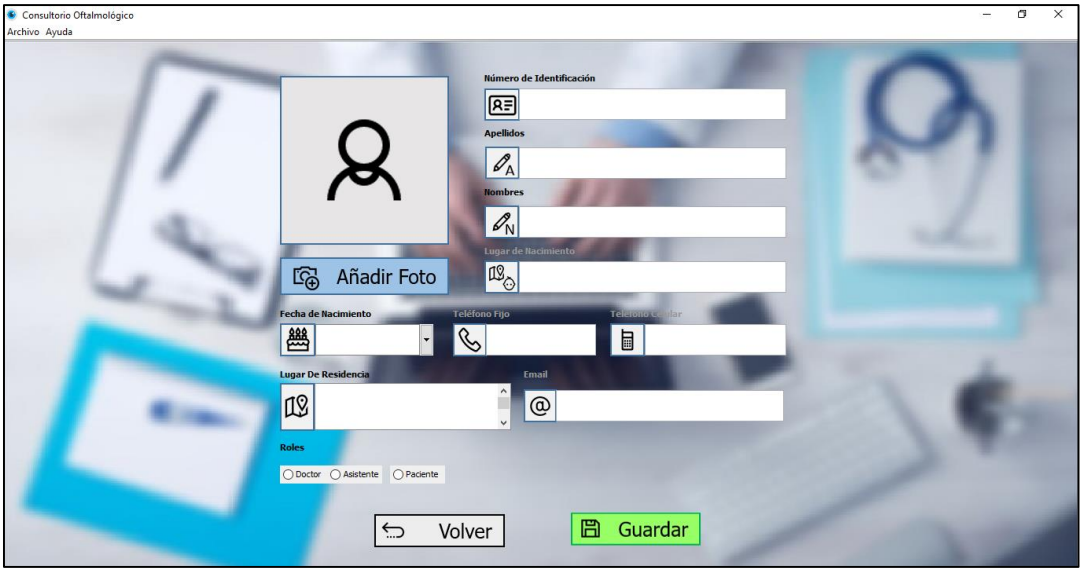

Figura 4.15. Pantalla ‘Ingresar Personas’ de usuarios con rol de ‘Doctor’.

Nota: Al ingresar un número de identificación se valida que este no exista en el sistema. De ser así se desplegará el siguiente mensaje:

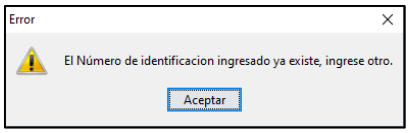

Figura 4.16. Mensaje de validación de número de identificación.

Al seleccionar la opción de 'Doctor' o Asistente se desplegarán dos cuadros para ingresar el usuario y contraseña de la nueva persona. Se validará que nombre de usuario ingresado no exista en el sistema y no exceda la longitud de diez caracteres.

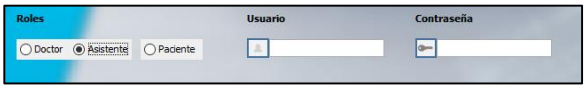

Figura 4.17. Cuadros de texto para ingresar usuario y contraseña.

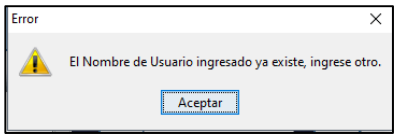

Figura 4.18. Mensaje de validación de nombre de usuario.

b) Para editar una persona se debe dar click en el botón tomate 'Modificar Persona'. Se desplegará una pantalla para buscar la persona que se desea editar por número de identificación. Una vez cargados los datos de la persona se podrán editar los mismos.

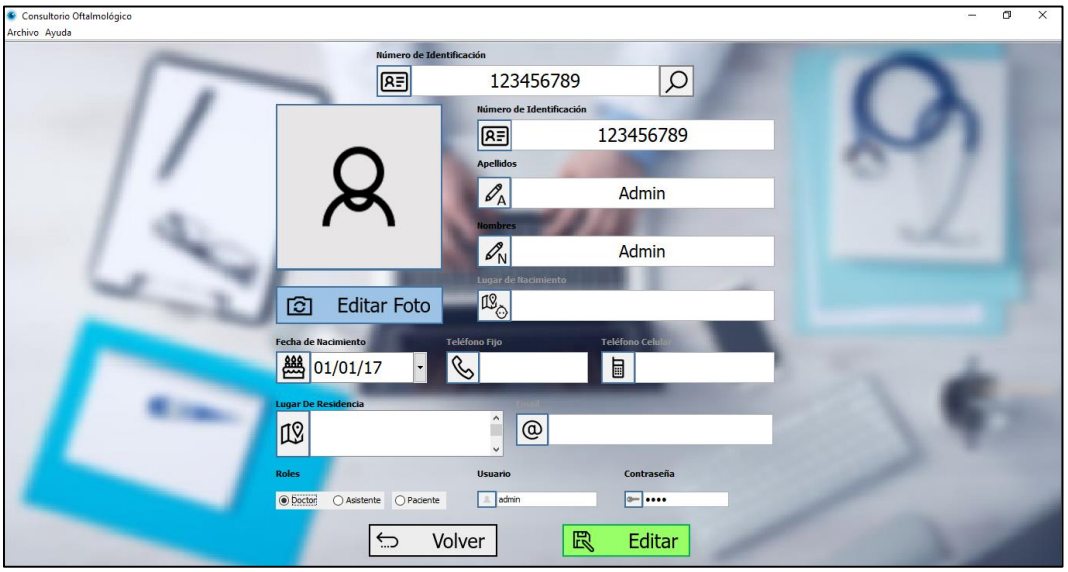

Figura 4.19. Pantalla 'Modificar Personas' de usuarios con rol de 'Doctor'.

Nota: Si no se encuentra el número de identificación ingresado se presentará el siguiente mensaje.

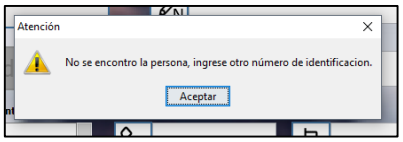

Figura 4.20. Mensaje de búsqueda de número de identificación sin resultados.

De igual manera que al ingresar una persona, si se modifica el usuario se validará que el nombre de este sea único.

Se debe tener en cuenta que el usuario que ya estaba creado en el sistema, es decir el usuario 'admin' no puede ser cambiado al rol de 'Asistente'.

c) Para eliminar una persona se debe dar click en el botón rojo 'Eliminar Personas'. Se desplegará una pantalla para buscar la persona a eliminar por número de identificación. Si se desea eliminar la persona que se debe presionar el botón 'Eliminar' y aplastar 'Ok' en el dialogo de confirmación.

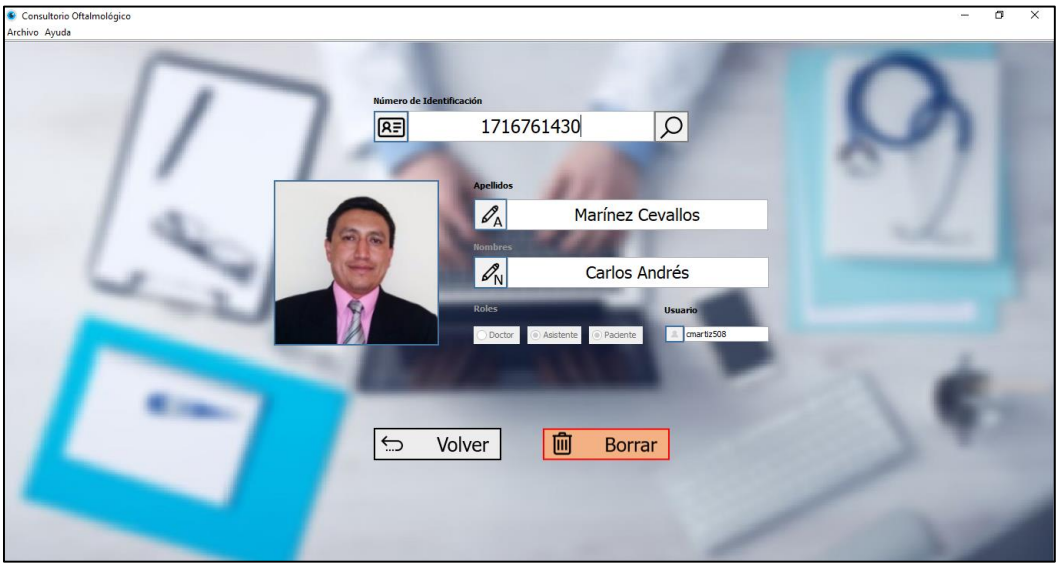

Figura 4.21. Pantalla 'Eliminar Personas' de usuarios con rol de 'Doctor'.

Nota: La persona que se encuentra conectada no se podrá eliminar a sí mismo.

Si no se encuentra el número de identificación de la persona ingresa se mostrará un mensaje como en la 'Modificación de Personas'.

d) Para consultar personas se debe dar click en el botón gris ‘Consultar Personas’. Se mostrará una ventana para buscar personas por número de identificación, apellidos o nombres. De ser una búsqueda general (por apellidos o nombres) se presentará en una tabla las personas que cumplan con el criterio de búsqueda. De ser una búsqueda detallada (por número de identificación) se mostrará la información de la persona buscada.

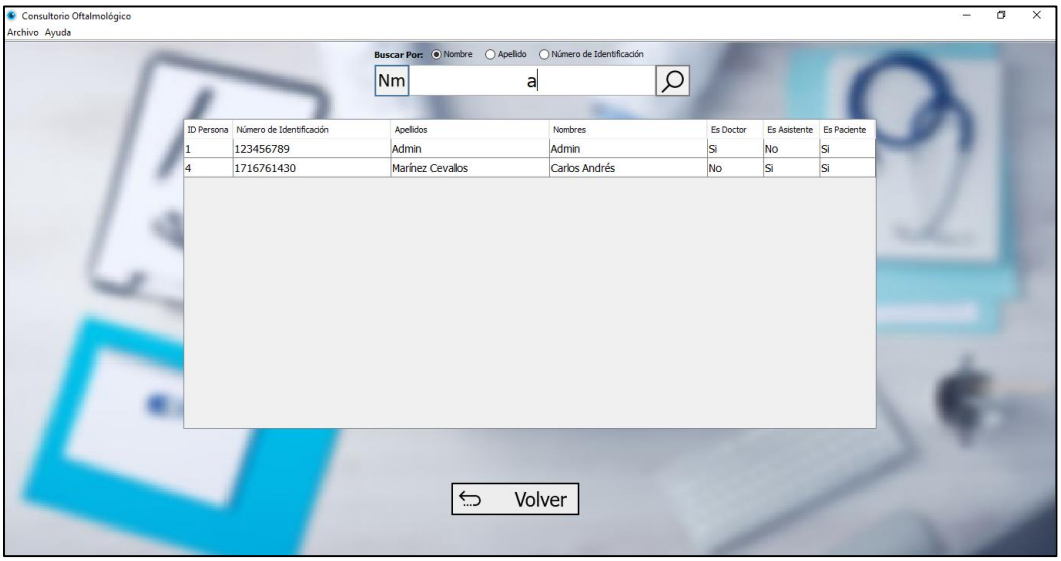

Figura 4.22. Pantalla ‘Consultar Personas’ de manera general.

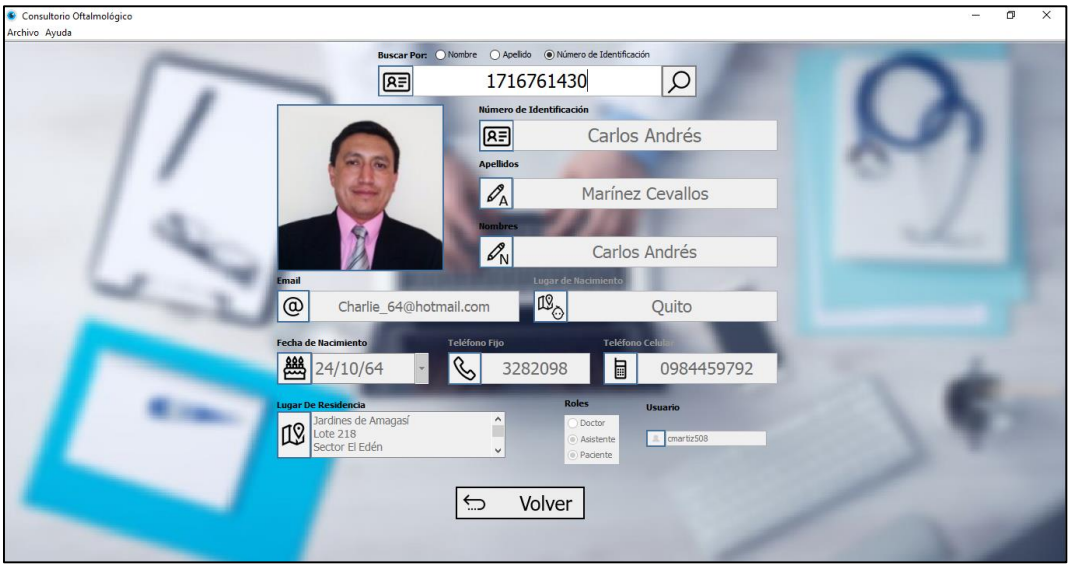

Figura 4.23. Pantalla ‘Consultar Personas’ de manera detallada.

4.3.1.2 Administrar Fichas Médicas.

Para la gestión de fichas médicas se debe acceder a la opción de ‘Administrar Fichas Médicas’. Luego se desplegará una ventana para modificar y consultar fichas médicas. En esta ventana también se podrá ingresar, modificar, eliminar y buscar Consultas.

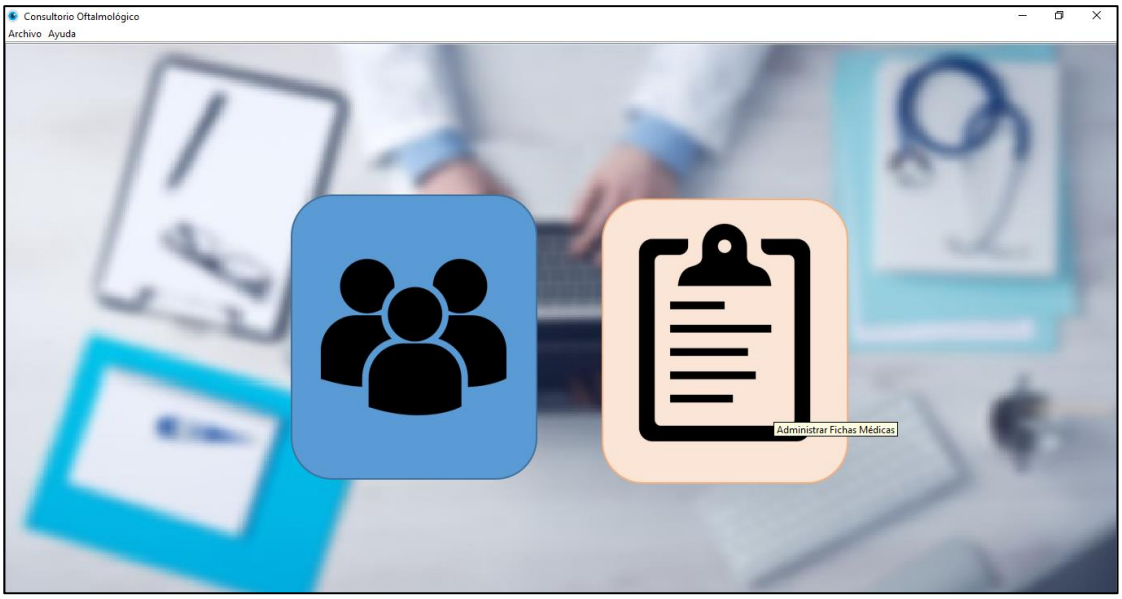

Figura 4.24. Pantalla de ‘Opciones’ de usuarios con rol de ‘Doctor’.

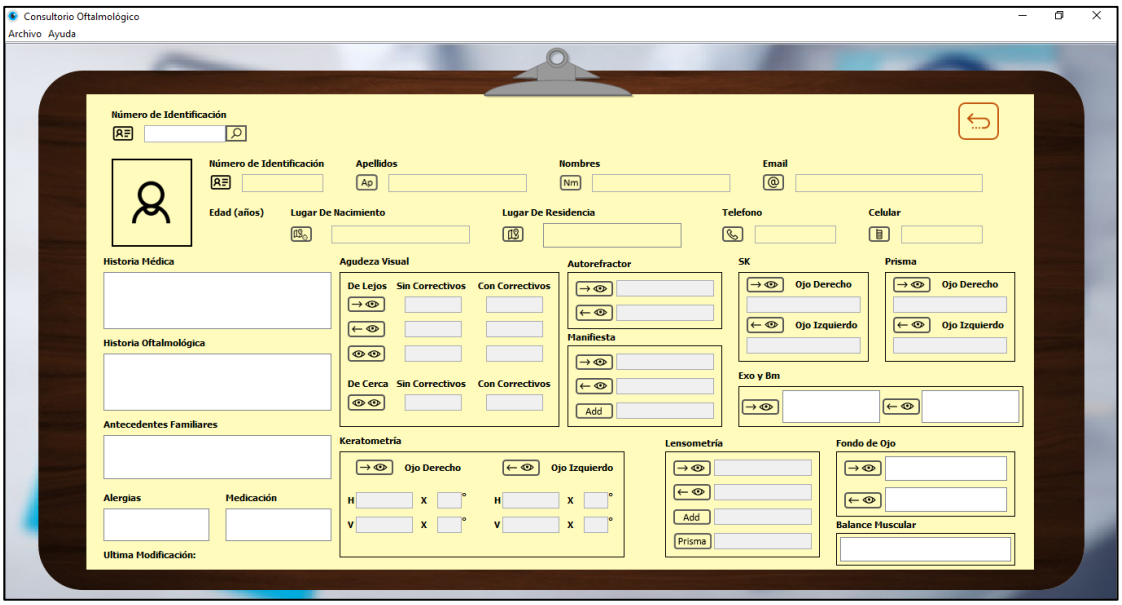

Figura 4.25. Pantalla ‘Administrar Fichas Médicas’ de usuarios con rol de ‘Doctor’.

a) Al crear una persona con rol de paciente se crea también su ficha médica pero con datos vacíos. Para modificar una ficha médica se debe buscar al paciente por su número de identificación. El sistema despliega la última ficha médica ingresada en el sistema.

A esta ficha se le pueden actualizar los datos y para guardar estos cambios se debe presionar el botón verde de la parte superior derecha ‘Guardar Ficha Médica’. Se mostrará un dialogo de confirmación en el que se informa que la ficha antigua será guardada en el historial.

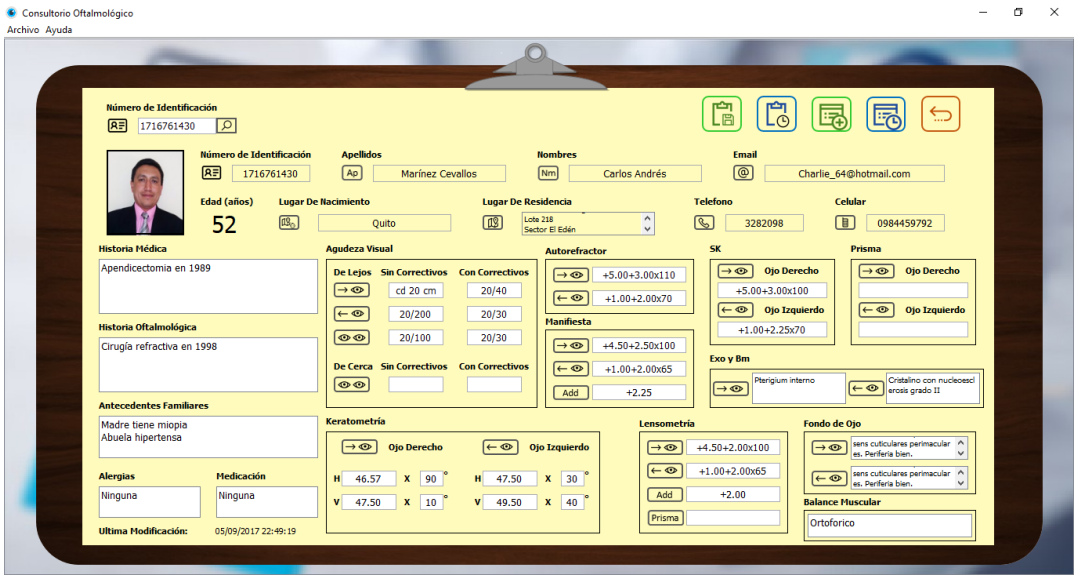

Figura 4.26. Ficha Médica de una Persona.

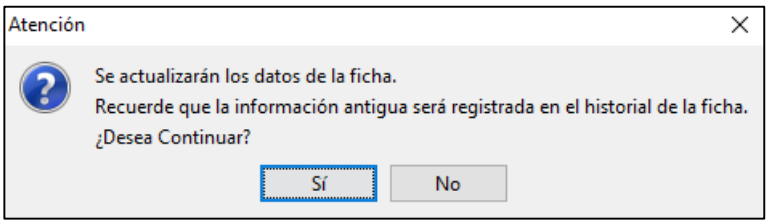

Figura 4.27. Dialogo de confirmación al guardar la ficha.

b) Para consultar todas las fichas médicas anteriores se debe dar click en el botón azul de la parte superior derecha ‘Historial Ficha Médica’. El sistema desplegará una ventana con el listado de todas las fichas médicas anteriores, listadas por el día y hora en el que fueron ingresados al sistema. Para ver el detalle de una ficha médica anterior se debe seleccionar una y dar click en el botón ‘Detalle’.

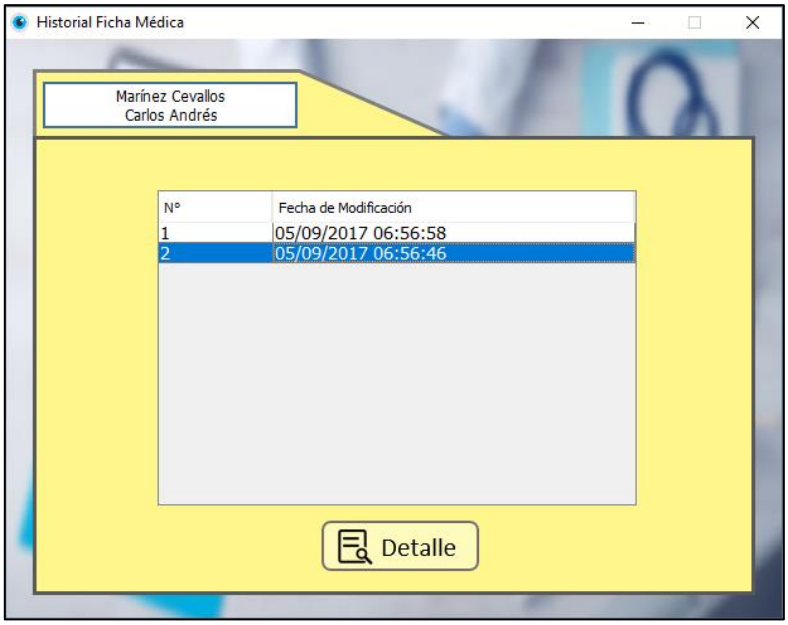

Figura 4.28. Ventana 'Historial Ficha Médica'.

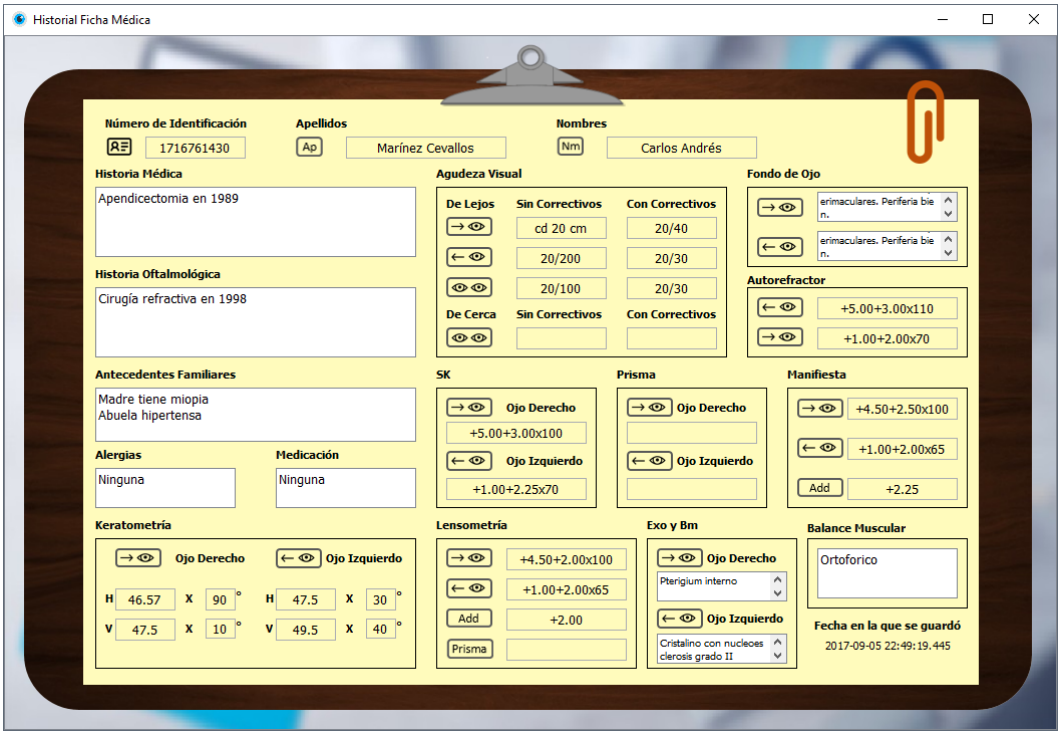

Figura 4.29. Ventana 'Detalle Historial Ficha Médica'.

4.3.1.3 Administrar Consultas

Para la administración de consultas se debe acceder a la ficha médica como se indica en el punto 4.3.1.2.

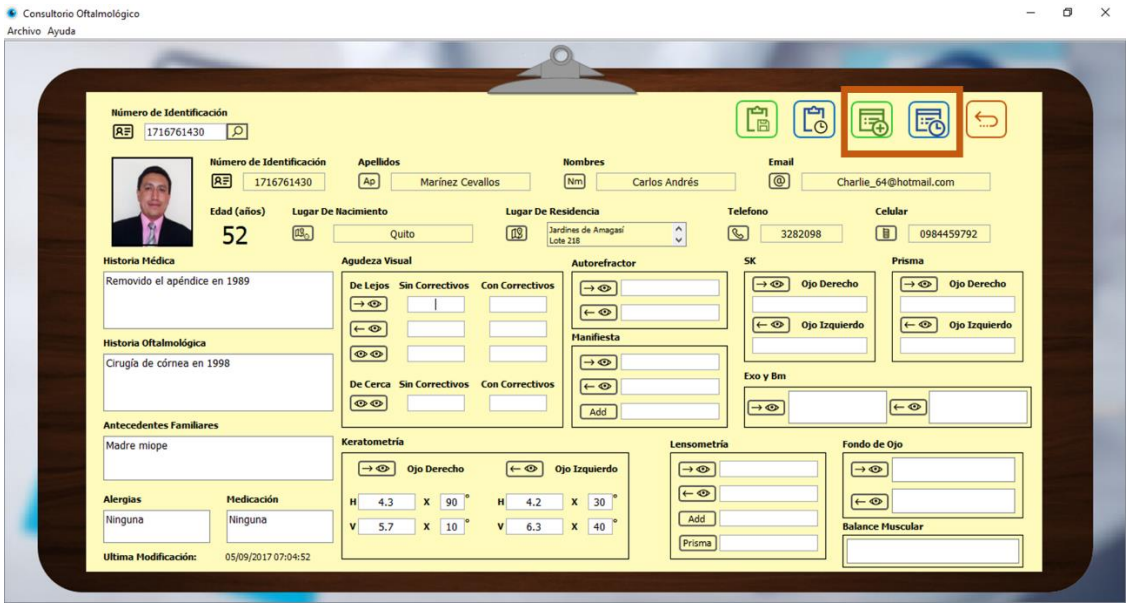

Figura 4.30. Opciones para administrar consultas.

a) Para ingresar una consulta se debe presionar el botón verde de la parte superior derecha ‘Ingresar Consulta’. El sistema despliega una ventana para ingresar la consulta. Una vez llenos los datos de la consulta se presiona el botón guardar y una vez guardada la consulta se muestra un dialogo preguntando si se desea imprimir dicha consulta.

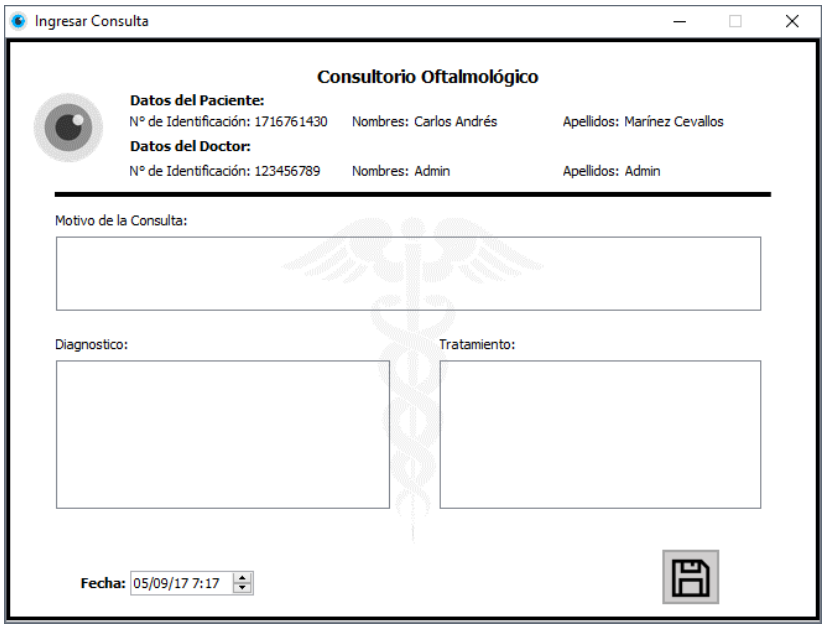

Figura 4.31. Ventana para Ingresar Consulta.

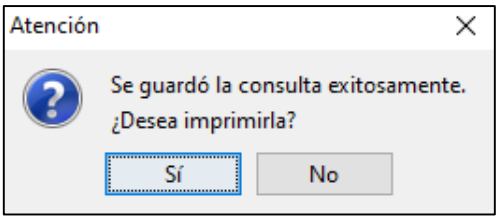

Figura 4.32. Dialogo de confirmación para imprimir consulta.

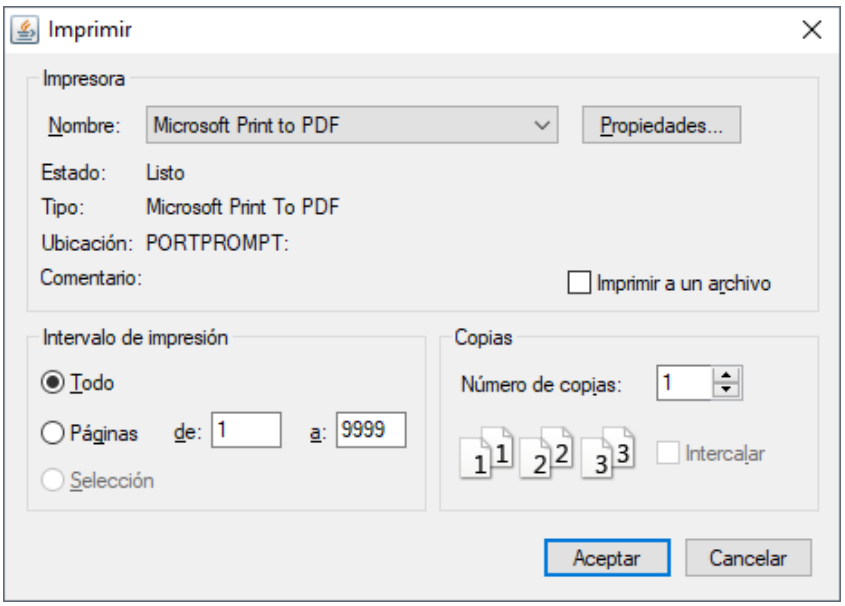

Figura 4.33. Dialogo para imprimir la consulta.

b) Para modificar consultas se debe presionar el botón azul de la parte superior derecha ‘Consultas Previas’. El sistema muestra un listado de las consultas hechas anteriormente. Luego se selecciona una consulta del listado y se da click en el botón ‘Editar.

Se muestra en una ventana la consulta seleccionada, en esta ventana los datos de la consulta pueden ser modificados. Una vez modificada la consulta se presiona el botón ‘Editar’. Cuando el sistema haya terminado de guardar la consulta se despliega un mensaje preguntando si se desea imprimir la nueva consulta modificada.

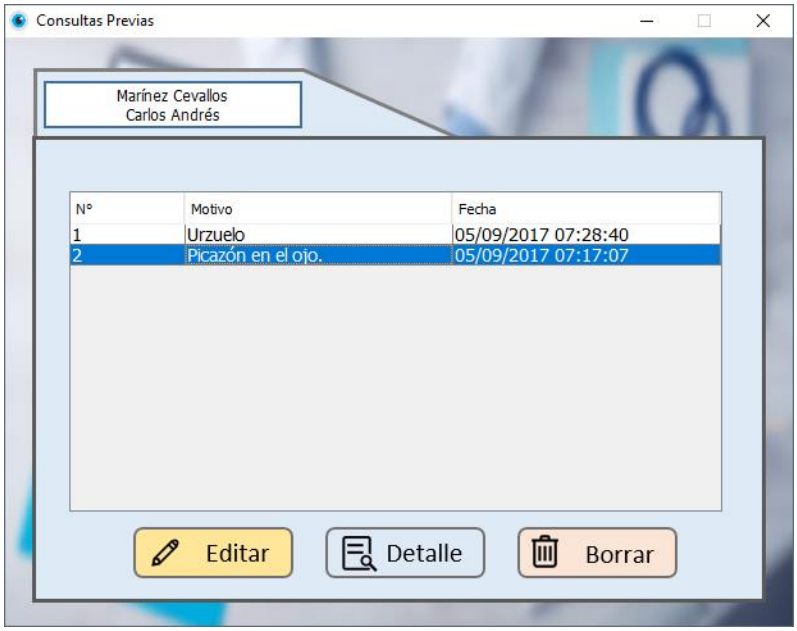

Figura 4.34. Ventana de 'Consultas Previas'.

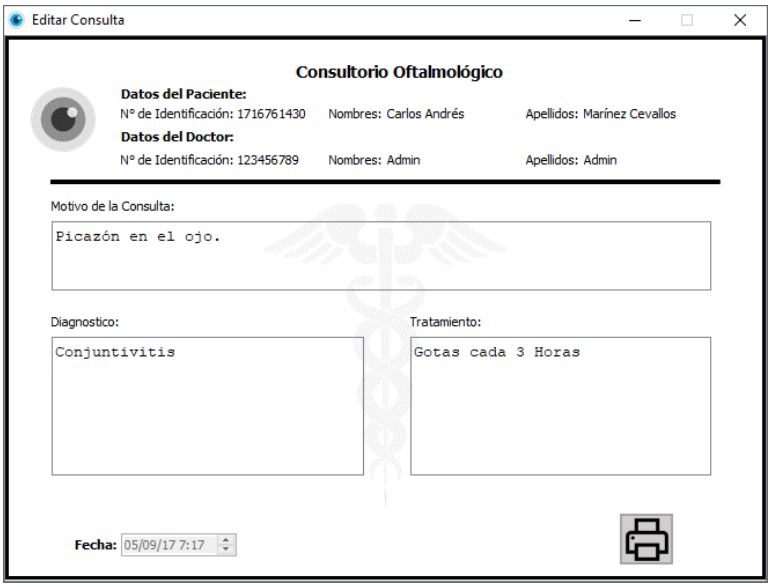

Figura 4.35. Ventana 'Editar Consulta'.

c) Para eliminar una consulta se selecciona una del listado y se presiona el botón eliminar. Se mostrará un dialogo de confirmación para eliminar la consulta.

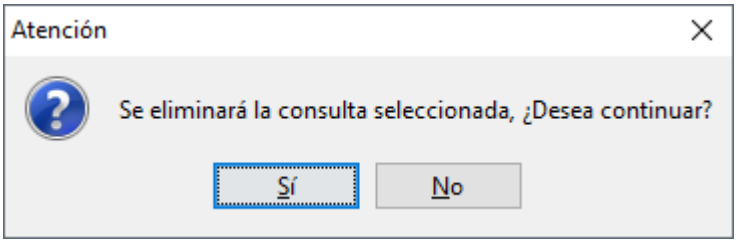

Figura 4.36. Dialogo de confirmación para eliminar una consulta.

d) Si solo se desea ver el contenido de una consulta y/o imprimirla se debe seleccionar una del listado y luego presionar el botón ‘Detalle’.

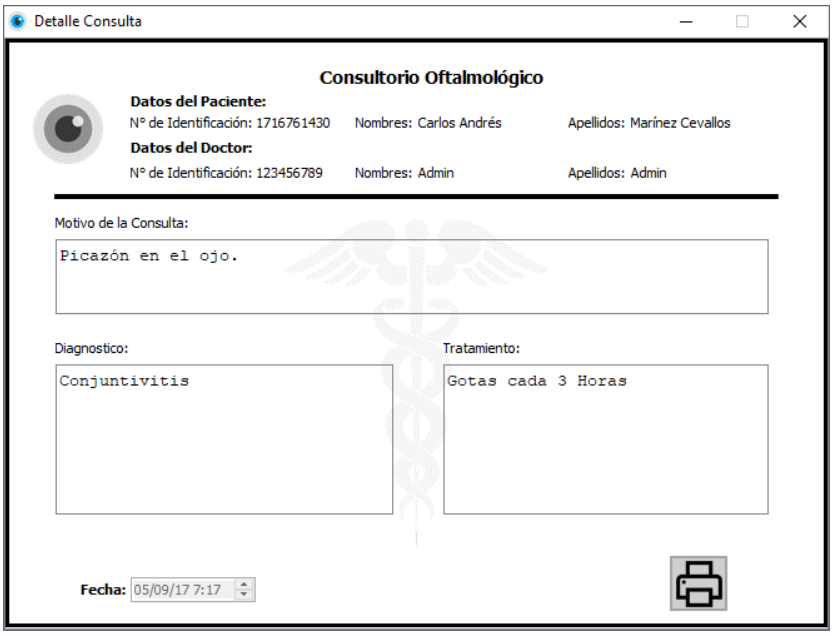

Figura 4.37. Ventana de detalle de una consulta.

4.3.2 Manual para asistentes.

A diferencia de los doctores, los asistentes únicamente pueden administrar personas, mas no fichas médicas. Las personas que sean ingresadas por un asistente serán siempre pacientes, es decir que un asistente no puede ingresar un doctor u otro asistente. Por último, los asistentes no podrán eliminar personas, ni tampoco modificar la información de doctores o asistentes ya existentes dentro del sistema.

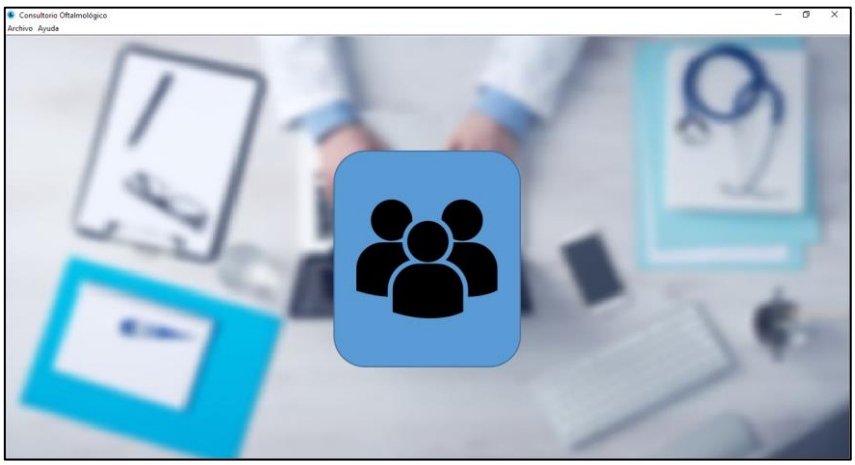

Figura 4.38. Pantalla de ‘Opciones’ de usuarios con rol de ‘Asistente’.

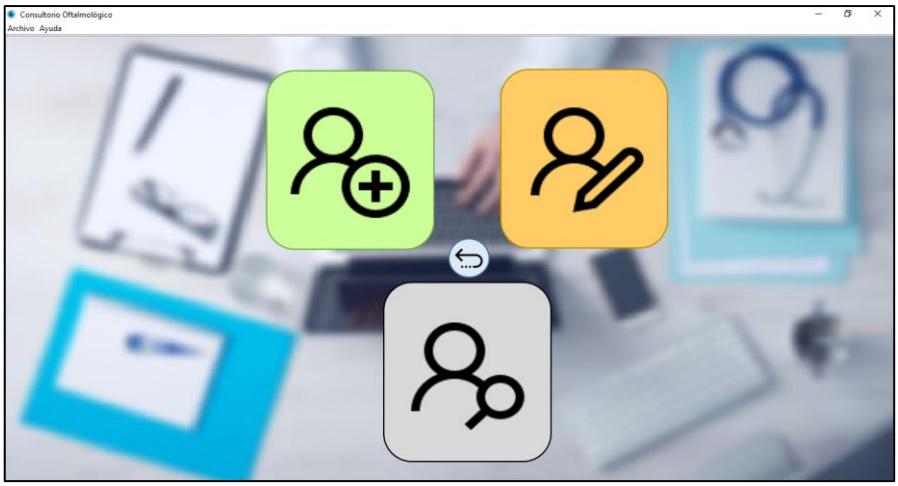

Figura 4.39. Pantalla ‘Administrar Personas’ de usuarios con rol de ‘Asistente’.

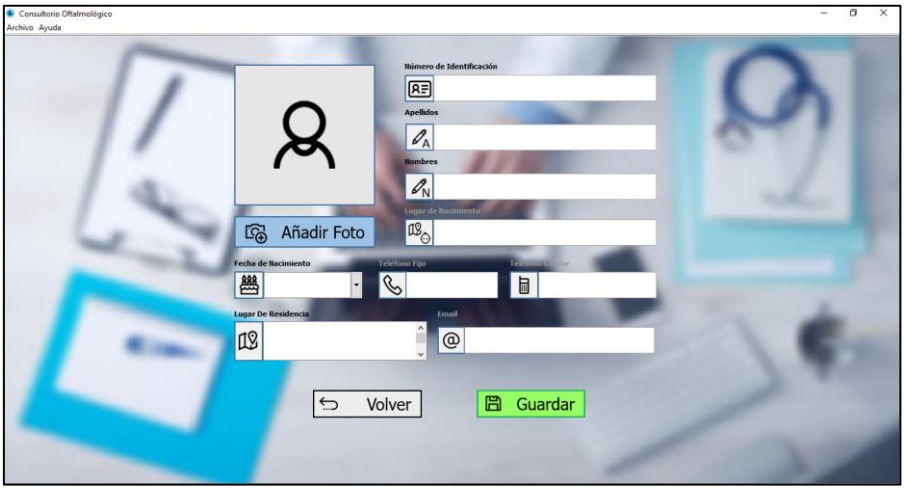

Figura 4.40. Pantalla ‘Ingresar Personas’ de usuarios con rol de ‘Asistente’.

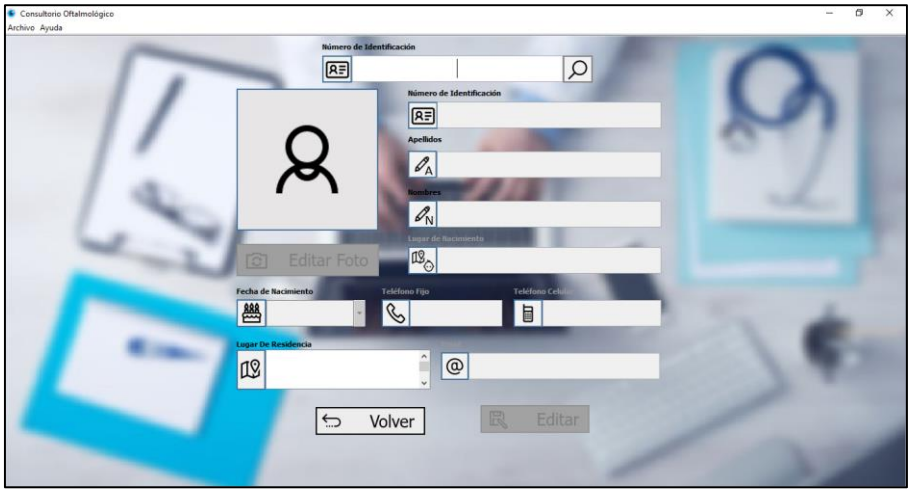

Figura 4.41. Pantalla ‘Modificar Personas’ de usuarios con rol de ‘Asistente’.

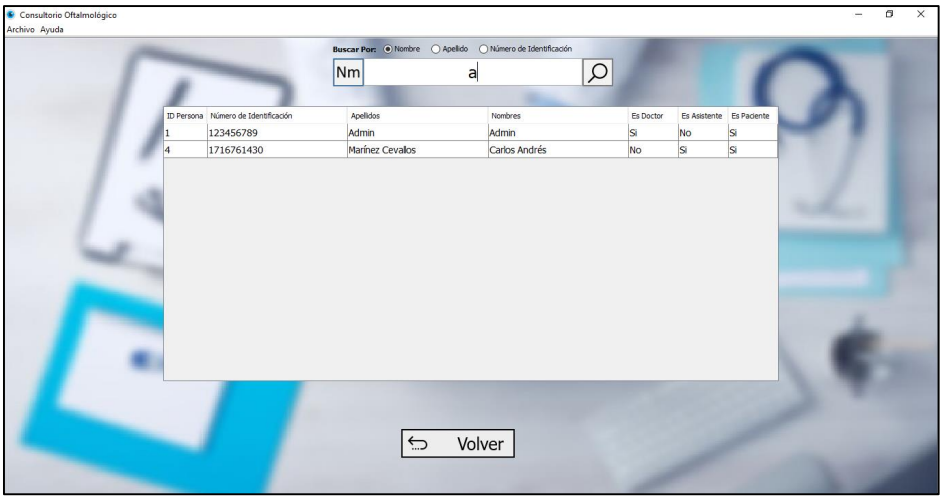

Figura 4.22. Pantalla ‘Consultar Personas’ de manera general.

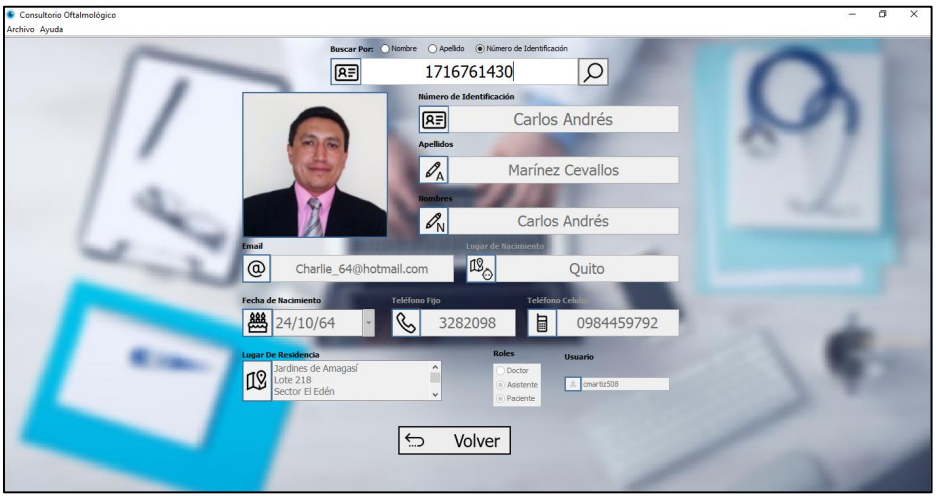

Figura 4.23. Pantalla ‘Consultar Personas’ de manera detallada.

4.3.3 Manual para doctores y asistentes.

Para cerrar sesión o salir del programa se debe dar click en ‘Archivo’. Ahí se encontrarán ambas opciones.

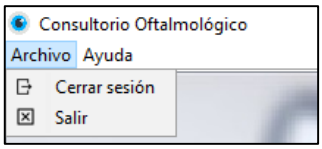

Figura 4.41. Opción de ‘Archivo’ en la barra de menú.

Se debe tener en cuenta que todas las acciones realizadas en el sistema serán registradas en un log de actividades.

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES

Este capítulo contiene las conclusiones y recomendaciones recogidas a lo largo del desarrollo de esta disertación.

5.1 CONCLUSIONES

- La oftalmología es un campo de la medicina, que al igual que otros campos, requiere llevar un historial o ficha médica de las personas para que estas sean diagnosticadas de la mejor forma.

- NetBeans es un IDE bastante completo, que permite el desarrollo de aplicaciones brindando herramientas y permitiendo la instalación e importación de complementos que facilitan la programación.

- La metodología RUP permite la elaboración de software de manera ordenada y bien documentada, lo que facilita el proceso de cambios en caso de requerirlos.

- Los diagramas UML proveen de elementos que ayudan a visualizar de manera clara los requerimientos del cliente y la forma en la que se va a diseñar y está diseñado el software.

- La metodología RUP provee un proceso iterativo que facilita la elaboración del software, dividiendo su desarrollo en fases que se van repitiendo hasta alcanzar un producto de calidad y que satisfaga las necesidades del cliente.

- La herramienta PowerDesigner permite realizar de manera fácil y rápida diagramas UML. También facilita la creación de la base de datos ya que utiliza el diagrama conceptual de la base de datos para generar el respectivo Script.

- La clase PrinterJob en Java se encarga de obtener el servicio de impresión del computador de manera fácil, permitiendo de esta forma realizar la impresión de cualquier contenedor en Java. En esta disertación fue utilizada para imprimir un jPanel.

- La utilización de 'Absolute Layout' en el diseño de un contenedor en Java permite organizar los elementos de cualquier forma, pero al cambiar el tamaño del contenedor, estos no se auto redimensionarán.

5.2 RECOMENDACIONES

- PostgreSQL es una base de datos completa que no tiene ningún costo, por lo que se recomienda su utilización en pequeñas y medianas empresas.

- Al realizar una aplicación de escritorio con Java SE y NetBeans se recomienda utilizar únicamente un jFrame y cargar los elementos de cada pantalla en un panel dentro de este jFrame. Realizar esto permitirá tener un solo ícono en la barra de tareas y será más amigable para el usuario.

- Se recomienda el uso de pgAdmin para la administración de las bases de datos de PostgreSQL, ya que provee de una interfaz gráfica que facilita la gestión de la base de datos.

- En cualquier programa que sea realizado con acceso a usuarios se aconseja crear en la base de datos un log de actividad. Esto permitirá llevar la evidencia de las acciones realizadas por los usuarios.

- Si se va a utilizar una gran cantidad de imágenes en un programa desarrollado con Java SE y NetBeans, se recomienda crear un paquete para almacenarlas dentro de este y así tener mejor organizado el código.

- Si se utiliza NetBeans para el desarrollo de la interfaz gráfica de una aplicación de escritorio, se recomienda utilizar jLabels como botones y cargarlos con una imagen, esto mejorará visualmente la interfaz ya que los botones que vienen incluidos en el IDE no son muy llamativos.

- Para transformar el ".jar" de una aplicación desarrollada en Java a ".exe" para que sea ejecutada en Windows se aconseja utilizar el programa "exe4j" ya que brinda la opción de configurar varias opciones de la ejecución de la aplicación, entre otras características.

BIBLIOGRAFÍA

- Bittner, K. (2008). *Use Case Modeling.* Boston, Massachussets, Estados Unidos: Addison-Wesley.
- Dantas, R. (2011). *NetBeans IDE 7 Cookbook.* Birmingham, Inglaterra: Packt Publishing Ltd.
- Favre, L. (2003). *UML And The Unified Process.* Hershey, Estados Unidos: IRM Press.
- Harrington, J. (2009). *Relational Database Design.* Burlington, Massachussets, Estados Unidos: Morgan Kaufmann/Elsevier.
- Heffelfinger, D. (2015). *Java EE 7 Development with NetBeans 8.* Birmingham, Inglaterra: Packt Publishing Ltd.
- Holt, J. (2007). *UML for Systems Engineering.* Londres, Inglaterra: Institution of Engineering and Technology.
- James, B., & Bron, A. (2012). *Lecture Notes (Ophthalmology)* (1 ed.). New York, Estados Unidos: John Wiley & Sons.
- Juba, S., Vannahme, A., & Volkov, A. (2015). *PostgreSQL for data architects.* Birmingham, Inglaterra: Packt Publishing.
- Kruchten, P. (2004). *The Rational Unified Process: An Introduction.* Boston, Estados Unidos: Addison-Wesley Professional. Recuperado el junio de 06 de 2016, de https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf
- Kruchten, P. (2009). *The rational unified process.* Munich, Alemania: Addison-Wesley.
- Olver, J., & Cassidy, L. (2005). *Ophthalmology at a glance* (1 ed.). Malden, Estados Unidos: Blackwell Science.
- Piñeiro, J. (2014). *Diseño de bases de datos relacionales.* Madrid, España: Paraninfo.
- Reilly, D., & Reilly, M. (2002). *Java network programming and distributed computing.* Boston, Estados Unidos: Addison-Wesley.

- Rossberg, J. (2014). *Beginning Application Lifecycle Management.* Berkeley, California, Estados Unidos: Apress.
- Schmuller, J. (2004). *UML.* Indianapolis, Indiana, Estados Unidos: Sams.
- Shaw, M. (2016). *Ophthalmic Nursing* (Quinta ed.). Boca Raton, Florida, Estados Unidos: CRC Press.
- Sommerville, I. (2005). *Ingeniería del software.* Madrid, España: Pearson Educación.
- Tsui, F., Karam, O., & Bernal, B. (2016). *Essentials of Software Engineering.* Burlington, Estados Unidos: Jones & Bartlett Learning.