PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

ESCUELA DE SISTEMAS Y COMPUTACIÓN

CONSTRUCCIÓN DE UN SISTEMA PARA LA GESTIÓN DE ESTUDIANTES DEL INSTITUTO ESPECIAL PARA NIÑOS "MARIANA DE JESÚS", USANDO LA METODOLOGÍA DE DESARROLLO DE SOFTWARE RUP

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERA DE SISTEMAS Y COMPUTACIÓN

LILIAN ESTEFANÍA RECALDE CARPIO

DIRECTOR:

INGENIERO FABIÁN DE LA CRUZ

QUITO, 2017

Dedicatoria

El presente trabajo de disertación está dedicado a Dios, por haberme regalado una vida llena de satisfacciones y brindarme fe para poder seguir adelante en todo momento.

A mi querida Madre por haber sido el impulso, fortaleza y apoyo más grande para que yo pudiera lograr mis sueños a pesar de las adversidades. Gracias por estar dispuesta a acompañarme en cada larga y agotadora noche de estudio, agotadoras noches en la que su compañía y la llegada de sus cafés eran para mí como agua en el desierto.

A mis familiares, a mi tía Nancita por siempre darme ese apoyo incondicional y de la cual aprendí muchas cosas, a mis primas que, con su amor y su gran amistad, han sido un gran apoyo y grandes amigas incondicionales en mi vida, a mis sobrinos, que con sus locuras y alegrías siempre logran siempre sacarme una sonrisa, a mi abuelita, quien a pesar que ya no está junto a mí, sé que siempre me cuida y me protege.

A mis maestros, Ing. Fabián De La Cruz por siempre estar pendiente, apoyándome y aconsejándome durante la carrera y en todo lo necesario para cumplir con éxito el presente trabajo de disertación. Ing. Eddy Sánchez por su apoyo incondicional y siempre esas ganas de ayudar. Ing. Edison Mora por haberme dado las pautas teóricas durante toda la carrera y su ayuda en la culminación del presente trabajo de disertación.

Y para finalizar, también agradezco a todos los que fueron mis compañeros de clase durante todos los niveles que corresponden a la carrera, ya que, gracias al compañerismo, amistad, apoyo y sin esperar nada a cambio compartieron su conocimiento, alegrías y tristezas.

A todas aquellas personas que durante todo este tiempo estuvieron a mi lado apoyándome y lograron que este sueño se haga realidad.

¡Gracias a todos!

Agradecimientos

Quisiera agradecer primeramente a Dios por haberme dado la capacidad y la oportunidad de estar culminando mis estudios de pregrado.

Agradezco a la Pontificia Universidad Católica del Ecuador por haberme acogido y brindado una educación de excelencia, formando mi vida en el ámbito profesional y personal. De igual forma agradezco a la Facultad de Ingeniería de la PUCE la oportunidad de haberme integrado en sus aulas y ayudar a consolidarme como una mejor persona para la sociedad a través de mi profesión.

A mi director de tesis, Ing. Fabián De La Cruz por su esfuerzo y dedicación, quien, con sus conocimientos, su experiencia, su paciencia y su motivación ha logrado en mí que pueda terminar mis estudios con éxito.

También me gustaría agradecer a todos aquellos que fueron mis profesores durante la carrera, los cuales me enseñaron muchas cosas tanto en lo teórico como en valores.

"Son muchas las personas que han formado parte de mi vida profesional a las que me encantaría agradecerles su amistad, consejos, apoyo, ánimo y compañía en los momentos más difíciles de mi vida. Algunas están aquí conmigo y otras en mis recuerdos y en mi corazón, sin importar en donde estén quiero darles las gracias por formar parte de mí, por todo lo que me han brindado y por todo su apoyo."

Para todos: Muchas Gracias.

Tabla de Contenidos

Lista de Tablas

Lista de Figuras

Introducción

El Instituto Especial 'Mariana de Jesús', fundado en abril de 1953, por el grupo de señoras 'La Fundación Amiga de los Ciegos', es una entidad que presta ayuda social a niños, adolescentes y adultos con problemas visuales, auditivos y otros (Machuca, 2014a).

Figura 1. Logo Fundación "Amiga de los Ciegos"
Elaborado por: (Machuca, 2014a).(González, 2012)

El Instituto funcionó en un local de la compañía de las Hijas de la Caridad hasta el año 1982, local que fue quedando muy pequeño para poder atender de manera adecuada, debido a la creciente demanda de alumnos. Gracias al apoyo de diferentes instituciones y a la labor de las Señoras de la Fundación, fue posible construir otro local que desde el año 1983 funciona en el sector de la Jipijapa, al norte de Quito. (Machuca, 2014a).

El Instituto dedica la mayor parte del tiempo a niños y jóvenes con deficiencias auditivas y visuales que llegan de todas las provincias del Ecuador. Además de contar con el servicio de Integración Educativa, es decir, brinda atención a estudiantes ciegos y de baja visión de escuelas y colegios de educación regular (Machuca, 2014a).

El principal objetivo del Instituto es brindar educación integral, ofreciendo a sus alumnos en forma adecuada la formación, habilitación y rehabilitación (Machuca, 2014b).

El Instituto cuenta con dos secciones:

- Deficientes Visuales
- Deficientes Auditivos

La educación brindada sigue una metodología especializada de acuerdo con el tipo de deficiencia en cada sección. (Machuca, 2014b).

Capítulo I Marco Teórico

En este capítulo, se revisarán los principales elementos teóricos que necesitaremos para poder realizar el proceso de desarrollo mediante la metodología RUP.

1.1. Objetivos

1.1.1. Objetivo General

Diseñar y desarrollar un sistema que gestione la información de los estudiantes, así también como reportes del Instituto Especial para Niños 'Mariana de Jesús'.

1.1.2. Objetivos Específicos

- Analizar los requerimientos de la institución mediante reuniones preliminares al inicio del proyecto,
- Definir la base de datos a utilizar, así como su diseño.
- Desarrollar el sistema informático de tipo Cliente – Servidor en el lenguaje de programación JAVA.
- Realizar las pruebas necesarias para el correcto funcionamiento del sistema.

1.2. Metodología de Desarrollo de Software

Una metodología de desarrollo de software está compuesta por una serie de pasos que deben seguir un orden lógico y ser realizados cuidadosamente, de forma que, a partir de nuevos requerimientos, su resultado sea un software por inaugurar (Sellarés, s. f.).

Figura 2. Ejemplo de Algunos Pasos de una Metodología de Desarrollo de Software

Fuente: Metodología de Software

Elaborado por: (González, 2012)

Elegir una metodología de forma correcta nos puede aportar garantía de calidad, así también como una manera de estimar y controlar costos. De manera que nos permita conocer el tiempo en que nos tardaremos en realizar el trabajo, y si es o no rentable (Sellarés, s. f.).

Al ser un proceso estructurado, la metodología organiza la forma en que el proyecto será realizado, controlando el avance del mismo, revisando que los resultados estén correctos para así proseguir, obteniendo una mejor eficiencia de recursos, es decir, "se invierte lo mínimo para obtener lo máximo a cambio" (Sellarés, s. f.).

1.2.1. Tipos de Metodologías de Desarrollo de Software

Se puede encontrar dos tipos principales de metodologías de desarrollo de software:

1.2.1.1. Metodologías Ligeras

Son metodologías prácticas, las cuales por lo general obvian, en su mayoría la documentación, son utilizadas especialmente en proyectos cuyos requisitos serán cambiados constantemente durante todo el proceso (Sellarés, s. f.).

1.2.1.2. Metodologías Pesadas

Son metodologías, en donde el proceso se encuentra mucho más controlado, que generan documentación previa a la instrumentación del proyecto, poniendo mayor énfasis en el análisis y el diseño del mismo. Dichas metodologías son más adecuadas para grandes proyectos o si su rendimiento y nivel de calidad son factores que pueden incidir en el éxito del proyecto (Sellarés, s. f.).

1.2.2. Ciclo de Vida del Software

Así también, cada metodología de desarrollo de software posee su ciclo de vida correspondiente. El ciclo de vida de un software se define como la serie de pasos a seguir por el proyecto, desde su inicio hasta su finalización y cierre, incluyendo al mismo los mantenimientos (Sellarés, s. f.).

1.2.2.1. Tipos

Se presentan diversos tipos de ciclo de vida que puede tomar una metodología de desarrollo de software:

- Cascada

  Su enfoque es secuencial. Cuando se presentan fallas de las fases anteriores, éstas deberán ser arregladas en la fase en la que se encuentre, y como regla general se debe proceder hacia adelante, teniendo en cuenta que no se puede regresar a ninguna fase anterior (Sellarés, s. f.).

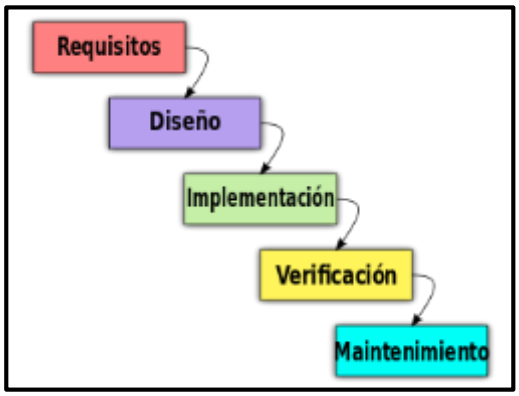

Figura 3. Ciclo de Vida de Software en Cascada

Fuente: Introducción a las Metodologías de Desarrollo de Software

Elaborado por: (Sellarés, s. f.)

- Prototipo

  Su enfoque es iterativo. Se basa en efectuar pequeños prototipos finales de la aplicación, de tal forma que sus funcionalidades se implementan por encima de la anterior versión, hasta poder alcanzar el producto definitivo y su entrega al cliente (Sellarés, s. f.).

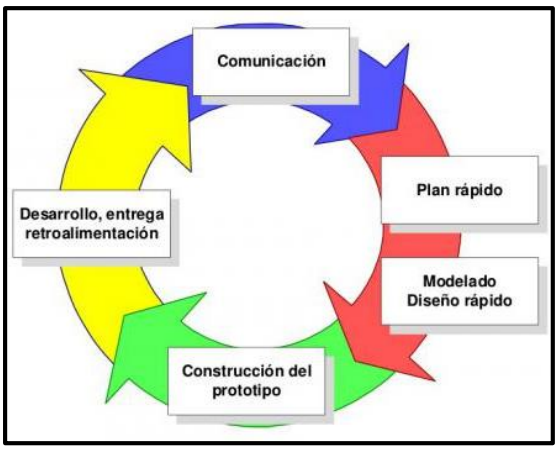

Figura 4. Ciclo de Vida de Software de Prototipo

Fuente: Introducción a las Metodologías de Desarrollo de Software

Elaborado por: (Sellarés, s. f.)

- Incremental

  Posee una combinación de los dos tipos de ciclos de vida mencionados anteriormente. Este ciclo de vida es similar al de prototipos, con la diferencia que se efectuarán mini-cascadas en cada iteración, de tal forma que atraviese por todas sus fases (Sellarés, s. f.).

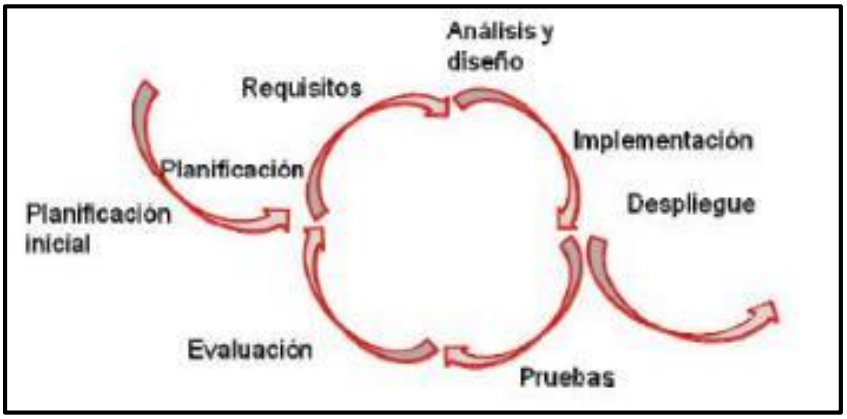

Figura 5. Ciclo de Vida de Software Incremental
Fuente: Introducción a las Metodologías de Desarrollo de Software
Elaborado por: (Sellarés, s. f.)

- Espiral

  Es otro enfoque combinado, pero la complejidad es mayor a los mencionados anteriormente. El proceso es similar al de un espiral, cada rotación simboliza una mini-cascada, y cada distancia radial el volumen del proyecto. Cuanto mayor volumen exista, más avanzado se encontrará el proyecto (Sellarés, s. f.).

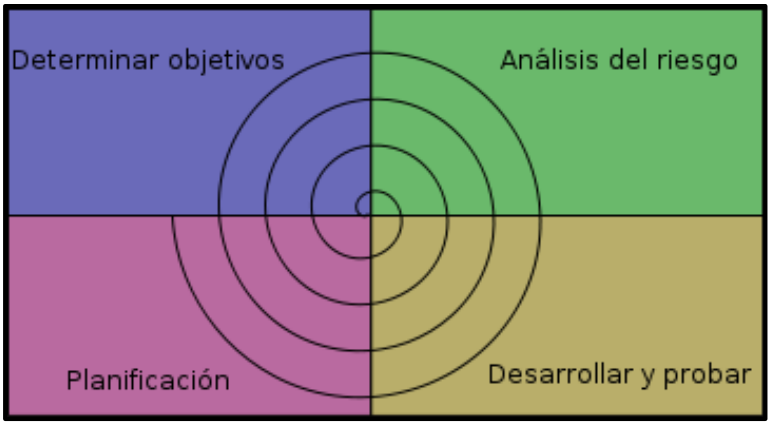

Figura 6. Ciclo de Vida de Software Espiral
Fuente: Introducción a las Metodologías de Desarrollo de Software
Elaborado por: (Sellarés, s. f.)

1.3. Metodologías de Desarrollo de Software Orientadas a Objetos

Una metodología de desarrollo de software orientada a objetos es un proceso para producir dicho software de manera organizada, con el uso de convenciones y técnicas predefinidas. Desde que se introdujo el concepto de que los objetos son entidades coherentes con identidad, estado y conducta; y que, además pueden tener herencia y polimorfismo, las metodologías orientadas a objetos incorporan dichos conceptos para poder definir reglas, normas, procedimientos, guías y notaciones, para de esta manera, alcanzar un producto de calidad que satisfaga las necesidades del cliente (Quintanilla, Peña, Montes, & Nuñez, 2015).

El Lenguaje Unificado 'UML' sirve para especificar, visualizar y documentar proyectos de desarrollo de software que se encuentren orientado a objetos. Dicho lenguaje no es un método de desarrollo, simplemente nos ayuda a tener una visión clara del diseño y a hacer más accesible para otros. UML está compuesto de varios elementos de esquematización, los cuales representan las distintas partes de un sistema software, y son utilizados para crear diagramas (Quintanilla et al., 2015).

1.3.1. Técnicas Orientadas a Objetos

1.3.1.1. Metodología OMT

Por sus siglas en inglés, Object Modeling Technique. El modelo es muy importante y se lo usa para lograr una abstracción, "en la cual el análisis está enfocado en el mundo real para un nivel de diseño" (Quintanilla et al., 2015).

Dicha metodología procede del trabajo de James Rumbaugh, y de sus colaboradores de LG, es una metodología completa pero demasiado complicada debido al grado de detalle que exige, por este motivo se recomienda el apoyo automatizado (Quintanilla et al., 2015).

OMT pone énfasis en el análisis y no en el desarrollo, mayor atención en los datos que en las funciones, para así proporcionar estabilidad al proceso de desarrollo. Utiliza una notación común para todas las fases, a través de tres modelos: estáticos, dinámicos y funcionales, que combinándolos proveerán de una descripción completa al sistema (Quintanilla et al., 2015).

La metodología OMT consta de tres fases:

- Análisis: El objetivo en esta fase será desarrollar un modelo de qué es lo que se va querer realizar en el sistema. El modelo se expresa en términos de objetos con sus respectivas relaciones, el flujo dinámico de control y las transformaciones funcionales.
- Diseño: "Es la estrategia de alto nivel para poder resolver el problema y cómo construir una solución". En esta fase se define la arquitectura del sistema, y es en la cual se toman las decisiones estratégicas.
- Implementación: En esta fase, el diseño de objetos se convierte finalmente en código (Quintanilla et al., 2015).

1.3.1.2. Metodología BOOCH

La metodología BOOCH consta de cuatro actividades:

- Estructura Lógica: Donde se incluyen los diagramas de clases y de objetos.
- Estructura Física: En la que se encuentran los diagramas de módulos y de procesos.
- Dinámica de Clases: Para poder conocer el diagrama de transición de estados.
- Dinámica de Instancias: Con el diagrama temporal del sistema.

1.3.1.3. Metodología RUP

Proceso Unificado de Desarrollo, por sus siglas es español, es el resultado de tres décadas de desarrollo. Esta metodología se presenta como un conjunto de actividades necesarias para transformar los requerimientos de un usuario en un sistema de software. Es un marco de trabajo genérico, el cual puede especializarse para una gran diversidad de dominios. RUP es la metodología estándar de la industria para la completa construcción del ciclo de ingeniería de software, tanto para sistemas tradicionales como para sistemas web (Quintanilla et al., 2015).

RUP unifica disciplinas en lo que se refiere a desarrollo de software, incluyendo así, el modelado de negocio, el manejo de requerimientos, los componentes de desarrollo, la ingeniería de datos, el manejo y la configuración de cambios, y finalmente, las pruebas, cubriendo todo el ciclo

de vida de los proyectos basados en la construcción de componentes y maximizando el uso del Lenguaje Unificado de Modeloado -UML- (Quintanilla et al., 2015).

En el Capítulo II El Proceso Unificado de Desarrollo de Software, se detalla a mayor profundidad esta metodología.

1.4. Base de Datos PostgreSQL

El sistema de base de datos relacional denominado PostgreSQL está basado en Open Source o en español Código Abierto, es decir, que el código fuente del sistema tiene que estar disponible para cualquier persona libre de cargos directos (Denzer, 2002).

Figura 7. Logo de Sistema de Base de Datos PostgreSQL
Fuente: PostgreSQL (https://www.postgresql.org/)

Las bases de datos relacionales son aquellas que permiten la manipulación de la información conforme a las reglas del álgebra relacional. La información es almacenada en tablas, las cuales pueden estar relacionadas con otras, y se usan identificadores únicos conocidos como llaves o keys. PostgreSQL usa el modelo cliente – servidor (Denzer, 2002).

1.4.1. Características

En la Tabla 1 se refleja algunas características del sistema de base de datos PostgreSQL, los cuales son una buena referencia para conocer aspectos de dicho sistema.

Característica	Valor
Versión	PostgreSQL 9.6.3
Licencia	BSD
Cumplimiento con estándar SQL	Alta
Velocidad	Media
Estabilidad	Alta
Integridad de datos	Si
Seguridad	Media
Soporte de Vistas	Si
Soporte Subconsultas	Si
Procedimientos almacenados	Si
Integridad referencial	Si
Interfaces de programación	ODBC, JDBC, C/C++, SQL embebido (en C), Tcl/Tk, Perl, Phyton, PHP
Transacciones	Si
Claves foráneas	Si
Backups en caliente	Si

Tabla 1. Características PostgreSQL

Fuente: PostgreSQL

Elaborado por: (Denzer, 2002)

1.4.2. Ventajas

Existen diferentes ventajas que pueden presentarse en el sistema de base de datos PostgreSQL, entre las cuales destacan:

- Sistema estable
- De alto rendimiento
- Gran flexibilidad
- El desarrollo o migración desde Access, Visual Basic, Foxpro, Visual Foxpro, C/C++, Visual C/C++, Delphi, entre otras, para que puedan ser usadas como servidores de Base de Datos

Por los puntos expuestos anteriormente, PostgreSQL puede convertirse en una gran alternativa al momento de elegir un sistema de bases de datos (Denzer, 2002).

1.5. Lenguaje de programación JAVA

Creado por Sun Microsystems, es un lenguaje que puede funcionar en distintos tipos de procesadores. («Lenguaje Java y Entorno de Desarrollo», s. f.).

Figura 8. Logo Lenguaje de Programación JAVA
Fuente: Java (https://java.com/es/)

1.5.1. Historia

El lenguaje de programación Java fue creado por Sun Microsystems Inc. en un proceso por etapas:

- Abril, 1991: Año en el que Sun crea un grupo de trabajo liderado por James Gosling, con el objetivo de crear un software para que funcione en dispositivos electrónicos inteligentes y televisión interactiva, dicho software se denominó el Proyecto Green, y el resultado fue el Lenguaje Oak (Mestras, 2004).
- 1994: Se crea WebRunner, después denominado HotJava, el cual era un navegador Web escrito en Java (Mestras, 2004).
- 23 de mayo de 1995: Java se distribuye gratuitamente, aparece SunWorld 95, el cual anunciaba la reléase Alpha de Java (Mestras, 2004).
- 1995: Sun lanzó sus primeras versiones de Java.

1.5.2. Características

Las características que ofrece el lenguaje de programación Java son:

- Simple

  "Java ofrece toda la funcionalidad de un lenguaje potente". Java añade características muy útiles como el garbage collector, o reciclador de memoria dinámica, es decir, no es necesario preocuparse de liberar

memoria, ya que éste se encarga de ello, liberando bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria. Además el intérprete de Java que existe al momento ocupa únicamente 215 Kb de RAM («Introducción a Java», s. f.).

- Orientado a Objetos

  Java trabaja con sus datos como objetos, y con las interfaces de los mismos. Soporta tres características propias del paradigma de la orientación a objetos: Encapsulación, Herencia y Polimorfismo.

- Distribuido

  Java está construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para poder acceder e interactuar con los protocolos http y ftp. Esto permite acceder a la información a través de la red con tanta facilidad como en los ficheros locales («Introducción a Java», s. f.).

- Robusto

  Java puede realizar verificaciones en busca de inconvenientes tanto en tiempo de compilación como en tiempo de ejecución. Esto ayuda a detectar errores rápidamente en el ciclo de desarrollo. Además Java exige a la declaración explicita de métodos, así reduciendo las posibilidades de tener errores («Introducción a Java», s. f.).

- Arquitectura Neutral

  "El compilador de Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará", es decir cualquier máquina que posea el sistema de ejecución (run - time) puede ejecutar dicho código, sin dar importancia el modo en que la máquina ha sido generada («Introducción a Java», s. f.).

- Seguro

  La seguridad en Java tiene dos aspectos:

  - Características como punteros o casting implícitos se eliminan para prevenir el acceso ilegal a la memoria.
  - Cuando se usa Java para la creación de un navegador, se combinan características de lenguaje con protecciones comunes aplicadas al propio navegador.

- Portable

  Java posee portabilidad básica, gracias a ser de arquitectura independiente. Además, "Java implementa estándares de portabilidad para facilitar el desarrollo". A más que las interfaces son construidas a través de un sistema abstracto de ventanas, las cuales pueden ser implementadas en entornos Unix, Pc o Mac («Introducción a Java», s. f.).

- Interpretado

  El intérprete de Java puede ejecutar directamente el código objeto. Aunque el compilador actual es bastante lento, por el momento no existen más compiladores específicos de Java para las distintas plataformas existentes («Introducción a Java», s. f.).

- Dinámico

  "Java no intenta conectar todos los módulos que comprenden una aplicación hasta su tiempo de ejecución" («Introducción a Java», s. f.).

1.6. Arquitectura en Tres Capas

Cuando se construye un software, su desarrollo debe ser realizado de manera ordenada, para asegurar un avance continuo, un producto final con calidad y poder realizar mejoras en un futuro, por lo que se utiliza diversas técnicas que dependen del tipo de software a desarrollar (Valle & Granados, s. f.).

Una de las técnicas más utilizadas se denomina 'Programación por Capas', la cual se basa en dividir el código fuente según su funcionalidad principal. Se debe seguir un conjunto de pasos en secuencia lógica, para realizar una correcta 'Programación por Capas'. Dichos pasos se deben definir según el tipo de proyecto a realizar para posteriormente ser examinados, asegurando que el modelo escogido cumpla con las normas necesarias para que el producto final sea agradable al usuario. Como último punto, se implementa el software de acuerdo al modelo seleccionado, facilitando su desarrollo al dividir la aplicación en módulos y capas fáciles de manejar (Valle & Granados, s. f.).

Las principales capas presentes en un modelo son:

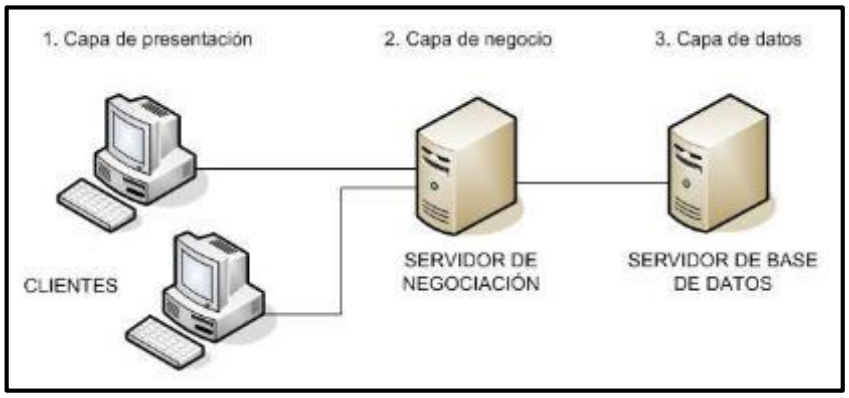

Figura 9. Arquitectura en Tres Capas
Fuente: («Arquitectura de tres niveles», s. f.)

1.6.1. Capa de Presentación

El objetivo principal de esta capa es facilitar al usuario la interacción con la aplicación. Para lograr este objetivo se manejan patrones que ya se encuentran definidos previamente dependiendo del tipo de aplicación y también para cada necesidad del usuario. Las interfaces que se presenten tienen como fin ser amigables y simples de usar, para facilitar al usuario final la manipulación del sistema y que el mismo pueda en un futuro brindar recomendaciones o mejoras (Valle & Granados, s. f.).

Las interfaces de una aplicación son realizadas de acuerdo a los requerimientos del usuario y la información que el mismo requiera, sin utilizar más campos de los necesarios, y de ésta manera satisfacer las necesidades del usuario, por lo que los requerimientos del usuarios deben ser claros y concisos (Valle & Granados, s. f.).

Hablando en términos técnicos, esta capa contiene los objetos encargados de la comunicación al usuario con el sistema mediante el intercambio de la información, capturando y desplegando los datos que son necesarios para poder realizar alguna tarea. Esta capa está conectada únicamente con la 'Capa de Negocio' (Valle & Granados, s. f.).

1.6.2. Capa de Negocio

En esta capa es donde se deben definir todas las reglas o funciones a ser ejecutadas para una correcta ejecución de la aplicación (Valle & Granados, s. f.).

Esta capa posee toda la lógica del programa, así también como la estructura de los datos y objetos que se encargan del manejo de la información existente, procesando las solicitudes que fueron realizadas en la capa anterior de presentación (Valle & Granados, s. f.).

La 'Capa de Negocio' se comunica con las otras dos capas para poder ejecutar las tareas solicitadas. Si la aplicación se debe comunicar con otros sistemas que actúan en conjunto, lo debe realizar mediante esta capa. Esta capa también se comunica con la capa de datos, debido a que debe obtener información existente o ingresar datos nuevos (Valle & Granados, s. f.).

En esta capa se utiliza un concepto llamado *encapsulamiento*, es decir, cuando recibe los datos ingresados mediante la 'Capa de Presentación', dichos datos son procesados, y a partir de eso crea objetos según lo que necesite realizar con dichos datos (Valle & Granados, s. f.).

Cuando los datos son encapsulados, se obtiene diferentes beneficios. Tales como:

- Mantener la consistencia de los datos
- Obtener información precisa de las bases de datos
- Ingresar únicamente la información necesaria, evitando datos repetidos

1.6.3. Capa de Datos

Es la capa encargada de realizar las transacciones con las bases de datos y con otros sistemas para la obtención o el ingreso de información al sistema (Valle & Granados, s. f.).

El manejo de los datos debe realizarse de una manera tal que exista consistencia en los mismos, tanto ingresando así también como extrayendo de la base de datos (Valle & Granados, s. f.).

En esta capa es donde se deben definir las consultas a realizar en las bases de datos, para que la información consultada sea enviada directamente a la 'Capa de

Negocio', y así pueda ser procesada e ingresada en los objetos que se necesite, dicha acción es denominada *encapsulamiento.*

Todos los elementos mencionados en este capítulo serán utilizados para poder realizar la presente disertación, el uso del lenguaje de programación orientado a objetos conocido como Java, servirá de mucha ayuda debido a las características que presenta (*véase la sección* 1.5.2), y finalmente se escogió como motor de base de datos a PostgreSQL debido a su flexibilidad, su modo de uso simple y además su entorno de desarrollo libre.

Capítulo II El Proceso Unificado de Desarrollo de Software

Es un proceso de desarrollo propuesto por 'Rational Software Corporation', sus creadores: Ivar Jacobson, Grady Booch y James Rumbaugh. El proceso es el resultado del esfuerzo y grandes estudios en las tres últimas décadas en desarrollo de software junto con la experiencia de sus creadores (Hormiga, 2007).

El Proceso Unificado es un proceso de desarrollo de software toma el conjunto de actividades necesarias con el fin de que a los requisitos del usuario transformarlos en un sistema software. (*véase* la Figura 10). RUP es un marco global y puede adaptarse a una gran variedad de tipos de sistemas. RUP está basado en componentes, donde el software se encuentra formado por componentes software interconectados a través de interfaces (Torossi, s. f.).

Figura 10. Un Proceso de Desarrollo de Software
Fuente: Proceso Unificado para Desarrollo de Software (RUP)
Elaborado por: (Torossi, s. f.)

2.1. Historia del Proceso Unificado

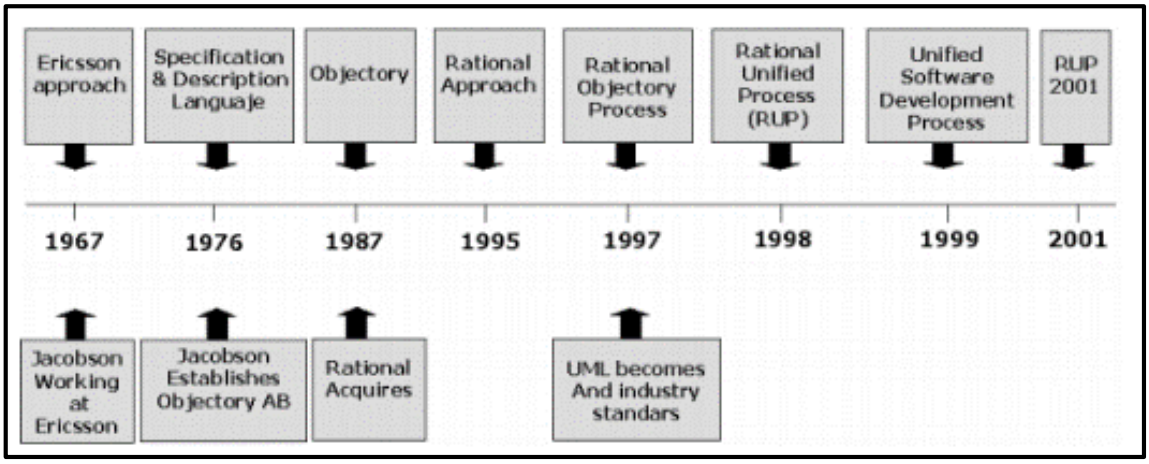

Figura 11. Historia RUP («Rational Unified Process (RUP)», s. f.).
Fuente: («Rational Unified Process (RUP)», s. f.)

"El Proceso Unificado está equilibrado por ser el producto final de tres décadas de desarrollo y uso práctico" (Jacobson, Booch, & Rumbaugh, 2000).

Como se refleja en la Figura 11., la historia de RUP es extensa, en los siguientes puntos se expondrá un pequeño resumen para poder entender mejor:

- 1967: Aparece la 'Metodología Ericsson' elaborada por Ivar Jacobson, la cual se aproximaba a lo que se quería denominar el desarrollo basado en componentes, el cual introdujo el concepto de 'Caso de Uso'.
- Entre 1987 a 1995: Ivar Jacobson funda la compañía Objectory AB ofertando el 'Proceso de Desarrollo Objetcory'.
- 1995: La compañía Rational Software Corporation adquiere Objectory AB.
- Entre 1995 a 1997: Basándose en 'Objectory 3.8' y del enfoque 'Rational', se desarrolla el ROP, o por sus siglas, en español el 'Proceso Objectory de Rational', y se adopta por el uso del Lenguaje de Modelado a UML.
- 1997: Rational Sofware, Booch, Jacobson y Rumbaugh incorporaron varios elementos que ayudarían a expandir el proceso, entre los cuales se puede destacar el flujo de trabajo conocido como 'Modelado de Negocio'
- Junio 1998: Se lanza el proceso de desarrollo RUP.

2.2. Principios clave

RUP está basado en seis principios clave, que se detallan a continuación:

2.2.1. Adaptar el proceso

Como su nombre lo dice, adaptar el proceso a las necesidades del cliente, para poder interactuar con el mismo. Tomando en cuenta que las características, el tamaño y el alcance del proyecto incidirán en el diseño del mismo («Metodología de RUP», s. f.).

2.2.2. Equilibrar prioridades

Los requerimientos que son presentados por el usuario pueden no ser entendibles o ser contradictorios. Por ello, se debe encontrar un punto clave de equilibro para satisfacer las necesidades del usuario. Equilibrando prioridades se podrá, en un futuro, corregir especificaciones para satisfacer al usuario final («Metodología de RUP», s. f.).

2.2.3. Demostrar valor iterativamente

Los proyectos deben entregarse en etapas iterativas. En cada iteración que se presente al usuario, se deberá recoger la opinión del mismo, para poder mantener estabilidad, calidad en el producto y evitar riesgos. La aprobación del usuario es punto clave para poder seguir avanzando en el proyecto con una dirección enfocada a satisfacer las necesidades del usuario («Metodología de RUP», s. f.).

2.2.4. Colaboración entre equipos

Para desarrollar un software, los miembros del equipo de desarrollo deben tener una buena comunicación para poder coordinar requisitos, dividir el trabajo equitativamente, desarrollar un producto de calidad, evaluar en cada iteración, realizar planes de los resultados obtenidos, etc. («Metodología de RUP», s. f.).

2.2.5. Elevar el nivel de abstracción

Para satisfacer de mejor manera las necesidades del usuario, se debe seguir una serie de pasos lógicos, evitando ir directamente de los requisitos a la codificación. Teniendo un alto nivel de abstracción se puede introducir patrones de software, marcos de referencia y representar visualmente a la arquitectura con el uso del lenguaje UML («Metodología de RUP», s. f.).

2.2.6. Enfocarse en la calidad

Se trata de mantener la calidad durante todo el proceso y no solo al final de cada iteración («Metodología de RUP», s. f.).

2.3. Características Esenciales de RUP

Las características principales de RUP son:

2.3.1. Dirigido por Casos de Uso

"Un sistema software ve la luz para dar servicios a sus usuarios. Por tanto, para construir un sistema software con éxito debemos conocer lo que sus futuros usuarios necesitan y desean" (Jacobson et al., 2000).

Los casos de uso sirven para capturar de mejor manera los requisitos, satisfaciendo las necesidades del usuario final eficientemente («Rational Unified Process (RUP)», s. f.).

Cuando hablamos de la metodología de desarrollo RUP, los casos de uso, además de especificar los requerimientos por parte del usuario, también brindan una guía de diseño, implementación y pruebas («Rational Unified Process (RUP)», s. f.).

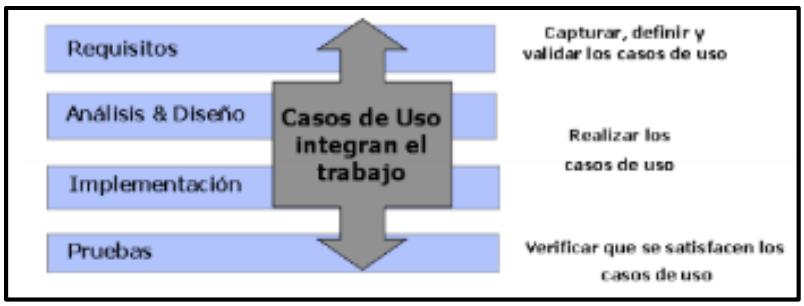

Figura 12. Los Casos de Uso Integran el Trabajo
Fuente: («Rational Unified Process (RUP)», s. f.)

2.3.2. Centrado en la arquitectura

"El concepto de arquitectura software incluye los aspectos estáticos y dinámicos más significativos del sistema" (Jacobson et al., 2000).

Si tanto los desarrolladores como los usuarios se centran en la arquitectura, se obtendrá una visión común y una perspectiva clara de la aplicación completa, controlando de mejor manera el desarrollo del producto.

La arquitectura en RUP está representada por medio de lo que se denominan vistas arquitectónicas. Esta idea la plasmó Philippe Kructen en un documento titulado "The 4+1 view model of architecture" (Hernández, 2006).

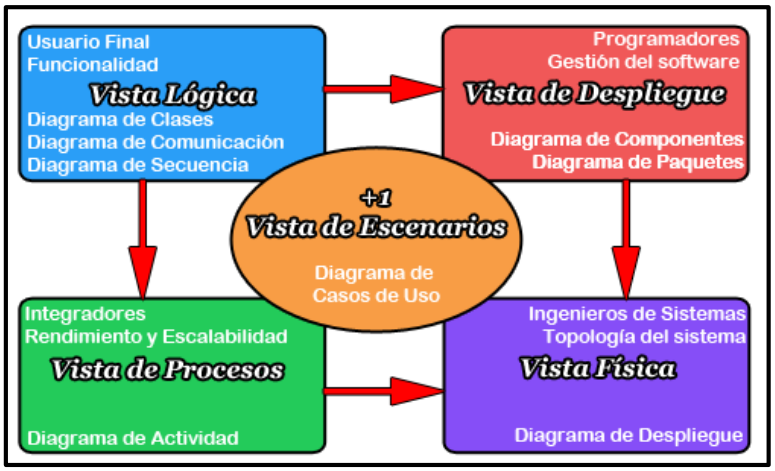

Figura 13. Vistas Arquitectónicas
Fuente: Modelo "4+1" vistas de Kruchten
Elaborado por: (Moya, 2012)

Como se indica en la Figura 13, dichas vistas arquitectónicas se representan por medio de conjuntos de diagramas, los cuales se representan en el Lenguaje Unificado de Modelado 'UML' (Hernández, 2006).

A continuación, se presenta una explicación de cada vista arquitectónica:

- Vista Lógica: Se representa la funcionalidad que el sistema proporciona al usuario final.
- Vista de Despliegue: Esta vista se ocupa de la gestión del software y se presenta el sistema desde la perspectiva que el desarrollador tiene.
- Vista de Procesos: Como su nombre lo dice, se puede visualizar los procesos existentes en el sistema y su forma de comunicación.
- Vista Física: Se observa los componentes físicos del sistema, así como conexiones físicas entre los componentes.
- '+1' Vista de Escenarios: Representada por los casos de uso, su objetivo principal es acoplar las anteriores cuatro vistas mencionadas.

2.3.3. Iterativo e Incremental

El dividir el proyecto en partes más pequeñas es muy práctico. Cada parte pequeña es una iteración que deben ser controladas para dar paso en el flujo de trabajo (Jacobson et al., 2000).

Teniendo un proceso iterativo e incremental, permite que exista un equilibro entre casos de uso y arquitectura, durante cada parte pequeña del proyecto («Rational Unified Process (RUP)», s. f.).

2.4. La Vida del Proceso Unificado

El Proceso Unificado es repetitivo durante una serie de ciclos. Cada ciclo concluye con una versión del producto para el usuario final (Jacobson et al., 2000).

Cada ciclo consta de cuatro fases: Inicio, Elaboración, Construcción y Transición (*véase la* Figura 14), y cada fase se subdivide en iteraciones como se ha dicho en el punto 2.3.3 (Jacobson et al., 2000)

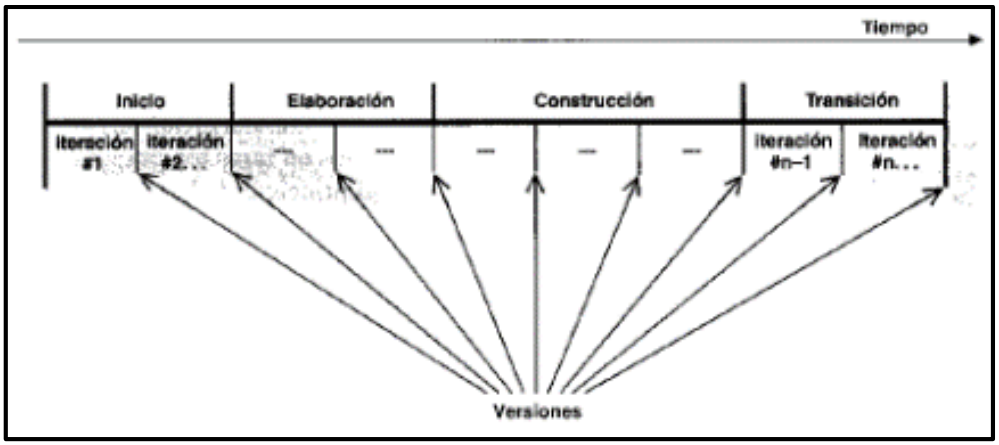

Figura 14. Un Ciclo con sus Fases e Iteraciones
Fuente: El Proceso Unificado de Desarrollo de Software
Elaborado por: (Jacobson et al., 2000).

2.4.1. Estructura del proceso

La estructura del proceso puede ser descrita en dos dimensiones o ejes:

- Eje Horizontal: En este eje se representa el tiempo, se lo considera como el eje que contiene los aspectos dinámicos del proceso y está expresado en ciclos, fases, iteraciones e hitos.
- Eje Vertical: Este eje representa los aspectos estáticos del proceso y describe las actividades, el flujo de trabajo y los roles.

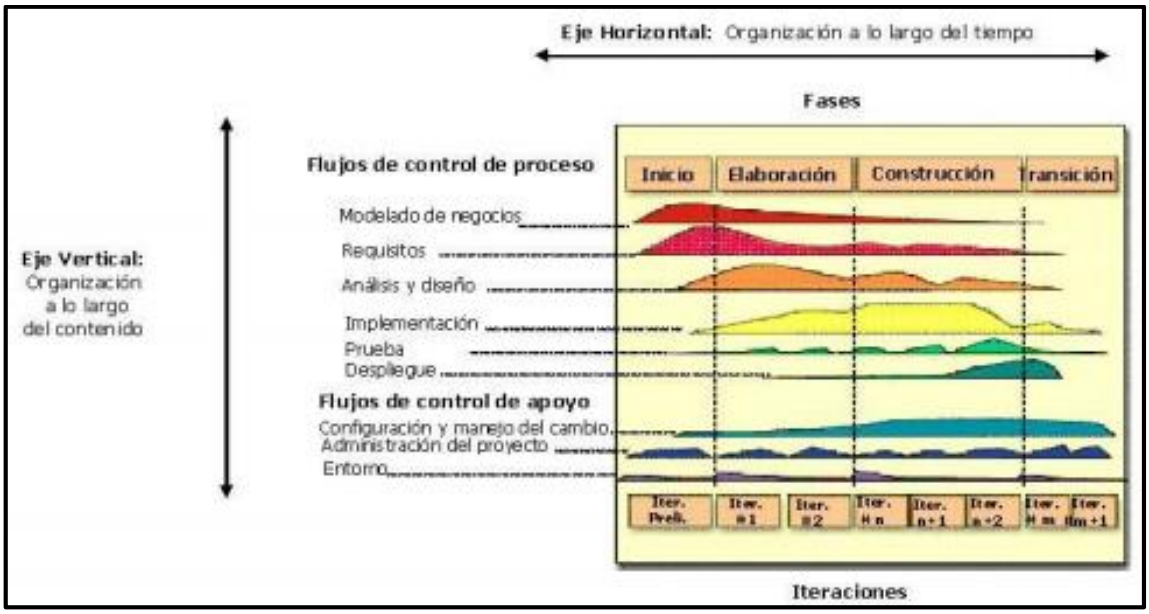

Figura 15. Estructura de RUP
Fuente: («Rational Unified Process (RUP)», s. f.)

2.4.2. Fases dentro de un ciclo

Como se dijo anteriormente, cada ciclo consta de cuatro fases, dichas fases se las detalla a continuación:

2.4.2.1. Fase de Inicio

El objetivo principal de la 'Fase de Inicio' es establecer objetivos principales y secundarios. En esta fase se define el modelo del negocio y el alcance del proyecto. Para lograr esto, se deben tener en cuenta todos los aspectos externos que interactuaran con el sistema (Actores). A los requerimientos del usuario se los debe expresar en términos funcionales (Casos de Uso), para que puedan ser diseñados. Además se desarrolla un plan de negocio para determinar los recursos que deben ser asignados al proyecto («Rational Unified Process (RUP)», s. f.).

2.4.2.2. Fase de Elaboración

El objetivo principal de la 'Fase de Elaboración' es examinar por completo el dominio del problema, definir la base arquitectónica, desarrollar un plan para todo el proyecto y eliminar los posibles riesgos («Rational Unified Process (RUP)», s. f.).

Esta es una fase considerada crítica del proyecto ya que aquí se realiza la mayor parte de la ingeniería, se asegura que la arquitectura, requerimientos y planificación sean estables y los hayan sido estudiados. De esta manera se puede determinar los costos y tiempos para completar el desarrollo («Rational Unified Process (RUP)», s. f.).

2.4.2.3. Fase de Construcción

El objetivo principal de esta fase es conseguir la operatividad del producto (Hormiga, 2007).

La capacidad operacional del producto se lo alcanza de forma incremental por medio de sucesivas iteraciones. A lo largo de esta fase todos los componentes, características y requisitos deben ser desarrollados, integrados y probados completamente al producto («Rational Unified Process (RUP)», s. f.).

2.4.2.4. Fase de Transición

El propósito de esta fase es poner en entrega el producto operando a los usuarios finales (Hormiga, 2007).

Para poder realizar la entrega del producto se requiere desarrollar nuevas versiones actualizadas, completar la documentación, preparar al usuario con respecto al manejo del producto, y generalmente tareas relacionadas con la configuración, instalación y facilidad de uso del producto («Rational Unified Process (RUP)», s. f.).

En la siguiente imagen se puede observar un resumen de las fases presentadas anteriormente con sus respectivos hitos:

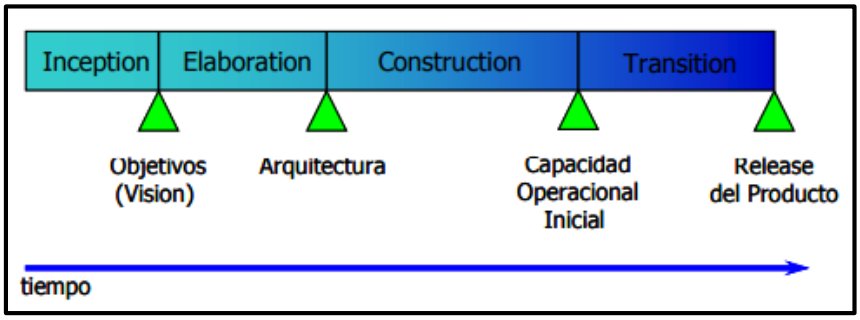

Figura 16. Fases e Hitos en RUP
Fuente: («Rational Unified Process (RUP)», s. f.)

2.4.3. Proceso Iterativo e Incremental

El proceso iterativo e incremental es donde se divide el trabajo en partes más pequeñas o mini proyectos. Así pues, cada mini proyecto se puede observar como una iteración («Rational Unified Process (RUP)», s. f.).

Como se puede observar en la Figura 17, es un claro ejemplo de cómo una iteración puede realizarse por medio de una cascada. En dicho modelo se atraviesa por las etapas fundamentales: Requisitos, Análisis, Diseño, Implementación y Pruebas («Rational Unified Process (RUP)», s. f.).

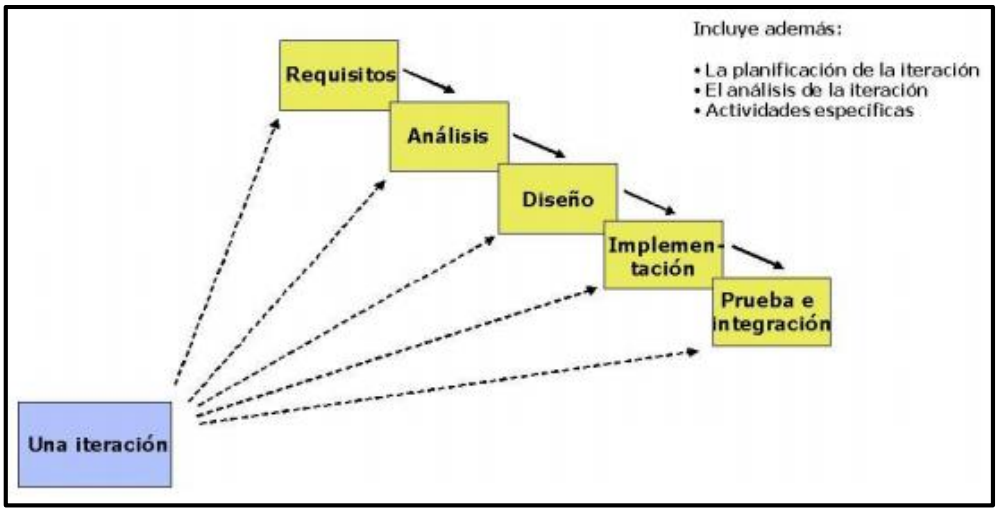

Figura 17. Una Iteración en RUP
Fuente: («Rational Unified Process (RUP)», s. f.)

2.5. UML

Lenguaje Unificado de Construcción de Modelos, o por sus siglas en inglés 'Unified Modeling Languaje', es una notación con la que se construyen sistemas mediante los conceptos orientados a objetos (Larman, 1999).

2.5.1. Diagramas de UML

El UML está compuesto por varios elementos gráficos que combinados forman diagramas. Estos diagramas tienen como función mostrar varias perspectivas de un sistema, lo que indica que estos diagramas muestran modelos que son "una representación simplificada de la realidad.", el modelo UML describe lo que puede hacer el sistema, más no como implementarlo («Diagramas del UML», s. f.).

A continuación, se detallan algunos de los diagramas del UML:

2.5.1.1. Diagrama de Clases

Representan una estructura estática de un sistema. Las cosas existentes y que nos rodean se agrupan en categorías, a las que se conoce como una clase, y al grupo de cosas se los denomina como atributos o propiedades, que poseen acciones similares («Diagramas del UML», s. f.).

Las clases se representan mediante el símbolo del rectángulo que se divide en tres filas:

- En la parte superior contiene el nombre de la clase
- La parte central los atributos
- La inferior contiene las acciones u operaciones.

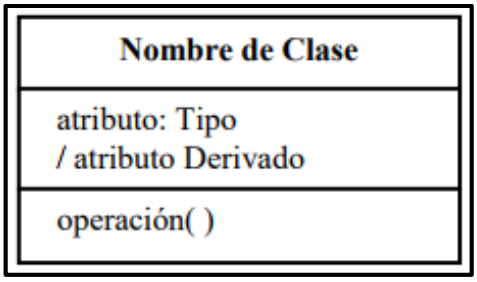

Figura 18. Clase Abstracta
Fuente: («Diagramas del UML», s. f.)

Un diagrama de clases está conformado por varios rectángulos que se encontrarán interconectados por líneas, las mismas que representan las asociaciones, es decir, las maneras en que las clases se relacionan entre sí («Diagramas del UML», s. f.).

- Asociaciones

  Las asociaciones representan a las relaciones estáticas entre clases. Su nomenclatura se puede observar en la Figura 19 («Diagramas del UML», s. f.).

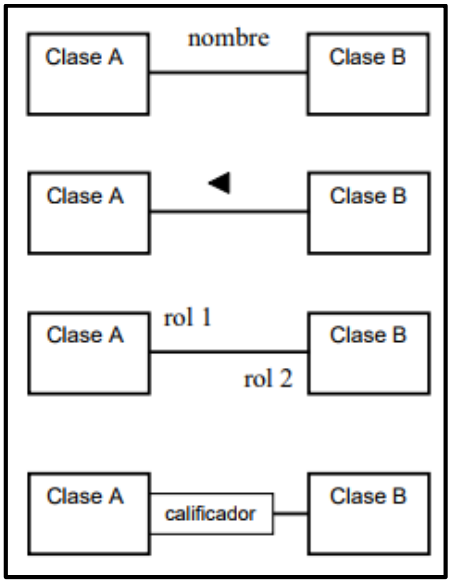

Figura 19. Asociaciones entre Clases
Fuente: («Diagramas del UML», s. f.)

- Multiplicidad

  La multiplicidad indica el número de instancias que posee una clase vinculadas a una de las instancias de otra clase («Diagramas del UML», s. f.).

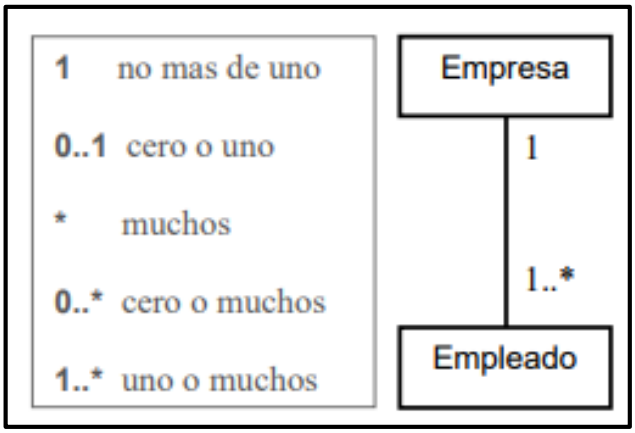

Figura 20. Multiplicidad entre Clases
Fuente: («Diagramas del UML», s. f.)

En la Figura 20 se puede observar la notación utilizada por la multiplicidad, así también como un ejemplo de la misma.

- Composición y Agregación

  La composición es un tipo especial de agregación, como se puede observar en la **¡Error! No se encuentra el origen de la referencia.** la clase 'Todo' posee una fuerte posesión a la clase 'Parte'. Mientras que la agregación es una relación en la que la clase 'Todo' juega un rol más importante que la clase 'Parte', pero ambas clases no dependen entre sí (véase la **¡Error! No se encuentra el origen de la referencia.**)

  ADDIN ZOTERO_ITEM CSL_CITATION {"citationID":"aoroc06gbf","properties":{"formattedCitation":"{\\rtf (\\uc0\\u171{}Diagramas del UML\\uc0\\u187{}, s.\\uc0\\u160{}f.)}","plainCitation":"(«Diagramas del UML», s. f.)"},"citationItems":[{"id":165,"uris":["http://zotero.org/users/local/iYTjpWZo/items/IZVUHUB5"],"uri":["http://zotero.org/users/local/iYTjpWZo/items/IZVUHUB5"],"itemData":{"id":165,"type":"webpage","title":"Diagramas del UML","URL":"http://festivaldealmagro.com/en/uploads/convocatorias/diagramas_del_uml.pdf","accessed":{"date-

parts":[["2017",7,17]]}}}],"schema":"https://github.com/citation-style-language/schema/raw/master/csl-citation.json"} («Diagramas del UML», s. f.).

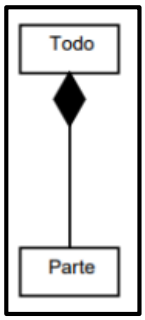

Figura 21. Composición entre Clases
Fuente: («Diagramas del UML», s. f.)

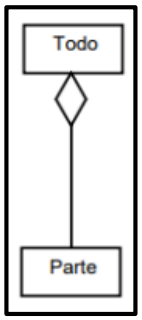

Figura 22. Agregación entre Clases
Fuente: («Diagramas del UML», s. f.)

- Generalización

  Se puede decir que la generalización es un segundo nombre para la herencia. Representa la relación existente entre dos clases. Como se puede observar en la Figura 23, la clase 'Específica' es una versión especializada de la clase 'General', es decir la primera clase heredará los atributos y acciones de la segunda clase («Diagramas del UML», s. f.).

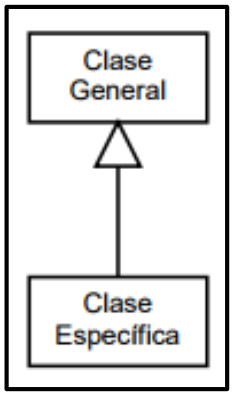

Figura 23. Generalización entre Clases
Fuente: («Diagramas del UML», s. f.)

2.5.1.2. Diagrama de Casos de Uso

Un caso de uso se define como una descripción de las acciones de un sistema, desde el punto de vista del usuario. Modelan la funcionalidad del sistema con

el uso de actores y casos de uso, en donde los casos de uso son funciones que proveen el sistema para sus usuarios («Diagramas del UML», s. f.).

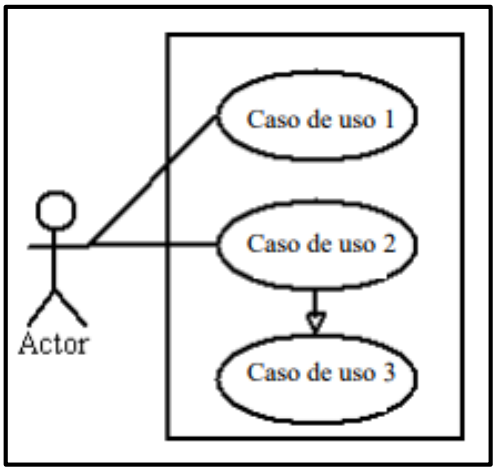

Figura 24. Ejemplo de Diagrama de Casos de Uso
Fuente: («Diagramas del UML», s. f.)

- Casos de Uso

  Se representan con óvalos. La etiqueta que se ubica dentro del óvalo indica la función del sistema («Diagramas del UML», s. f.).

Figura 25. Representación de un Caso de Uso
Fuente: («Diagramas del UML», s. f.)

- Actores

  Los actores representan a los usuarios del sistema («Diagramas del UML», s. f.).

Figura 26. Representación de un Actor
Fuente: («Diagramas del UML», s. f.)

- Relaciones

Existen relaciones entre un actor y un caso de uso que son representados mediante una línea simple. Indican que un caso de uso depende de otro para poder cumplir con una tarea específica («Diagramas del UML», s. f.).

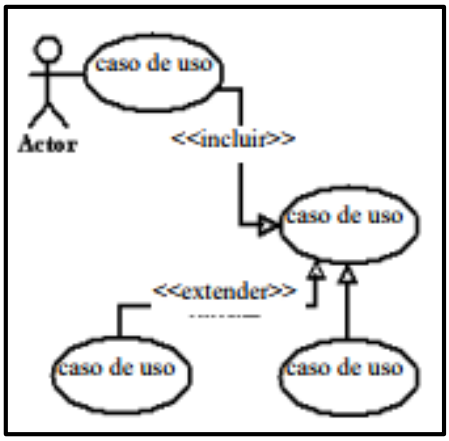

Figura 27. Ejemplo de Relaciones
Fuente: («Diagramas del UML», s. f.)

2.5.1.3. Diagrama de Secuencias

Los diagramas de secuencia representan cuando los objetos interactúan entre sí durante un tiempo en un sistema funcional («Diagramas del UML», s. f.).

- Rol de la Clase

  Describe la manera en que un objeto va a comportarse en el contexto («Diagramas del UML», s. f.).

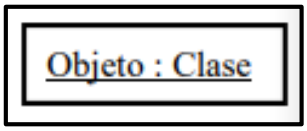

Figura 28. Representación de un Rol de Clase
Fuente: («Diagramas del UML», s. f.)

- Activación

  En un diagrama de secuencias existen los cuadros de activación, los cuales representan el tiempo que necesita un objeto para completar una tarea («Diagramas del UML», s. f.).

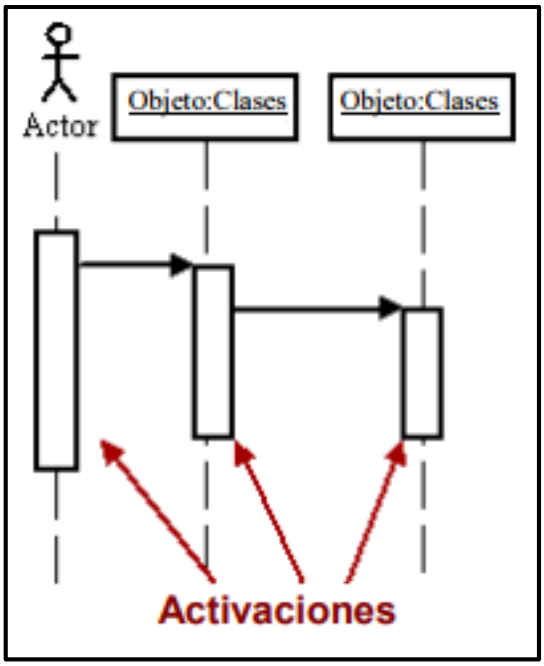

Figura 29. Representación de Activaciones

Fuente: («Diagramas del UML», s. f.)

- Mensajes

  Son flechas que representan las forma de comunicarse entre objetos. («Diagramas del UML», s. f.).

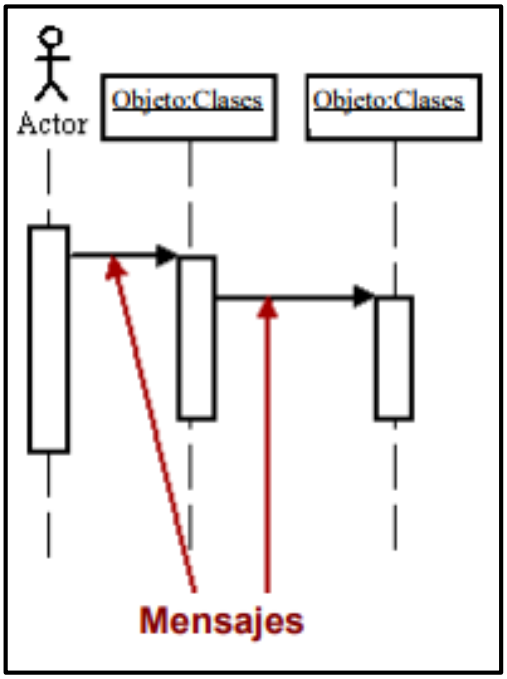

Figura 30. Representación de Mensajes

Fuente: («Diagramas del UML», s. f.)

En la siguiente figura se puede observar la notación de los mensajes y cada uno de su significado:

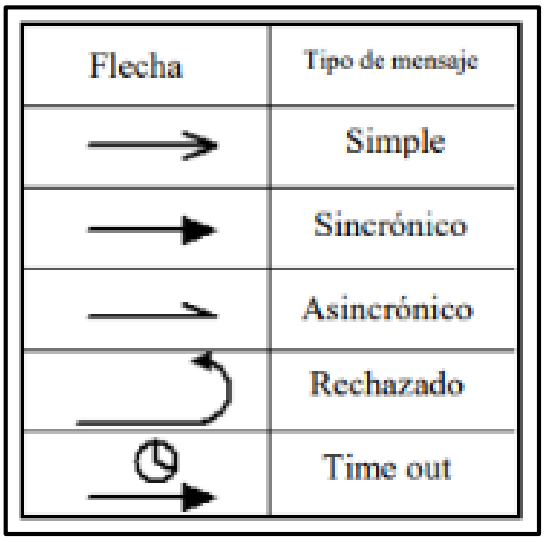

Figura 31. Notación de Mensajes y su Significado

Fuente: («Diagramas del UML», s. f.)

- Líneas de Vida

  Son aquellas que se encuentran representadas verticalmente y en línea de puntos, éstas indican la presencia del objeto durante su tiempo («Diagramas del UML», s. f.).

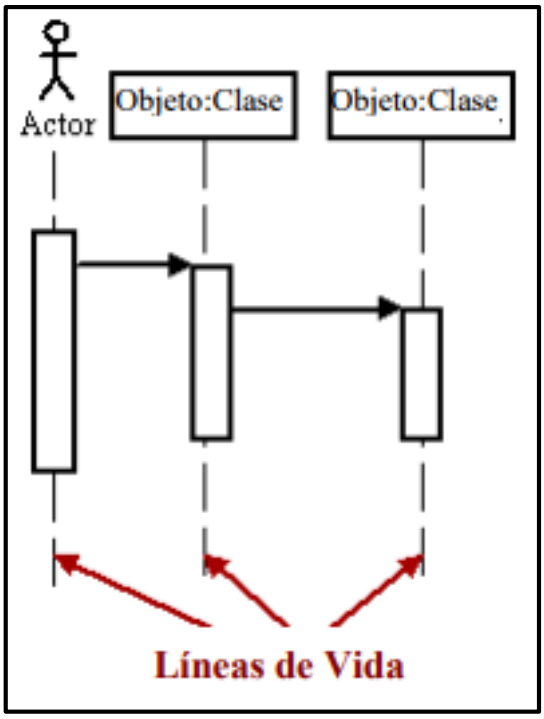

Figura 32. Representación de Líneas de Vida

Fuente: («Diagramas del UML», s. f.)

2.5.1.4. Diagrama de Actividades

Estos diagramas ilustran el comportamiento dinámico que posee un sistema por medio del modelado del flujo que ocurre entre actividades. Una actividad representa una acción del sistema («Diagramas del UML», s. f.).

- Estados de Acción

  Son la representación de las acciones no interrumpidas de los objetos («Diagramas del UML», s. f.).

Figura 33. Representación de Estados de Acción
Fuente: («Diagramas del UML», s. f.)

- Flujo de la Acción

  Exponen la relación que existe entre dos estados de acción y son representados con flechas («Diagramas del UML», s. f.).

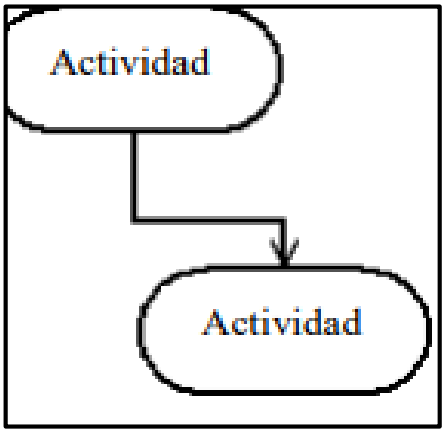

Figura 34. Representación de Flujo de la Acción
Fuente: («Diagramas del UML», s. f.)

- Flujo de Objetos

  Se refiere a la creación y modificación de los objetos por parte de las actividades («Diagramas del UML», s. f.).

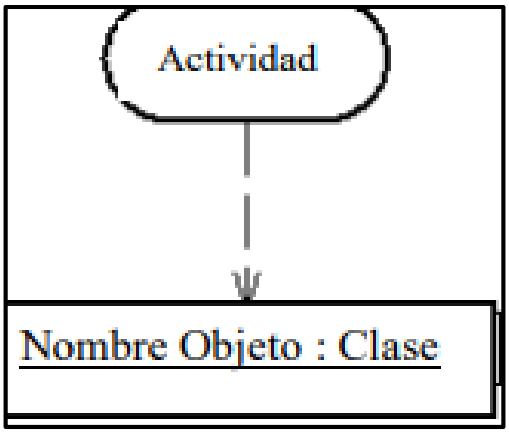

Figura 35. Representación de Flujo de Objetos

Fuente: («Diagramas del UML», s. f.)

- Estados

Estado Inicial

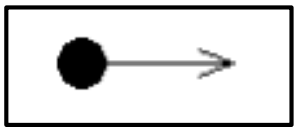

Figura 36. Representación de Estado Inicial

Fuente: («Diagramas del UML», s. f.)

Estado Final

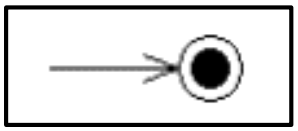

Figura 37. Representación de Estado Final

Fuente: («Diagramas del UML», s. f.)

- Marcos de Responsabilidad

Dichos marcos aglomeran a las actividades relacionadas entre sí en una misma columna («Diagramas del UML», s. f.).

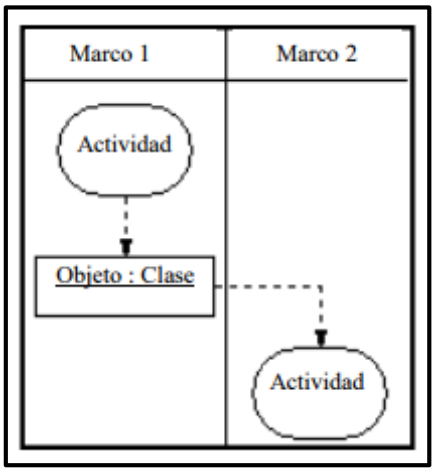

Figura 38. Representación de Marcos de Responsabilidad

Fuente: («Diagramas del UML», s. f.)

2.5.1.5. Diagrama de Componentes

Un diagrama de componentes es aquel que describe la organización de los componentes físicos de un sistema. Un componente es un bloque de construcción física del sistema («Diagramas del UML», s. f.).

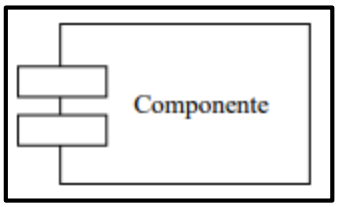

Figura 39. Representación de un Componente
Fuente: («Diagramas del UML», s. f.)

Los elementos que se han mencionado en este capítulo junto con lo presentado en el Capítulo I Marco Teórico, serán utilizados en la presente disertación para la correcta especificación de requerimientos y de diseño mediante el uso de los diagramas UML, representando requerimientos funcionales y casos de uso que tendrá el proyecto a realizarse. Además, serán utilizados para representar los diagramas en los cuales se evidenciará como estará estructurada la base de datos. Todo lo que se ha mencionado será realizado conforme la metodología de desarrollo RUP que especifica en sus diferentes fases con sus respectivos hitos (*véase la sección* 2.4).

Capítulo III Caso de Estudio: Instituto Especial para Niños "Mariana de Jesús" - Ciclo 1

En este capítulo se realizará las clases y procesos correspondientes al primer ciclo con la metodología de desarrollo de software RUP.

3.1. Fase de Inicio – Ciclo 1

3.1.1. Descripción General

3.1.1.1. Justificación

Actualmente, la innovación de la tecnología ha desembocado en el uso de los sistemas de información y éstos han cambiado la forma que operan las empresas.

Teniendo en cuenta el presente proyecto sobre el diseño y desarrollo de un sistema para el Instituto Especial para Niños "Mariana de Jesús", debe decirse que la motivación principal radica en la necesidad que ha evidenciado esta entidad para poder manejar de manera adecuada la información de sus estudiantes.

A raíz de dicha necesidad, este proyecto se justifica en los procesos que deben ser mejorados, y se lo realiza por que se observó una necesidad en el manejo de la información que amerita intervenciones de los sistemas de información.

Esta propuesta se hace para brindar una solución tecnológica que pueda mantener almacenada la información de cada uno de los estudiantes y poder llevar un control histórico de los mismos, planeando un cronograma de ejecución, una implementación, y la documentación correspondiente al mismo.

Personalmente contribuye a generar conocimientos, y aplicar los contenidos vistos en la carrera.

3.1.1.2. Estrategia

El presente proyecto constará de dos ciclos de desarrollo siguiendo la metodología de desarrollo de software RUP. En el primer ciclo serán realizados los principales módulos que el sistema dispondrá, siendo los mismos:

- La gestión de usuarios
- La gestión de estudiantes
- La gestión de historias clínicas
- La gestión de diagnósticos
- La gestión de prescripciones

Se ejecutarán reuniones con el cliente para verificar el avance del proyecto.

En el segundo ciclo serán realizados los dos módulos restantes, siendo éstos:

- La gestión de datos estadísticos
- La gestión de reportes

Dicho proyecto será desarrollado utilizando el IDE de Java NetBeans 8.2 y la base de datos a utilizar será PostgreSQL.

3.1.2. Especificación de Requerimientos – Ciclo 1

3.1.2.1. Requerimientos Funcionales - Ciclo 1

Los requerimientos funcionales serán los siguientes:

RF1. El sistema permitirá gestionar usuarios
RF2. El sistema permitirá gestionar estudiantes
RF3. El sistema permitirá gestionar historias clínicas
RF4. El sistema permitirá gestionar diagnósticos
RF5. El sistema permitirá gestionar prescripciones

Los requerimientos funcionales mencionados anteriormente son representados en el diagrama general de la Figura 40, y serán detallados más adelante con el uso de los diagramas de caso de uso de UML (*véase la sección* 2.5.1.2).

3.1.2.1.1 Diagrama General de Casos de Uso - Ciclo 1

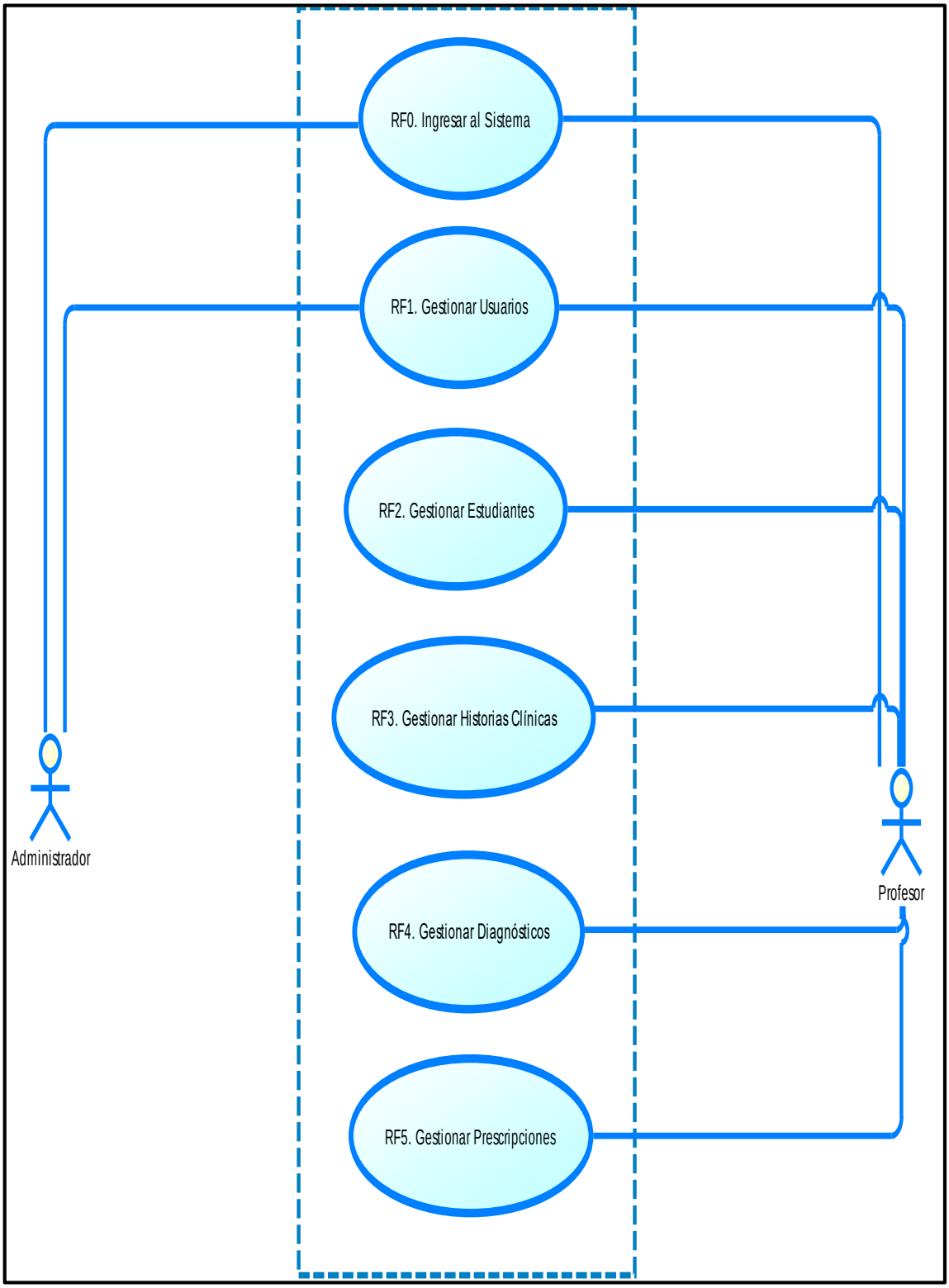

Figura 40. Diagrama General de Casos de Uso – Ciclo 1

Elaborado por: (Recalde, 2017)

3.1.2.1.2 Diagramas Específicos – Siguiente Nivel

RF1. Gestionar Usuarios

Descripción: El administrador será capaz de ingresar, modificar, eliminar y consultar usuarios. Mientas que los profesores únicamente podrán modificar datos de su información correspondiente.

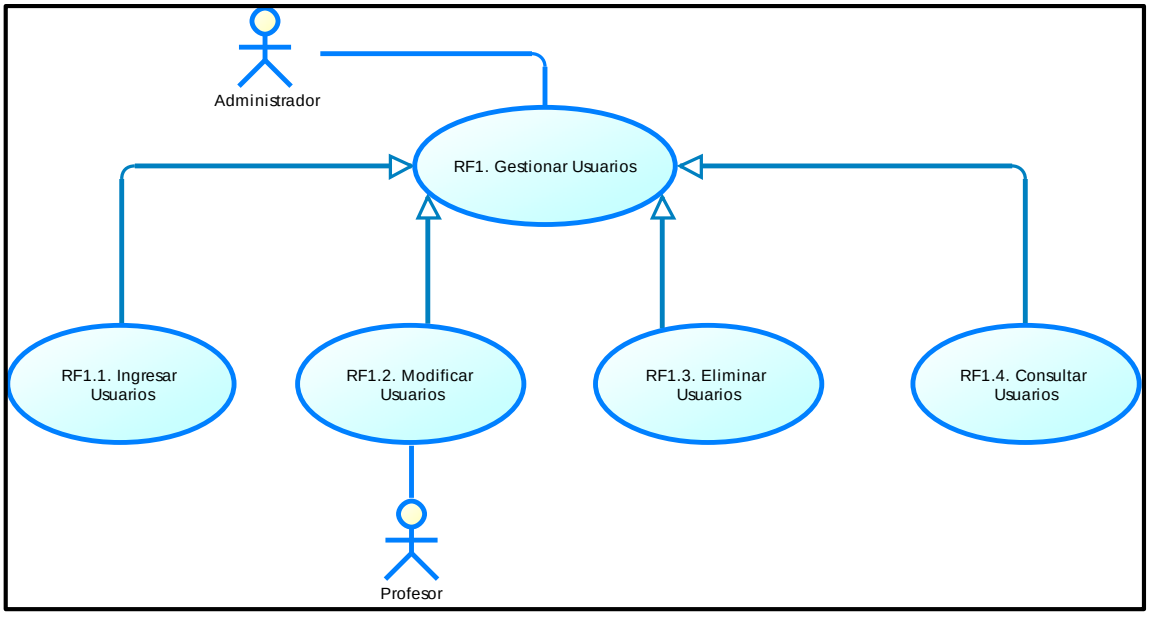

Figura 41. Diagrama Específico de Caso de Uso RF1. Gestionar Usuarios

Elaborado por: (Recalde, 2017)

RF2. Gestionar Estudiantes

Descripción: El profesor será capaz de ingresar, modificar y eliminar estudiantes. Además, podrá realizar consultas generales o por parámetro de los mismos.

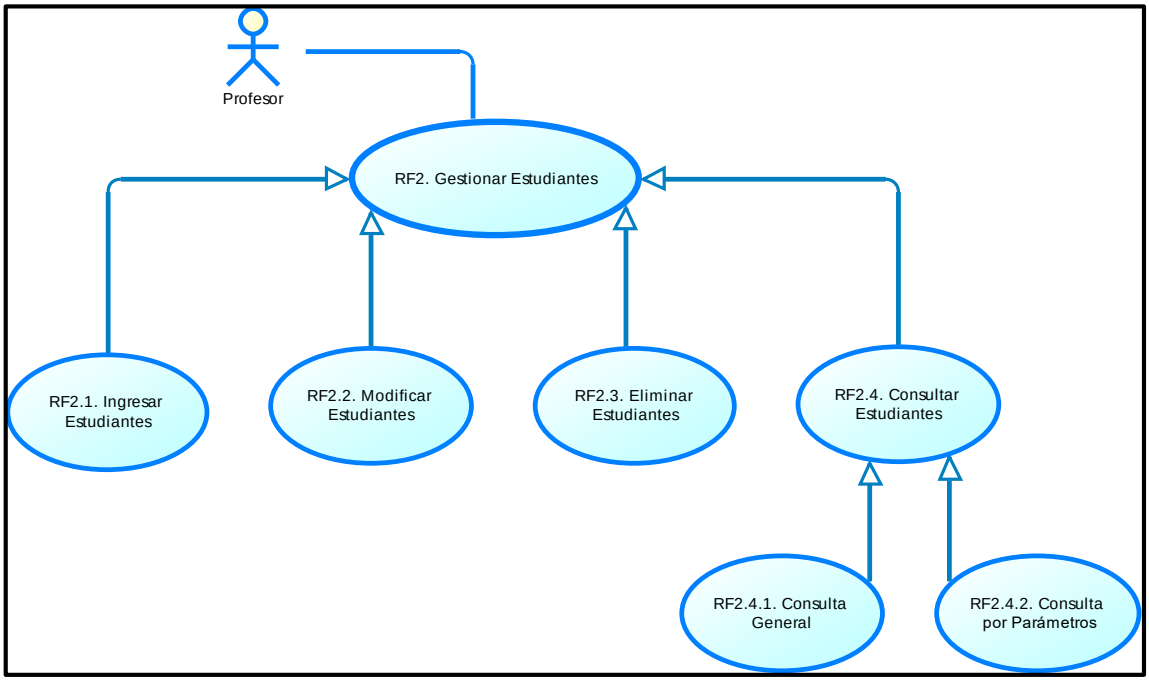

Figura 42. Diagrama Específico de Caso de Uso RF2. Gestionar Estudiantes

Elaborado por: (Recalde, 2017)

RF3. Gestionar Historias Clínicas

Descripción: El profesor será capaz de ingresar, modificar y eliminar historias clínicas.

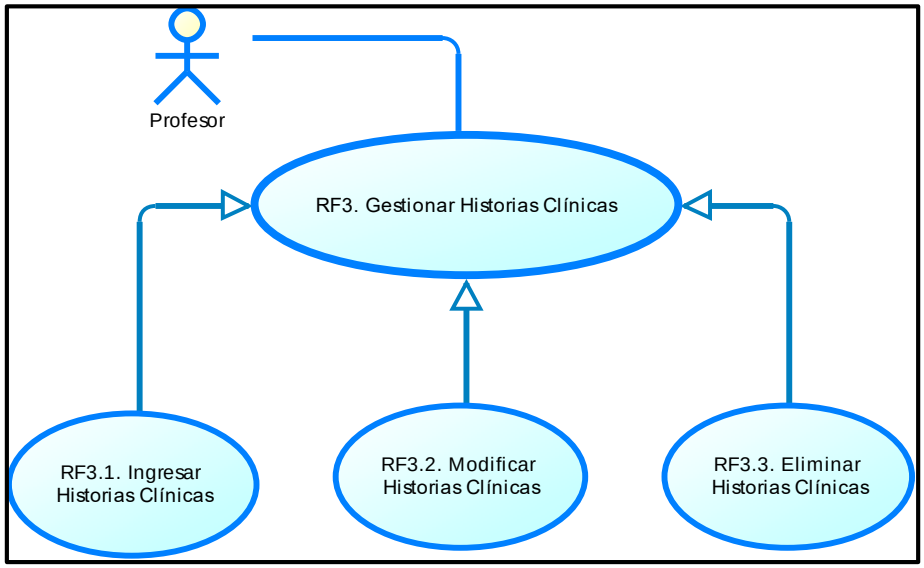

Figura 43. Diagrama Específico de Caso de Uso RF3. Gestionar Historias Clínicas

Elaborado por: (Recalde, 2017)

RF4. Gestionar Diagnósticos

Descripción: El profesor será capaz de ingresar, modificar y eliminar diagnósticos de las historias clínicas.

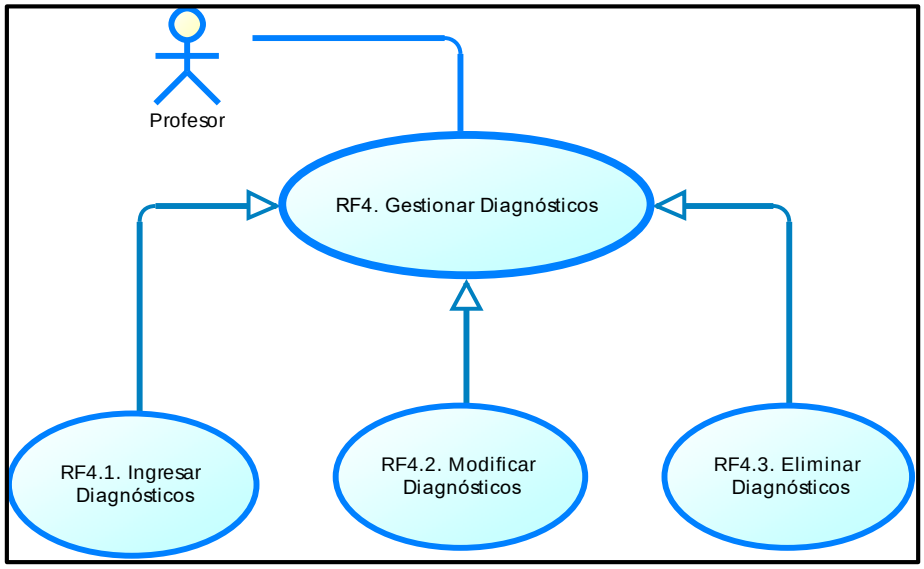

Figura 44. Diagrama Específico de Caso de Uso RF4. Gestionar Diagnósticos

Elaborado por: (Recalde, 2017)

RF5. Gestionar Prescripciones

Descripción: El profesor será capaz de ingresar, modificar y eliminar prescripciones de las historias clínicas.

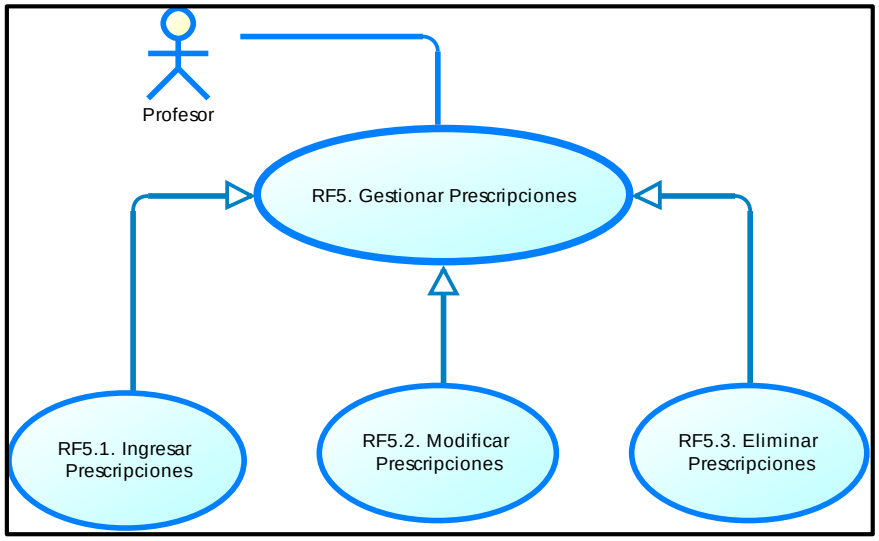

Figura 45. Diagrama Específico de Caso de Uso RF5. Gestionar Prescripciones

Elaborado por: (Recalde, 2017)

3.1.2.1.3 Diagramas de Actividades

Se ha tomado en cuenta el principal proceso que el sistema poseerá, siendo el mismo la gestión de historias clínicas. En el siguiente diagrama se refleja el funcionamiento.

DA1. Diagrama de Actividades – Gestión de Historias Clínicas

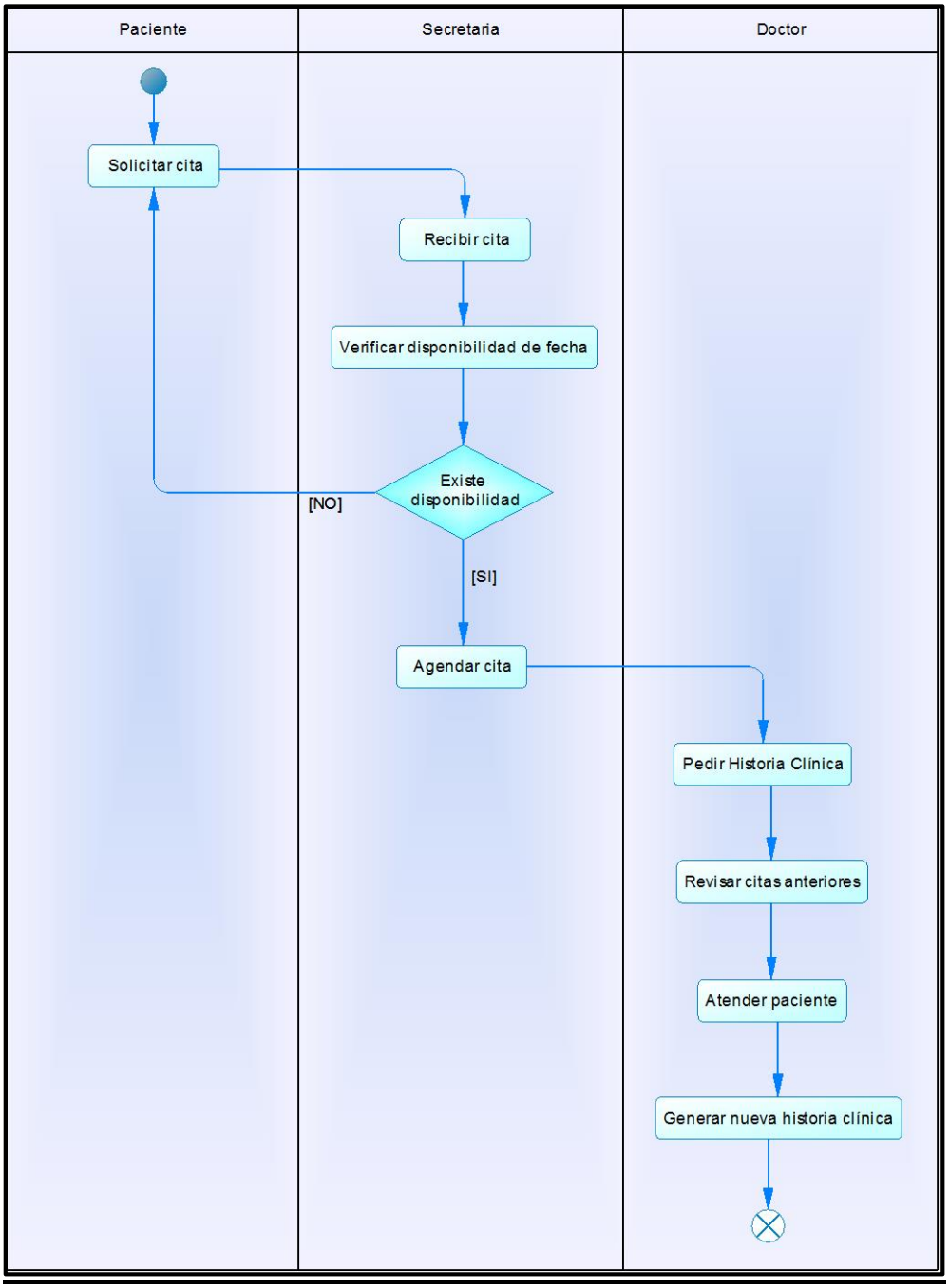

Figura 46. Diagrama de Actividades – Gestión de Historias Clínicas

Elaborado por: (Recalde, 2017)

En la Figura 46 podemos observar la gestión de historias clínicas que se puede estar utilizando en la actualidad en la Fundación, la misma que nos indica que podrían ser automatizados dos procesos:

- El proceso de reserva
- El proceso de gestión de fichas

La principal necesidad de la fundación al momento es la automatización del proceso de gestión de fichas, es por esto que se ha decidido realizar dicha automatización en la presente disertación.

3.1.2.2. Otros Requerimientos

El sistema que se va a desarrollar necesita de ciertos requerimientos a nivel de software como de hardware tanto por parte del cliente para su correcto funcionamiento como por los desarrolladores para su desarrollo, actualizaciones, etc.

3.1.2.2.1 Requerimientos de Software

	Cliente	**Desarrollador**
Sistema Operativo	Windows XP o superior	Windows 8.1 o superior
IDE	—	Entornos de desarrollo para Java
JDK	Java SE 8.0	Java SE 8.0

Tabla 2. Requerimientos de Software

Elaborado por: (Recalde, 2017)

3.1.2.2.2 Requerimientos de Hardware

	Cliente	**Desarrollador**
Procesador	Intel Centrino o superior	Intel Inside i5 7ma Generación o superior
Memoria RAM	4Gb o superior	8Gb o superior
Unidad de Disco Duro	600Mb o superior	1Tb o superior
Tarjeta de Video	—	AMD Radeon R7 4Gb

Tabla 3. Requerimientos de Hardware

Elaborado por: (Recalde, 2017)

3.1.2.3. Estimación de Riesgos

3.1.2.3.1 Formulario ITL

Formulario ITL

Número	Fecha	Riesgo/Problema	Fecha FU	Resuelto
1	25/09/2017	Problema	30/09/2017	30/09/2017
Descripción:	Problemas para comprender por completo JAVA			
Número	Fecha	Riesgo/Problema	Fecha FU	Resuelto
2	27/09/2017	Riesgo	03/10/2017	05/10/2017
Descripción:	Cliente cambie requerimientos			
Número	Fecha	Riesgo/Problema	Fecha FU	Resuelto
3	27/09/2017	Problema	20/10/2017	23/10/2017
Descripción:	Usuarios finales insatisfechos			
Número	Fecha	Riesgo/Problema	Fecha FU	Resuelto
4	27/09/2017	Riesgo	20/10/2017	20/10/2017
Descripción:	Falta de reuniones con los usuarios finales			
Número	Fecha	Riesgo/Problema	Fecha FU	Resuelto
5	27/09/2017	Problema	08/11/2017	08/11/2017
Descripción:	Cambio de director de fundación			
Número	Fecha	Riesgo/Problema	Fecha FU	Resuelto
6	24/09/2017	Riesgo	30/09/2017	02/10/2017
Descripción:	Mala definición de requerimientos por parte del usuario			

Tabla 4. Formulario ITL - Estimación de Riesgos

Elaborado por: (Recalde, 2017)

3.1.2.4. Prototipo de Interfaz

En el siguiente apartado se muestra un prototipo ejemplo que tendrá el sistema a realizar

- Sistema de Ingreso

  El sistema contará con una ventana de login o ingreso, en la cual los usuarios deberán ingresar sus credenciales, el sistema los validará, y si son correctos y existen el sistema le permitirá el acceso al mismo.

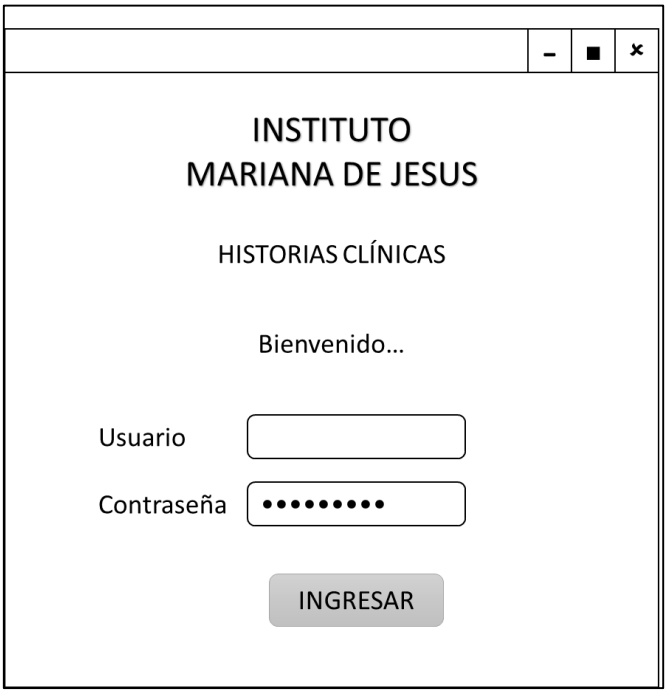

Figura 47. Prototipo de Interfaz del Sistema de Ingreso
Elaborado por: (Recalde, 2017)

- Menú Principal

  Al momento los principales módulos a presentarse en el menú principal son los que se muestran en la siguiente figura. Cada uno de los mismos tendrán acceso de acuerdo con el tipo de usuario que se encuentre loggeado y los cuales fueron explicados diagramas UML de casos de uso en el apartado 3.1.2.1.2

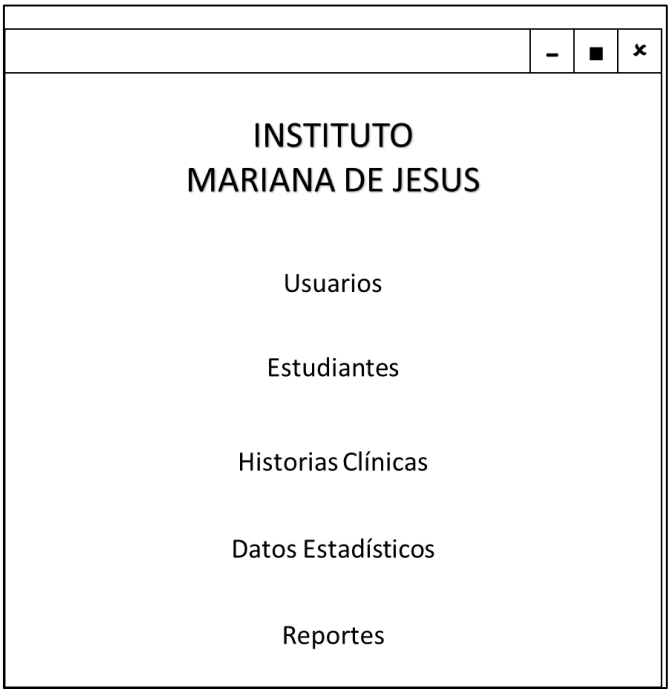

Figura 48. Prototipo de Interfaz del Menú Principal

Elaborado por: (Recalde, 2017)

- Gestión de Historias Clínicas

  Según los requerimientos especificados por parte del usuario, la siguiente figura muestra un prototipo de la ventana de Gestión de Historias Clínicas.

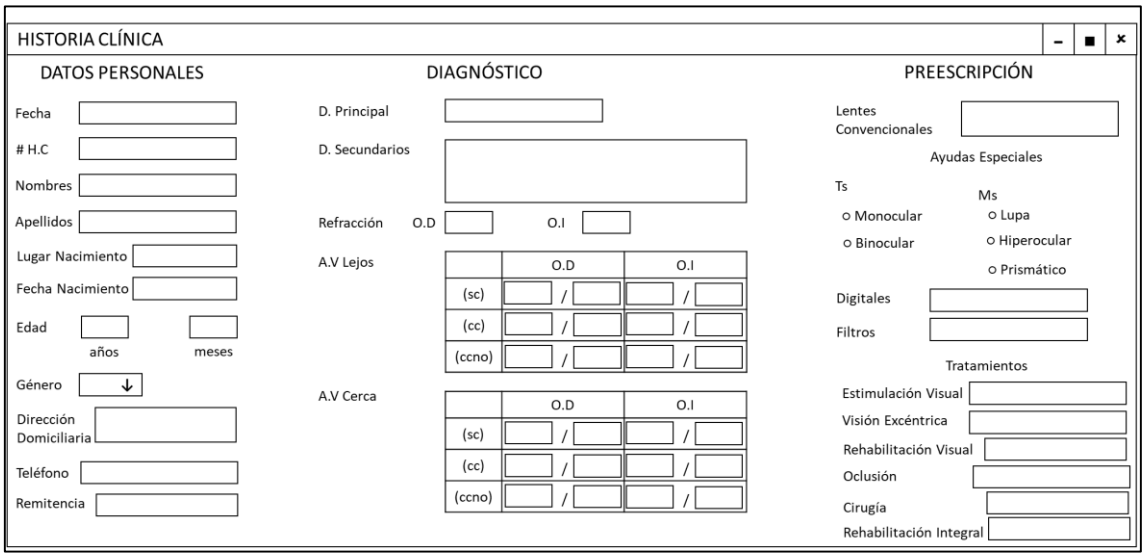

Figura 49. Prototipo de Interfaz de Gestión de Historias Clínicas

Elaborado por: (Recalde, 2017)

3.2. Fase de Elaboración – Ciclo 1

3.2.1. Diseño Conceptual de Clases

Una vez que hemos recopilado todos los requerimientos, el siguiente paso es crear un esquema o diseño conceptual que servirá para la base de datos, como se observa en la Figura 50 es un modelo de datos conceptual de bajo nivel, donde se encuentran identificadas las siguientes clases:

- Agudeza Visual Cerca
- Agudeza Visual Lejos
- Ayudas Especiales
- Datos Familiares
- Diagnóstico
- Estudiante
- Historia Clínica
- Prescripción
- Tratamientos

- Usuario

Siendo identificadas a las clases Estudiante e Historia Clínica como las dos principales.

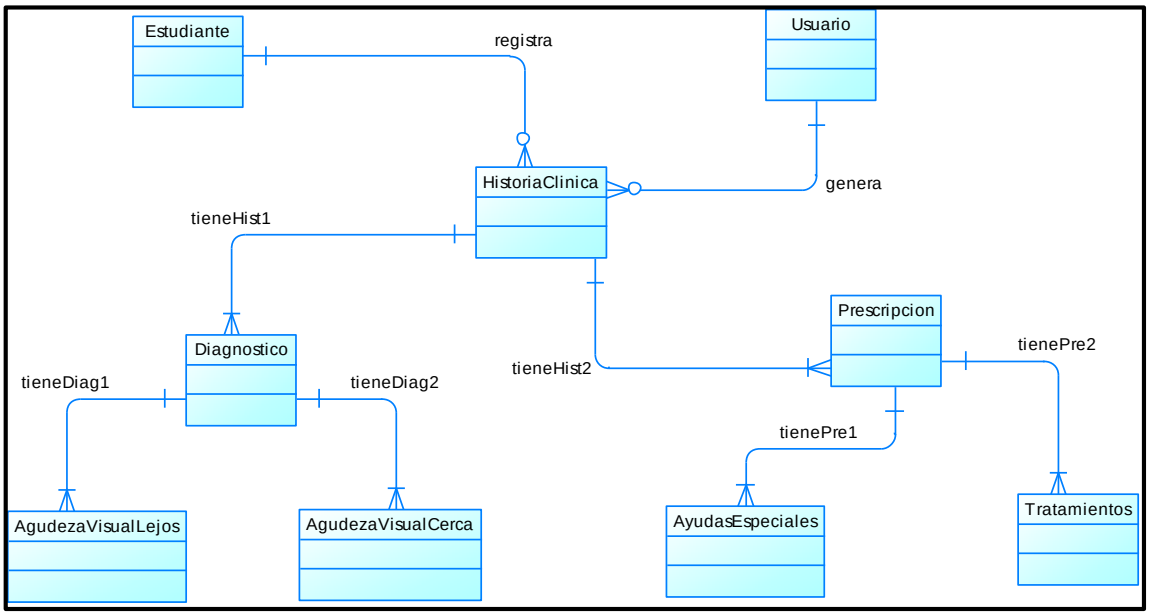

Figura 50. Diseño Conceptual de la Base de Datos

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 3. Diseño Conceptual, que se encuentra en el CD.

3.2.2. Definición de ciclos de desarrollo y tiempos

Como se explicó en el apartado 0 los principales requerimientos serán realizados en el primer ciclo, debido a que gracias a éstos se puede obtener los datos para poder realizar los dos requerimientos restantes del sistema.

3.2.2.1. Formulario STRAT

Formulario STRAT

Referencia	Funciones	LOC de Ciclo 1		Horas de Ciclo 1	
		1	2	1	2
F1.0	Gestionar usuario				
F1.1	Ingresar Usuarios	115		2	
F1.2	Modificar Usuarios	95		2	
F1.3	Eliminar Usuarios	100		1,5	
F1.4	Consultar Usuarios	100		1,5	
F2.0	Gestionar Estudiantes				
F2.1	Ingresar Estudiantes	95		2,5	
F2.2	Modificar Estudiantes	105		2,5	
F2.3	Eliminar Estudiantes	100		2	
F2.4	Consultar Estudiantes	130		1,5	
F3.0	Gestionar Historias Clínicas				
F3.1	Ingresar Historias Clínicas	90		2	
F3.2	Modificar Historias Clínicas	75		2,5	
F3.3	Eliminar Historias Clínicas	70		2	
F4.0	Gestionar Diagnósticos				
F4.1	Ingresar Diagnósticos	110		3	
F4.2	Modificar Diagnósticos	105		2,5	
F4.3	Eliminar Diagnósticos	105		2,5	
F5.0	Gestionar Prescripciones				
F5.1	Ingresar Prescripciones	130		3	
F5.2	Modificar Prescripciones	125		3	
F5.3	Eliminar Prescripciones	110		2,5	
F6.0	Gestionar Datos Estadísticos				
F6.1	Consultar Datos Estadísticos		130		5,0
F7.0	Gestionar Reportes				
F7.1	Generar Reporte de Estudiantes por Parámetro		100		3,5
F7.2	Generar Reporte de Historias Clínicas por Parámetro		100		3,5
F7.3	Generar Reporte de Datos Estadísticos por Parámetro		100		3,5
	TOTAL	1.760	430	38,5	15.5

Tabla 5. Formulario STRAT – Definición de Ciclos de Desarrollo y Tiempos

Elaborado por: (Recalde, 2017)

3.2.3. Diagrama de Clases – Ciclo 1

En el siguiente apartado se muestran los diagramas de clases en las tres capas que existirán en la arquitectura del sistema, basado en lo mencionado en el apartado 1.6, se dispone de la capa de presentación, la capa de dominio del problema y la capa de manejo de datos.

3.2.3.1. Capa de Manejo de Datos – Ciclo 1

3.2.3.1.1 Diagrama Conceptual de Clases de Manejo de Datos – Ciclo 1

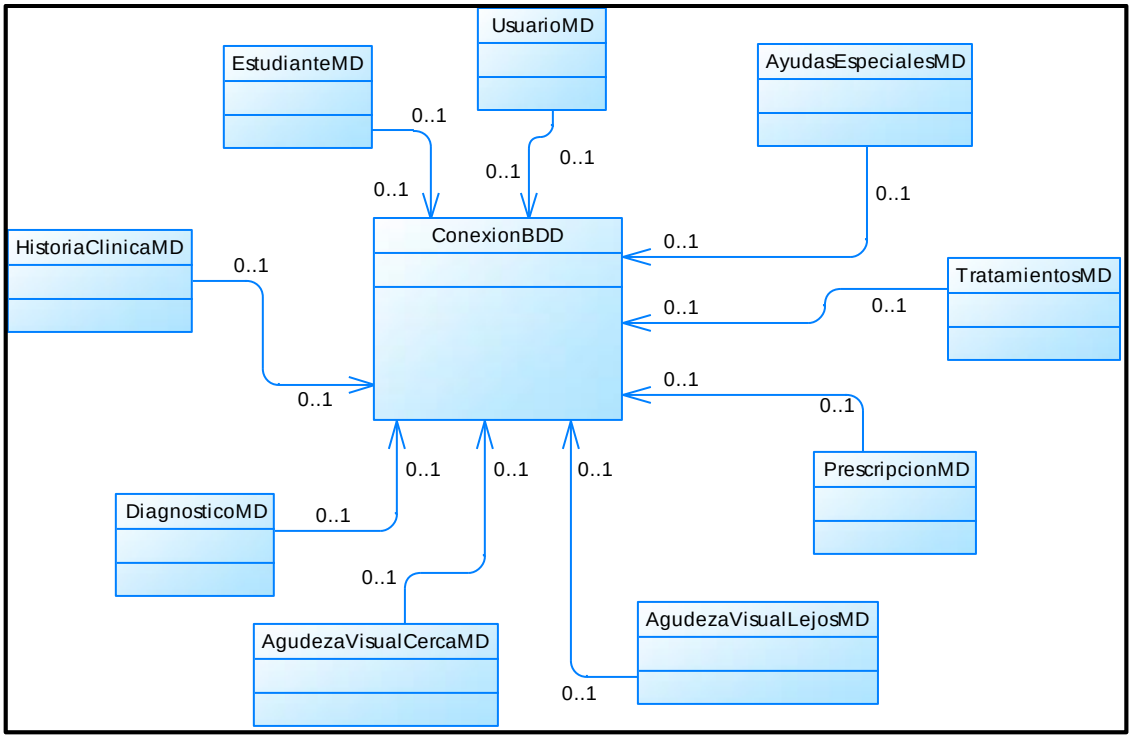

Figura 51. Diagrama Conceptual de Clases de Manejo de Datos – Ciclo 1

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 4. Diagramas de Clases → 1. Manejo de Datos, que se encuentra en el CD.

3.2.3.1.2 Diagrama de Clases de Manejo de Datos – Ciclo 1

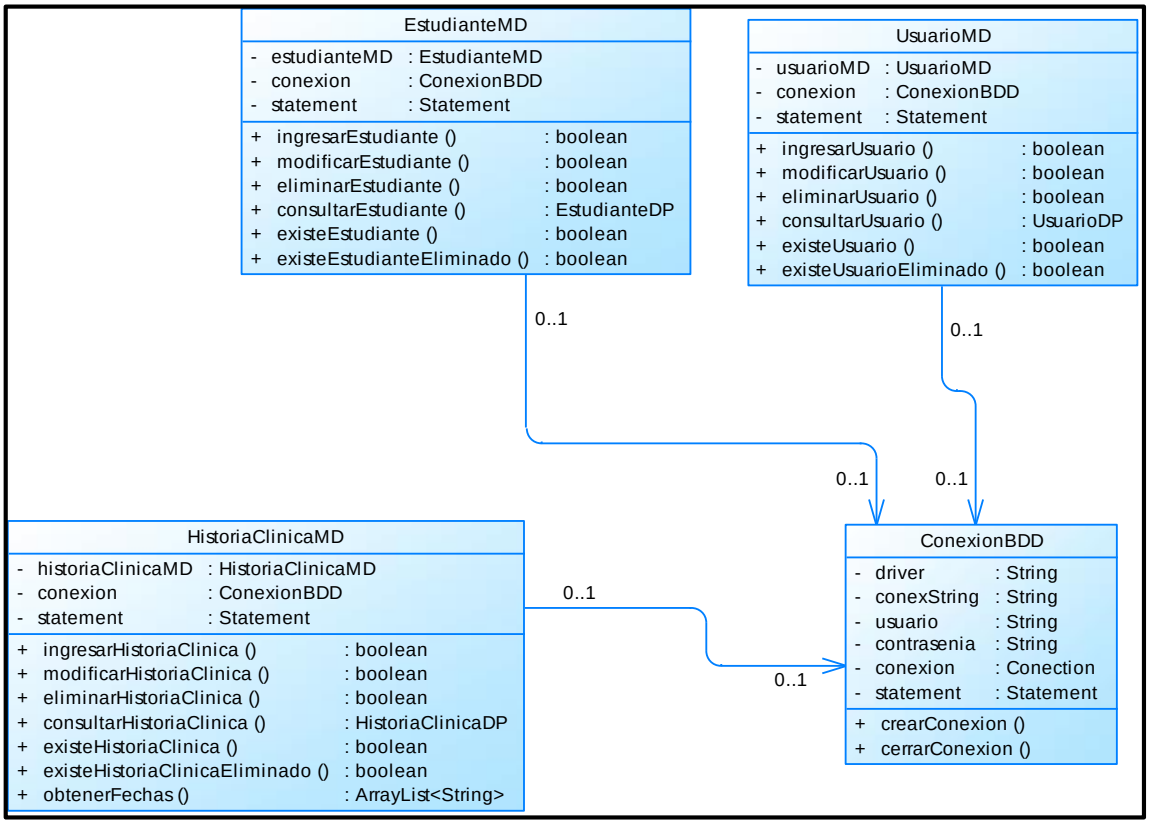

Figura 52. Diagrama de Clases de Manejo de Datos – Ciclo 1

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 4. Diagramas de Clases → 1. Manejo de Datos, que se encuentra en el CD.

3.2.3.2. Capa de Dominio del Problema – Ciclo 1

3.2.3.2.1 Diagrama de Conceptual Clases del Dominio del Problema – Ciclo 1

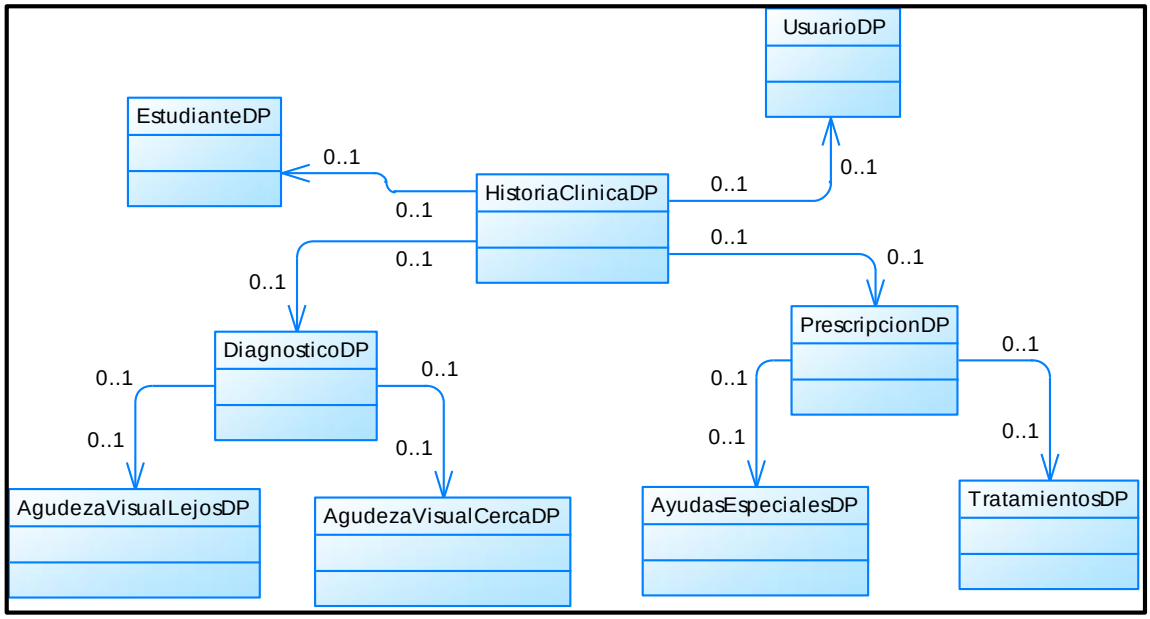

Figura 53. Diagrama Conceptual de Clases del Dominio del Problema – Ciclo 1
Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 4. Diagramas de Clases → 2. Dominio del Problema, que se encuentra en el CD.

3.2.3.2.2 Diagrama de Clases del Dominio del Problema – Ciclo 1

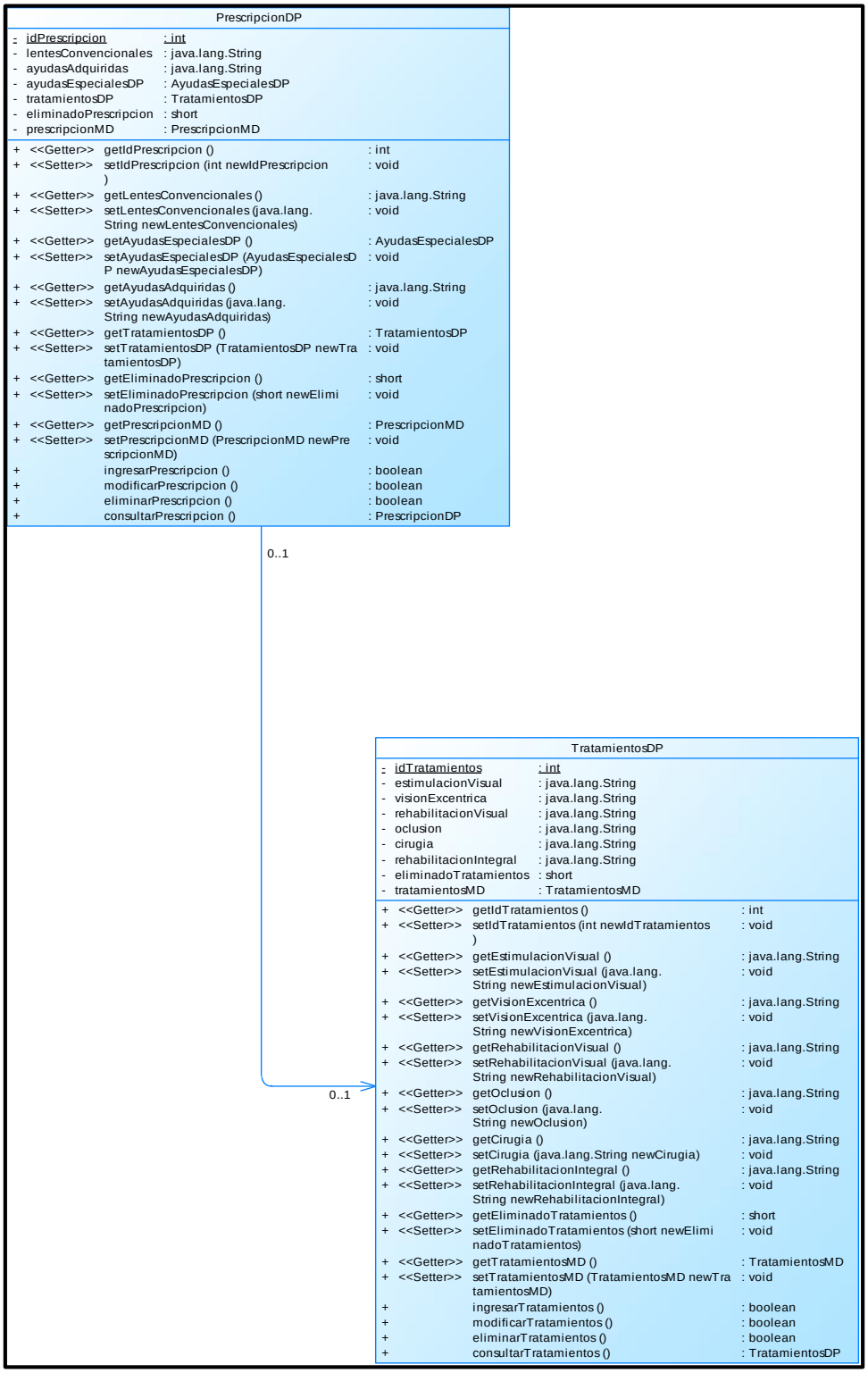

Figura 54. Diagrama de Clases del Dominio del Problema – Ciclo 1

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 4. Diagramas de Clases → 2. Dominio del Problema, que se encuentra en el CD.

3.2.3.3. Capa GUI o de Presentación – Ciclo 1

3.2.3.3.1 Diagrama de Clases GUI – Ciclo 1

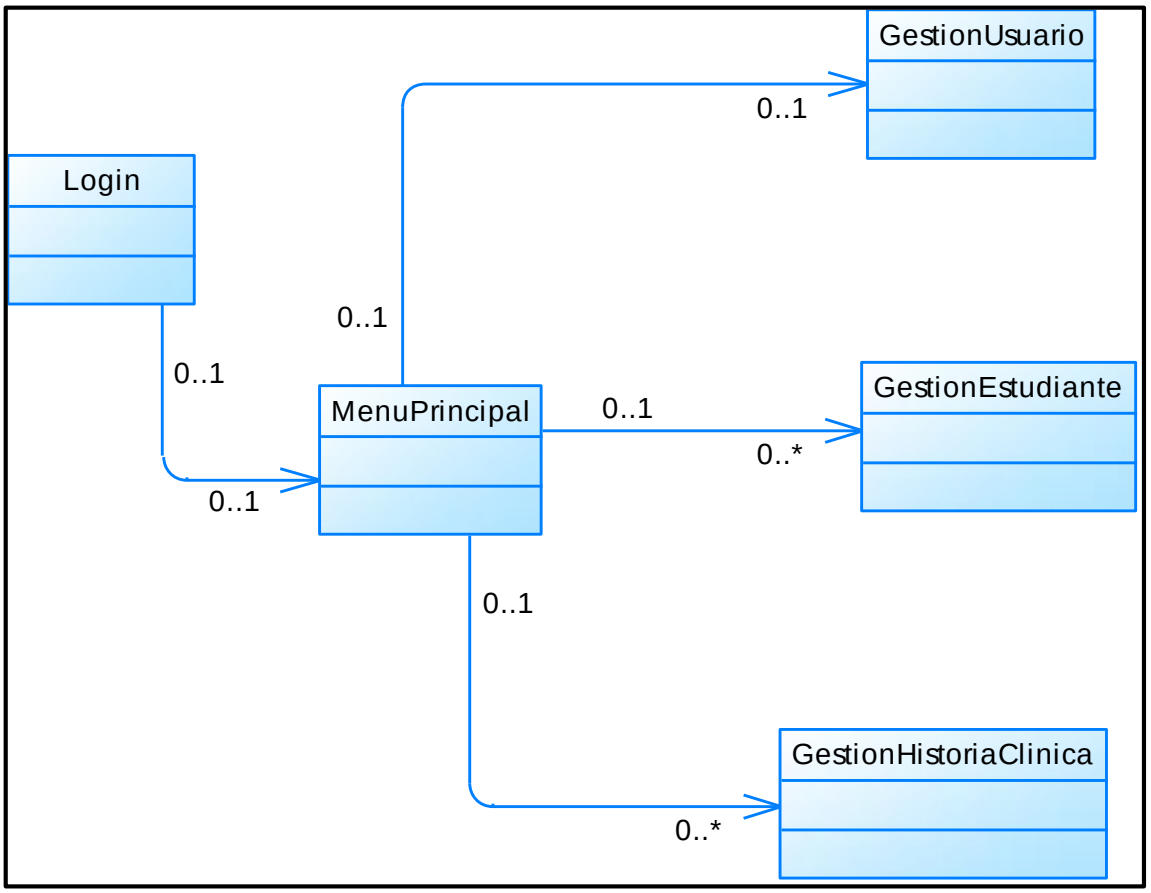

Figura 55. Diagrama de Clases GUI – Ciclo 1

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 4. Diagramas de Clases → 3. GUI, que se encuentra en el CD.

3.2.4. Diagramas Entidad – Relación de la Base de Datos

A continuación, se presentan los diagramas de la estructura que tendrá la base de datos a los tres niveles.

3.2.4.1. Nivel Conceptual

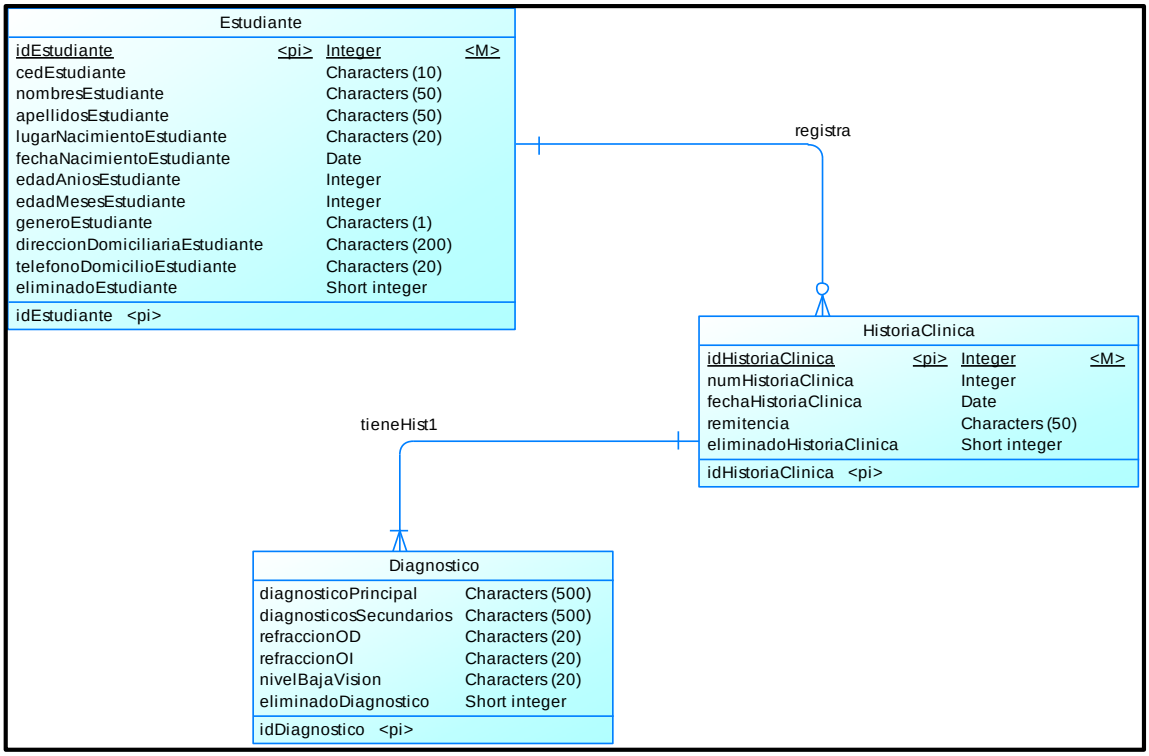

Figura 56. Diagrama Entidad - Relación de la Base de Datos a Nivel Conceptual

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 6. Diagramas Entidad – Relación → 1. Diagrama E-R Conceptual, que se encuentra en el CD.

3.2.4.2. Nivel Lógico

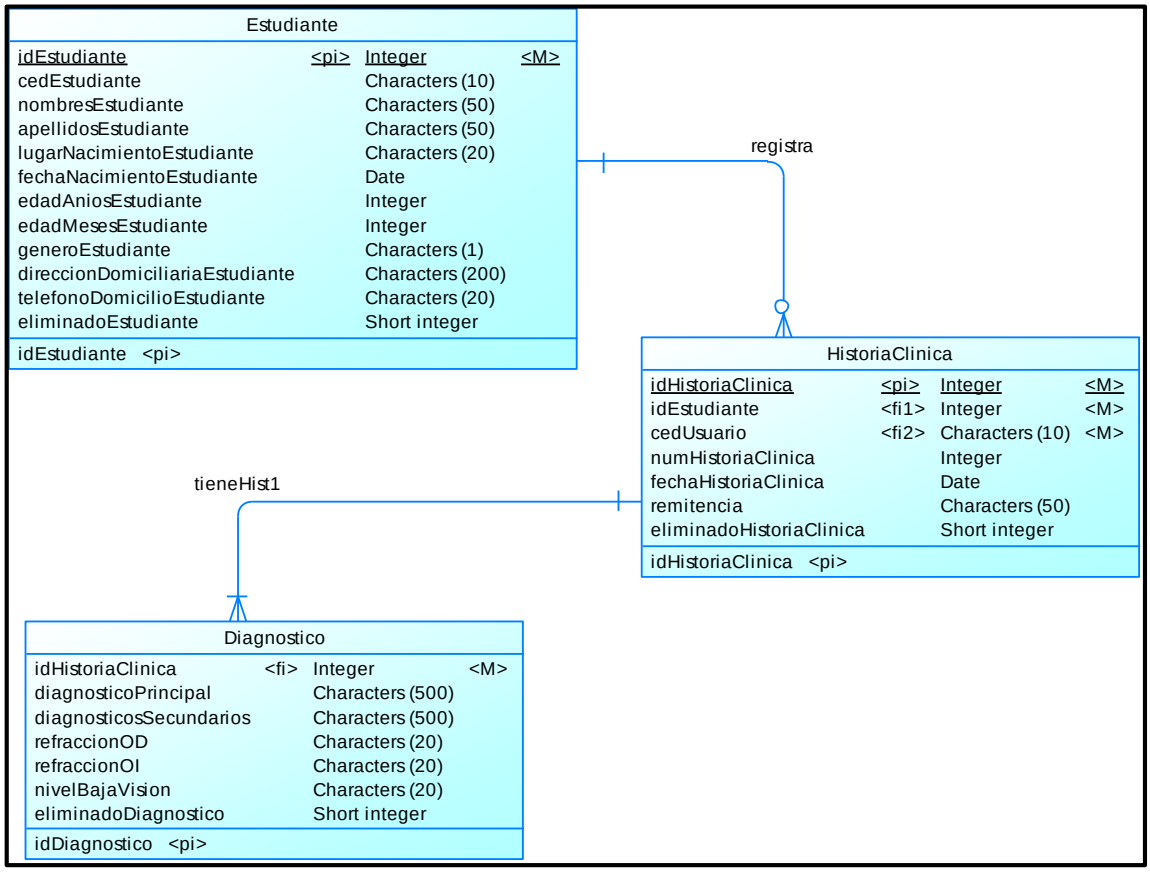

Figura 57. Diagrama Entidad - Relación de la Base de Datos a Nivel Lógico

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 6. Diagramas Entidad – Relación → 2. Diagrama E-R Lógico, que se encuentra en el CD.

3.2.4.3. Nivel Físico

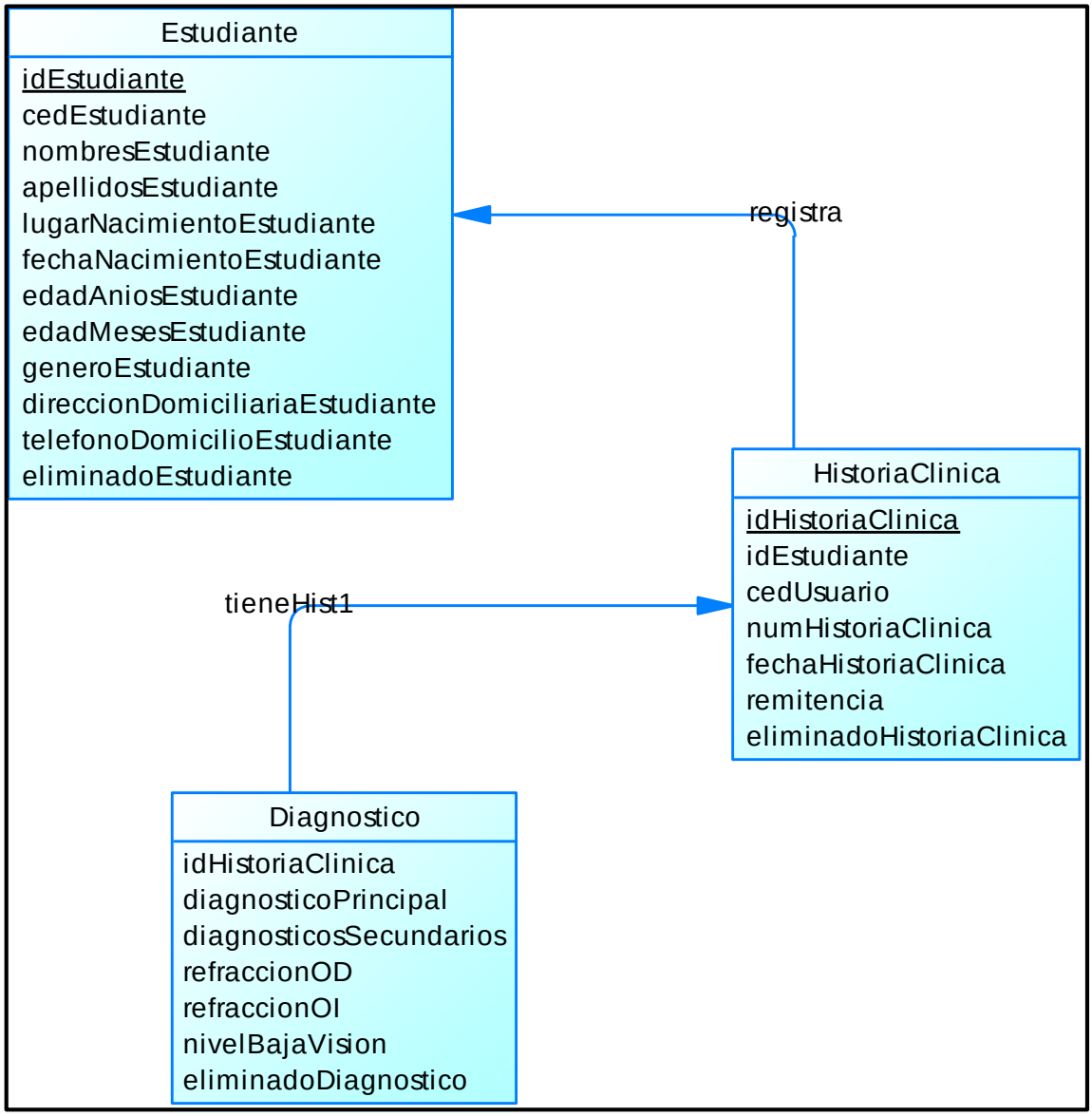

Figura 58. Diagrama Entidad - Relación de la Base de Datos a Nivel Físico

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 6. Diagramas Entidad – Relación → 3. Diagrama E-R Físico, que se encuentra en el CD.

3.2.5. Diagrama de Paquetes – Ciclo 1

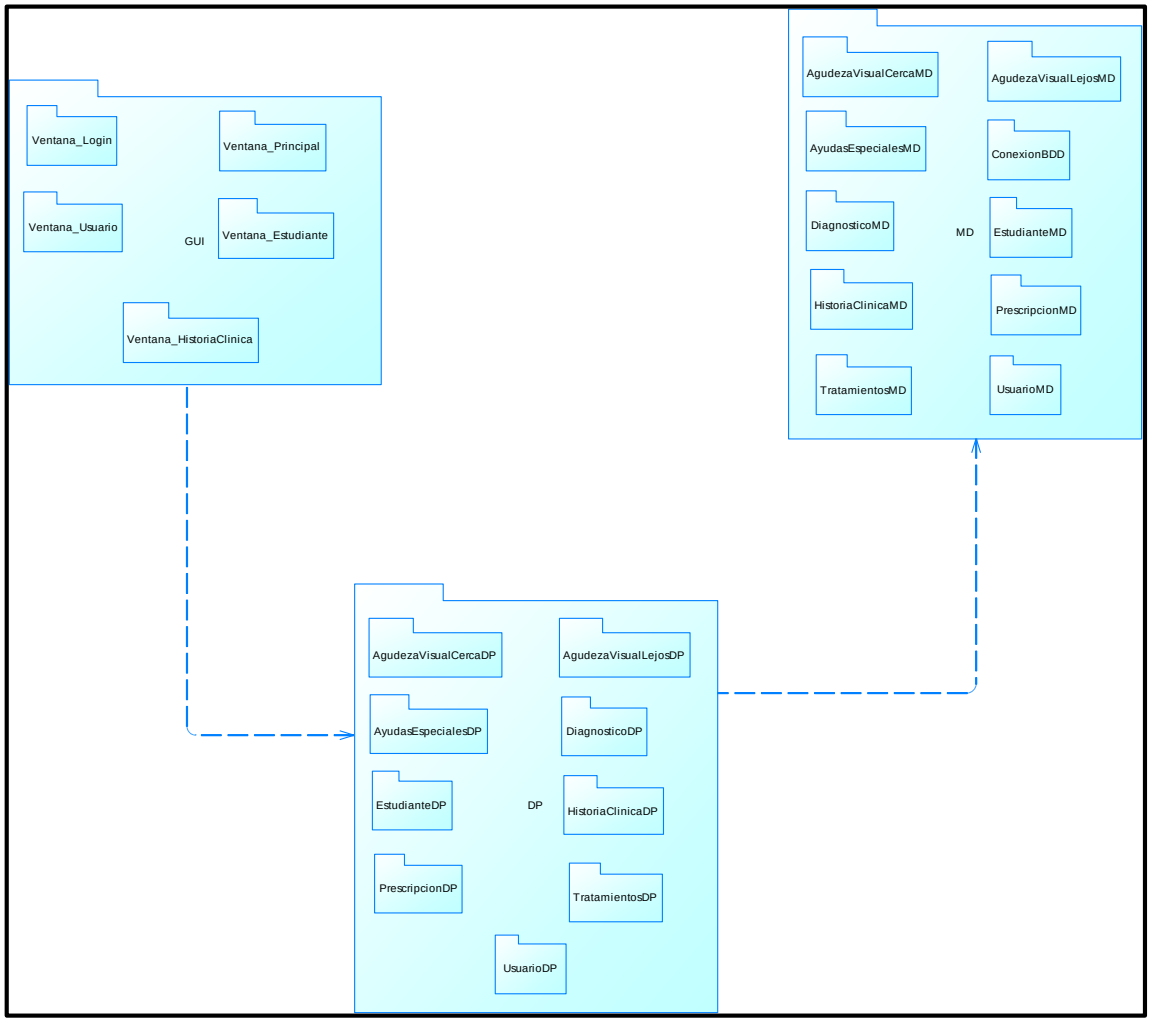

Figura 59. Diagrama de Paquetes – Ciclo 1

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 7. Diagrama de Paquetes, que se encuentra en el CD.

3.2.6. Diagramas de Secuencia

Los siguientes diagramas reflejan la estructura de los principales procesos del sistema, es decir como el usuario interactuará con el sistema, y como el mismo procesa y responde.

3.2.6.1. Diagrama de Secuencia – Gestión de Usuarios

En la Figura 60 se observará la interacción entre el usuario y el sistema para el proceso de inicio de sesión y el manejo del CRUD de usuarios.

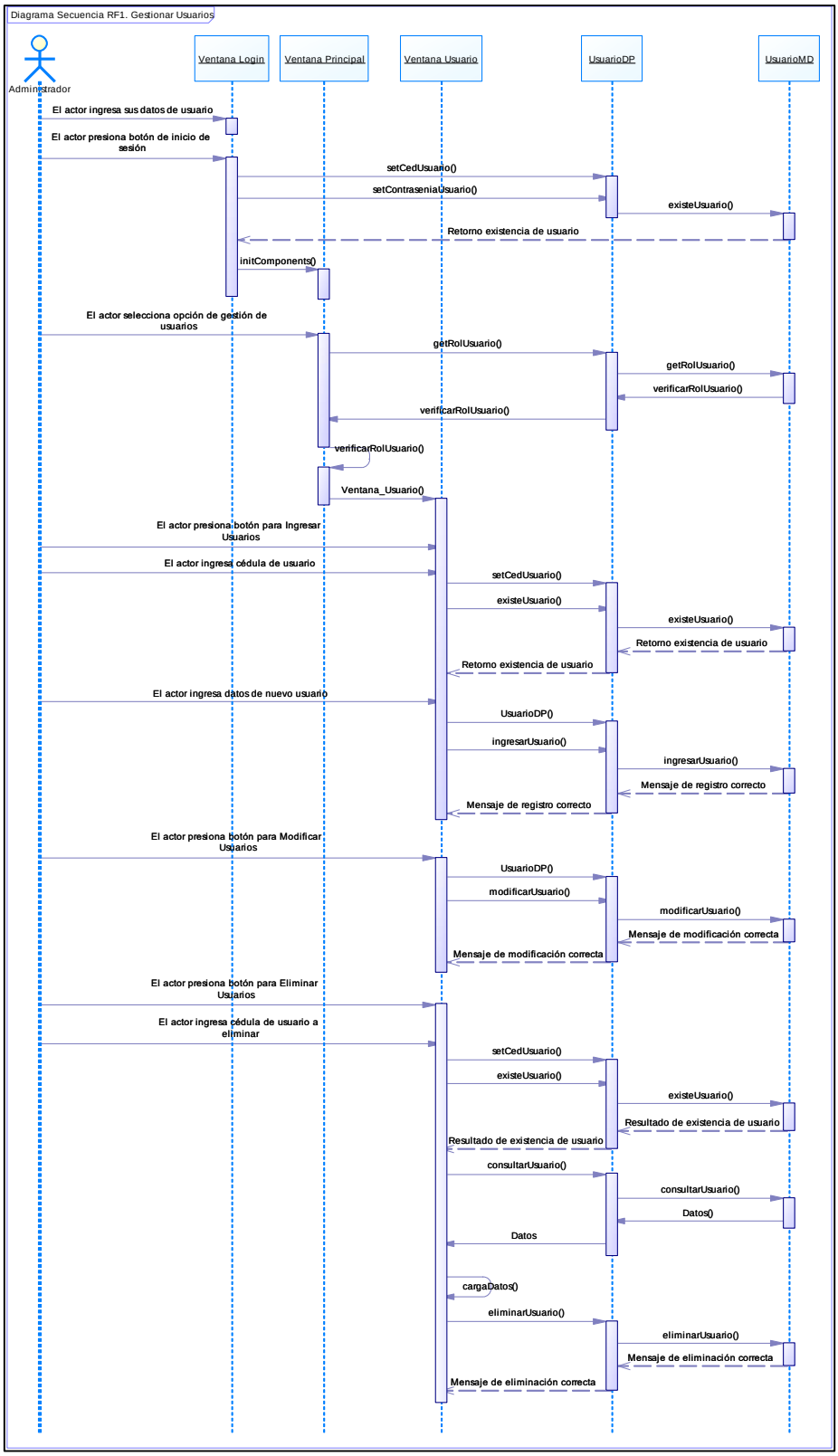

Figura 60. Diagrama de Secuencia – Gestión de Usuarios

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 8. Diagramas de Secuencia → 1. Gestionar Usuarios, que se encuentra en el CD.

3.2.6.2. Diagrama de Secuencia – Gestión de Historias Clínicas

En la siguiente figura se muestra la interacción entre el usuario y el sistema para el ingreso de historias clínicas.

- RF3.1. Ingresar Historias Clínicas

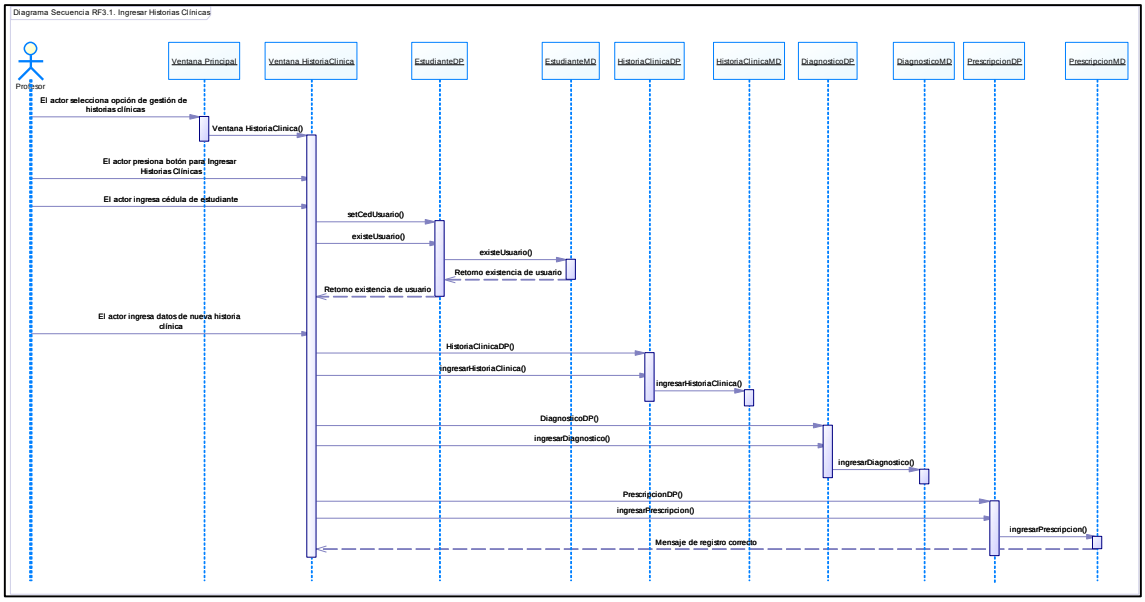

Figura 61. Diagrama de Secuencia - RF3.1. Ingresar Historias Clínicas
Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 8. Diagramas de Secuencia → 2. Gestionar Historias Clínicas, que se encuentra en el CD.

3.2.7. Plan de Pruebas del Sistema – Ciclo 1

Las pruebas del sistema sirven para asegurar el apropiado uso y manejo del sistema y de todas sus funcionalidades, se enfocan en los requerimientos que fueron provistos por el usuario al inicio del proyecto y que pueden ser tomados de los casos de uso, ejecutando los mismos con datos válidos y erróneos (Londoño, 2005).

3.2.7.1. RF1. Gestionar Usuarios

Entradas	Resultados Esperados	Caso de Uso	Estado
Selección de opción Gestión de Usuarios	Despliegue de Ventana de Usuarios	RF1	
Verificación de cédula	Sistema valida que la cédula sea correcta	RF1	
Verificación de cédulas existentes	Sistema valida que la cédula ingresada exista en la base de datos	RF1	
Verificación de elección de rol de usuario	Sistema valida que se seleccione un tipo de rol para un usuario	RF1	
Verificación de contraseña	Sistema valida que las contraseñas ingresadas coincidan	RF1	
Verificación de usuario loggeado	Sistema valida que el usuario a eliminar no se encuentre loggeado	RF1	
Ingresar Usuario	Datos almacenados correctamente	RF1.1	
Modificar Usuario	Datos modificados correctamente	RF1.2	
Eliminar Usuario	Datos eliminados correctamente	RF1.3	
Consultar Usuario	Visualización de datos	RF1.4	

Tabla 6. Plan de Pruebas del Sistema para RF1. Gestionar Usuarios

Elaborado por: (Recalde, 2017)

3.2.7.2. RF2. Gestionar Estudiantes

Entradas	Resultados Esperados	Caso de Uso	Estado
Selección de opción Gestión de Estudiantes	Despliegue de Ventana de Estudiantes	RF2	
Verificación de número de historia clínica	Sistema valida que el número de historia clínica sea correcto	RF2	
Verificación de historias clínicas existentes	Sistema valida que el número de historia clínica ingresada exista en la base de datos	RF2	
Verificación de elección de género	Sistema valida que se seleccione un tipo de género para el estudiante	RF2	
Ingresar Estudiante	Datos almacenados correctamente	RF2.1	
Modificar Estudiante	Datos modificados correctamente	RF2.2	
Eliminar Estudiante	Datos eliminados correctamente	RF2.3	
Consultar Estudiante	Visualización de datos	RF2.4	

Tabla 7. Plan de Pruebas del Sistema para RF2. Gestionar Estudiantes

Elaborado por: (Recalde, 2017)

3.2.7.3. RF3. Gestionar Historias Clínicas - RF4. Gestionar Diagnósticos – RF5. Gestionar Prescripciones

Entradas	Resultados Esperados	Caso de Uso	Estado
Selección de opción Gestión de Historias Clínicas	Despliegue de Ventana de Historias Clínicas	RF3 RF4 RF5	
Verificación de existencia de historias clínicas de estudiante seleccionado	Sistema valida que el estudiante seleccionado posea historias clínicas	RF3	
Verificación de historias clínicas existentes	Sistema valida que el número de historia clínica ingresada exista en la base de datos	RF3	
Verificación de datos obligatorios	Sistema valida que los campos obligatorios se encuentren llenos	RF3 RF4 RF5	
Verificación de longitud de cadenas	Sistema verifica que la longitud de los datos no sobrepase el tamaño especificado en la BDD	RF3 RF4 RF5	
Ingresar Historia Clínica	Datos almacenados correctamente	RF3.1 RF4.1 RF5.1	
Modificar Historia Clínica	Datos modificados correctamente	RF3.1 RF4.1 RF5.1	
Eliminar Historia Clínica	Datos eliminados correctamente	RF3.3	
Consultar Historia Clínica	Visualización de datos	RF3.1 RF4.1 RF5.1	

Tabla 8. Plan de Pruebas del Sistema para RF3. Gestionar Historias Clínicas – RF4. Gestionar Diagnósticos – RF5. Gestionar Prescripciones

Elaborado por: (Recalde, 2017)

En el siguiente apartado se explicará la manera en la que fueron desarrollados los módulos correspondientes al ciclo 1, tanto a nivel visual como a en codificación. Debido a la cantidad de código que los módulos poseen, no se presenta el mismo en su totalidad, únicamente se expondrá el código fundamental que se utilizó a lo largo del desarrollo. Además, se presenta la fase final del primer ciclo con su correspondiente informe.

3.3. Fase de Desarrollo – Ciclo 1

3.3.1. Diseño e Implementación de Interfaces Gráficas – Ciclo 1

3.3.1.1. Interfaz de Inicio de Sesión

Figura 62. Interfaz de Inicio de Sesión

Elaborado por: (Recalde, 2017)

3.3.1.2. Interfaz de Ventana o Menú Principal – Ciclo 1

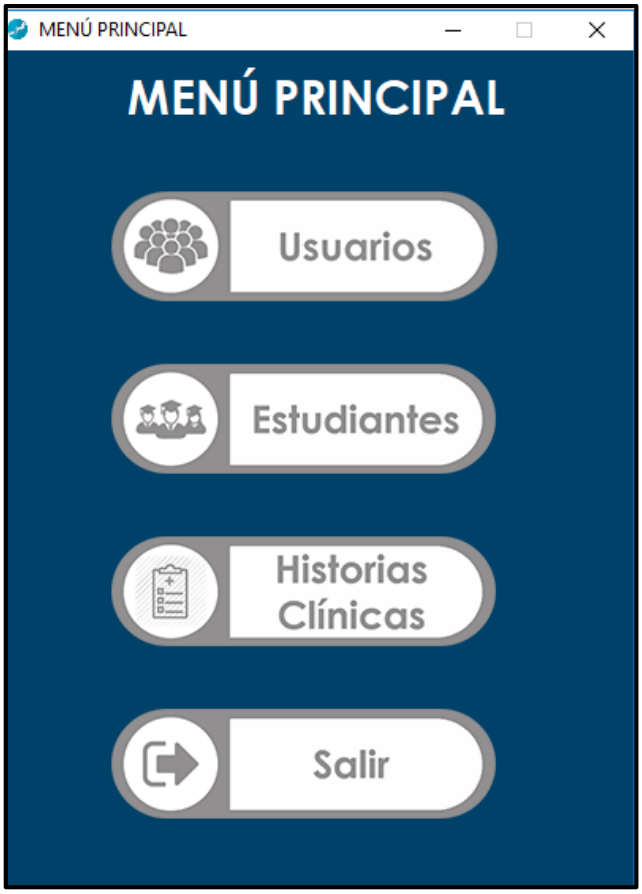

Figura 63. Interfaz de Ventana o Menú Principal

Elaborado por: (Recalde, 2017)

3.3.1.3. Interfaz Ventana Gestión de Usuarios

Figura 64. Interfaz Ventana Gestión de Usuarios

Elaborado por: (Recalde, 2017)

3.3.1.4. Interfaz Ventana Gestión de Estudiantes

Figura 65. Interfaz Ventana Gestión de Estudiantes

Elaborado por: (Recalde, 2017)

3.3.1.5. Interfaz Ventana Gestión de Historias Clínicas

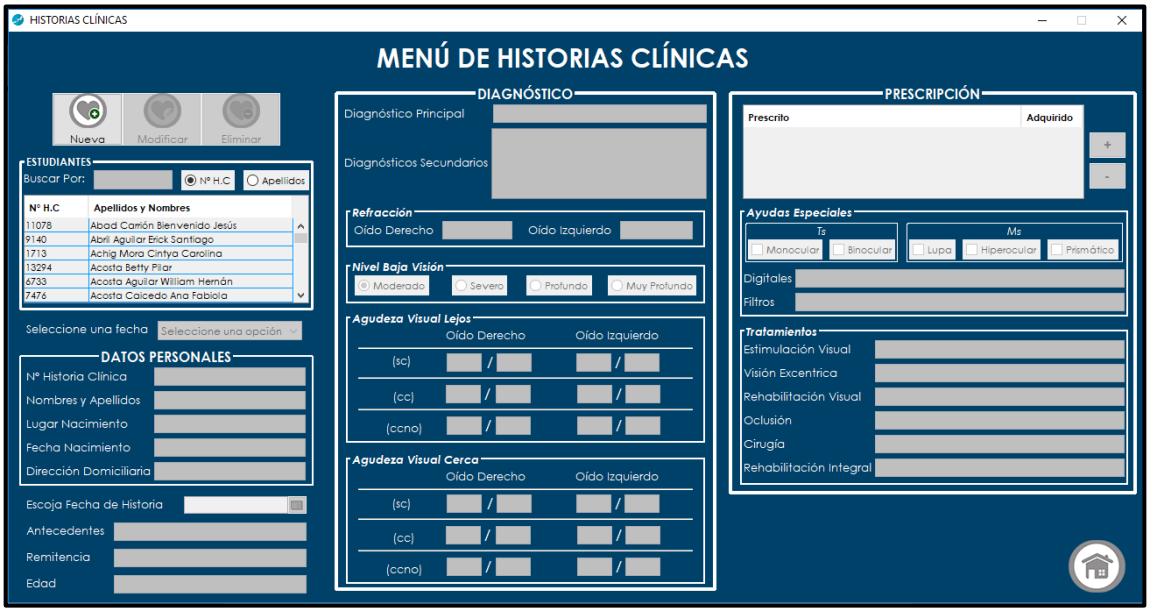

Figura 66. Interfaz Ventana Gestión de Historias Clínicas

Elaborado por: (Recalde, 2017)

3.3.2. Código GUI y Explicación – Ciclo 1

3.3.2.1. Validación de Inicio de Sesión

Para la validación de inicio de sesión se inicializa un objeto UsuarioDP definido anteriormente, mediante el cual se realiza la consulta de la existencia en la base de datos y si su contraseña es la correcta. En caso de no existir el usuario o si la contraseña no coincide con la almacenada en la base de datos se presenta el siguiente cuadro diálogo de aviso:

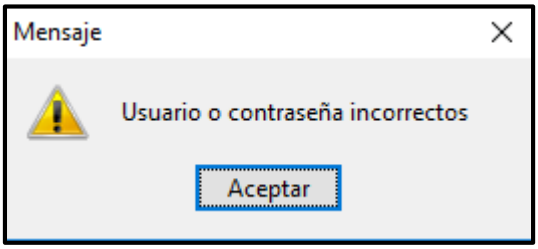

Figura 67. Validación de usuario y contraseña

Elaborado por: (Recalde, 2017)

- Código

```
private void btnIniciarSesionActionPerformed(java.awt.event.ActionEvent evt) {
    String usuario = txtUsuario.getText();
    String contrasenia = new String(pssContrasenia.getPassword());
    UsuarioDP usuarioDP = new UsuarioDP(usuario,contrasenia);
    usuarioDP.setLogeado(1);
    usuarioDP.loggeado();
    usuarioDP = usuarioDP.consultarUsuarioContrasenia();
    if(usuarioDP.getCedUsuario().compareTo("") == 0 && usuarioDP.getContraseniaUsuario().compareTo("") == 0)
    {
        JOptionPane.showMessageDialog(this, "Usuario o contraseña incorrectos", "Mensaje",
                JOptionPane.WARNING_MESSAGE);
        btnIniciarSesion.setBackground(Color.LIGHT_GRAY);
    }
    else
    {
        VentanaPrincipal ventanaPrincipal = new VentanaPrincipal();
        usuarioDP.setContraseniaUsuario("");
        ventanaPrincipal.setUsuarioDP(usuarioDP);
        this.dispose();
        ventanaPrincipal.setVisible(true);
```

Figura 68. Código de Validación de Usuario y Contraseña

Elaborado por: (Recalde, 2017)

3.3.2.2. Cambio de Imágenes

Al momento de pasar el mouse en las diferentes opciones del sistema, las mismas cambian de color para que el usuario pueda identificar fácilmente la opción que puede ser seleccionada.

Sin pasar el mouse

Figura 69. Opción sin pasar el mouse

Elaborado por: (Recalde, 2017)

Al pasar el mouse

Figura 70. Opción al pasar el mouse

Elaborado por: (Recalde, 2017)

- Código

```
private void jLabelUsuariosMouseEntered(java.awt.event.MouseEvent evt) {
    ImageIcon imageIcon = new ImageIcon(getClass().getResource("/IMG/bannerUsuariosSeleccionado.png"));
    jLabelUsuarios.setIcon(imageIcon);
}

private void jLabelUsuariosMouseExited(java.awt.event.MouseEvent evt) {
    ImageIcon imageIcon = new ImageIcon(getClass().getResource("/IMG/bannerUsuarios.png"));
    jLabelUsuarios.setIcon(imageIcon);
```

Figura 71. Código de Cambio de Imagen

Elaborado por: (Recalde, 2017)

3.3.2.3. Validaciones de Campos

Los campos tanto numéricos como de caracteres del GUI se encuentran validados de la siguiente manera:

Tipo de Validación	Validación
Validaciones de ingreso	Tanto en campos numéricos como en el campo para ingresar la cédula y los números de historias clínicas, no se permite el ingreso de caracteres que no sean dígitos. La validación mencionada se la realiza mediante el evento KeyTyped [1] de java.
Longitud de cadenas	Para poder conservar la integridad de los datos, el sistema valida la longitud máxima permitida en cada campo de tipo de caracteres, de acuerdo con lo establecido en la base de datos, de forma que la información que se ingrese sea lógica, además así evitar posibles problemas con la base de datos. De la misma manera, dicha validación se realiza mediante el evento KeyTyped de java.

Tabla 9. Validaciones de campos
Elaborado por: (Recalde, 2017)

- Código validaciones de ingreso

```
private void txtNumHistoriaClinicaKeyTyped(java.awt.event.KeyEvent evt) {
    char caracter = evt.getKeyChar();
    if(((caracter < '0') || (caracter > '9')) && (caracter != '\b' /*corre
    {
       evt.consume();
    }
```

Figura 72. Código Validaciones de Ingreso
Elaborado por: (Recalde, 2017)

[1] Método de eventos tipo KeyListener, este método se ejecuta cada vez que se presiona y se suelta una tecla.

- Código validaciones de longitud de cadena

```
private void txtRemitenciaKeyTyped(java.awt.event.KeyEvent evt) {
    if(txtRemitencia.getText().length() == 50)
        JOptionPane.showMessageDialog(this, "No se admiten más carcateres", "Mensaje", JOptionPane.INFORMATION_MESSAGE);
```

Figura 73. Código Validaciones de Longitud de Cadena

Elaborado por: (Recalde, 2017)

- Aviso de validación de longitud de cadena

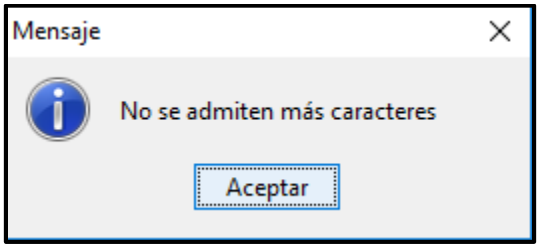

Figura 74. Aviso Validación de Longitud de Cadena

Elaborado por: (Recalde, 2017)

3.3.2.4. Validaciones de Nuevos Registros de Usuarios y Estudiantes

En estos casos, sobre todo se valida que los usuarios o estudiantes nuevos no estén aún en la base de datos mediante su identificación única, siendo la misma la cédula en el caso de usuarios y el número de historia clínica en el caso de estudiantes, dicha validación se la realiza mediante el evento FocusLost[2].

En cada caso, tanto para insertar o modificar, se inicializa un nuevo objeto, UsuarioDP en el caso de usuarios y EstudianteDP en el caso de estudiantes; se utilizan los métodos SET[3] para setear cada uno de los mismos con sus respectivas identificaciones y se procede a utilizar las funciones para verificar la existencia de usuarios o de estudiantes. Se utilizó campos lógicos en la eliminación de los mismos, es decir, cuando se desea eliminar un usuario o un estudiante, no se eliminará por completo su información, caso contrario el estado lógico del campo cambiará, esto se realizó a pedido del cliente,

[2] Método de eventos tipo FocusListener, este método se ejecuta cada vez el componente actual, pierde el foco, es decir cuando se deja de utilizar el componente.
[3] Aquellos que proporcionan acceso a las propiedades de un objeto. Un método SET tiene un tipo de retorno VOID, y toma un parámetro del tipo adecuado para asignar a la variable miembro asociada.

entonces cuando se ingrese una cédula o un número de historia que ya fue ingresado anteriormente, se podrá reestablecer la información del mismo, si el usuario así lo desea.

- Código de validación de existencia

```
private void txtCedulaFocusLost(java.awt.event.FocusEvent evt) {
    usuarioDP = new UsuarioDP();
    String cedula = txtCedula.getText();
    if(insertar){
        if(!cedula.isEmpty()){
            usuarioDP.setCedUsuario(txtCedula.getText());
            if(usuarioDP.existeUsuario())
                JOptionPane.showMessageDialog(this, "El usuario con cédula N° "+txtCedula.getText()+" ya existe",
                        "Mensaje", JOptionPane.WARNING_MESSAGE);
            else if(usuarioDP.existeUsuarioEliminado()){
                int resp = JOptionPane.showConfirmDialog(this, "El usuario con cédula N° "+txtCedula.getText()
                        +" ya ha sido ingresado anteriormente\n¿Desea Restaurarlo?", "Advertencia",
                        JOptionPane.YES_NO_OPTION);
                if (JOptionPane.YES_NO_OPTION == resp){
                    usuarioDP.setEliminadoUsuario(0);
                    usuarioDP.eliminarUsuario();
                    insertar = false;
                    limpiar();
                    cambiarColorNoEditable();
                    btnAceptarIngresar.setVisible(false);
                    btnCancelarIngresar.setVisible(false);
                    noEditable();
                    btnModificarUsuario.setEnabled(true);
                    btnEliminarUsuario.setEnabled(true);
                }
            }
            else if(!usuarioDP.existeUsuario()){
                txtNombres.setEditable(true);
```

Figura 75. Código Validaciones de Existencia en la Base de Datos

Elaborado por: (Recalde, 2017)

- Avisos
  - Aviso de usuario ya existente en la base de datos

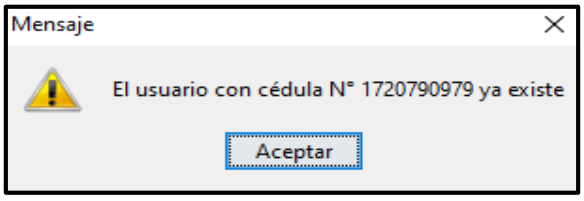

Figura 76. Aviso Validación Usuario Existente

Elaborado por: (Recalde, 2017)

  - Aviso para usuarios anteriormente ya ingresados

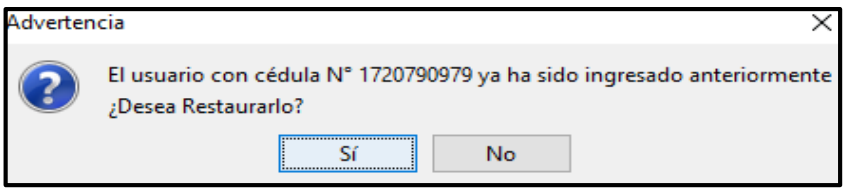

Figura 77. Aviso Validación Usuario Anteriormente Ingresado

Elaborado por: (Recalde, 2017)

3.3.2.5. Validaciones de Eliminación de Usuarios y Estudiantes

Si se desea eliminar un usuario o un estudiante, el sistema presentará una alerta para que el usuario Acepte o Cancele la eliminación del estudiante o usuario escogido. Si el usuario Acepta, se setea al objeto UsuarioDP o EstudianteDP, dependiendo del caso, su identificación, es decir, la cédula en el caso de usuarios o el número de historia clínica en el caso de estudiantes; se setea también el campo lógico de cada caso y se procede a su eliminación.

En el caso de que el usuario ingresado al sistema sea de tipo de rol Administrador y por error desee eliminar a su propio usuario, el sistema presentará una alerta de que no se puede eliminar al usuario que en ese momento se encuentra ingresado al sistema.

- Código de validación de eliminación

```
private void btnEliminarUsuarioActionPerformed(java.awt.event.ActionEvent evt) {
    int fila = jTableUsuarios.getSelectedRow();
    if (fila==-1)
        JOptionPane.showMessageDialog(this, "Seleccione un usuario a eliminar", "Mensaje", JOptionPane.WARNING_MESSAGE);
    else{
        usuarioDP.setCedUsuario(txtCedula.getText());
        if(!usuarioDP.consultarUsuarioLogeado()){
            btnNuevoUsuario.setEnabled(false);
            btnModificarUsuario.setEnabled(false);
            int resp = JOptionPane.showConfirmDialog(this, "¿Está seguro de eliminar el usuario con cédula N° "
                    +txtCedula.getText()+"?",
                "Mensaje", JOptionPane.YES_NO_OPTION);
            if (JOptionPane.YES_NO_OPTION == resp){
                usuarioDP.setCedUsuario(txtCedula.getText());
                usuarioDP.setEliminadoUsuario(1);
                if(usuarioDP.eliminarUsuario()){
                    JOptionPane.showMessageDialog(this, "Usuario eliminado correctamente", "Mensaje",
                            JOptionPane.INFORMATION_MESSAGE);
                    limpiar();
                    usuarioDP.cargarDatos(jTableUsuarios);
                    btnNuevoUsuario.setEnabled(true);
                    btnModificarUsuario.setEnabled(true);
                }
                else
                    JOptionPane.showMessageDialog(this, "Error inesperado, por favor reinicie el sistema", "Mensaje", JO
            }
            else{
```

Figura 78. Código Validación de Eliminación

Elaborado por: (Recalde, 2017)

- Avisos
  - Aviso de alerta de confirmación de usuario a eliminar

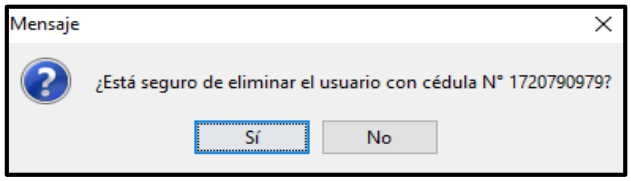

Figura 79. Aviso de Alerta de Confirmación de Usuario a Eliminar

Elaborado por: (Recalde, 2017)

- Aviso de alerta de usuario ingresado al sistema

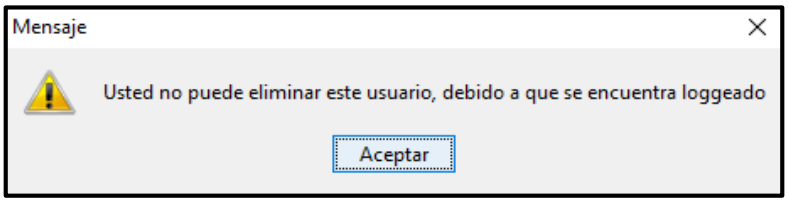

Figura 80. Aviso de Alerta de Usuario Loggeado

Elaborado por: (Recalde, 2017)

3.3.2.6. Validaciones de Presentación de Historias Clínicas

Al momento de abrir el módulo de historias clínicas se presentará una JTable[4] o tabla con los estudiantes no eliminados que al momento existan, la cual al momento de seleccionar un estudiante de la misma y si existen historias clínicas para dicho estudiante, aquí se seteará el id del estudiante seleccionado para proceder a la búsqueda de historias clínicas, de ser así, se cargarán las fechas de las mismas en un jComboBox[5] o seleccionador, a partir de ésto, el usuario podrá seleccionar una fecha para que se le presente la información de la historia clínica correspondiente a dicha fecha, con el seteo de la fecha de historia clínica para buscar la información de la misma.

- Código de presentación de historias clínicas

```
else{
    String cedula = jTableEstudiantes.getValueAt(fila, 0).toString();
    estudianteDP.setnumHistoriaClinicaEstudiante(cedula);
    int id = estudianteDP.obtenerIDEstudiante();
    historiaClinicaDP.setIdEstudiante(id);
    estudianteDP = estudianteDP.consultarEstudiante();
    fN = estudianteDP.getFechaNacimientoEstudiante().toString();
    String genero = "";
    txtCedula.setText(cedula);
    txtNombresApellidos.setText(estudianteDP.getNombresEstudiante()+" "+estudianteDP.getApellidosEstudiante());
    txtLugarNacimiento.setText(estudianteDP.getLugarNacimientoEstudiante());
    txtFechaNacimiento.setText(estudianteDP.getFechaNacimientoEstudiante().toString());
    txtEdad.setText(estudianteDP.getEdadAniosEstudiante()+" años, "+estudianteDP.getEdadMesesEstudiante()+" meses");
    if(estudianteDP.getGeneroEstudiante().equals("F"))
        genero = "Femenino (F)";
    else
        genero = "Masculino (M)";
    txtGenero.setText(estudianteDP.getGeneroEstudiante());
```

Figura 81. Código Presentación de Historias Clínicas

Elaborado por: (Recalde, 2017)

[4] Componente visual de java que nos permite dibujar una tabla.
[5] Lista desplegable hacia abajo que permite elegir entre varias opciones diferentes.

- Avisos
  - Avisos de no existencia de historias clínicas para un estudiante

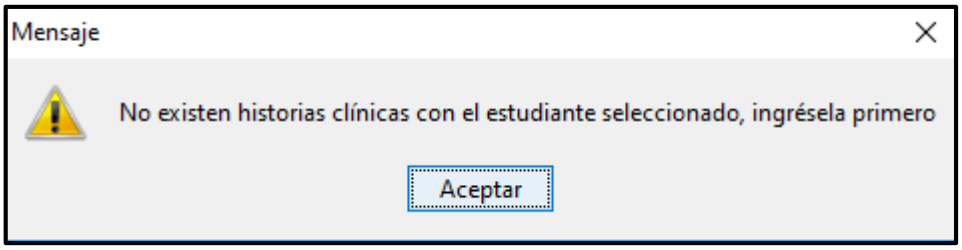

Figura 82. Aviso de no Existencia de Historias Clínicas

Elaborado por: (Recalde, 2017)

  - Aviso de existencia de historias clínicas, para seleccionar una fecha

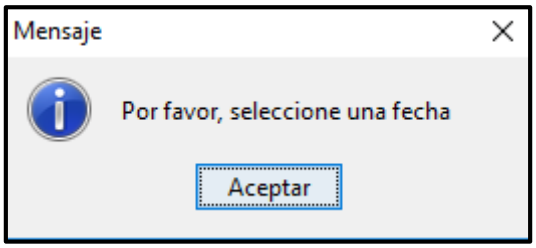

Figura 83. Aviso de Existencia de Historias Clínicas para la Selección de una Fecha

Elaborado por: (Recalde, 2017)

3.3.2.7. Validaciones de Ingreso, Modificación y Eliminación de Historias Clínicas

En el apartado anterior se explicó la presentación de los estudiantes existentes en una tabla, a partir de la selección que el usuario realice se podrá ingresar una nueva historia para el estudiante seleccionado, o bien modificar o eliminar una ficha seleccionada a partir del seleccionador de fechas. En los casos de ingresar y modificar es necesario el ingreso de ciertos campos obligatorios. En todos los casos de advertencias se generan avisos para alertar al usuario.

- Código de ingreso de historias clínicas

```
private void btnNuevaHistoriaActionPerformed(java.awt.event.ActionEvent evt) {
    modificar = false;
    insertar = true;
    int fila = jTableEstudiantes.getSelectedRow();
    if (fila==-1 || fila==jTableEstudiantes.getRowCount()-1)
    JOptionPane.showMessageDialog(this, "Seleccione un estudiante para insertar una nueva historia clínica",
            "Mensaje", JOptionPane.WARNING_MESSAGE);
    else{
        calcularAnios();
        txtEdad.setText(edadAnios+" años "+edadMeses+" mese");
        btnModificarHistoria.setEnabled(false);
        btnEliminarHistoria.setEnabled(false);
        limpiar();
        buttonGroup2.clearSelection();
        cambiarColorEditable();
        editable();
        btnAceptarIngresar.setVisible(true);
        btnCancelarIngresar.setVisible(true);
    }
}
```

Figura 84. Código para Ingreso de Historias Clínicas

Elaborado por: (Recalde, 2017)

- Código de modificación de historias clínicas

```
private void btnModificarHistoriaActionPerformed(java.awt.event.ActionEvent evt) {
    btnNuevaHistoria.setEnabled(false);
    btnEliminarHistoria.setEnabled(false);
    int resp = JOptionPane.showConfirmDialog(this, "¿Está Seguro de Modificar la Historia Clínica Actual?",
            "Advertencia", JOptionPane.YES_NO_OPTION);
    if(resp == JOptionPane.YES_OPTION){
        modificar = true;
        cambiarColorEditable();
        editable();
        btnAceptarModificar.setVisible(true);
        btnCancelarModificar.setVisible(true);
    }
    else{
        btnNuevaHistoria.setEnabled(true);
        btnEliminarHistoria.setEnabled(true);
    }
}
```

Figura 85. Código para Modificación de Historias Clínicas

Elaborado por: (Recalde, 2017)

- Código de eliminación de historias clínicas

```
private void btnEliminarHistoriaActionPerformed(java.awt.event.ActionEvent evt) {
    btnNuevaHistoria.setEnabled(false);
    btnModificarHistoria.setEnabled(false);
    int resp = JOptionPane.showConfirmDialog(this, "¿Está Seguro de Eliminar la Historia Clínica Actual?",
            "Advertencia", JOptionPane.YES_NO_OPTION);
    if(resp == JOptionPane.YES_OPTION){
        eliminar = true;
        historiaClinicaDP.setIdHistoriaClinica(id_H);
        historiaClinicaDP.setEliminadoHistoriaClinica(1);
        if(historiaClinicaDP.eliminarHistoriaClinica()){
            eliminar = false;
            JOptionPane.showMessageDialog(this, "Historia Clínica Eliminada Correctamente", "Mensaje", JOptionPane
            limpiar();
            limpiarDatosPersonales();
            cbFechas.setSelectedIndex(0);
            cbFechas.setEnabled(false);
            btnNuevaHistoria.setEnabled(true);
        }
    }
    else{
        btnNuevaHistoria.setEnabled(true);
        btnModificarHistoria.setEnabled(true);
```

Figura 86. Código para Eliminación de Historias Clínicas

Elaborado por: (Recalde, 2017)

- Avisos
  - Aviso en caso de no haber seleccionado un estudiante y se desee ingresar una nueva historia clínica

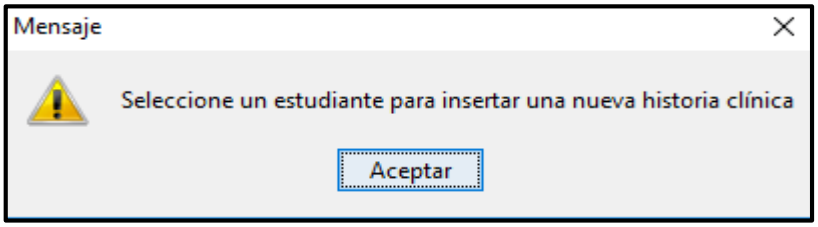

Figura 87. Advertencia para el Ingreso de una Nueva Historia Clínica

Elaborado por: (Recalde, 2017)

  - Aviso de campos obligatorios

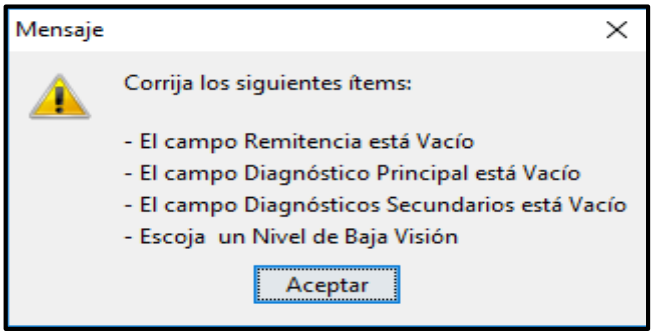

Figura 88. Aviso de Campos Obligatorios

Elaborado por: (Recalde, 2017)

- Aviso de confirmación de datos correctos

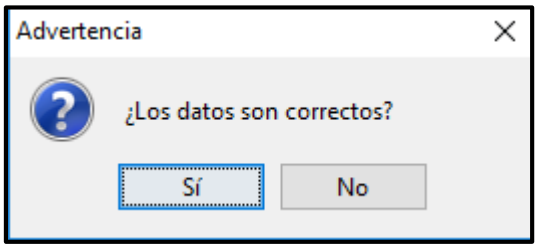

Figura 89. Aviso de Confirmación de Datos Correctos

Elaborado por: (Recalde, 2017)

- Aviso de ingreso correcto de historia clínica

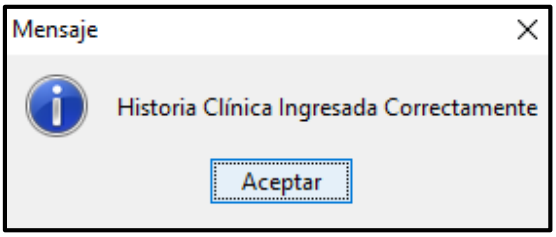

Figura 90. Aviso de Ingreso Correcto de Historias Clínicas

Elaborado por: (Recalde, 2017)

- Aviso de confirmación de modificación de historia clínica

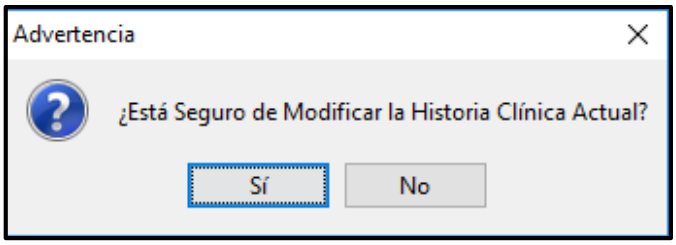

Figura 91. Aviso de Confirmación de Modificación de Historias Clínicas

Elaborado por: (Recalde, 2017)

- Aviso de modificación correcta de historia clínica

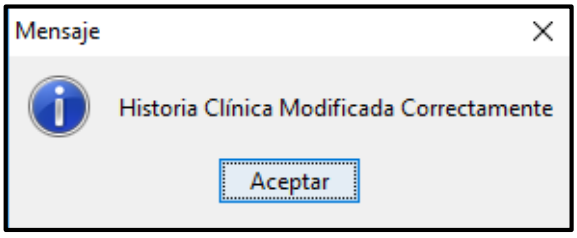

Figura 92. Aviso de Modificación Correcta de Historias Clínicas

Elaborado por: (Recalde, 2017)

- Aviso de confirmación de eliminación de historia clínica

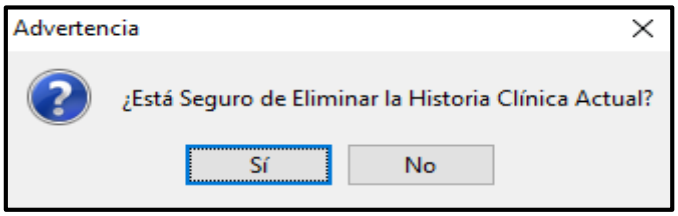

Figura 93. Aviso de Confirmación de Eliminación de Historias Clínicas

Elaborado por: (Recalde, 2017)

- Aviso de eliminación correcta de historia clínica

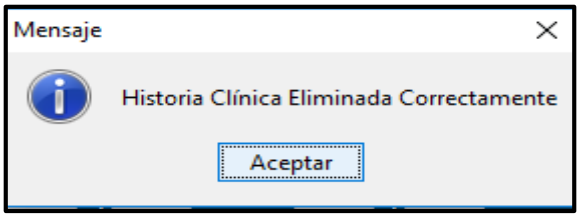

Figura 94. Aviso de Eliminación Correcta de Historias Clínicas

Elaborado por: (Recalde, 2017)

3.3.3. Clases DP y Explicación – Ciclo 1

Las clases DP contienen los diferentes métodos utilizados en el sistema, tales como obtenerID(), obtenerFechas(), etc. Además, contienen los métodos GETTER y SETTER de cada uno de los atributos de cada clase, y los métodos básicos para realizar el CRUD de cada clase.

Para explicar este apartado, se tomó como ejemplo la clase EstudianteDP.

3.3.3.1. Código de Instanciación de Clase EstudianteDP

```
public class EstudianteDP {

    private int idEstudiante;
    private String numHistoriaClinicaEstudiante;
    private String nombresEstudiante;
    private String apellidosEstudiante;
    private String lugarNacimientoEstudiante;
    private Date fechaNacimientoEstudiante;
    private int edadAniosEstudiante;
    private int edadMesesEstudiante;
    private String generoEstudiante;
```

Figura 95. Código de Instanciación de Clase EstudianteDP

Elaborado por: (Recalde, 2017)

3.3.3.2. Código de Constructor de Clase EstudianteDP

```
public EstudianteDP(int idEstudiante, String numHistoriaClinicaEstudiante, String nombresEstudiante,
        String apellidosEstudiante, String lugarNacimientoEstudiante, Date fechaNacimientoEstudiante,
        int edadAniosEstudiante, int edadMesesEstudiante, String generoEstudiante,
        String direccionDomiciliariaEstudiante, String telefonoDomicilioEstudiante, int eliminadoEstudiante) {
    this.idEstudiante = idEstudiante;
    this.numHistoriaClinicaEstudiante = numHistoriaClinicaEstudiante;
    this.nombresEstudiante = nombresEstudiante;
    this.apellidosEstudiante = apellidosEstudiante;
    this.lugarNacimientoEstudiante = lugarNacimientoEstudiante;
    this.fechaNacimientoEstudiante = fechaNacimientoEstudiante;
    this.edadAniosEstudiante = edadAniosEstudiante;
    this.edadMesesEstudiante = edadMesesEstudiante;
    this.generoEstudiante = generoEstudiante;
    this.direccionDomiciliariaEstudiante = direccionDomiciliariaEstudiante;
    this.telefonoDomicilioEstudiante = telefonoDomicilioEstudiante;
    this.eliminadoEstudiante = eliminadoEstudiante;
```

Figura 96. Código de Constructor de Clase EstudianteDP

Elaborado por: (Recalde, 2017)

3.3.3.3. Código de GETTER y SETTER de un Atributo de Clase EstudianteDP

```
public String getNombresEstudiante() {
    return nombresEstudiante;
}

public void setNombresEstudiante(String nombresEstudiante) {
    this.nombresEstudiante = nombresEstudiante;
```

Figura 97. Código de GETTER y SETTER de un Atributo de Clase EstudianteDP

Elaborado por: (Recalde, 2017)

3.3.3.4. Códigos de Métodos Básicos de CRUD hacia Clase EstudianteMD

3.3.3.4.1 Código de Ingreso de Estudiante

```
public boolean ingresarEstudiante(){
    this.estudianteMD = new EstudianteMD(this);
    return estudianteMD.ingresarEstudiante();
```

Figura 98. Código de Ingreso de Estudiante en Clase EstudianteDP

Elaborado por: (Recalde, 2017)

3.3.3.4.2 Código de Modificación de Estudiante

```
public boolean modificarEstudiante(){
    this.estudianteMD = new EstudianteMD(this);
    return estudianteMD.modificarEstudiante();
```

Figura 99. Código de Modificación de Estudiante en Clase EstudianteDP

Elaborado por: (Recalde, 2017)

3.3.3.4.3 Código de Eliminación de Estudiante

```
public boolean eliminarEstudiante(){
    this.estudianteMD = new EstudianteMD(this);
    return estudianteMD.eliminarEstudiante();
```

Figura 100. Código de Eliminación de Estudiante en Clase EstudianteDP

Elaborado por: (Recalde, 2017)

3.3.3.4.4 Código de Consulta de Estudiante

```
public EstudianteDP consultarEstudiante(){
    this.estudianteMD = new EstudianteMD(this);
    return estudianteMD.consultarEstudiante();
```

Figura 101. Código de Consulta de Estudiante en Clase EstudianteDP

Elaborado por: (Recalde, 2017)

3.3.3.4.5 Código de Verificación de Existencia de Estudiante

```
public boolean existeEstudiante(){
    this.estudianteMD = new EstudianteMD(this);
    return estudianteMD.existeEstudiante();
```

Figura 102. Código de Verificación de Existencia de Estudiante en Clase EstudianteDP

Elaborado por: (Recalde, 2017)

3.3.3.4.6 Código de Verificación de Existencia de Estudiante Eliminado

```
public boolean existeEstudianteEliminado(){
    this.estudianteMD = new EstudianteMD(this);
    return estudianteMD.existeEstudianteEliminado();
```

Figura 103. Código de Verificación de Existencia de Estudiante Eliminado en Clase EstudianteDP

Elaborado por: (Recalde, 2017)

3.3.3.4.7 Código de Carga de Datos en JTable

```
public void cargarDatos(JTable tabla){
    this.estudianteMD = new EstudianteMD(this);
    estudianteMD.cargarDatos(tabla);
```

Figura 104. Código de Carga de Datos en JTable en Clase EstudianteDP

Elaborado por: (Recalde, 2017)

3.3.3.4.8 Código de Obtención de ID de Estudiante

```
public int obtenerIDEstudiante(){
    this.estudianteMD = new EstudianteMD(this);
    return estudianteMD.obtenerIDEstudiante();
```

Figura 105. Código de Obtención de ID de Estudiante EstudianteDP

Elaborado por: (Recalde, 2017)

3.3.4. Clases MD y Explicación – Ciclo 1

Las clases MD poseen su objeto de conexión a la base de datos 'ConexionBDD', mismo que se encarga de gestionar la conexión con la base de datos a utilizarse. Además, cuenta con los métodos básicos para realizar el CRUD en cada clase, y otros métodos requeridos y se encuentran especificados en las clases DP. Igual que en el caso anterior se tomó como ejemplo la clase EstudianteMD. Y se tomará también en cuenta la clase de conexión a la base de datos ConexionBDD para su explicación.

3.3.4.1. Clase EstudianteMD

3.3.4.1.1 Código de Instanciación de Clase EstudianteMD

```
public class EstudianteMD {
    EstudianteDP estudianteDP;
    ConexionBDD conexionBDD;
    Statement statement;
```

Figura 106. Código de Instanciación de Clase EstudianteMD

Elaborado por: (Recalde, 2017)

3.3.4.1.2 Código de Constructor de Clase EstudianteMD

```
public EstudianteMD(EstudianteDP estudianteDP) {
    this.estudianteDP = estudianteDP;
    conexionBDD = new ConexionBDD();
```

Figura 107. Código de Constructor de Clase EstudianteMD

Elaborado por: (Recalde, 2017)

3.3.4.1.3 Códigos de Métodos Básicos de CRUD en Clase EstudianteMD

- Código de Ingreso de Estudiante

```
public boolean ingresarEstudiante(){
    boolean retorno = false;
    this.statement = conexionBDD.crearConexion();
    try {
        String query = "INSERT INTO estudiante(numhistoriaclinicaestudiante, nombresestudiante, apellidosestudiante, "
                + "lugarnacimientoestudiante, fechanacimientoestudiante, edadaniosestudiante, "
                + "edadmesesestudiante, generoestudiante, direccciondomiciliariaestudiante, "
                + "telefonodomicilioestudiante, eliminadoestudiante, provinciaestudiante, cantonestudiante) "
                + "VALUES ('"+estudianteDP.getnumHistoriaClinicaEstudiante()+"', '"+estudianteDP.getNombresEstudiante()+"'
                +estudianteDP.getApellidosEstudiante()+"', '"+estudianteDP.getLugarNacimientoEstudiante()+"', '"
                +estudianteDP.getFechaNacimientoEstudiante()+"', "+estudianteDP.getEdadAniosEstudiante()+", "
                +estudianteDP.getEdadMesesEstudiante()+", '"+estudianteDP.getGeneroEstudiante()+"', '"
                +estudianteDP.getDireccionDomiciliariaEstudiante()+"', '"
                +estudianteDP.getTelefonoDomicilioEstudiante()+"', "+estudianteDP.getEliminadoEstudiante()+", '"
                +estudianteDP.getProvinciaestudiante()+"', '"+estudianteDP.getCantonestudiante()+"')";
        statement.executeUpdate(query);
        retorno = true;
        conexionBDD.cerrarConexion();
    } catch (SQLException ex) {
        System.out.println(ex);
        retorno = false;
    }
    return retorno;
```

Figura 108. Código de Ingreso de Estudiante en Clase EstudianteMD

Elaborado por: (Recalde, 2017)

- Código de Modificación de Estudiante

```
public boolean modificarEstudiante(){
    boolean retorno = false;
    this.statement = conexionBDD.crearConexion();
    try {
        String query = "UPDATE estudiante SET nombresestudiante='"+estudianteDP.getNombresEstudiante()
                +"', apellidosestudiante='"+estudianteDP.getApellidosEstudiante()
                +"', lugarnacimientoestudiante='"+estudianteDP.getLugarNacimientoEstudiante()
                +"', fechanacimientoestudiante='"+estudianteDP.getFechaNacimientoEstudiante()
                +"', edadaniosestudiante="+estudianteDP.getEdadAniosEstudiante()
                +", edadmesesestudiante="+estudianteDP.getEdadMesesEstudiante()
                +", generoestudiante='"+estudianteDP.getGeneroEstudiante()
                +"', direccciondomiciliariaestudiante='"+estudianteDP.getDireccionDomiciliariaEstudiante()
                +"', telefonodomicilioestudiante='"+estudianteDP.getTelefonoDomicilioEstudiante()
                +"', eliminadoestudiante=0 "
                +", provinciaestudiante='"+estudianteDP.getProvinciaestudiante()
                +"', cantonestudiante='"+estudianteDP.getCantonestudiante()
                +"' WHERE numhistoriaclinicaestudiante='"+estudianteDP.getnumHistoriaClinicaEstudiante()+"'";
        statement.executeUpdate(query);
        retorno = true;
        conexionBDD.cerrarConexion();
    } catch (SQLException ex) {
        System.out.println(ex);
        retorno = false;
    }
    return retorno;
```

Figura 109. Código de Modificación de Estudiante en Clase EstudianteMD

Elaborado por: (Recalde, 2017)

- Código de Eliminación de Estudiante

```
public boolean eliminarEstudiante(){
    boolean retorno = false;
    this.statement = conexionBDD.crearConexion();
    try {
        String query = "UPDATE estudiante SET eliminadoestudiante="+estudianteDP.getEliminadoEstudiante()
                +" WHERE numhistoriaclinicaestudiante='"+estudianteDP.getnumHistoriaClinicaEstudiante()+"'";
        statement.executeUpdate(query);
        retorno = true;
        conexionBDD.cerrarConexion();
    } catch (SQLException ex) {
        System.out.println(ex);
        retorno = false;
    }
    return retorno;
```

Figura 110. Código de Eliminación de Estudiante en Clase EstudianteMD

Elaborado por: (Recalde, 2017)

- Código de Consulta de Estudiante

```
public EstudianteDP consultarEstudiante(){
    EstudianteDP retorno = new EstudianteDP();
    this.statement = conexionBDD.crearConexion();
    try {
        String query = "SELECT * FROM estudiante WHERE numhistoriaclinicaestudiante = '"
                +estudianteDP.getnumHistoriaClinicaEstudiante()+"' and eliminadoestudiante = 0";
        ResultSet resultSet = statement.executeQuery(query);
        while(resultSet.next())
        {
            retorno.setnumHistoriaClinicaEstudiante(resultSet.getString("numhistoriaclinicaestudiante"));
            retorno.setNombresEstudiante(resultSet.getString("nombresestudiante").trim());
            retorno.setApellidosEstudiante(resultSet.getString("apellidosestudiante").trim());
            retorno.setLugarNacimientoEstudiante(resultSet.getString("lugarnacimientoestudiante").trim());
            retorno.setFechaNacimientoEstudiante(resultSet.getDate("fechanacimientoestudiante"));
            retorno.setEdadAniosEstudiante(resultSet.getInt("edadaniosestudiante"));
            retorno.setEdadMesesEstudiante(resultSet.getInt("edadmesesestudiante"));
            retorno.setGeneroEstudiante(resultSet.getString("generoestudiante").trim());
            retorno.setDireccionDomiciliariaEstudiante(resultSet.getString("direccciondomiciliariaestudiante").trim());
            retorno.setTelefonoDomicilioEstudiante(resultSet.getString("telefonodomicilioestudiante").trim());
            retorno.setProvinciaestudiante(resultSet.getString("provinciaestudiante"));
            retorno.setCantonestudiante(resultSet.getString("cantonestudiante"));
        }
         conexionBDD.cerrarConexion();
        return retorno;
    } catch (SQLException ex) {
        System.out.println(ex);
    }
    return retorno;
```

Figura 111. Código de Consulta de Estudiante en Clase EstudianteMD

Elaborado por: (Recalde, 2017)

- Código de Verificación de Existencia de Estudiante

```
public boolean existeEstudiante(){
    boolean retorno = true;
    this.statement = conexionBDD.crearConexion();
    String cedula = "";
    try {
        String query = "SELECT * FROM estudiante WHERE numhistoriaclinicaestudiante = '"
                +estudianteDP.getnumHistoriaClinicaEstudiante()+"' and eliminadoestudiante = 0";
        ResultSet resultSet = statement.executeQuery(query);
        while(resultSet.next())
        {
            cedula = resultSet.getString("numhistoriaclinicaestudiante");
        }
        conexionBDD.cerrarConexion();
        if(cedula.equals(""))
            return false;
    } catch (SQLException ex) {
        System.out.println(ex);
    }
    return retorno;
}
```

Figura 112. Código de Verificación de Existencia de Estudiante en Clase EstudianteMD

Elaborado por: (Recalde, 2017)

- Código de Verificación de Existencia de Estudiante Eliminado

```
public boolean existeEstudianteEliminado(){
    boolean retorno = true;
    this.statement = conexionBDD.crearConexion();
    String cedula = "";
    try {
        String query = "SELECT * FROM estudiante WHERE numhistoriaclinicaestudiante = '"
                +estudianteDP.getnumHistoriaClinicaEstudiante()+"' and eliminadoestudiante = 1";
        ResultSet resultSet = statement.executeQuery(query);
        while(resultSet.next())
        {
            cedula = resultSet.getString("numhistoriaclinicaestudiante");
        }
        conexionBDD.cerrarConexion();
        if(cedula.equals(""))
            return false;
    } catch (SQLException ex) {
        System.out.println(ex);
    }
    return retorno;
}
```

Figura 113. Código de Verificación de Existencia de Estudiante Eliminado en Clase EstudianteMD

Elaborado por: (Recalde, 2017)

- Código de Carga de Datos en JTable

```
public void cargarDatos(JTable tabla){
    x = 0;
    String[] columnas = {"N° Historia Clínica", "Nombres","Apellidos","Fecha Nacimiento", "Edad", "Género"};
    String [][] datos = {{"",""}};
    modelo = new DefaultTableModel(datos, columnas);
    tabla.setModel(modelo);
    this.statement = conexionBDD.crearConexion();
    try
    {
        String query = "SELECT * FROM estudiante WHERE eliminadoestudiante = 0 ORDER BY idestudiante";
        ResultSet resultSet = statement.executeQuery(query);
        while(resultSet.next())
        {
            String cedula = resultSet.getString("numhistoriaclinicaestudiante").trim();
            String nombres = resultSet.getString("nombresestudiante").trim();
            String apellidos = resultSet.getString("apellidosestudiante").trim();
            String lugarNacimiento = resultSet.getString("lugarnacimientoestudiante").trim();
            Date fechaNacimiento = resultSet.getDate("fechanacimientoestudiante");
            int edad = resultSet.getInt("edadaniosestudiante");
            String genero = resultSet.getString("generoestudiante");
            modelo.insertRow(x++, new Object[]{cedula,nombres,apellidos,fechaNacimiento,edad,genero});
        }
        conexionBDD.cerrarConexion();
    }
    catch(SQLException e)
```

Figura 114. Código de Carga de Datos en JTable en Clase EstudianteMD

Elaborado por: (Recalde, 2017)

- Código de Obtención de ID de Estudiante

```
public int obtenerIDEstudiante(){
    int retorno = -1;
    this.statement = conexionBDD.crearConexion();
    try {
        String query = "SELECT idestudiante FROM estudiante WHERE numhistoriaclinicaestudiante = '"
                +estudianteDP.getnumHistoriaClinicaEstudiante()+"' AND eliminadoestudiante = 0";
        ResultSet resultSet = statement.executeQuery(query);
        while(resultSet.next())
        {
            retorno = resultSet.getInt("idestudiante");
        }
        conexionBDD.cerrarConexion();
    } catch (SQLException ex) {
        System.out.println(ex);
    }
    return retorno;
}
```

Figura 115. Código de Obtención de ID de Estudiante en Clase EstudianteMD

Elaborado por: (Recalde, 2017)

3.3.4.2. Clase de Conexión a la Base de Datos

En esta clase se gestiona la conexión a la base de datos, definiendo así el driver encargado de realizar la conexión, la IP del servidor, el puerto, el usuario y la contraseña que están asociados a la base de datos.

3.3.4.2.1 Código de Instanciación de Clase ConexionBDD

```
public class ConexionBDD {
    private Connection conexion;
    private Statement statement;
```

Figura 116. Código de Instanciación de Clase ConexionBDD

Elaborado por: (Recalde, 2017)

3.3.4.2.2 Código de Método de Conexión a Base de Datos

```
public Statement crearConexion(){
    String driver = "org.postgresql.Driver"; //nombre de driver
    String conexString = "jdbc:postgresql://localhost:5432/FundacionTesis"; //nombr
    String usuario = "postgres";
    String contrasenia = "admin";

    try {
        cerrarConexion();
        Class.forName(driver);
        conexion = DriverManager.getConnection(conexString, usuario, contrasenia);
        conexion.setAutoCommit(false);
        statement = conexion.createStatement();
        //Si la conexion fue realizada con exito, muestra el sgte mensaje.
        System.out.println("Conexión a la base de datos realizada con exito!");
    } catch (SQLException | ClassNotFoundException ex) {
        System.out.println("Error al conectar a la base de datos");
    }
    return statement;
}
```

Figura 117. Código de Método de Conexión a la Base de Datos en Clase ConexionBDD

Elaborado por: (Recalde, 2017)

3.3.4.2.3 Código de Método de Cierre de Conexión con Base de Datos

```
public void cerrarConexion() throws SQLException{
    if (statement != null) {
        statement.close();
        statement = null;
    }
    if (conexion != null) {
        conexion.commit();
        conexion.close();
        conexion = null;
```

Figura 118. Código de Método de Cierre de Conexión con Base de Datos en Clase ConexionBDD

Elaborado por: (Recalde, 2017)

3.4. Fase de Transición – Ciclo 1

En este apartado se presentará las pruebas realizadas en el sistema, y su correspondiente informe firmado por el cliente, asegurando que el producto final funciona de manera correcta.

3.4.1. Pruebas de Caja Blanca – Ciclo 1

Conocidas también como pruebas estructurales. Con el conocimiento del código y siguiendo su estructura lógica, se consigue diseñar pruebas destinadas a comprobar que el código hace correctamente lo que el diseño de bajo nivel indica, y otras pruebas que logren demostrar que no se comporte debidamente ante determinadas situaciones (Luna, 2009).

Entradas	Resultados Esperados	Caso de Uso	Estado
Despliegue de ventanas	Despliegue de Ventanas correctamente		☑
Verificación de inicio de sesión correcto	Sistema valida que el usuario y la contraseña ingresadas coincidan con los de la base de datos	RF1	☑
Verificación de ingreso de usuarios, estudiantes e historias clínicas	Sistema almacena datos correctamente	RF1.1 RF2.1 RF3.1 RF4.1 RF5.1	☑
Verificación de modificación de usuarios, estudiantes e historias clínicas	Sistema modifica datos correctamente	RF1.2 RF2.2 RF3.2 RF4.2 RF5.2	☑
Verificación de eliminación de usuarios, estudiantes e historias clínicas	Sistema elimina datos correctamente	RF1.3 RF2.3 RF3.3 RF4.3 RF5.3	☑
Verificación de visualización de datos	Sistema realiza la visualización de datos correctamente	RF1.4 RF2.4 RF3.4 RF4.4 RF5.4	☑

Tabla 10. Pruebas de Caja Blanca – Ciclo 1

Elaborado por: (Recalde, 2017)

3.4.2. Pruebas de Caja Negra – Ciclo 1

También conocidas como pruebas funcionales. Tomando en cuenta los requisitos funcionales de muy alto nivel se pueden diseñar este tipo de pruebas, que se aplican sobre el sistema sin la necesidad de conocer la estructura interna del mismo. Se emplea un determinado conjunto de datos de entrada y se observa las salidas producidas para determinar si la función se está ejecutando correctamente. Herramientas básicas de este tipo de pruebas son observar la funcionalidad y contrastar con su especificación (Luna, 2009).

Entradas	Resultados Esperados	Caso de Uso	Estado
Verificación de dato numérico de cédula y número de historia	Sistema impide el ingreso de caracteres no numéricos		☑
Verificación de datos textuales	Sistema verifica la longitud de los campos textuales con el fin de no sobrepasar a lo especificado en la base de datos y poder mantener la integridad de los datos		☑
Verificación de existencias de estudiantes antes de ingresar historias clínicas	Sistema verifica la existencia de estudiantes antes de ingresar historias clínicas	RF3 RF4 RF5	☑
Verificación de existencia de historias clínicas antes de la posibilidad de modificar o eliminar las mismas	Sistema verifica la existencia de historias clínicas antes de poder modificarla o eliminarla	RF3 RF4 RF5	☑

Tabla 11. Pruebas de Caja Negra – Ciclo 1

Elaborado por: (Recalde, 2017)

3.4.3. Informe de Pruebas del Sistema

Plan de Pruebas del Sistema – Ciclo 1

1.1. RF1. Gestionar Usuarios

Entradas	Resultados Esperados	Caso de Uso	Resultado
Selección de opción Gestión de Usuarios	Despliegue de Ventana de Usuarios	RF1	✓
Verificación de cédula	Sistema valida que la cédula sea correcta	RF1	✓
Verificación de cédulas existentes	Sistema valida que la cédula ingresada exista en la base de datos	RF1	✓
Verificación de elección de rol de usuario	Sistema valida que se seleccione un tipo de rol para un usuario	RF1	✓
Verificación de contraseña	Sistema valida que las contraseñas ingresadas coincidan	RF1	✓
Verificación de usuario loggeado	Sistema valida que el usuario a eliminar no se encuentre loggeado	RF1	✓
Ingresar Usuario	Datos almacenados correctamente	RF1.1	✓
Modificar Usuario	Datos modificados correctamente	RF1.2	✓
Eliminar Usuario	Datos eliminados correctamente	RF1.3	✓
Consultar Usuario	Visualización de datos	RF1.4	✓

Firma Representante Fundación "Mariana de Jesús"

Firma Estudiante PUCE
Lilian Recalde

Figura 119. Informe de Pruebas del Sistema 1 de 3 – Ciclo 1

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 6. Pruebas del Sistema → 1. Pruebas del Sistema – Ciclo 1, que se encuentra en el CD.

1.2. RF2. Gestionar Estudiantes

Entradas	Resultados Esperados	Caso de Uso	Resultado
Selección de opción Gestión de Estudiantes	Despliegue de Ventana de Estudiantes	RF2	✓
Verificación de número de historia clínica	Sistema valida que el número de historia clínica sea correcto	RF2	✓
Verificación de historias clínicas existentes	Sistema valida que el número de historia clínica ingresada exista en la base de datos	RF2	✓
Verificación de elección de género	Sistema valida que se seleccione un tipo de género para el estudiante	RF2	✓
Ingresar Estudiante	Datos almacenados correctamente	RF2.1	✓
Modificar Estudiante	Datos modificados correctamente	RF2.2	✓
Eliminar Estudiante	Datos eliminados correctamente	RF2.3	✓
Consultar Estudiante	Visualización de datos	RF2.4	✓

FUNDACIÓN "AMIGA DE LOS CIEGOS" Instituto Especial "MARIANA DE JESÚS" Quito - 1953

Firma Representante Fundación
"Mariana de Jesús"

Recalde C

Firma Estudiante PUCE
Lilian Recalde

Figura 120. Informe de Pruebas del Sistema 2 de 3 – Ciclo 1

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 6. Pruebas del Sistema → 1. Pruebas del Sistema – Ciclo 1, que se encuentra en el CD.

1.3. RF3. Gestionar Historias Clínicas - RF4. Gestionar Diagnósticos – RF5. Gestionar Prescripciones

Entradas	Resultados Esperados	Caso de Uso	Resultado
Selección de opción Gestión de Historias Clínicas	Despliegue de Ventana de Historias Clínicas	RF3 RF4 RF5	✓
Verificación de existencia de historias clínicas de estudiante seleccionado	Sistema valida que el estudiante seleccionado posea historias clínicas	RF3	✓
Verificación de historias clínicas existentes	Sistema valida que el número de historia clínica ingresada exista en la base de datos	RF3	✓
Verificación de datos obligatorios	Sistema valida que los campos obligatorios se encuentren llenos	RF3 RF4 RF5	✓
Verificación de longitud de cadenas	Sistema verifica que la longitud de los datos no sobrepase el tamaño especificado en la BDD	RF3 RF4 RF5	✓
Ingresar Historia Clínica	Datos almacenados correctamente	RF3.1 RF4.1 RF5.1	✓
Modificar Historia Clínica	Datos modificados correctamente	RF3.1 RF4.1 RF5.1	✓
Eliminar Historia Clínica	Datos eliminados correctamente	RF3.3	✓
Consultar Historia Clínica	Visualización de datos	RF3.1 RF4.1 RF5.1	✓

FUNDACIÓN "AMIGA DE LOS CIEGOS" Instituto Especial "MARIANA DE JESÚS" Quito 1983

Firma Representante Fundación "Mariana de Jesús"

Lili Recalde

Firma Estudiante PUCE
Lilian Recalde

Figura 121. Informe de Pruebas del Sistema 3 de 3 – Ciclo 1

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 6. Pruebas del Sistema → 1. Pruebas del Sistema – Ciclo 1, que se encuentra en el CD.

Capítulo IV Caso de Estudio: Instituto Especial para Niños "Mariana de Jesús" - Ciclo 2

En este capítulo se realizará las clases y procesos correspondientes al segundo ciclo con la metodología de desarrollo de software RUP.

4.1. Fase de Inicio – Ciclo 2

4.1.1. Estrategia

El presente proyecto constará de dos ciclos de desarrollo siguiendo la metodología de desarrollo de software RUP. En el primer ciclo serán realizados los principales módulos que el sistema dispondrá, siendo los mismos:

- La gestión de usuarios
- La gestión de estudiantes
- La gestión de historias clínicas
- La gestión de diagnósticos
- La gestión de prescripciones

Se ejecutarán reuniones con el cliente para verificar el avance del proyecto.

En el segundo ciclo serán realizados los dos módulos restantes, siendo éstos:

- La gestión de datos estadísticos
- La gestión de reportes

Dicho proyecto será desarrollado utilizando el IDE de Java NetBeans 8.2 y la base de datos a utilizar será PostgreSQL.

4.1.2. Especificación de Requerimientos – Ciclo 2

4.1.2.1. Requerimientos Funcionales – Ciclo 2

Los requerimientos funcionales serán los siguientes:

RF1. El sistema permitirá gestionar usuarios
RF2. El sistema permitirá gestionar estudiantes
RF3. El sistema permitirá gestionar historias clínicas
RF4. El sistema permitirá gestionar diagnósticos
RF5. El sistema permitirá gestionar prescripciones
RF6. El sistema permitirá gestionar datos estadísticos
RF7. El sistema permitirá gestionar reportes

Los requerimientos funcionales mencionados anteriormente son representados en el diagrama general de la Figura 122, y serán detallados más adelante con el uso de los diagramas de caso de uso de UML (*véase la sección* 2.5.1.2).

4.1.2.1.1 Diagrama General de Casos de Uso – Ciclo 2

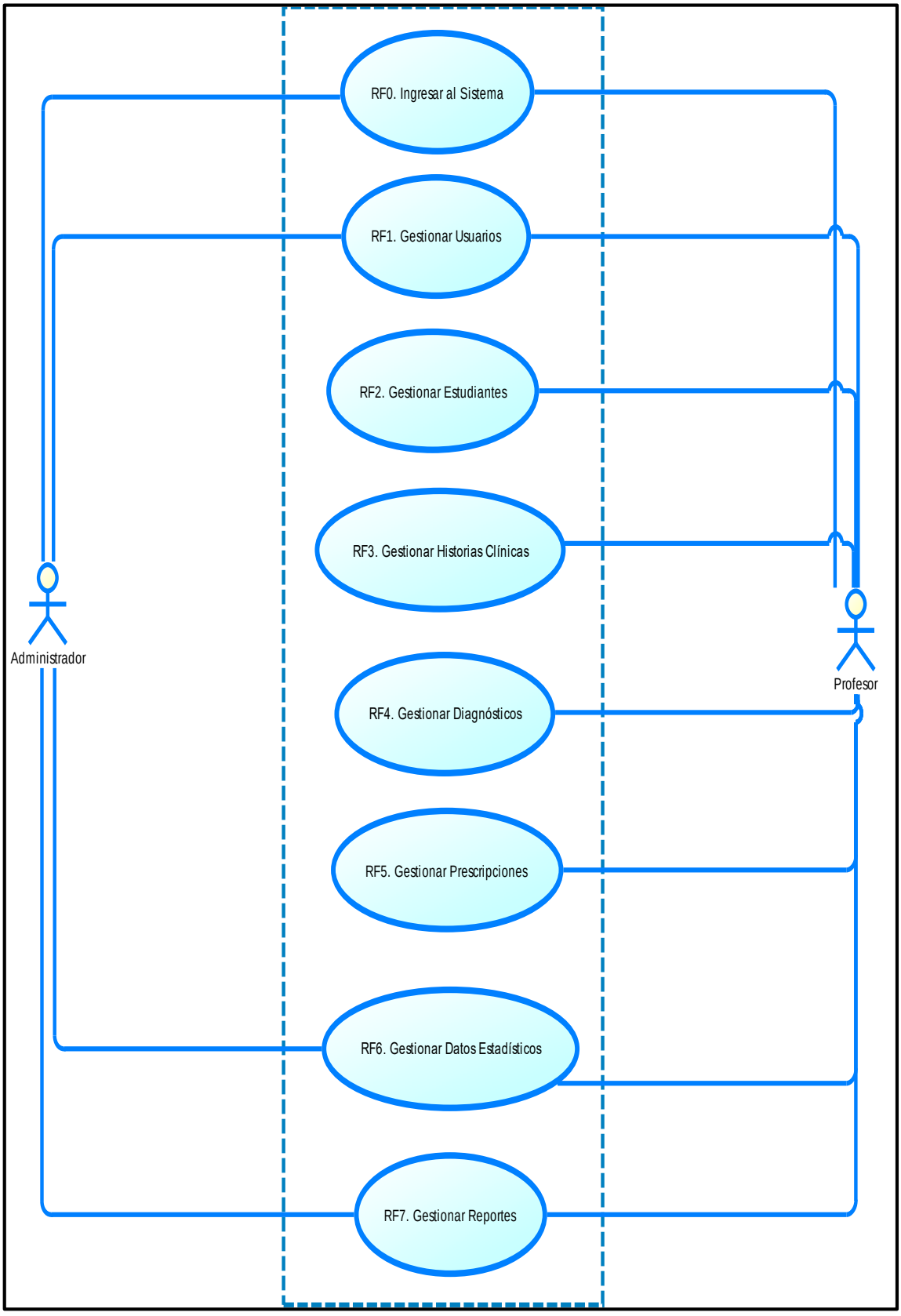

Figura 122. Diagrama General de Casos de Uso – Ciclo 2

Elaborado por: (Recalde, 2017)

4.1.2.1.2 Diagramas Específicos – Siguiente Nivel

RF6. Gestionar Datos Estadísticos

Descripción: El administrador y el profesor serán capaces de poder consultar los datos estadísticos. Dichos datos estadísticos son el resultado de la recopilación de la información que se encuentre en ciertos campos de las historias clínicas, tanto en los diagnósticos como en las prescripciones.

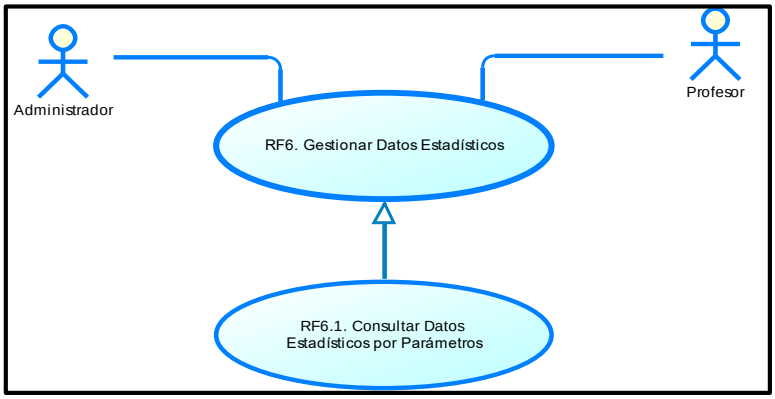

Figura 123. Diagrama Específico de Caso de Uso RF6. Gestionar Datos Estadísticos
Elaborado por: (Recalde, 2017)

RF7. Gestionar Reportes

Descripción: El administrador y el profesor serán capaces de poder generar diferentes tipos de reportes por parámetro.

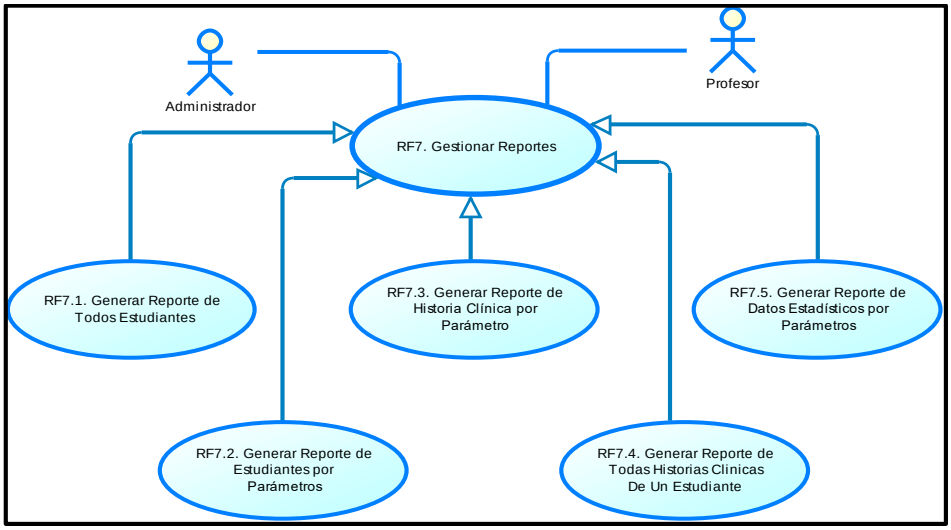

Figura 124. Diagrama Específico de Caso de Uso RF7. Gestionar Reportes
Elaborado por: (Recalde, 2017)

4.1.2.2. Otros Requerimientos

Los requerimientos se mantienen como los explicados anteriormente en el apartado 3.1.2.2

4.2. Fase de Elaboración – Ciclo 2

4.2.1. Diagrama de Clases – Ciclo 2

En el siguiente apartado se muestran los diagramas de clases en las tres capas que existirán en la arquitectura del sistema, basado en lo mencionado en el apartado 1.6, se dispone de la capa de presentación, la capa de dominio del problema y la capa de manejo de datos.

4.2.1.1. Capa de Manejo de Datos – Ciclo 2

4.2.1.1.1 Diagrama Conceptual de Clases de Manejo de Datos – Ciclo 2

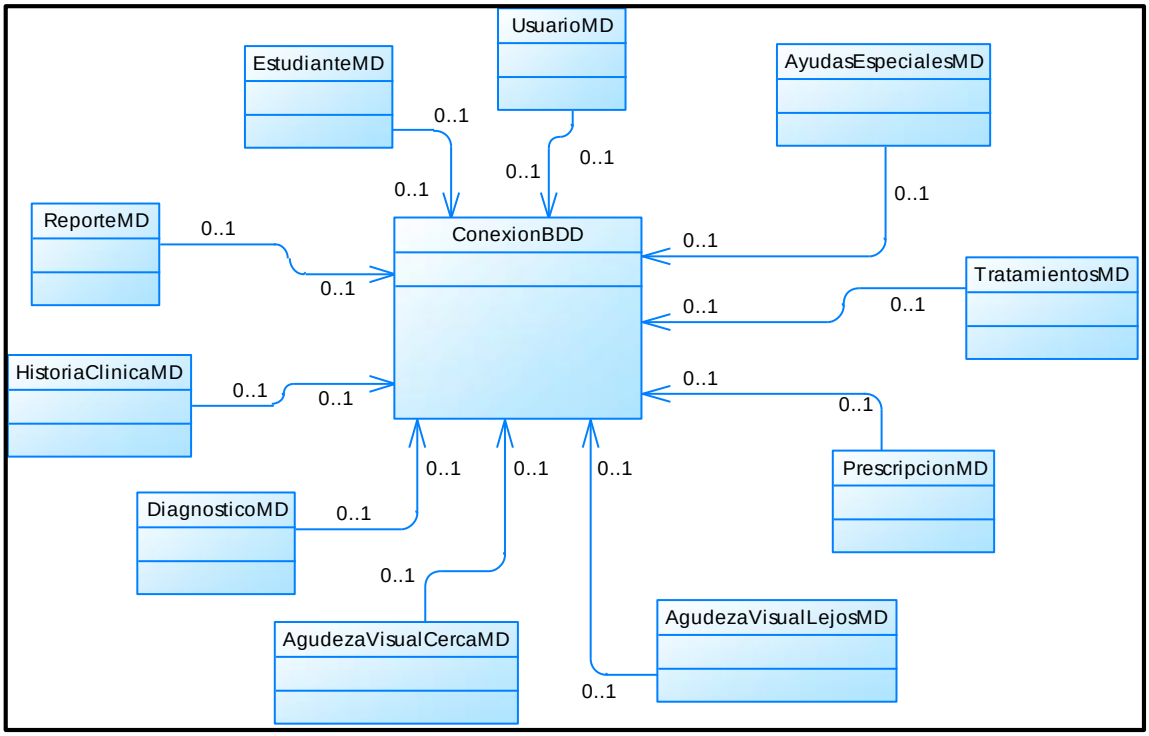

Figura 125. Diagrama Conceptual de Clases de Manejo de Datos – Ciclo 2

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 4. Diagramas de Clases → 1. Manejo de Datos, que se encuentra en el CD.

4.2.1.1.2 Diagrama de Clases de Manejo de Datos – Ciclo 2

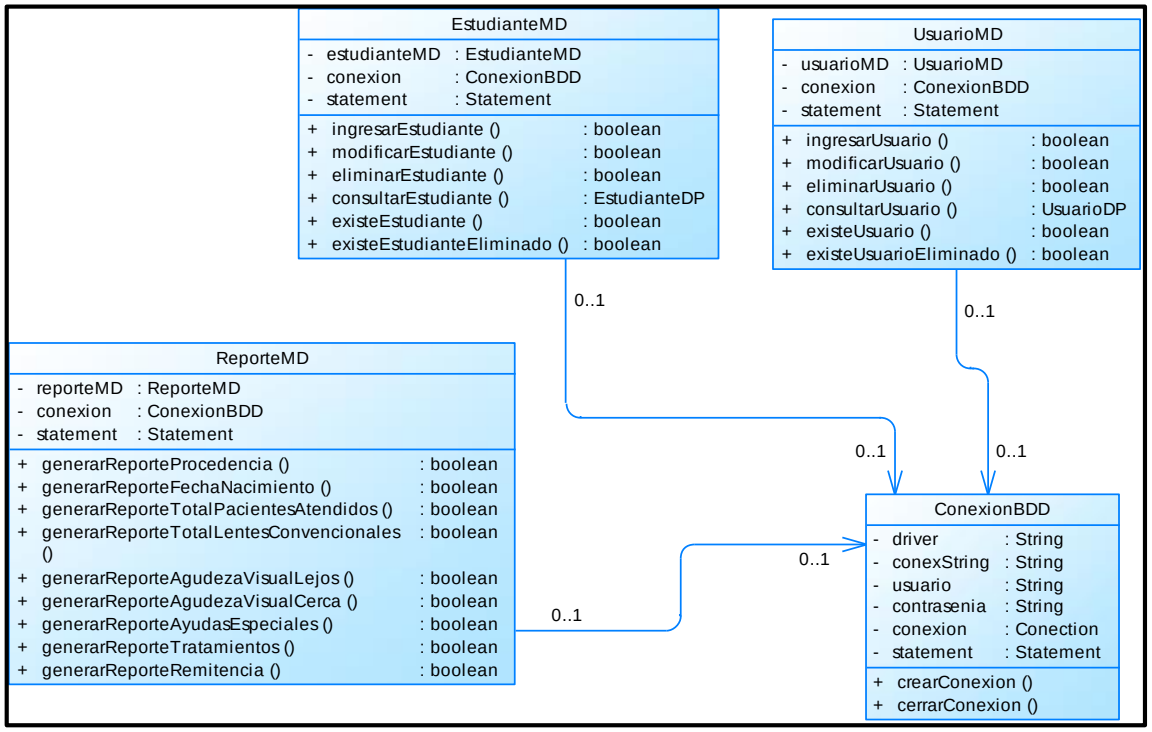

Figura 126. Diagrama de Clases de Manejo de Datos – Ciclo 2

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 4. Diagramas de Clases → 1. Manejo de Datos, que se encuentra en el CD.

4.2.1.2. Capa de Dominio del Problema – Ciclo 2

4.2.1.2.1 Diagrama de Conceptual Clases del Dominio del Problema – Ciclo 2

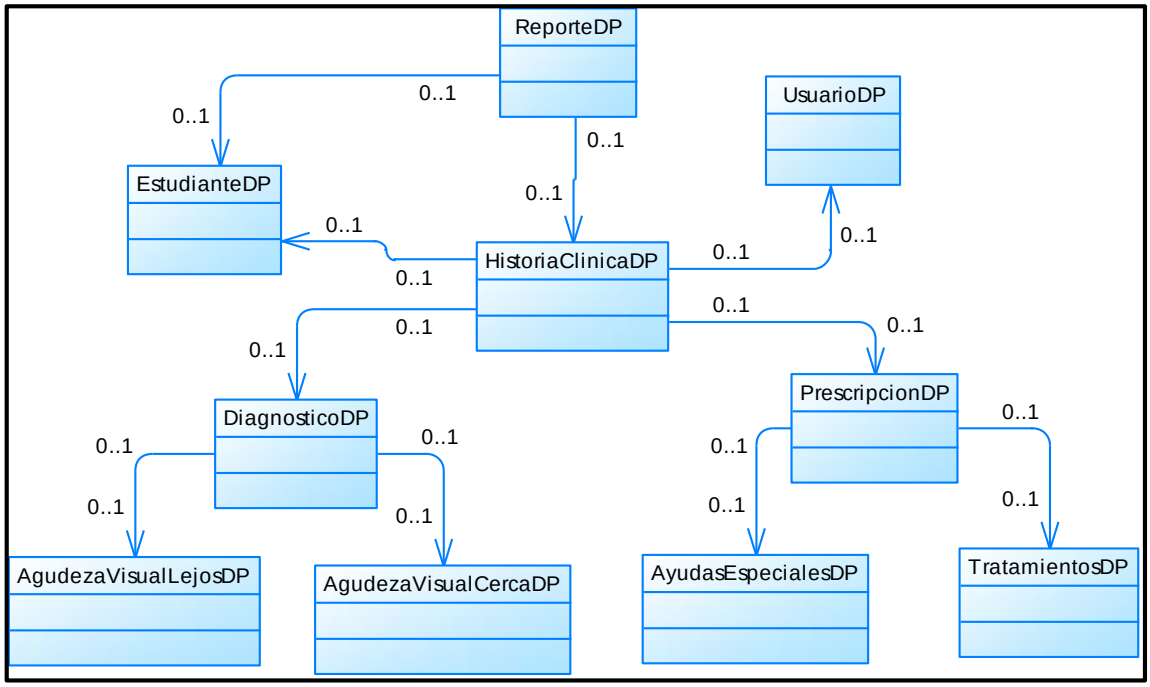

Figura 127. Diagrama Conceptual de Clases del Dominio del Problema – Ciclo 2

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 4. Diagramas de Clases → 2. Dominio del Problema, que se encuentra en el CD.

4.2.1.2.2 Diagrama de Clases del Dominio del Problema – Ciclo 2

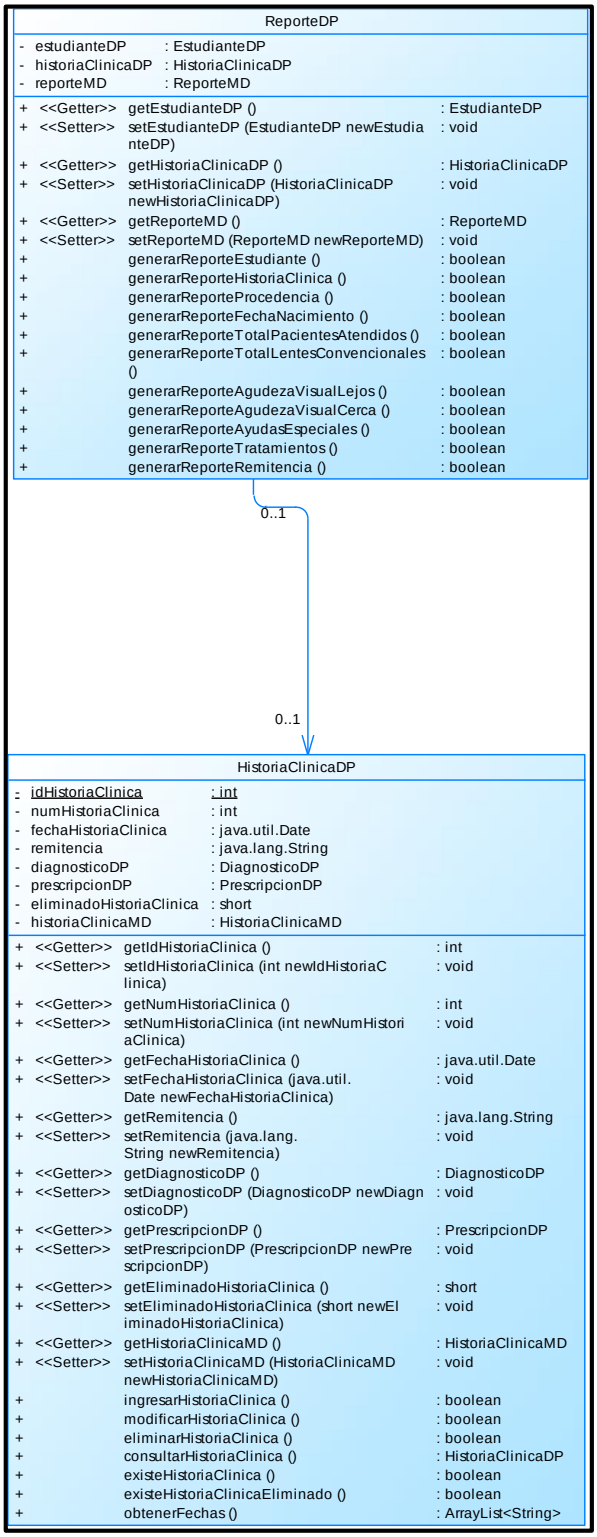

Figura 128. Diagrama de Clases del Dominio del Problema – Ciclo 2

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 4. Diagramas de Clases → 2. Dominio del Problema, que se encuentra en el CD.

4.2.1.3. Capa GUI o de Presentación – Ciclo 2

4.2.1.3.1 Diagrama de Clases GUI – Ciclo 2

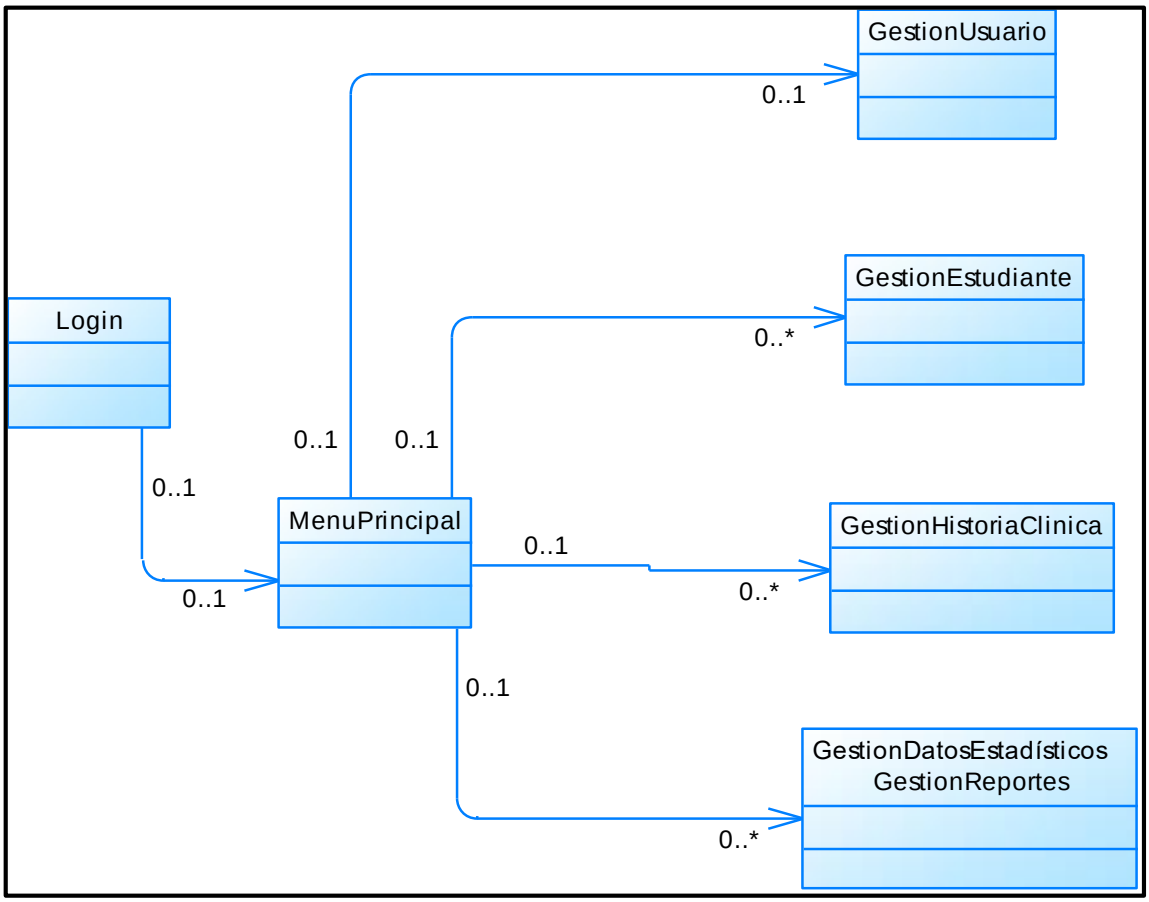

Figura 129. Diagrama de Clases GUI – Ciclo 2

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 4. Diagramas de Clases → 3. GUI, que se encuentra en el CD.

4.2.2. Diagrama de Paquetes – Ciclo 2

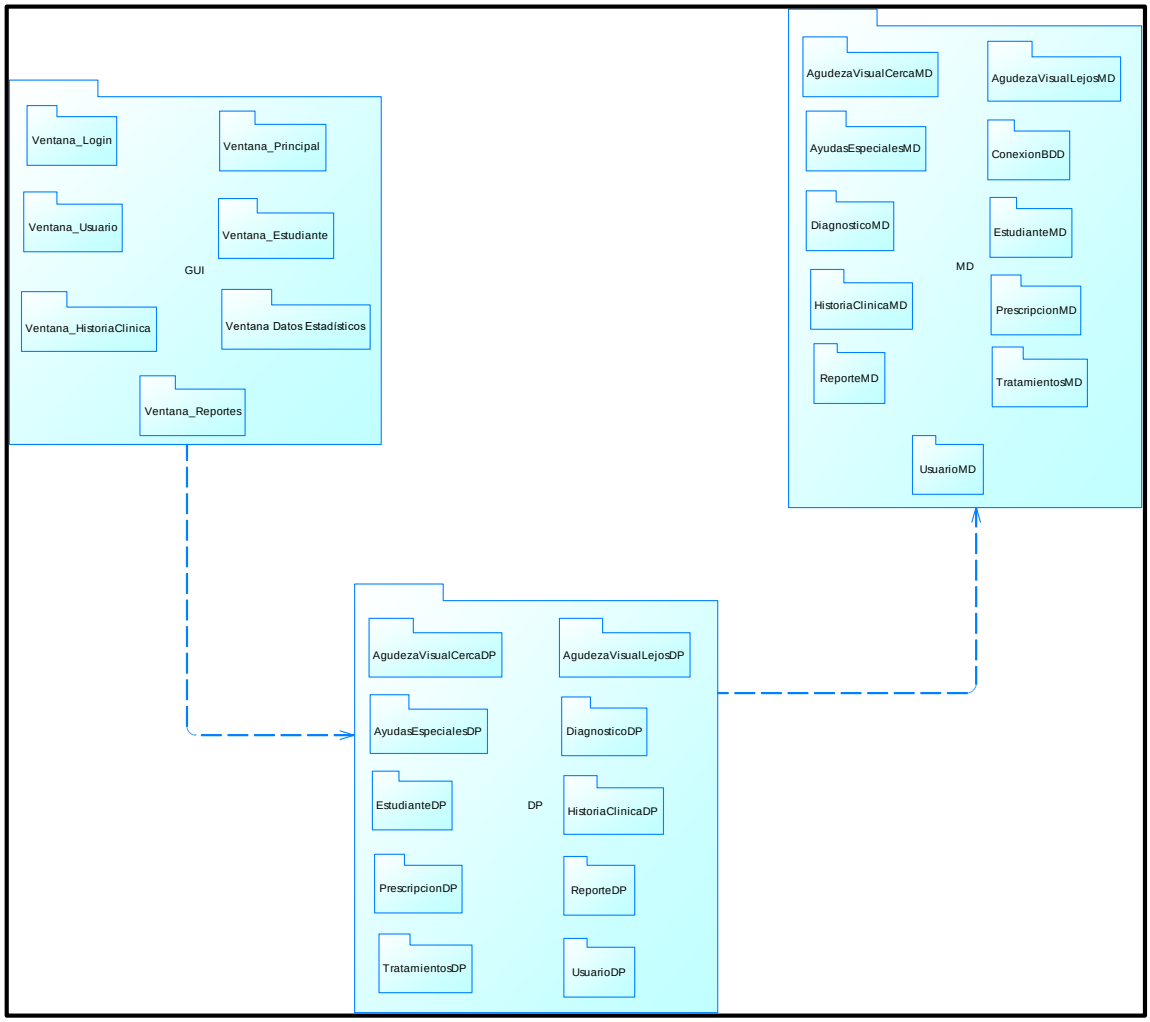

Figura 130. Diagrama de Paquetes – Ciclo 2

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 1. Diagramas → 7. Diagramas de Paquetes, que se encuentra en el CD.

4.2.3. Plan de Pruebas del Sistema – Ciclo 2

4.2.3.1. RF6. Gestionar Datos Estadísticos

Entradas	Resultados Esperados	Caso de Uso	Estado
Selección de opción Gestión de Datos Estadísticos y Reportes	Despliegue de Ventana de Datos Estadísticos y Reportes	RF6 RF7	
Verificación de datos estadísticos generados correctamente	Sistema genera correctamente datos estadísticos a partir de los datos de la base de datos	RF6 RF6.1	
Verificación de generación correcta de gráficos estadísticos	Sistema genera correctamente gráficos estadísticos	RF6 RF6.1	

Tabla 12. Plan de Pruebas del Sistema para RF6. Gestionar Datos Estadísticos

Elaborado por: (Recalde, 2017)

4.2.3.2. RF7. Gestionar Reportes

Entradas	Resultados Esperados	Caso de Uso	Estado
Selección de opción Gestión de Datos Estadísticos y Reportes	Despliegue de Ventana de Datos Estadísticos y Reportes	RF6 RF7	
Verificación de generación de reportes de datos estadísticos	Sistema genera correctamente reportes de datos estadísticos	RF7.5	
Verificación de generación de reporte de todos los estudiantes existentes	Sistema genera correctamente reporte de todos los estudiantes existentes	RF7.1	
Verificación de selección de un estudiante para generar reporte de estudiante por parámetro	Sistema verifica que se ha seleccionado un estudiante para la generación de estudiante	RF7	
Verificación de generación de reporte de estudiante por parámetro	Sistema genera correctamente reporte de estudiante seleccionado por parámetro	RF7.2	
Verificación de existencia de historias clínicas de un estudiante	Sistema verifica la existencia de historias clínicas de estudiante seleccionado	RF7	
Verificación de generación de reporte de historias clínica por parámetro	Sistema genera correctamente reporte de historia clínica por parámetro	RF7.3	
Verificación de generación de reporte de todas historias clínicas de un estudiante	Sistema genera correctamente reporte de todas las historias clínicas de un estudiante	FR7.4	

Tabla 13. Plan de Pruebas del Sistema para RF7. Gestionar Reportes

Elaborado por: (Recalde, 2017)

4.3. Fase de Desarrollo – Ciclo 2

En el siguiente apartado se explica la manera en la que fue desarrollado los módulos del ciclo 2, tanto a nivel visual como a en codificación. Debido a la cantidad de código que el sistema posee, no se presenta al mismo en su totalidad, únicamente se expondrá el código fundamental que se utilizan a lo largo de su desarrollo.

4.3.1. Diseño e Implementación de Interfaces Gráficas – Ciclo 2

4.3.1.1. Interfaz de Ventana o Menú Principal – Ciclo 2

Figura 131. Interfaz de Ventana o Menú Principal

Elaborado por: (Recalde, 2017)

4.3.1.2. Interfaz Ventana Gestión Datos Estadísticos y Reportes

Figura 132. Interfaz Ventana Gestión Datos Estadísticos y Reportes

Elaborado por: (Recalde, 2017)

4.3.1.3. Interfaz Ejemplo de Datos Estadísticos

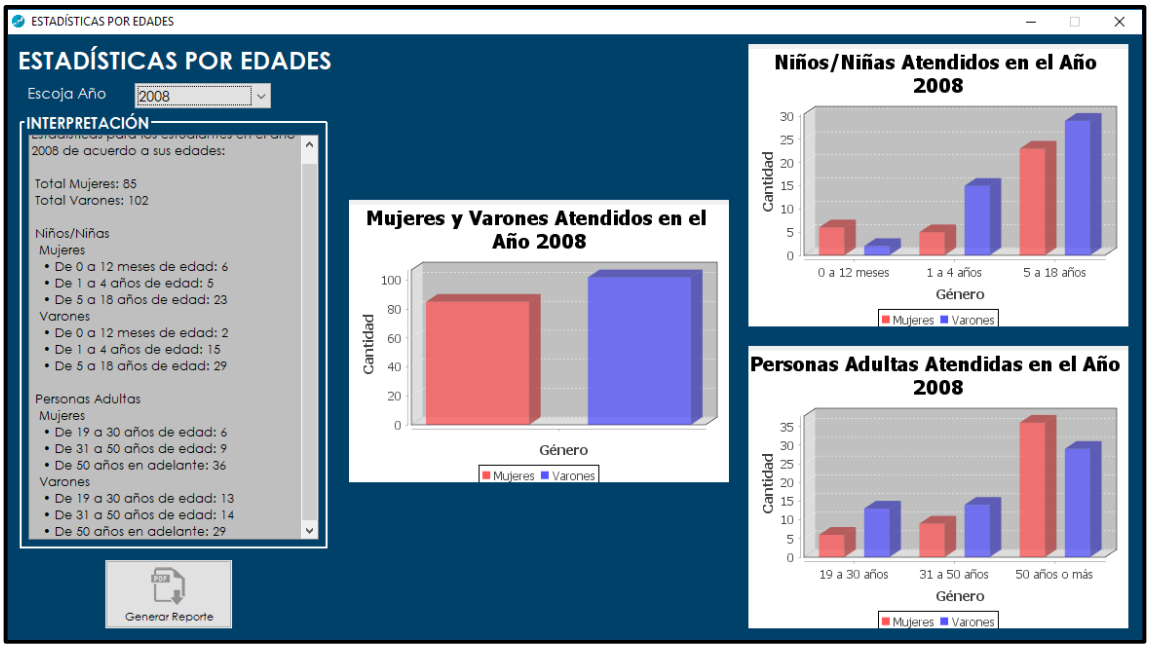

Figura 133. Interfaz Ejemplo de Datos Estadísticos

Elaborado por: (Recalde, 2017)

4.3.1.4. Interfaz Gestión de Reporte de Estudiante por Parámetro

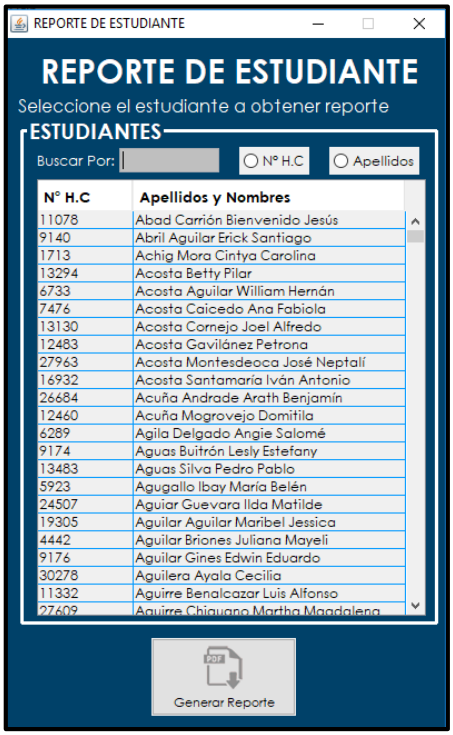

Figura 134. Interfaz Gestión de Reporte de Estudiante por Parámetro

Elaborado por: (Recalde, 2017)

4.3.1.5. Interfaz Gestión de Reportes de Historias Clínicas por Estudiante

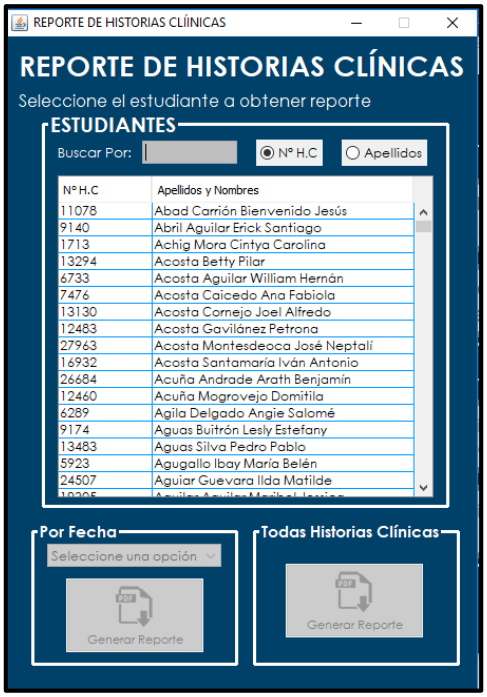

Figura 135. Interfaz Gestión de Reportes de Historias Clínicas por Estudiante

Elaborado por: (Recalde, 2017)

4.3.1.6. Interfaz Gestión de Reportes de Historias Clínicas por Año

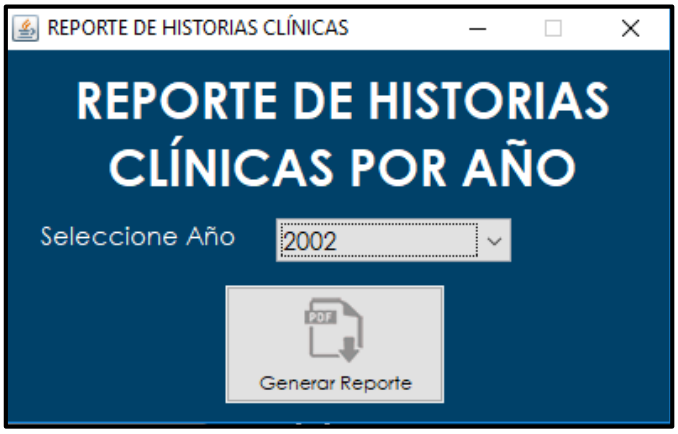

Figura 136. Interfaz Gestión de Reportes de Historias Clínicas por Año
Elaborado por: (Recalde, 2017)

4.3.2. Código GUI y Explicación – Ciclo 2

Para la generación de los gráficos presentados en el sistema se utilizó la librería JFreeChart, la cual es una librería de Java de gráficos 100% gratuita, que facilita a los desarrolladores la generación y visualización de gráficos de calidad profesional en las distintas aplicaciones realizadas. Entre las características de esta librería constan:

- Una API[6] consistente y bien documentada que admite una extensa gama de tipos de gráficos.
- Un diseño flexible, fácil de extender, y se dirige tanto a aplicaciones del lado del servidor como del cliente.
- Soporte para varios tipos de salida, incluidos componentes Swing y JavaFX, archivos de imagen incluyendo PNG y JPEG. También formatos de archivos de gráficos vectoriales, incluidos PDF, EPS y SVG.

JFreeChart es de código abierto, es decir, de software libre. Se distribuye bajo los términos de la Licencia Pública General Reducida (LGPL) de GNU, que permite el uso en aplicaciones propietarias («JFreeChart», s. f.).

[6] Conjunto de reglas y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano - software.

4.3.2.1. Ejemplo de Gráfico JFreeChart

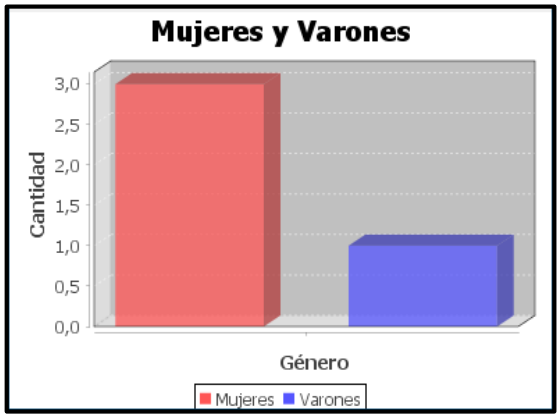

Figura 137. Ejemplo de Gráfico JFreeChart

Elaborado por: (Recalde, 2017)

- Código

```
DefaultCategoryDataset datos2;
datos2 = new DefaultCategoryDataset();
datos2.addValue(Integer.parseInt(MujeresVarones.get(1).get(0)), MujeresVarones.get(0).get(0), "");
datos2.addValue(Integer.parseInt(MujeresVarones.get(1).get(1)), MujeresVarones.get(0).get(1), "");
barraGeneral = ChartFactory.createBarChart3D("Mujeres y Varones", "Género","Cantidad",datos2,PlotOrientation.
      VERTICAL,true,true,true);
BufferedImage graficoBarra2=barraGeneral.createBufferedImage(panelGeneral.getWidth(), panelGeneral.getHeight());
lblGeneral.setSize(panelGeneral.getSize());
lblGeneral.setIcon(new ImageIcon(graficoBarra2));
panelGeneral.updateUI();
```

Figura 138. Código de Generación de Gráfico JFreeChart

Elaborado por: (Recalde, 2017)

4.3.2.2. Validación Selección de Estudiante

Para generar el reporte de estudiante por parámetro se valida primero que se haya seleccionado un estudiante para poder generar dicho reporte. En caso de no ser así, se presentará el siguiente cuadro de diálogo de aviso:

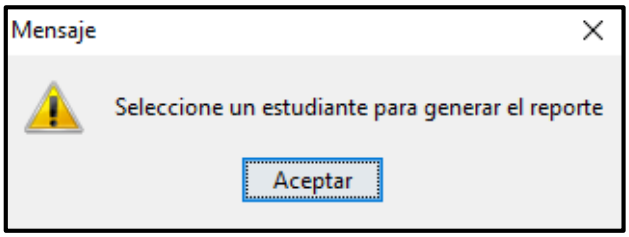

Figura 139. Aviso de Selección de Estudiante

Elaborado por: (Recalde, 2017)

- Código

```
private void btnReporteEstudianteActionPerformed(java.awt.event.ActionEvent evt) {
    int fila = jTableEstudiantes.getSelectedRow();
    if (fila==-1 || fila==jTableEstudiantes.getRowCount()-1)
    JOptionPane.showMessageDialog(jFrameEstudianteParametro, "Seleccione un estudiante para generar el reporte",
            "Mensaje", JOptionPane.WARNING_MESSAGE);
    else{
        String cedula = jTableEstudiantes.getValueAt(fila, 0).toString();
        estudianteDP.setnumHistoriaClinicaEstudiante(cedula);
        reporteDP = new ReporteDP();
        reporteDP.setEstudianteDP(estudianteDP);
        String ruta = "", rutal = "", ruta2 = "";
        if(fcSeleccionArchivo.showSaveDialog(jFrameEstudianteParametro) == JFileChooser.APPROVE_OPTION)
        {
            File archivo = fcSeleccionArchivo.getSelectedFile();
            ruta = fcSeleccionArchivo.getSelectedFile().getAbsolutePath();
            rutal = fcSeleccionArchivo.getSelectedFile().getName();
            if(new File(ruta).exists())
            {
                int resp = JOptionPane.showConfirmDialog(jFrameEstudianteParametro, rutal+" ya existe\n¿Quieres re
                "Mensaje", JOptionPane.YES_NO_OPTION);
                if (JOptionPane.YES_NO_OPTION == resp){
                    if(reporteDP.generarReporteEstudiante(archivo))
                    {
                        JOptionPane.showMessageDialog(jFrameEstudianteParametro, "Reporte Generado con Éxito");
                    }
                    else
                        JOptionPane.showMessageDialog(jFrameEstudianteParametro, "Error de sistema porfavor reinic
```

Figura 140. Código de Validación de Selección de Estudiante

Elaborado por: (Recalde, 2017)

4.3.2.3. Aviso de Reporte Generado con Éxito

Cuando se generen los reportes correctamente se generará el siguiente aviso, con el fin de que el usuario sepa que su reporte fue generado y pueda acceder al mismo.

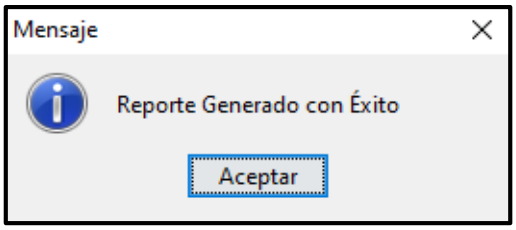

Figura 141. Aviso de Reporte Generado Exitosamente

Elaborado por: (Recalde, 2017)

4.3.3. Clase DP y Explicación – Ciclo 2

Se utilizó una única clase DP llamada ReporteDP la cual contiene principalmente los métodos que van a la clase MD y consultan los valores para la generación de los gráficos estadísticos que el sistema posee, además de los métodos para la de reportes.

Para la creación de los reportes se utilizó la librería llamada iText que es una bibioteca Open Source o de Código Abierto y que sirve para la creación y manipulación de archivos PDF, RTF y HTML en Java. Desarrollada por Bruno Lowagie, Paulo Saores, entre otros; y se la distribuye con la licencia Affero General Public versión 3.

El soporte que brinda iText para un PDF, es bastante extenso. Dicho hace que sobrelleve formas basadas en PKI de PDF, cifrado de 40bit y 128bit y ejecute corrección de colores («iText», 2013).

4.3.3.1. Código de Instanciación de Clase ReporteDP

```
public class ReporteDP {
    private EstudianteDP estudianteDP;
    private HistoriaClinicaDP historiaClinicaDP;
    DiagnosticoDP diagnosticoDP = new DiagnosticoDP();
    AgudezaVisualLejosDP agudezaVisualLejosDP = new AgudezaVisualLejosDP();
    AgudezaVisualCercaDP agudezaVisualCercaDP = new AgudezaVisualCercaDP();
    PrescripcionDP prescripcionDP = new PrescripcionDP();
    AyudasEspecialesDP ayudasEspecialesDP = new AyudasEspecialesDP();
    TratamientosDP tratamientosDP = new TratamientosDP();
    private ReporteMD reporteMD;
```

Figura 142. Código de Instanciación de Clase ReporteDP

Elaborado por: (Recalde, 2017)

4.3.3.2. Código de GETTER y SETTER de Clase ReporteDP

```
public EstudianteDP getEstudianteDP() {
    return estudianteDP;
}

public void setEstudianteDP(EstudianteDP estudianteDP) {
    this.estudianteDP = estudianteDP;
}

public HistoriaClinicaDP getHistoriaClinicaDP() {
    return historiaClinicaDP;
}

public void setHistoriaClinicaDP(HistoriaClinicaDP historiaClinicaDP) {
    this.historiaClinicaDP = historiaClinicaDP;
}

public ReporteMD getReporteMD() {
    return reporteMD;
}

public void setReporteMD(ReporteMD reporteMD) {
    this.reporteMD = reporteMD;
}
```

Figura 143. Código de GETTER y SETTER de Clase ReporteDP

Elaborado por: (Recalde, 2017)

4.3.3.3. Código Ejemplo de Método que Retorna Valores para la Generación de Gráficos Estadísticos

```
public ArrayList<ArrayList<String>> obtenerCantones(){
    this.reporteMD = new ReporteMD(this);
    return reporteMD.obtenerCantones();
```

Figura 144. Código Ejemplo de Métodos que Retorna Valores para la Generación de Gráficos Estadísticos en Clase ReporteDP

Elaborado por: (Recalde, 2017)

4.3.3.4. Código de Creación de Reportes

4.3.3.4.1 Código de Creación de Documento

```
public boolean generarReporteEstudiante(File archivo){
    boolean retorno = true;
    estudianteDP = estudianteDP.consultarEstudiante();
    fN = estudianteDP.getFechaNacimientoEstudiante().toString();
    calcularAnios();
    try
    {
        Document documento = new Document(PageSize.A4, 50, 50, 60, 40);
        PdfWriter writer = null;
        try {
            writer = PdfWriter.getInstance(documento, new FileOutputStream(archivo.getAbsolutePath()+".pdf"));
        } catch (FileNotFoundException | DocumentException ex) {
            ex.getMessage();
            return false;
```

Figura 145. Código de Creación de Documento en Clase ReporteDP

Elaborado por: (Recalde, 2017)

4.3.3.4.2 Código de Instanciación de Documento

```
documento.open();
documento.addAuthor("FundacionMarianaJesus");
documento.addTitle("REPORTE ESTUDIANTES");
```

Figura 146. Código de Instanciación de Documento en Clase ReporteDP

Elaborado por: (Recalde, 2017)

4.3.3.4.3 Código de Creación de Título de Documento

```
Paragraph parrafo0 = new Paragraph();
parrafo0.setAlignment(Element.ALIGN_CENTER);
parrafo0.setFont(FontFactory.getFont("Times New Roman", 15, Font.BOLD, BaseColor.RED));
parrafo0.add("REPORTE ESTUDIANTIL");
```

Figura 147. Código de Creación de Título de Documento en Clase ReporteDP

Elaborado por: (Recalde, 2017)

4.3.3.4.4 Código de Creación de Texto de Documento

```
Paragraph parrafo1 = new Paragraph();
parrafo1.setAlignment(Element.ALIGN_JUSTIFIED);
parrafo1.setFont(FontFactory.getFont("Times New Roman", 16, Font.BOLD, BaseColor.BLACK));
parrafo1.add("\n\nEstudiante: "+estudianteDP.getNombresEstudiante()+" "+estudianteDP.getApellidosEstudiante()+

Paragraph parrafo2 = new Paragraph();
parrafo2.setFont(FontFactory.getFont("Times New Roman", 12, Font.BOLD, BaseColor.BLACK));
parrafo2.add("\n   Número de Historia Clínica\n\n");
PdfPCell cell1 = new PdfPCell(parrafo2);
cell1.setHorizontalAlignment(Element.ALIGN_CENTER);
cell1.setVerticalAlignment(Element.ALIGN_MIDDLE);
PdfPCell cell2 = new PdfPCell(new Phrase(estudianteDP.getnumHistoriaClinicaEstudiante().trim()));
cell2.setVerticalAlignment(Element.ALIGN_MIDDLE);
table3.addCell(cell1);
table3.addCell(cell2);

Paragraph parrafo3 = new Paragraph();
parrafo3.setFont(FontFactory.getFont("Times New Roman", 12, Font.BOLD, BaseColor.BLACK));
parrafo3.add("\n   Apellidos\n\n");
PdfPCell cell3 = new PdfPCell(parrafo3);
cell3.setHorizontalAlignment(Element.ALIGN_CENTER);
cell3.setVerticalAlignment(Element.ALIGN_MIDDLE);
PdfPCell cell4 = new PdfPCell(new Phrase(estudianteDP.getApellidosEstudiante().trim()));
cell4.setVerticalAlignment(Element.ALIGN_MIDDLE);
table3.addCell(cell3);
table3.addCell(cell4);
```

Figura 148. Código de Creación de Texto de Documento en Clase ReporteDP

Elaborado por: (Recalde, 2017)

4.3.3.4.5 Código de Estructura de Documento

```
try {
    documento.add(parrafo0);
    documento.add(parrafo1);
    documento.add(table2);
} catch (DocumentException ex) {
    return false;
```

Figura 149. Código de Estructura de Documento en Clase ReporteDP

Elaborado por: (Recalde, 2017)

4.3.4. Clases MD y Explicación – Ciclo 2

La única clase MD para los reportes llamada ReporteMD, posee su objeto de conexión a la base de datos 'ConexionBDD' y tiene como fin la consulta de los datos que son necesarios para la creación del módulo de datos estadísticos

4.3.4.1. Código de Instanciación de Clase ReporteMD

```
public class ReporteMD {
    HistoriaClinicaDP historiaClinicaDP;
    ReporteDP reporteDP;
    ConexionBDD conexionBDD;
    Statement statement;
```

Figura 150. Código de Instanciación de Clase ReporteMD

Elaborado por: (Recalde, 2017)

4.3.4.2. Código de Constructor de Clase ReporteMD

```
public ReporteMD(ReporteDP reporteDP) {
    this.reporteDP = reporteDP;
    conexionBDD = new ConexionBDD();
}
```

Figura 151. Código de Constructor de Clase ReporteMD

Elaborado por: (Recalde, 2017)

4.3.4.3. Código de GETTER y SETTER de Clase ReporteMD

```
public HistoriaClinicaDP getHistoriaClinicaDP() {
    return historiaClinicaDP;
}

public void setHistoriaClinicaDP(HistoriaClinicaDP historiaClinicaDP) {
    this.historiaClinicaDP = historiaClinicaDP;
}
```

Figura 152. Código de GETTER y SETTER Clase ReporteMD

Elaborado por: (Recalde, 2017)

4.3.4.4. Código Ejemplo de Método de Consulta de Datos

```
public ArrayList<String> obtenerProvincias(){
    this.statement = conexionBDD.crearConexion();
    ArrayList<String> retorno = new ArrayList<>();
    try {
        String query = "SELECT provinciaestudiante FROM estudiante WHERE eliminadoestudiante = 0";
        ResultSet resultSet = statement.executeQuery(query);
        while(resultSet.next())
        {
            retorno.add(resultSet.getString("provinciaestudiante").trim());
        }

        conexionBDD.cerrarConexion();
        return retorno;
    } catch (SQLException ex) {
        System.out.println(ex);
    }
    return retorno;
}
```

Figura 153. Código Ejemplo de Método de Consulta de Datos

Elaborado por: (Recalde, 2017)

4.4. Fase de Transición – Ciclo 2

En este apartado se presentará las pruebas realizadas en el sistema, y su correspondiente informe firmado por el cliente, asegurando que el producto final funciona de manera correcta.

4.4.1. Pruebas de Caja Blanca – Ciclo 2

Entradas	Resultados Esperados	Caso de Uso	Estado
Despliegue de ventanas	Despliegue de Ventanas correctamente		☑
Verificación de Generación de Reportes	Sistema genera reportes correctamente	RF7 RF7.1 RF7.2 RF7.3 RF7.4 RF7.5	☑

Tabla 14. Pruebas de Caja Blanca – Ciclo 2

Elaborado por: (Recalde, 2017)

4.4.2. Pruebas de Caja Negra – Ciclo 2

Entradas	Resultados Esperados	Caso de Uso	Estado
Verificación de existencia de estudiantes para generar reportes	Sistema verifica la existencia de estudiantes para la generación de reportes de los mismos	RF7.1 RF7.2	☑
Verificación de existencia de historias clínicas para generar reportes	Sistema verifica la existencia de historias clínicas para la generación de reportes de las mismas	RF7.3 RF7.4	☑

Tabla 15. Pruebas de Caja Negra – Ciclo 2

Elaborado por: (Recalde, 2017)

4.4.3. Informe de Pruebas del Sistema

Plan de Pruebas del Sistema – Ciclo 2

2.1. RF6. Gestionar Datos Estadísticos

Entradas	Resultados Esperados	Caso de Uso	Resultado
Selección de opción Gestión de Datos Estadísticos y Reportes	Despliegue de Ventana de Datos Estadísticos y Reportes	RF6 RF7	✓
Verificación de datos estadísticos generados correctamente	Sistema genera correctamente datos estadísticos a partir de los datos de la base de datos	RF6 RF6.1	✓
Verificación de generación correcta de gráficos estadísticos	Sistema genera correctamente gráficos estadísticos	RF6 RF6.1	✓

FUNDACIÓN "AMIGA DE LOS CIEGOS" Instituto Especial "MARIANA DE JESÚS"

Firma Representante Fundación "Mariana de Jesús"

Firma Estudiante PUCE
Lilian Recalde

Figura 154. Informe de Pruebas del Sistema 1 de 2 – Ciclo 2

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 6. Pruebas del Sistema → 2. Pruebas del Sistema – Ciclo 2, que se encuentra en el CD.

2.2. RF7. Gestionar Reportes

Entradas	Resultados Esperados	Caso de Uso	Resultado
Selección de opción Gestión de Datos Estadísticos y Reportes	Despliegue de Ventana de Datos Estadísticos y Reportes	RF6 RF7	✓
Verificación de generación de reportes de datos estadísticos	Sistema genera correctamente reportes de datos estadísticos	RF7.5	✓
Verificación de generación de reporte de todos los estudiantes existentes	Sistema genera correctamente reporte de todos los estudiantes existentes	RF7.1	✓
Verificación de selección de un estudiante para generar reporte de estudiante por parámetro	Sistema verifica que se ha seleccionado un estudiante para la generación de estudiante	RF7	✓
Verificación de generación de reporte de estudiante por parámetro	Sistema genera correctamente reporte de estudiante seleccionado por parámetro	RF7.2	✓
Verificación de existencia de historias clínicas de un estudiante	Sistema verifica la existencia de historias clínicas de estudiante seleccionado	RF7	✓
Verificación de generación de reporte de historias clínica por parámetro	Sistema genera correctamente reporte de historia clínica por parámetro	RF7.3	✓
Verificación de generación de reporte de todas historias clínicas de un estudiante	Sistema genera correctamente reporte de todas las historias clínicas de un estudiante	FR7.4	✓

FUNDACIÓN "AMIGA DE LOS CIEGOS" Instituto Especial "MARIANA DE JESÚS" Quito - 1953

Firma Representante Fundación "Mariana de Jesús"

Lili Recalde C

Firma Estudiante PUCE
Lilian Recalde

Figura 155. Informe de Pruebas del Sistema 2 de 2 – Ciclo 2

Elaborado por: (Recalde, 2017)

Para mayor detalle favor revisar el anexo 6. Pruebas del Sistema → 2. Pruebas del Sistema – Ciclo 2, que se encuentra en el CD.

Capítulo V Conclusiones y Recomendaciones

5.1. Conclusiones

- El uso de la metodología de desarrollo de software RUP fue de mucha ayuda en la organización del desarrollo completo del sistema, debido a que la misma es confiable, mantiene integridad y estabilidad.

- Al dividir el desarrollo del sistema en dos ciclos, se logró una mejor eficiencia en el proceso, ya que al realizar los módulos principales en el primer ciclo se pudo realizar fácilmente los módulos del segundo ciclo, en vista que los mismos toman los datos necesarios de los módulos del primer ciclo, es decir, se siguió una secuencia lógica de pasos para lograr el completo desarrollo de software.

- Los sistemas computacionales deben ser diseñados y desarrollados de acuerdo con las especificaciones de los requerimientos que el usuario final otorgue.

- El lenguaje de programación JAVA al ser un lenguaje multiplataforma y gracias a su máquina virtual, permite que los sistemas desarrollados en el mismo puedan ser replicados en cualquier sistema operativo.

- El uso de librerías .jar, tales como jCalendar, iText o JFreeChart facilitaron el desarrollo del sistema, ya que otorgaron al desarrollador las herramientas necesarias para poder completar apropiadamente el proyecto.

- Realizar validaciones en los campos necesarios para el ingreso de datos por parte del usuario es fundamental para poder mantener la integridad de los mismos.

- El principal objetivo del sistema presentado en la presente disertación es que los usuarios del Instituto Especial 'Mariana de Jesús' puedan manejar de manera correcta la información de sus estudiantes y de las historias clínicas de los mismos. Además de poder visualizar datos estadísticos requeridos por el Instituto, así también como la generación correcta de reportes.

5.2. Recomendaciones

- El sistema presentado en la presente disertación se encuentra desarrollado en una plataforma de tipo cliente – servidor, se puede recomendar que, en un futuro y para ampliar el sistema, sea migrado a un sistema de tipo Web, con esto se podría reutilizar los paquetes MD y DP, y únicamente cambiar el paquete GUI.

- Se podrían realizar respaldos automáticos de la información desde el sistema cada día, semanalmente o mensualmente, como requiera el cliente, al momento solo se puede respaldar la información desde pgAdmin 4, el cual es la interfaz de usuario de PostgreSQL.

- Al momento, el sistema maneja información de estudiantes y de las historias clínicas de los mismos, se puede ampliar dicho sistema manejando además la información académica de los estudiantes del Instituto, para poder manejar notas, asistencia y registros académicos en general.

- Debido a los limitados recursos del Instituto, el mismo no puede contar con un servidor, y por lo tanto el sistema se encuentra programado de manera local, para futuras versiones y si el Instituto tiene la oportunidad de tener un servidor, se podría migrar la base de datos al servidor, y desde la red acceder a la misma, se tendría que cambiar la clase 'conexionBDD', debido a que ésta, gestiona todas las conexiones del sistema con la base de datos.

- Actualmente, y en mi opinión personal, el Instituto maneja de manera incorrecta la información de sus estudiantes para la Gestión de Historias Clínicas, ya que los mismos no cuentan con un campo para el Número de Cédula de Ciudadanía, es por esto que el único identificador que usan para sus estudiantes son los número de historias clínicas, y pueden existir números de historias clínicas repetidas en diferentes estudiantes, es por esto que se recomendó al Instituto que en un futuro tomen en cuenta el Número de Cédula de sus estudiantes como identificadores únicos, de ésta manera se lograría mayor efectividad y tener menos errores.

Anexos

1. Diagramas
- **1.1.** Diagrama General
- **1.2.** Diagramas Específicos
  - **1.2.1.** RF1. Gestionar Usuarios
  - **1.2.2.** RF2. Gestionar Estudiantes
  - **1.2.3.** RF3. Gestionar Historias Clínicas
  - **1.2.4.** RF4. Gestionar Diagnósticos
  - **1.2.5.** RF5. Gestionar Prescripciones
  - **1.2.6.** FR6. Gestionar Datos Estadísticos
  - **1.2.7.** RF7. Gestionar Reportes
- **1.3.** Diseño Conceptual
- **1.4.** Diagramas de Clases
  - **1.4.1.** Manejo de Datos
  - **1.4.2.** Dominio del Problema
  - **1.4.3.** GUI
- **1.5.** Diagramas de Actividades
- **1.6.** Diagramas Entidad – Relación de la Base de Datos
  - **1.6.1.** Diagrama E-R Conceptual
  - **1.6.2.** Diagrama E-R Lógico
  - **1.6.3.** Diagrama E-R Físico
- **1.7.** Diagrama de Paquetes
- **1.8.** Diagramas de Secuencia
  - **1.8.1.** RF1. Gestionar Usuarios
  - **1.8.2.** RF3. Gestionar Historias Clínicas

2. Especificación de Requerimientos

3. Manual de Usuario

4. Instalador

5. Programa en JAVA

6. Pruebas del Sistema
- **6.1.** Pruebas del Sistema – Ciclo 1
- **6.2.** Pruebas del Sistema – Ciclo 2

7. Certificados Instituto Especial 'Mariana de Jesús'

Bibliografía

[1] Arquitectura de tres niveles. (s. f.). Recuperado 7 de julio de 2017, a partir de https://www.ecured.cu/Arquitectura_de_tres_niveles

[2] Denzer, P. (2002, octubre 23). PostgreSQL. Recuperado 6 de julio de 2017, a partir de http://profesores.elo.utfsm.cl/~agv/elo330/2s02/projects/denzer/informe.pdf

[3] Diagramas del UML. (s. f.). Recuperado 17 de julio de 2017, a partir de http://festivaldealmagro.com/en/uploads/convocatorias/diagramas_del_uml.pdf

[4] González, K. A. (2012, febrero 22). Metodología de Softaware: Mapa conceptual «Metodología de desarrollo de Software». Recuperado a partir de http://kerenavi.blogspot.com/2012/02/mapa-conceptual-metodologia-de.html

[5] Hernández, S. D. G. (2006, mayo). *Flujo de trabajo del análisis y diseño de Rup.pdf*. Recuperado a partir de http://repository.uaeh.edu.mx/bitstream/bitstream/handle/123456789/11150/Flujo%20de%20trabajo%20del%20an%C3%A1lisis%20y%20dise%C3%B1o%20de%20Rup.pdf?sequence=1

[6] Hormiga, J. (2007, mayo 1). Proceso Unificado para Desarrollo de Software (RUP). Recuperado 7 de mayo de 2017, a partir de http://artemisa.unicauca.edu.co/~gramirez/archivos/AnotacionesRUP.pdf

[7] Introducción a Java. (s. f.). Recuperado 6 de julio de 2017, a partir de http://personales.upv.es/igil/java.PDF

[8] iText. (2013, noviembre 16). Recuperado 17 de noviembre de 2017, a partir de http://java-white-box.blogspot.com/2013/11/itext-introduccion-itext-que-es-el.html

[9] Jacobson, I., Booch, G., & Rumbaugh, J. (2000). *El Proceso Unificado de Desarrollo de Software*. Madrid: PEARSON EDUCACIÓN. S.A.

[10] JFreeChart. (s. f.). Recuperado 24 de octubre de 2017, a partir de http://www.jfree.org/jfreechart/

[11] Larman, C. (1999). *UML y Patrones* (Primera). PEARSON.

[12] Lenguaje Java y Entorno de Desarrollo. (s. f.). Recuperado 6 de julio de 2017, a partir de http://www.jtech.ua.es/j2ee/2006-2007/doc/sesion01-apuntes.pdf

[13] Londoño, J. H. A. (2005, miércoles, abril). Ingeniería de Software: TIPOS DE PRUEBAS DE SOFTWARE. Recuperado a partir de http://ing-sw.blogspot.com/2005/04/tipos-de-pruebas-de-software.html

[14] Luna, J. M. (2009, marzo 6). Pruebas de Caja Negra y Caja Blanca. Recuperado 23 de octubre de 2017, a partir de http://ingenierogestion.blogspot.com/2009/06/pruebas-de-caja-negra-y-caja-blanca.html

[15] Machuca, X. (2014a, diciembre 11). Conoce al «Instituto Especial Mariana de Jesús». Recuperado 22 de junio de 2017, a partir de https://iclosedmyeyes.wordpress.com/2014/12/11/instituto-especial-mariana-de-jesus/

[16] Machuca, X. (2014b, diciembre 11). Metodologías de Aprendizaje en el Instituto Mariana de Jesús. Recuperado 22 de junio de 2017, a partir de https://iclosedmyeyes.wordpress.com/2014/12/11/metodologias-de-aprendizaje-en-el-instituto-mariana-de-jesus/

[17] Mestras, J. P. (2004). Introducción al lenguaje Java. Recuperado 6 de julio de 2017, a partir de https://www.fdi.ucm.es/profesor/jpavon/poo/02IntroJava.pdf

[18] Metodología de RUP. (s. f.). Recuperado 1 de julio de 2017, a partir de https://metodologiasoo.wikispaces.com/Metodolog%C3%ADa+de+Rup

[19] Moya, R. (2012, marzo 31). Modelo «4+1» vistas de Kruchten. Recuperado 1 de julio de 2017, a partir de https://jarroba.com/modelo-41-vistas-de-kruchten-para-dummies/

[20] Quintanilla, A., Peña, L., Montes, E., & Nuñez, J. (2015). Que es la Metodología orientada a objetos - Introducción a UML, 31.

[21] Rational Unified Process (RUP). (s. f.). Recuperado 1 de julio de 2017, a partir de http://ima.udg.edu/~sellares/EINF-ES2/Present1011/MetodoPesadesRUP.pdf

[22] Recalde, L. (2017). *Construcción de un Sistema para la Gestión de Estudiantes del Instituto Especial para Niños «Mariana de Jesús», usando la Metodología de Desarrollo de Software RUP*. Pontificia Universidad Católica del Ecuador, Quito - Ecuador.

[23] Sellarés, J. A. (s. f.). Introducción a las Metodologías de Desarrollo de Software. Recuperado a partir de http://ima.udg.edu/~sellares/EINF-ES2/Present1011/MetodoPesadesDocumentacio.pdf

[24] Torossi, G. (s. f.). El Proceso Unificado de Desarrollo de Software. Recuperado 7 de mayo de 2017, a partir de http://dsc.itmorelia.edu.mx/~jcolivares/courses/pm10a/rup.pdf

[25] Valle, R. J. V. D., & Granados, J. P. M. (s. f.). Programación en Capas. Recuperado a partir de http://www.di-mare.com/adolfo/cursos/2007-2/pp-3capas.pdf