PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

ESCUELA DE SISTEMAS

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

"DISEÑO Y DESARROLLO DE UN SISTEMA COMPUTACIONAL DE REGISTROS BIO-PSICOSOCIALES Y TERAPIA OCUPACIONAL PARA NIÑOS Y JÓVENES CON DISCAPACIDAD INTELECTUAL PARA LA FUNDACIÓN IEPNI USANDO LA METODOLOGÍA DEL PROCESO UNIFICADO RACIONAL RUP"

ANDRÉS SEBASTIÁN SÁNCHEZ PÁEZ

DIRECTOR: INGENIERO FABIÁN DE LA CRUZ

CIUDAD: QUITO AÑO: 2017

Dedicatoria

A Dios

Por haberme permitido llegar hasta este punto y haberme dado salud para lograr mis objetivos, además de su infinita bondad y amor.

A mi madre Lorena.
Por haberme apoyado en todo momento, por sus consejos, sus valores, por la motivación constante que me ha permitido ser una persona de bien, pero más que nada, por su amor.

A mi padre Eddy.
Por los ejemplos de perseverancia y constancia que lo caracterizan y que me ha infundado siempre, por el valor mostrado para salir adelante, por todas las grandes enseñanzas que me ha dado y por su amor incondicional.

A mis familiares
A mi tía Anita por siempre darme ese apoyo incondicional y de la cual aprendí muchas cosas, a mis abuelitos por siempre cuidarme y darme tanto cariño, en general a toda mi familia por siempre estar ahí cuando los necesitaba.

A mis Maestros
Ing. Fabián de la Cruz por siempre estar pendiente, ayudándome y aconsejándome durante la carrera y en todo lo necesario para cumplir con éxito la tesis. Ing. Oswaldo Espinoza por su apoyo incondicional y siempre esas ganas de ayudar. Ing. Javier Cóndor por haberme dado las pautas teóricas durante la carrera y su ayuda en la culminación de la tesis.

A mis amigos
Gabriel, Sebastián, Miguel, David, Juan Carlos, Gianfranco y Lilian por siempre estar conmigo en las buenas y en las malas, su amistad incondicional y ese enorme apoyo que siempre me dieron cuando más lo necesitaba.

Finalmente a los maestros, aquellos que marcaron cada etapa de nuestro camino universitario, y que me ayudaron en asesorías y dudas presentadas en la elaboración de la tesis

Agradecimientos

En el presente trabajo de disertación quisiera agradecer primeramente a Dios por haberme dado la capacidad y la oportunidad de estar culminando mis estudios de pregrado.

A la Pontificia Universidad Católica del Ecuador por haberme brindado tantas oportunidades de aprender nuevas cosas y permitirme conocer a muchas personas invaluables.

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 los 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 todas sus bendiciones."

Para todos: Muchas Gracias.

Índice

Lista de Tablas

Lista de Figuras

CAPÍTULO 1: MARCO TEÓRICO

A través de este capítulo se revisarán los principales elementos teóricos que se necesitarán para realizar el proceso de desarrollo mediante la metodología RUP.

1.1. Objetivos

1.1.1. Objetivo General

Diseñar y desarrollar un sistema que gestione fichas y reportes, de terapia ocupacional y bio-psicosociales para niños y jóvenes, con discapacidad intelectual, del Instituto Educativo y Psicoterapéutico del Niño (IEPNI), utilizando la metodología RUP.

1.1.2. Objetivos Específicos

- Analizar las fichas que la institución utiliza para decidir cuál es la mejor forma de automatizarlas mediante un sistema computacional.
- Levantar los requerimientos de la institución, mediante reuniones preliminares al comienzo del proyecto.
- Diseñar la base de datos que será utilizada por el sistema.
- Diseñar el sistema, basado en estructura orientada a objetos.
- Desarrollar el sistema en el lenguaje de programación JAVA.
- Probar el sistema una vez finalizado.

1.2. Arquitectura Tres Capas

También conocida como arquitectura Multiple-Tier, "*emerge para cubrir las limitaciones de la arquitectura de Dos-Tiers,*" dado que esta arquitectura comprende una capa entre la capa de interfaz y la capa de servidor de base de datos. La arquitectura Tres Capas ha sido diseñada para brindar un mejor funcionamiento en los sistemas al tener una gran carga de usuarios y brindar flexibilidad (Cruz, Italo, & Echeverria).

La arquitectura de tres capas está formada por las siguientes capas:

1.2.1. Capa de Datos

La capa de datos es la responsable de interactuar con los datos del sistema y usuarios. Su principal función es guardar y retornar los datos, de la base de datos, a la capa de lógica del negocio ("Arquitectura de tres niveles - EcuRed", s/f).

1.2.2. Capa de Lógica del Negocio

Esta capa contiene las funciones que ejecuta el sistema, se obtiene las solicitudes de los usuarios, se procesa la información y se retornan las respuestas después de ejecutar el proceso. Tiene el nombre de lógica del negocio ya que, en esta capa se establecen las leyes que se deben satisfacer en el sistema. Esta capa es la intermediaria entre la capa de presentación y la capa de datos, la lógica de negocio recibe las solicitudes desde la capa de presentación y devuelve resultados procesados, con la capa de datos se conecta la lógica del negocio para gestionar los datos de la base de datos ("Arquitectura de tres niveles - EcuRed", s/f).

1.2.3. Capa de Presentación

Se encarga de que el usuario pueda interactuar con las funcionalidades del sistema, el usuario a través de esta capa puede obtener información del sistema o viceversa el sistema puede obtener información del usuario ("Arquitectura de tres niveles - EcuRed", s/f).

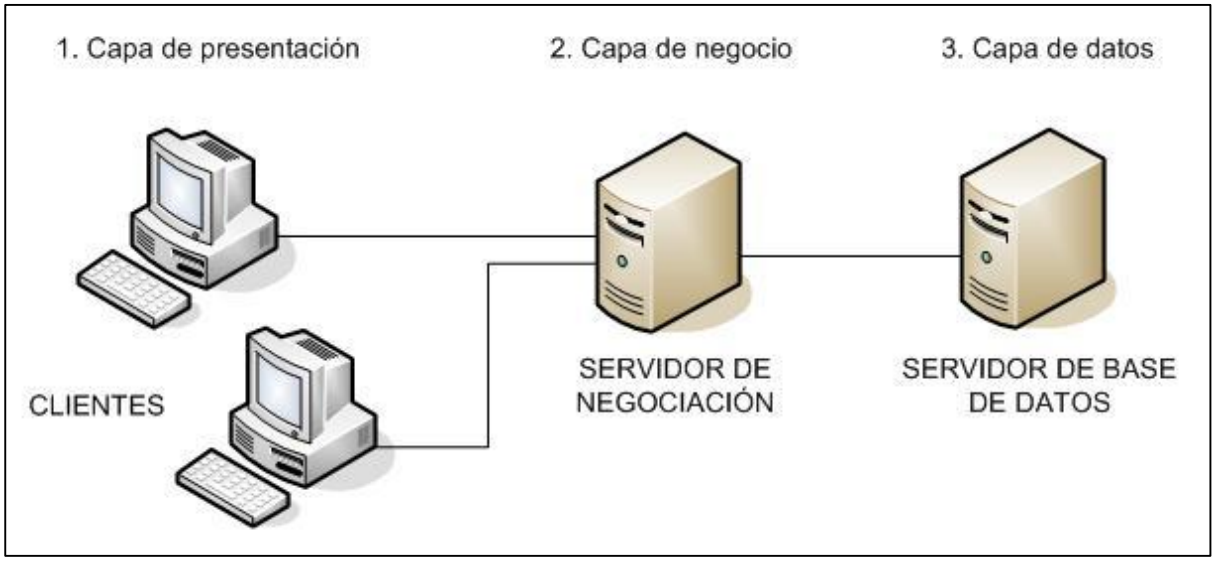

Figura 1: "Arquitectura de tres niveles" (EcuRed, s/f)

1.3.Paradigma de Programación Orientada a Objetos

"*La tecnología Orientada* a objetos se define como una metodología de diseño de software que modela las características de objetos reales o abstractos por medio del uso de clases y objetos" (De los Ángeles, 2003).

- Objeto

Los objetos es una particularización de una clase, es una entidad que posee atributos y efectúa acciones. Un objeto tiene sus atributos en variables y realiza acciones a través de funciones o métodos (De los Ángeles, 2003).

- Clase

"*Es un molde o bien prototipo en donde se definen los atributos (Variables) y las acciones (Métodos) comunes de una entidad*" (De los Ángeles, 2003).

1.3.1. Bases Fundamentales de la Programación Orientada a Objetos

Abstracción

Es un proceso que consiste en identificar las propiedades más importantes de un objeto y separarlas de las que son menos importantes. Es decir se determina los atributos y comportamientos que definen al objeto para luego convertirlo en un objeto de software (De los Ángeles, 2003).

Modularidad

Consiste en dividir la aplicación en varios módulos (Clases, Paquetes o bibliotecas), esto facilita la creación de la aplicación ya que se afronta el problema general en partes más pequeñas ("TPM | Tutorial de Programación Multiplataforma", s/f).

Encapsulamiento

Esta propiedad nos permite mantener la información de un objeto como desconocida para las demás entidades. "*Gracias al encapsulamiento se pueden definir los métodos y atributos de la clase* para que los objetos instanciados trabajen como unidades independientes de los demás objetos." Es decir, el encapsulamiento permite que se realice la modularidad y nos da una seguridad para prohibir la mala manipulación de los objetos.

Jerarquía

Existen dos relaciones jerárquicas principales en la Programación Orientada a Objetos, la de nominada "Es un" (Herencia) o "es parte de" (Agregación):

- Herencia o Generalización

  Es una propiedad que nos permite definir una clase a partir de otra clase más general. Es decir estamos definiendo un sub-clase de una super-clase.

- Agregación o inclusión

  "Es un agrupamiento lógico de objetos relacionados entre sí dentro de una clase"

Polimorfismo

El polimorfismo es una propiedad mediante la cual una entidad puede tomar diferentes formas (De los Ángeles, 2003).

1.4. Selección de la Plataforma

Para el presente proyecto se seleccionó una plataforma Windows ya que es el sistema más comúnmente usado por los usuarios finales.

1.4.1. Windows

Windows es un sistema operativo y por la definición de un sistema operativo, Windows es el programa principal de una computadora, el cual es el encargado de controlar su funcionamiento, los programas y la información que el dispositivo informático tenga ("Computación I: Unidad III", 2017).

1.4.2. Características de Windows

- Entorno gráfico sobre una pantalla.
- Controla el funcionamiento de los dispositivos periféricos con los que cuenta la computadora.
- Traduce los comandos a acciones realizadas con teclado o ratón.
- Realiza procesos multitareas.
- Registra el lugar donde se encuentran almacenados los archivos que han sido guardados en su computadora.
- Ejecuta los programas de aplicación que realizan tareas específicas.
- Pone la memoria de la computadora a disposición de los programas en ejecución de manera más eficiente.
- Dirige el tráfico de los datos que entran y salen del procesador de su computadora para que no se generen conflictos en su procesamiento.

1.5. Base de Datos

1.5.1. Bases de Datos Relacionales

"*Una base de datos relacional es un repositorio de datos*." IBM fue la precursora de las bases de datos relacionales al lanzar su primer prototipo de base de datos en su laboratorio de investigación San Jose. Además de crear el predecesor del lenguaje SQL el lenguaje Sequel. Una base de datos relacional está basada en relaciones, las cuales tienen un nombre único para su representación. Estas relaciones representan una dependencia entre las entidades que se manejan dentro de un modelo de base de datos Entidad-Relación (Silberschatz, Korth, & Sudarshan, 2002).

1.5.2. Lenguaje SQL

Es un lenguaje que permite realizar consultas estructuradas, está catalogado como un lenguaje de alto nivel y como un lenguaje estándar para bases de datos relacionales. SQL está comprendido por tres tipos de sentencias, los cuales son: Lenguaje de definición de Datos (DDL), Lenguaje de Manipulación de datos (DML) y Lenguaje de Control de datos (DCL) (Garcia, s/f).

1.5.2.1. Lenguaje de Definición de Datos (DDL)

"*Son una agrupación de sentencias SQL que permiten la definición y declaración de objetos de la base de datos.*" Estos objetos pueden ser: Database, Table, View, Index, Precedure, Trigger, Rule, Domain y Default (Garcia, s/f).

1.5.2.2. Lenguaje de Manipulación de Datos (DML)

"Grupo de sentencias SQL que permiten manipular los datos que están almacenados en las bases de datos a nivel de filas y/o columnas." Esto quiere decir que las DML permiten realizar consultas, modificaciones, eliminaciones o inserciones en las tablas de las bases de datos. Los comandos DML son: Insert, Update, Delete y Select (Garcia, s/f).

1.5.2.3. Lenguaje de Control de Datos (DCL)

Agrupación de sentencias SQL que permiten realizar un control sobre las funciones administrativas que realiza un motor de base de datos como: Atomicidad y Seguridad. Los comandos DCL son los siguientes: Commit Transaction, Rollback Transaction, Grant y Revoke (Garcia, s/f).

1.6. Lenguaje de Programación

El lenguaje seleccionado para el proyecto de desarrollo es el lenguaje JAVA por el motivo de ser uno de los lenguajes más usados en el mercado y por su facilidad de estructuración de programas.

1.6.1. JAVA

Java comienza siendo un lenguaje llamado OAK, el cual fue creado por James Gosling y Bill Joy. Oak fue un lenguaje diseñado para crear software para la televisión interactiva. Oak contaba con las siguientes características:

- Pequeño
- Robusto
- Independiente de la máquina
- Orientado a objetos

Al ver fracasar el proyecto de televisión y junto él, el lenguaje de programación el lenguaje Oak se empezó con el desarrollo del lenguaje JAVA bajo el lema "La red es la computadora." Los criterios para desarrollar JAVA fueron:

- Independiente de la máquina
- Seguro para trabajar en red
- Potente para sustituir código nativo

1.6.2. Características del Lenguaje

Java es un lenguaje que es compilado e interpretado. Todo programa al compilarse genera un archivo en bytecode que es interpretado por la máquina virtual, gracias a esto es lo que consigue la multiplataforma ya que el código no es directamente interpretado por el sistema operativo si no por una máquina virtual que depende del sistema operativo pero consigue interpretar el código Java para que el sistema operativo nativo lo ejecute. Java es un lenguaje orientado a objetos, teniendo una sintaxis muy parecida a la de C y C++ (Belmonte, 2005).

Java con respecto a la seguridad ha tomado varias medidas tanto por el lado del programador como de la ejecución en la máquina virtual. Del lado del programador se realiza una comprobación muy estricta de tipos durante la compilación, evitando así los desbordamientos de la pila, problema muy frecuente en otros lenguajes (Belmonte, 2005).

Al crear una instancia en Java se necesita del operador new() el cual permite que un recolector de basura de Java libere la memoria ocupada por objetos que ya no se encuentren referenciados, esto se realiza a través de la máquina virtual de Java que interpreta el código java y gestiona la memoria dinámicamente. Esto viene estrechamente relacionado con el uso de punteros que son una fuente de problemas a la hora de programar por sus referencias a memoria, para solucionar esto, Java creo las referencias para no utilizar punteros, las cuales permiten el acceso a las instancias de clase (Belmonte, 2005).

Las excepciones que se dan en el tiempo de ejecución a las cuales todo programador debe enfrentar pueden ser solucionadas de una manera más fácil gracias a los procedimientos que Java define para solucionar estos problemas, uno de los procedimientos principales para el tratamiento de excepciones es que Java antes de la compilación comprueba que el código no viole ninguna restricción de seguridad del sistema en el cual se va a ejecutar lo que además aumenta la seguridad del lenguaje al no permitir que se violente el sistema base (Belmonte, 2005).

1.6.3. Herramientas de Desarrollo

En el mundo de Java las herramientas de desarrollo se conocen como JDK que por sus siglas en inglés significa Java Development Kit dentro de este conjunto de herramientas se encuentra un compilador de Java conocido como Javac, podemos encontrar la máquina virtual de java que permite la ejecución de programas escritos en Java, una herramienta de documentación de código conocida como javadoc y por último una herramienta que permite empaquetar el código en un archivo .jar. Un detalle fundamental para poder utilizar estas herramientas es señalar a la máquina virtual en donde puede encontrar las librerías que no se encuentran en el paquete básico, esto se realiza mediante el comando CLASSPATH (Belmonte, 2005).

CAPÍTULO 2: METODOLOGÍA DE DESARROLLO RUP

En el capítulo siguiente se revisaran los fundamentos teóricos de la metodología RUP que permitirán el correcto desarrollo de software.

Es un proceso de ingeniería de software que provee una manera disciplinada al momento de asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que cumpla con las necesidades de los usuarios finales, con tiempos y presupuestos predecibles (Rational, 2011).

2.1. Definición

RUP es un proceso de producto que se desarrolla y mantiene por la empresa Rational Software, un equipo de desarrollo de RUP trabaja junto con los clientes, grupos de productos Rational y la organización de consultoría Rational, para asegurar que el proceso es constantemente actualizado y mejorado. RUP mejora la productividad de un equipo, proveyendo a cada miembro del equipo con un acceso fácil al conocimiento basado en líneas base, marcos de trabajo y herramientas para el desarrollo de actividades críticas. Al tener una base de conocimiento igual aseguramos que todos los miembros del equipo comparten un lenguaje, procesos y una visión común de cómo desarrollar (Rational, 2011).

RUP tiene actividades que crean y mantienen modelos, en vez de enfocarse en la producción de una larga cantidad de documentación, el proceso RUP enfatiza el desarrollo y mantenimiento de modelos que representan sistemas de software que se encuentran en desarrollo. Una de las principales características del RUP es que es una guía para la utilización del lenguaje de modelado unificado UML. Cuenta con una gran variedad de herramientas que automatizan partes del proceso. Estas son usadas para crear y mantener los modelos del proceso de ingeniería de software como: modelos visuales, programación, pruebas, entre otros (Rational, 2011).

RUP es un proceso configurable ya que se puede adaptar para pequeños equipos de desarrollo así como organizaciones que manejan grandes equipos de desarrollo, el proceso unificado se encuentra fundamentado en una simple y clara arquitectura la cual puede ser modificada dependiendo de la situación (Rational, 2011).

2.1.1. UML

UML es un lenguaje de modelado que comprende una serie de modelos y elementos gráficos que pueden ser combinados para conformar diagramas. Al ser un lenguaje se tienen diversas reglas para crear los diferentes 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."*

2.1.1.1. Diagrama de Clases

Los diagramas de clases representan una estructura estática del sistema. Las cosas que existen se aglomeran en categorías. Cada clase viene a ser una categoría o grupo de cosas que poseen atributos y realizan acciones similares (Diagramas UML, s/f).

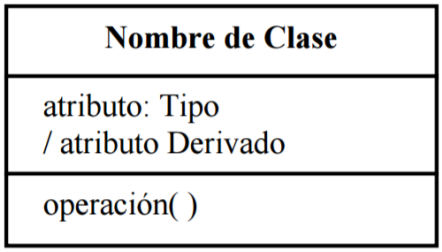

Figura 2: Diagramas de Clases (Diagramas UML, s/f)

Multiplicidad

Son símbolos que indican el número de instancias de una clase vinculada a otra clase. En la Figura 4 se puede ver un ejemplo de multiplicidad.

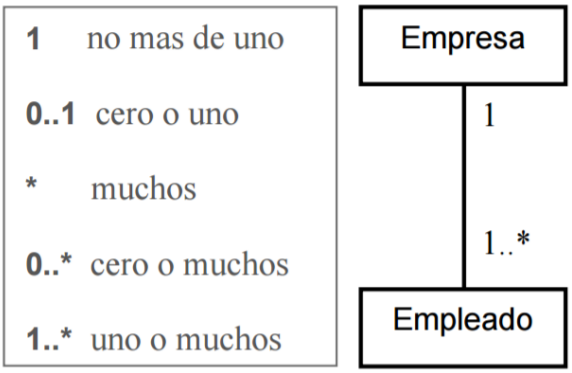

Figura 5: Multiplicidad entre clases (Diagramas UML, s/f)

Asociaciones

Las asociaciones representan las relaciones estáticas de las clases y su nomenclatura se las puede ver en la Figura 3.

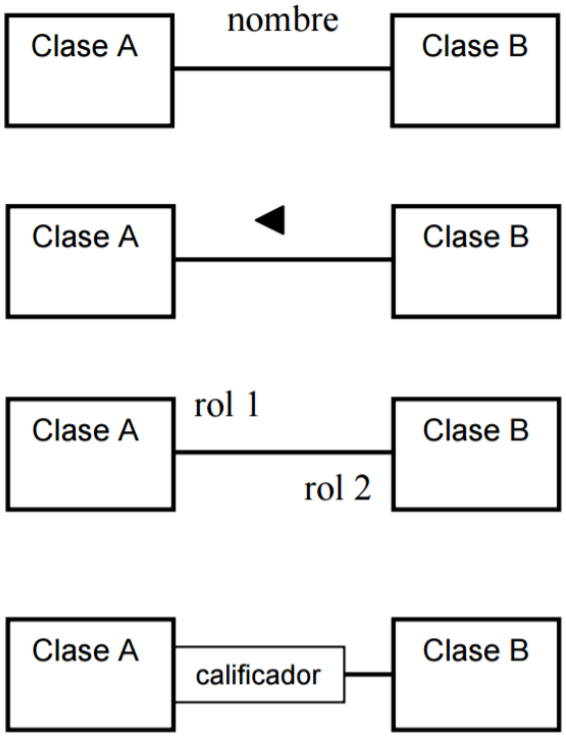

Figura 4: Asociaciones entre clases (Diagramas UML, s/f)

Composición y Agregación

La agregación es una relación en la que la clase "Todo" juega un rol más importante que la clase "Parte" pero las dos clases no son dependientes una de otra, se grafica mediante el robo vacío apuntado hacia la clase Todo. La composición es un tipo especial de agregación que denota que la clase Todo es poseedora de la Clase Parte, se grafica con un rombo pintado apuntando hacia la clase Todo, para entender mejor lo antes mencionado se puede observar el ejemplo en la Figura 5.

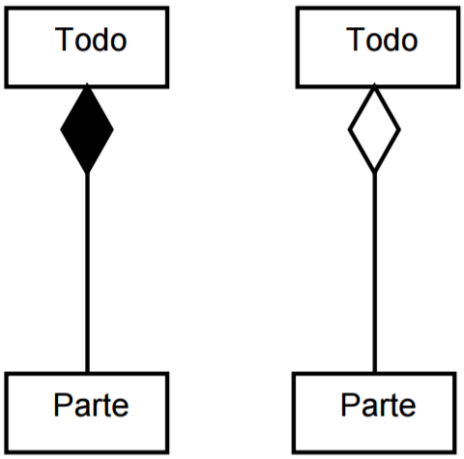

Figura 3: Composición y Agregación (Diagramas UML, s/f)

Generalización

Básicamente es la herencia de clases, que representa una relación entre dos clases en donde una clase específica hereda los atributos y acciones de una clase más general. Ejemplo Figura 6.

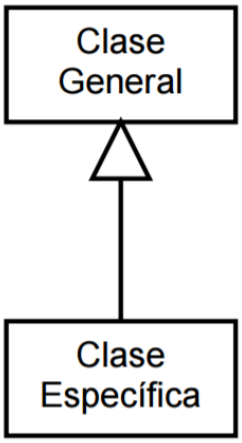

Figura 4: Generalización entre clases (Diagramas UML, s/f)

2.1.1.2. Casos de Uso

Los casos de uso son descripciones de acciones que realiza un sistema desde el punto de vista del usuario. Modelan la funcionalidad del sistema usando actores y casos, donde los casos son funciones del sistema para los usuarios, se podrá ver un ejemplo de un caso de uso con sus actores y casos respectivos en la Figura 7. (*Diagramas UML*, s/f).

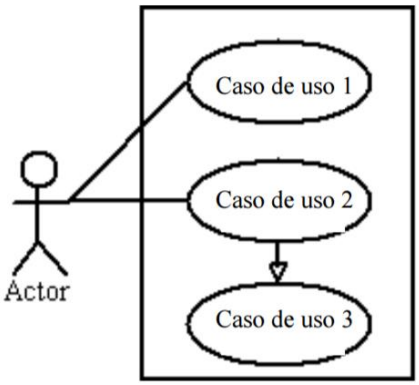

Figura 5: Diagramas de casos de uso (Diagramas UML, s/f)

2.1.2. Seis Buenas Prácticas Para un Desarrollo Efectivo

RUP describe como desplegar comercialmente el desarrollo de software para equipos de desarrollo, estas son las llamadas las buenas prácticas. Las buenas prácticas se encuentran categorizadas en seis temas diferentes:

Desarrollo de Software Iterativo

Dado que en la actualidad se manejan sistemas de software sofisticados, no es posible definir el problema por completo, diseñar una solución completa, desarrollar el software y luego probar el producto al final como se realiza en la metodología de cascada. Una metodología iterativa es necesaria ya que permite un entendimiento creciente del problema a través de refinamientos sucesivos para lograr una solución efectiva en múltiples iteraciones (Rational, 2011).

Gestionar Requerimientos

El proceso racional unificado describe cómo organizar y documentar funcionalidades y restricciones, realiza un seguimiento y documenta las decisiones y puede capturar fácilmente los requerimientos del usuario. Las nociones de los casos de uso y los escenarios del proceso han probado ser una excelente forma de capturar los requerimientos funcionales y asegurar que estos lleguen al diseño del sistema (Rational, 2011).

Usar Arquitecturas Basadas en Componentes

El proceso se enfoca en desarrollo previo y líneas base para poder tener una robusta arquitectura, antes de revisar los recursos para un desarrollo a gran escala. Describe

como diseñar una arquitectura flexible, que se acomode al cambio y que sea entendible (Rational, 2011).

Modelos de Software Visuales

El proceso muestra como modelar el software visualmente para capturar la estructura y el comportamiento de arquitecturas y componentes, esto te permite esconder detalles y escribir código usando bloques de construcción visuales (Rational, 2011).

Verificar la Calidad del Software

La calidad debe ser revisada con sumo respeto a los requerimientos basados en la confiabilidad, funcionalidad, capacidad de la aplicación y capacidades del sistema.

Control de Cambio

La capacidad de realizar un seguimiento de los cambios es fundamental en un ambiente en donde el cambio es inevitable, este proceso describe como controlar, seguir y monitorear los cambios para poder tener un desarrollo iterativo exitoso (Rational, 2011).

2.2. Estructura del Proceso

El proceso puede ser descrito en dos dimensiones en donde el eje horizontal muestra el tiempo y muestra un aspecto dinámico del proceso mientras es ejecutado, y está expresado en ciclos, fases, iteraciones e hitos. El eje vertical representa el aspecto estático del proceso, describe las actividades, artefactos y flujos de trabajo un ejemplo del flujo del RUP se lo puede ver en la figura 8. (Rational, 2011).

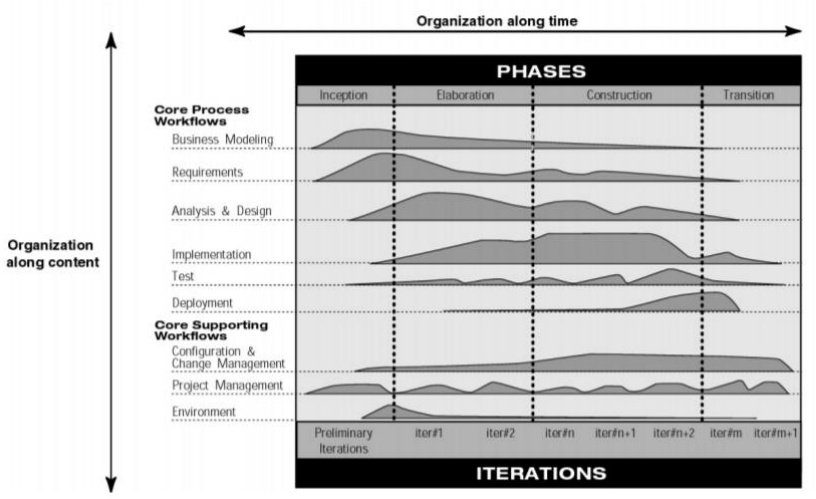

Figura 6: Ciclo de vida en RUP (Rational, 2011).

El ciclo de vida de RUP se divide en cuatro fases consecutivas que son: inicio, elaboración, construcción y transición.

2.2.1. Inicio

Durante esta fase se establece el caso de negocio para el sistema y se delimita el alcance del proyecto, para lograr esto se debe identificar todas las entidades externas con las que el sistema va a interactuar (Actores) y definir la naturaleza de esta interacción en un alto nivel de abstracción, esto involucra identificar todos los casos de uso y describir los más significantes (Rational, 2011).

Los resultados de la fase inicial son:

- Un documento de Visión: una visión general del núcleo del proyecto, sus requerimientos, características clave y restricciones principales.
- Un modelo de casos de uso inicial (10% - 20% completo).
- Un glosario del proyecto inicial.
- Un caso de negocio inicial, que incluye el contexto del negocio, criterios de éxito y previsiones financieras.
- Un documento inicial de riesgos.
- Un plan del proyecto, mostrando fases e iteraciones.
- Un modelo de negocio si es necesario.
- Uno o varios prototipos.

Al final de la fase de inicio se encuentra el primer hito principal del proyecto, llamado objetivos del ciclo de vida. Se deben considerar ciertos aspectos, para poder seguir a la siguiente fase, que son:

- Consenso con los interesados tanto en la definición y costos/tiempos estimados.
- Entendimiento de los requerimientos como evidencia de la fidelidad de los casos de uso.
- Credibilidad en el costo/tiempo estimados, prioridades, riesgos y proceso de desarrollo.
- Revisión de cualquier prototipo arquitectónico que se haya desarrollado.
- Gastos reales versus gastos planeados.

Nota: El proyecto puede llegar a ser cancelado o completamente modificado si falla al momento de pasar este hito.

2.2.2. Fase de Elaboración

El principal objetivo de la fase de elaboración es analizar el problema, establecer una base arquitectónica, desarrollar el plan del proyecto y eliminar los elementos de alto riesgo del proyecto. Esta fase es considerada la fase 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, 2011).

Los resultados de la fase de elaboración son:

- Modelo de Casos de uso
- Requerimientos no funcionales y requerimientos que no están asociados con los casos de uso.
- Descripción de la arquitectura de software.
- Un prototipo ejecutable
- Una lista de riesgos revisada
- Un plan de desarrollo para todo el proyecto
- Un preliminar manual de usuario (Opcional)

Al final de la fase se encuentra el segundo hito importante del proyecto llamado arquitectura de ciclo de vida. En este punto se examina a detalle los objetivos del sistema y su alcance, la elección de arquitectura y la resolución de riesgos más significativos (Rational, 2011).

Los criterios de evaluación de la fase de elaboración involucran las respuestas a estas preguntas:

- ¿Es la visión del producto estable?
- ¿Es la arquitectura estable?
- ¿La demostración ejecutable muestra que los elementos de mayor riesgo han sido identificados y solucionados?
- ¿Es el plan para la fase de construcción lo suficientemente detallado y preciso?
- ¿Todos los interesados están de acuerdo que la visión actual puede ser alcanzada si el plan es ejecutado para desarrollar el sistema completo?
- ¿Es el plan actual de uso de recursos aceptable frente al plan de gastos?

Nota: El proyecto puede llegar a ser cancelado o completamente modificado si falla en cualquiera de estos puntos.

2.2.3. Fase de Construcción

En la fase de construcción todos los componentes y características de la aplicación son desarrollados e integrados en el producto. La fase de construcción es básicamente el proceso de fabricación de software, el cual se enfoca en manejar recursos y controlar operaciones para optimizar costos, tiempos y calidad (Rational, 2011).

Los resultados de la fase de Construcción son:

- El producto de software integrado en las plataformas adecuadas.
- Los manuales de usuario.
- Una descripción de la versión actual.

Al final de la fase de construcción nos encontramos con el tercer hito importante del proyecto llamado capacidad operativa inicial. En este punto se decide si el software, las instalaciones y los usuarios están listos para poner operativo el sistema, sin exponer a riesgos al proyecto. Comúnmente a esta parte se la conoce como lanzamiento beta (Rational, 2011).

Criterios de evaluación de la fase de construcción involucran responder estas preguntas:

- ¿Es el producto estable y suficientemente maduro como para ser usado por los usuarios?
- ¿Los interesados están listos para la transición al software?
- ¿Es el plan actual de uso de recursos aceptable frente al plan de gastos?

Nota: La transición puede ser pospuesta si el proyecto falla en pasar este hito.

2.2.4. Fase de Transición

El propósito principal de la fase de transición es pasar el producto de software a la comunidad de usuarios, una vez que ha sido entregado a los usuarios finales usualmente se generan ciertos problemas que requieren que se realicen nuevas entregas del producto de software. La fase de transición se enfoca en las actividades requeridas para poner el software en manos de los usuarios, usualmente esta fase tiene varias iteraciones (Rational, 2011).

Esta fase incluye los siguientes puntos:

- Pruebas beta, que permiten validar el nuevo sistema a las expectativas de los usuarios.
- Conversión de bases de datos operacionales.
- Entrenamiento de usuarios y técnicos de mantenimiento de software.
- Enviar el producto a los departamentos de marketing, distribución y ventas.

Los objetivos principales de la fase de transición son:

- Lograr que el usuario pueda solucionar los problemas por sí solo.
- Lograr un producto final rápidamente y con costos eficientes.

Al final de la fase de transición nos encontramos con el último hito importante del proyecto, llamado lanzamiento del producto. En este punto se decide si los objetivos fueron cumplidos y si se debería empezar un nuevo ciclo de desarrollo (Rational, 2011).

La primera evaluación de la fase de transición involucra responder a estas preguntas:

- ¿Está el usuario satisfecho?
- ¿Es el plan actual de uso de recursos aceptable frente al plan de gastos?

2.2.5. Iteraciones

Cada fase en el proceso de desarrollo unificado RUP puede ser dividida en varias iteraciones. Donde una iteración es: "*un bucle de desarrollo completo que tiene como resultado un entregable (interno o externo) de un producto ejecutable, el cual es un subconjunto de producto final, que va creciendo incrementalmente en cada iteración.*"

Beneficios de un desarrollo iterativo:

- Los riesgos son mitigados antes.
- El cambio es manejable
- Mayor nivel de reúso
- El equipo del proyecto puede aprender en el camino
- Mejor calidad general

CAPÍTULO 3: INICIO

En el siguiente capítulo se realizará lo correspondiente a la fase de inicio de la metodología de desarrollo RUP. Lo que implica básicamente realizar un análisis de requerimientos y realizar un diseño inicial para verificar la viabilidad del proyecto.

3.1. Definición General del Problema

3.1.1. Antecedentes y Motivación

Se ha decidido realizar el diseño y desarrollo de una aplicación que agilite el proceso de registro y obtención de reportes de fichas psicológicas y estudiantiles de la fundación IEPNI, para brindar mayor comodidad a los administradores y profesores al momento de gestionar la información estudiantil. Se propone realizar un repositorio de datos que podrá ser gestionada mediante una aplicación cliente servidor.

La institución hoy en día maneja una gran cantidad de información la cual se encuentra archivada en grandes carpetas y libreros lo que dificulta mucho generar reportes y buscar la información puntual.

3.1.2. Estrategia

El proyecto constará de dos ciclos de desarrollo siguiendo la metodología de desarrollo RUP. En el primer ciclo se realizará el desarrollo completo del software, una vez finalizado el software será revisado por el cliente para especificar qué cambios deban ser realizados. En el ciclo dos se tomarán en cuenta todas las opiniones del cliente y se procederá a realizar dichos cambios en el software.

La aplicación será desarrollada utilizando el IDE de Java Netbeans 8.0.2 y la base de datos a utilizar será PostgreSQL.

3.2. Requerimientos Funcionales

Los requerimientos funcionales serán detallados mediante casos de uso de UML, los cuales estarán divididos en dos niveles de abstracción para el mejor entendimiento de los mismos.

3.2.1. Diagrama General

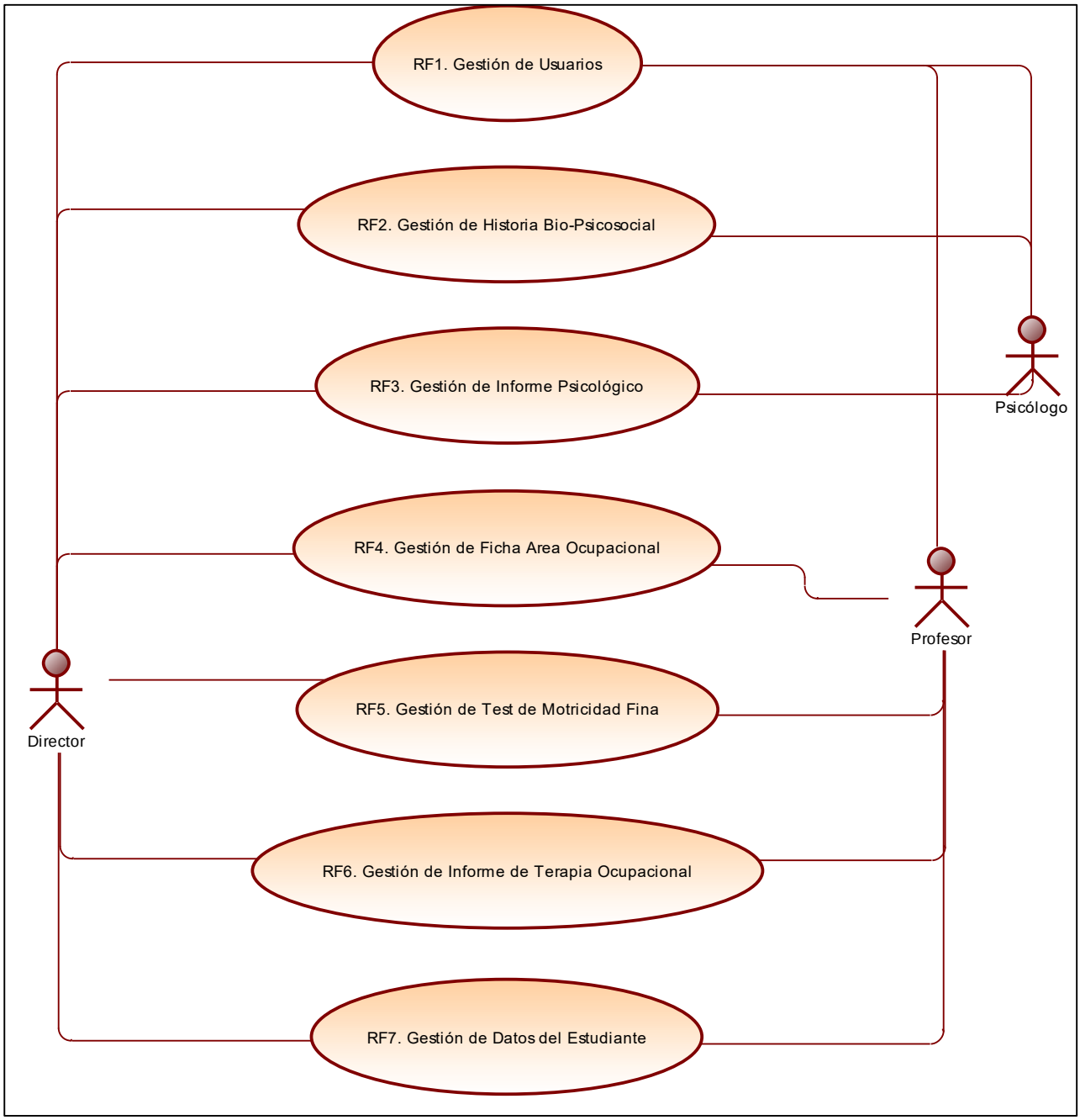

Figura 7: Diagrama General de Casos de Uso (Sánchez, 2017)

3.2.2. Diagramas Específicos

RF1. Gestión de Usuarios

El director será capaz de añadir, eliminar o modificar usuarios del sistema, mientras que los profesores y psicólogos solo podrán modificar sus contraseñas.

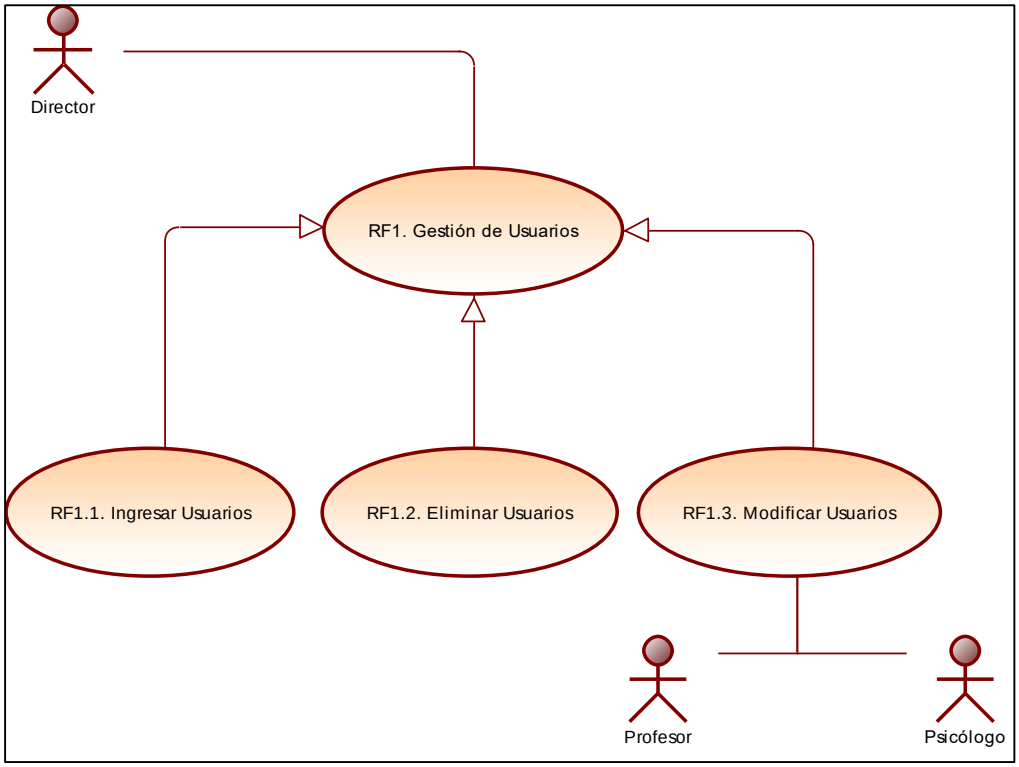

Figura 8: Diagrama de Caso de Uso RF1. Gestión de Usuarios (Sánchez, 2017)

RF2. Gestión de Historia Bio-Psicosocial

El psicólogo podrá añadir, eliminar, modificar y generar reportes de las fichas de historia bio-psicosocial, mientras que el director solo podrá generar reportes.

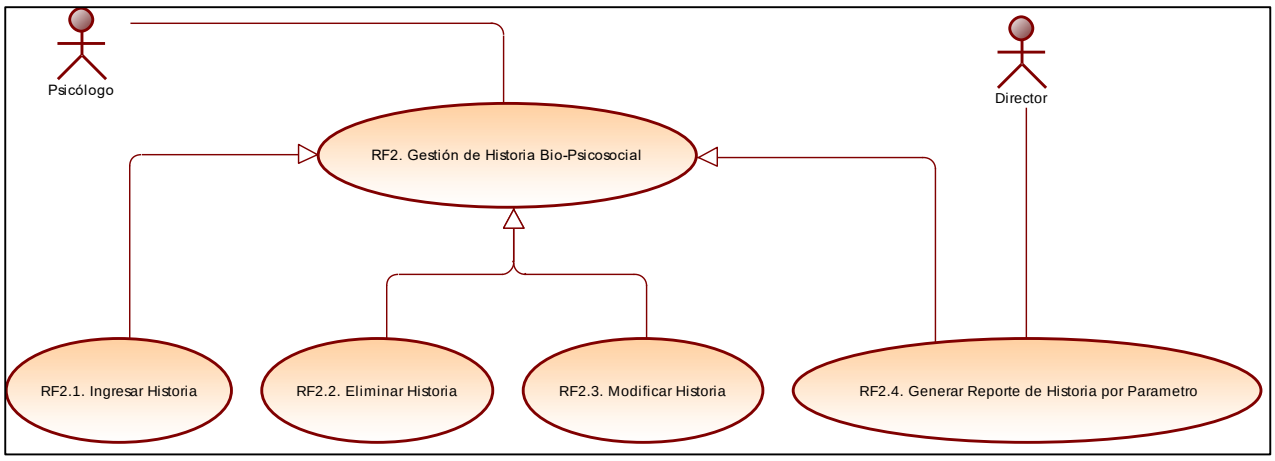

Figura 9: Diagrama de Caso de Uso RF2. Gestión de Historia Bio-Psicosocial (Sánchez, 2017)

RF3. Gestión de Informe Psicológico

El psicólogo podrá añadir, eliminar, modificar y generar reportes de las fichas del informe psicológico, mientras que el director solo podrá generar reportes.

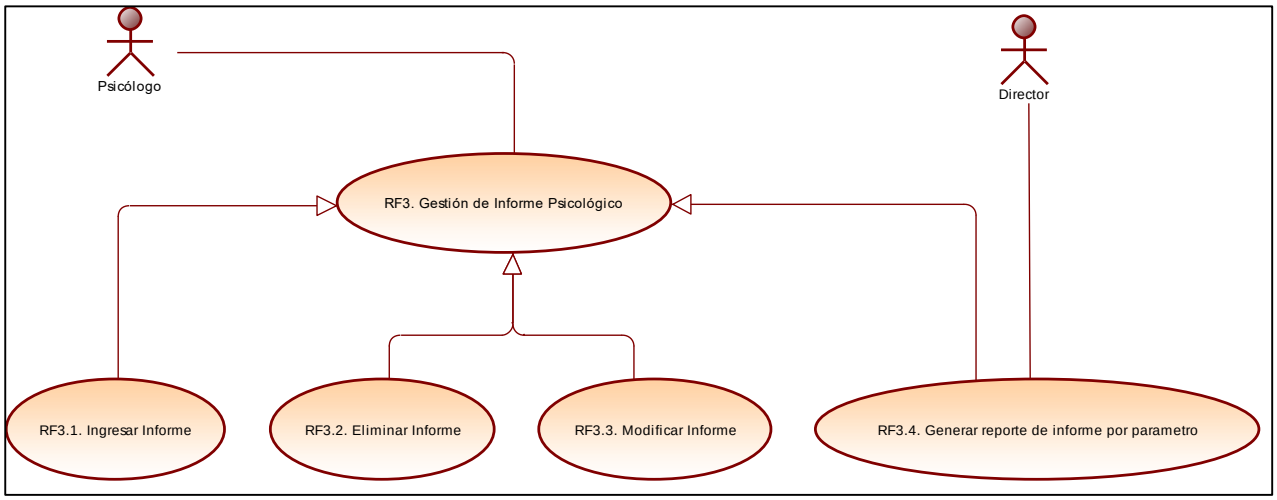

Figura 10: Diagrama de Caso de Uso RF3. Gestión de Informe Psicológico (Sánchez, 2017)

RF4. Gestión de Ficha de Área Ocupacional

El profesor podrá añadir, eliminar, modificar y generar reportes de las fichas del área ocupacional, mientras que el director solo podrá generar reportes.

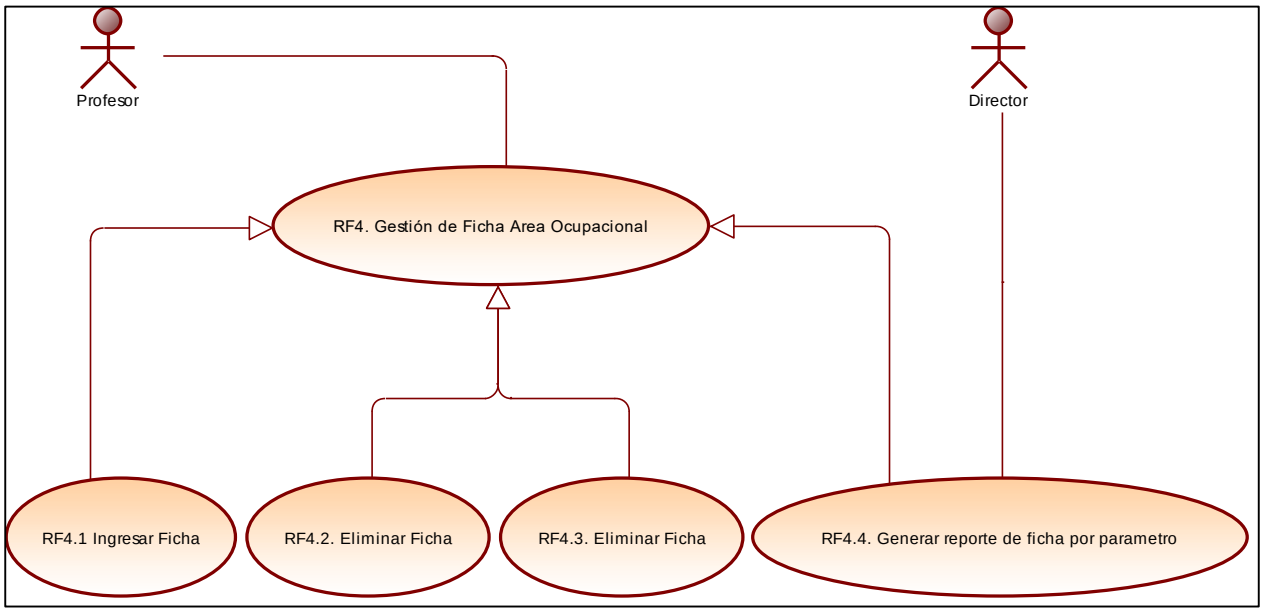

Figura 11: Diagrama de Caso de Uso RF4. Gestión de Ficha de Área Ocupacional (Sánchez, 2017)

RF5. Gestión de Test de Motricidad Fina

El profesor podrá añadir, eliminar, modificar y generar reportes de las fichas del test de motricidad fina, mientras que el director solo podrá generar reportes.

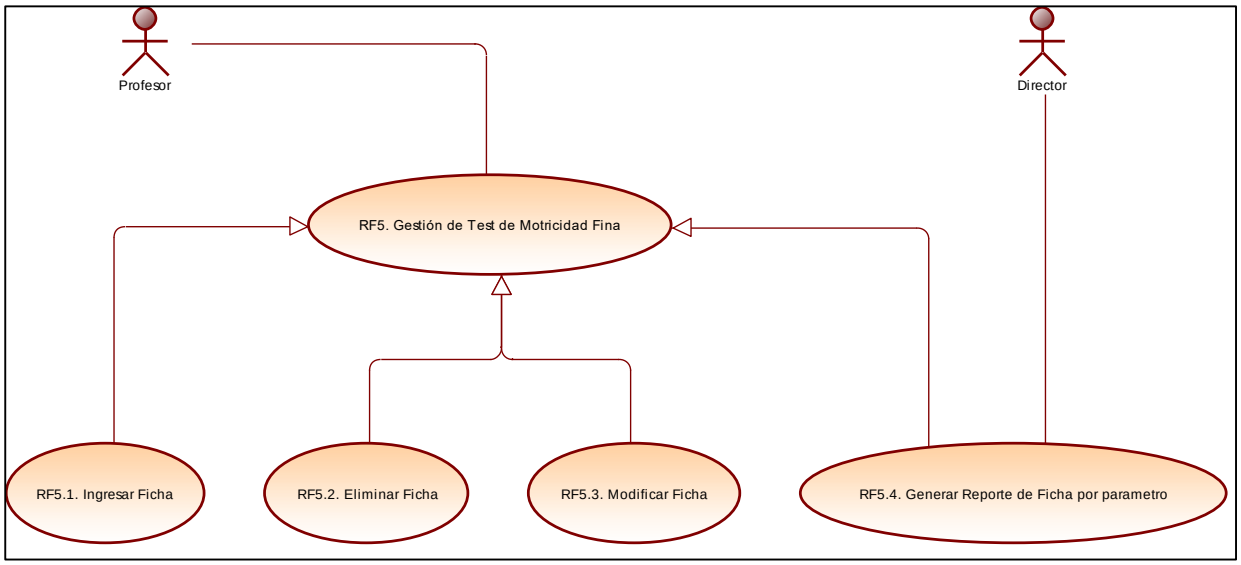

Figura 12: Diagrama de Caso de Uso RF5. Gestión de Test de Motricidad Fina (Sánchez, 2017)

RF6. Gestión de Informe de Terapia Ocupacional

El profesor podrá añadir, eliminar, modificar y generar reportes del informe de terapia ocupacional, mientras que el director solo podrá generar reportes.

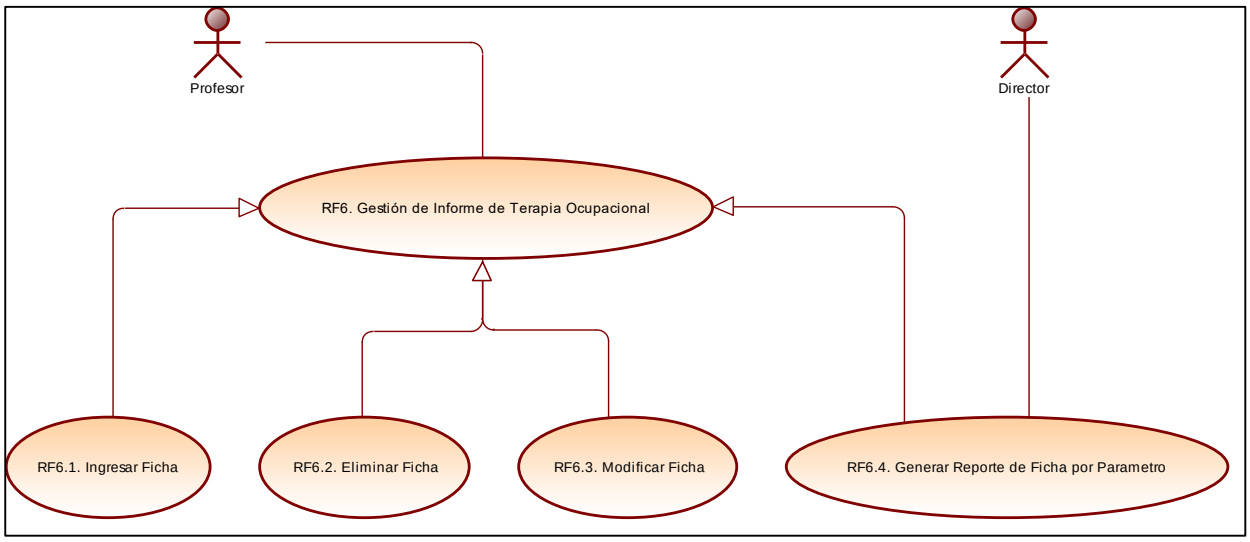

Figura 13: Diagrama de Caso de Uso RF6. Gestión de Informe de Terapia Ocupacional (Sánchez, 2017)

RF7. Gestión de Datos del Estudiante

El profesor podrá añadir, eliminar, modificar y generar reportes de los datos del estudiante, mientras que el director solo podrá generar reportes.

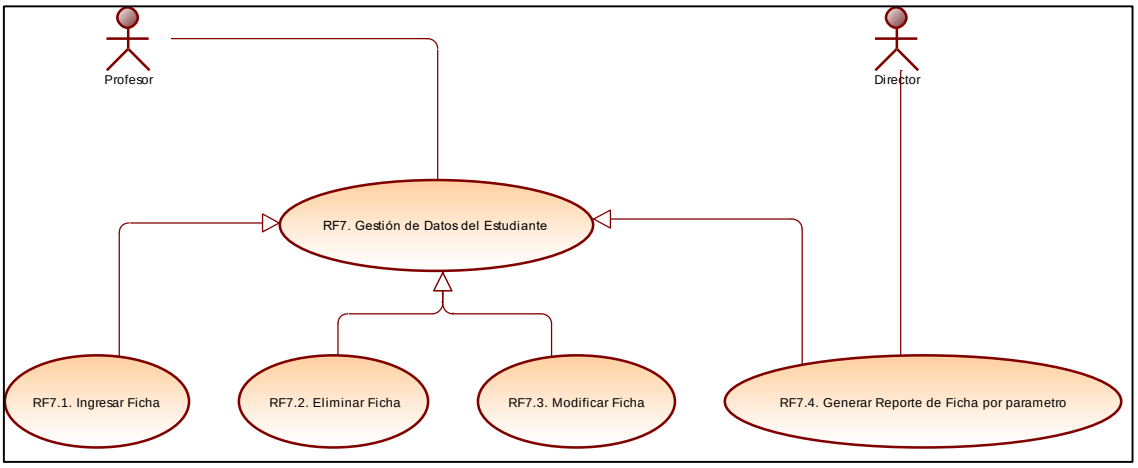

Figura 14: Diagrama de Caso de Uso RF7. Gestión de Datos del Estudiante (Sánchez, 2017)

3.2.3. Diagrama de Actividades

El sistema cuenta con dos procesos principales que son el manejo de fichas y el manejo de usuarios, en los dos diagramas siguientes se podrá observar cómo funcionan cada uno de estos procesos.

DA1. Diagrama de Actividades de Usuarios

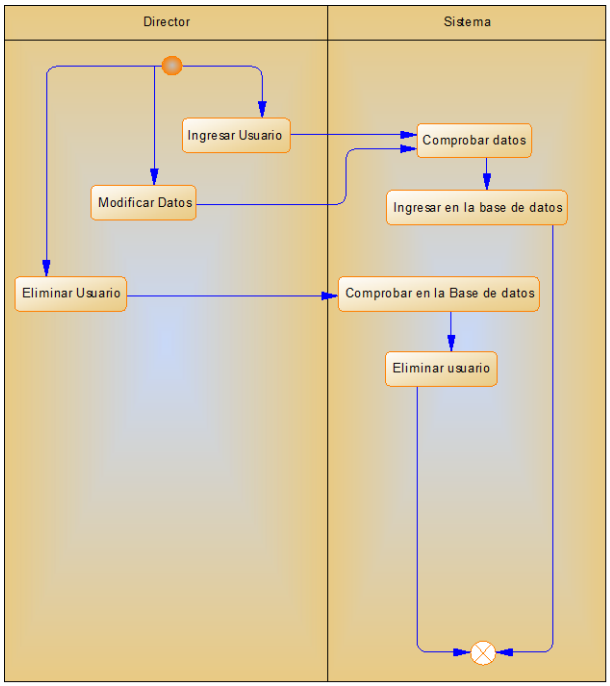

Figura 15: Diagrama de Actividades DA1. Diagrama de Actividades de Usuarios (Sánchez, 2017)

DA2. Diagrama de Actividades de Gestión de Fichas

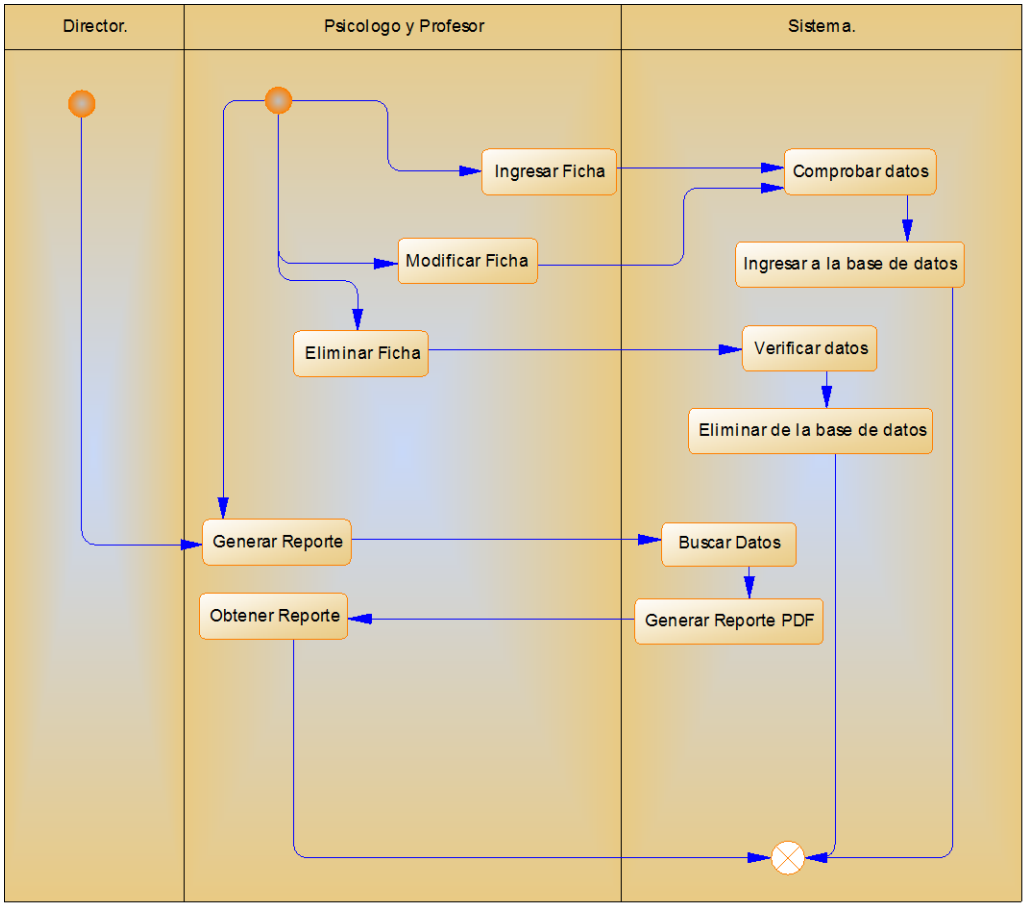

Figura 16: Diagrama de Actividades DA2. Diagrama de Actividades de Gestión de Fichas (Sánchez, 2017)

3.3. Otros Requerimientos

El sistema necesita de requerimientos tanto de software como de hardware para ser desarrollado y a su vez el cliente también necesita tener algunos requerimientos para que funcione el programa ya desarrollado.

3.3.1. Requerimientos de Hardware

	Cliente	**Desarrolladores**
Procesador	Intel Centrino	Intel Inside i7 5ta Gen
RAM	4Gb	12Gb
HDD	200Mb	1Tb
Tarjeta de Video	N/A	AMD Radeon R7 4GB

Tabla 1: Requerimientos de Hardware (Sánchez, 2017)

3.3.2. Requerimientos de Software

	Cliente	**Desarrolladores**
Sistema Operativo	Windows XP	Windows 10
JDK	Java SE 8.0	Java SE 8.0

Tabla 2: Requerimientos de Software (Sánchez, 2017)

3.4. Diseño Conceptual

En la Figura 19 podemos observar el diagrama conceptual de la base de datos el cual nos muestra cómo estará estructurada la base de datos. Para el sistema actual se consideró dos tablas principales que son Estudiante y Usuario ya que ellos son los que están relacionados con las tablas de cada una de las fichas. A su vez algunas fichas se subdividieron en partes más pequeñas para lograr un mejor entendimiento de la estructura de la base de datos.

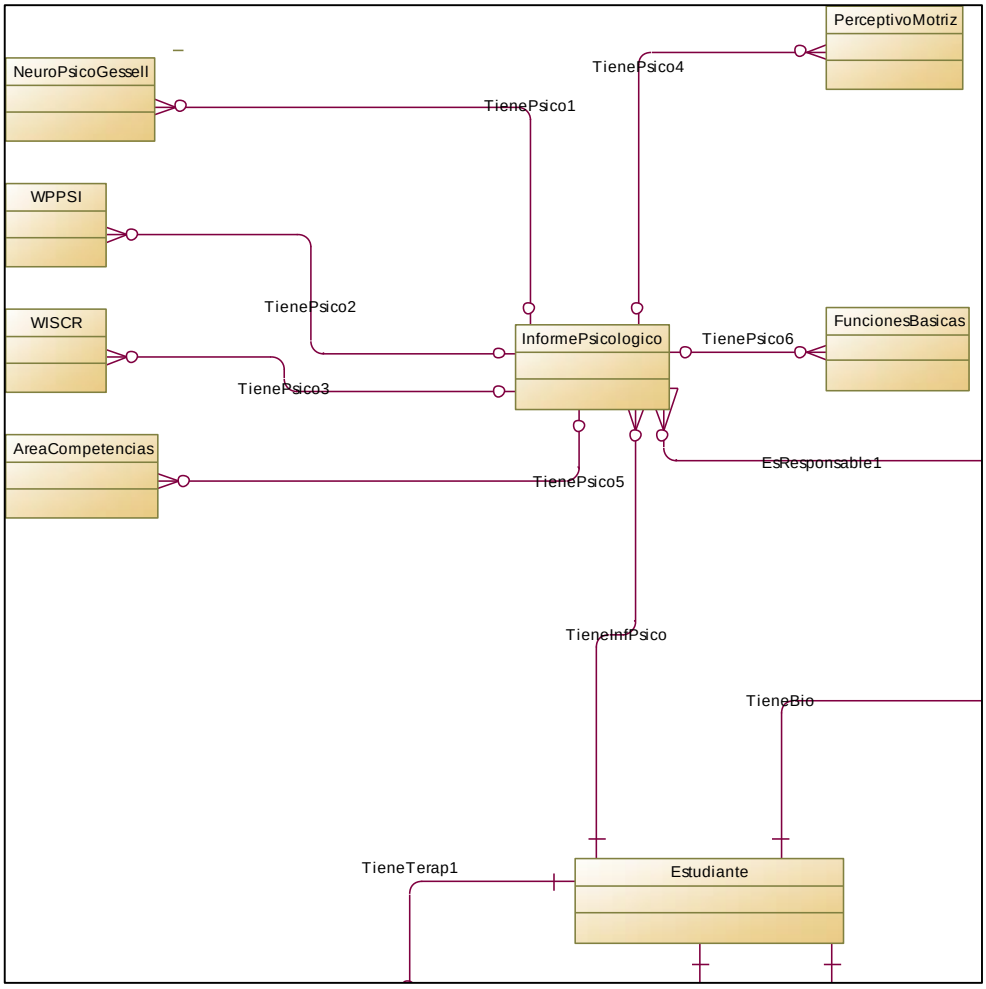

Figura 17: Diagrama Conceptual de la Base de Datos (Sánchez, 2017)

Para mayor detalle por favor revisar el Anexo1. Modelo Conceptual BDD que se encuentra en el CD.

CAPÍTULO 4: ELABORACIÓN

En el siguiente capítulo se explicará la estructura del sistema y de la base de datos, mediante los diagramas de clases, Entidad/Relación y secuencia.

4.1. Diagrama de Clases

Se muestran los diagramas de clases de las tres capas existentes en el sistema, el diagrama de ventanas de usuario GUI, diagrama de dominio del problema del sistema y por último la capa de manejo de datos.

4.1.1. Diagrama de Clases GUI

La Figura 20 muestra cómo va a estar estructurado el manejo de clases de ventanas del sistema.

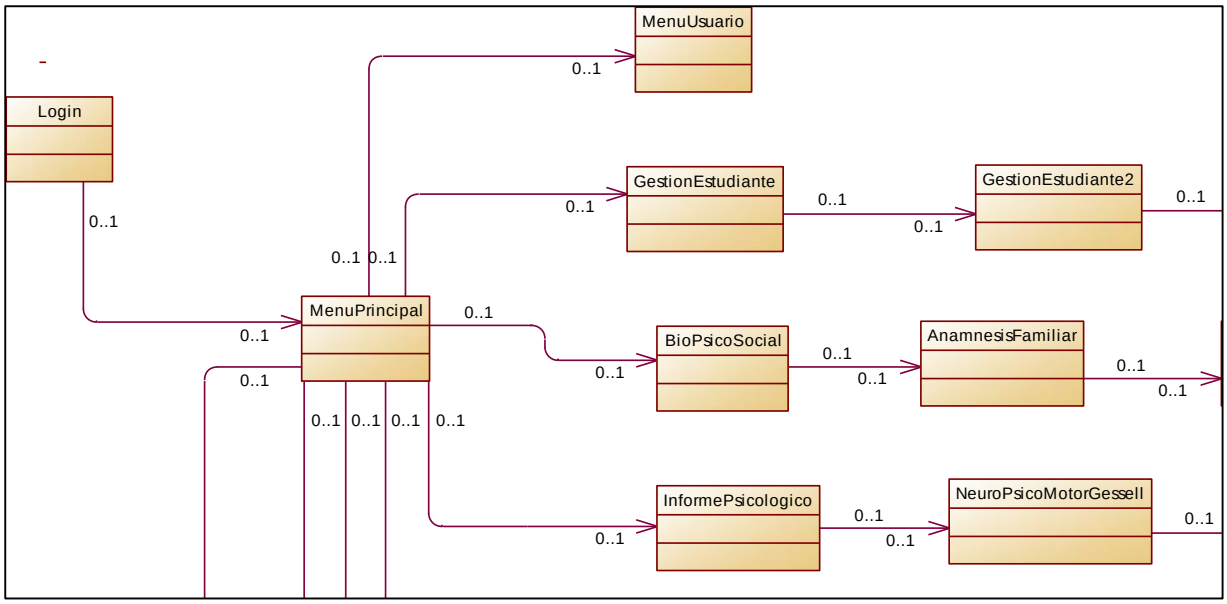

Figura 18: Diagrama de Clases GUI (Sánchez, 2017)

Para mayor detalle por favor revisar el anexo 2. Diagrama de clases y secuencia, que se encuentra en el CD.

4.1.2. Diagrama de Clases del Dominio del Problema Conceptual

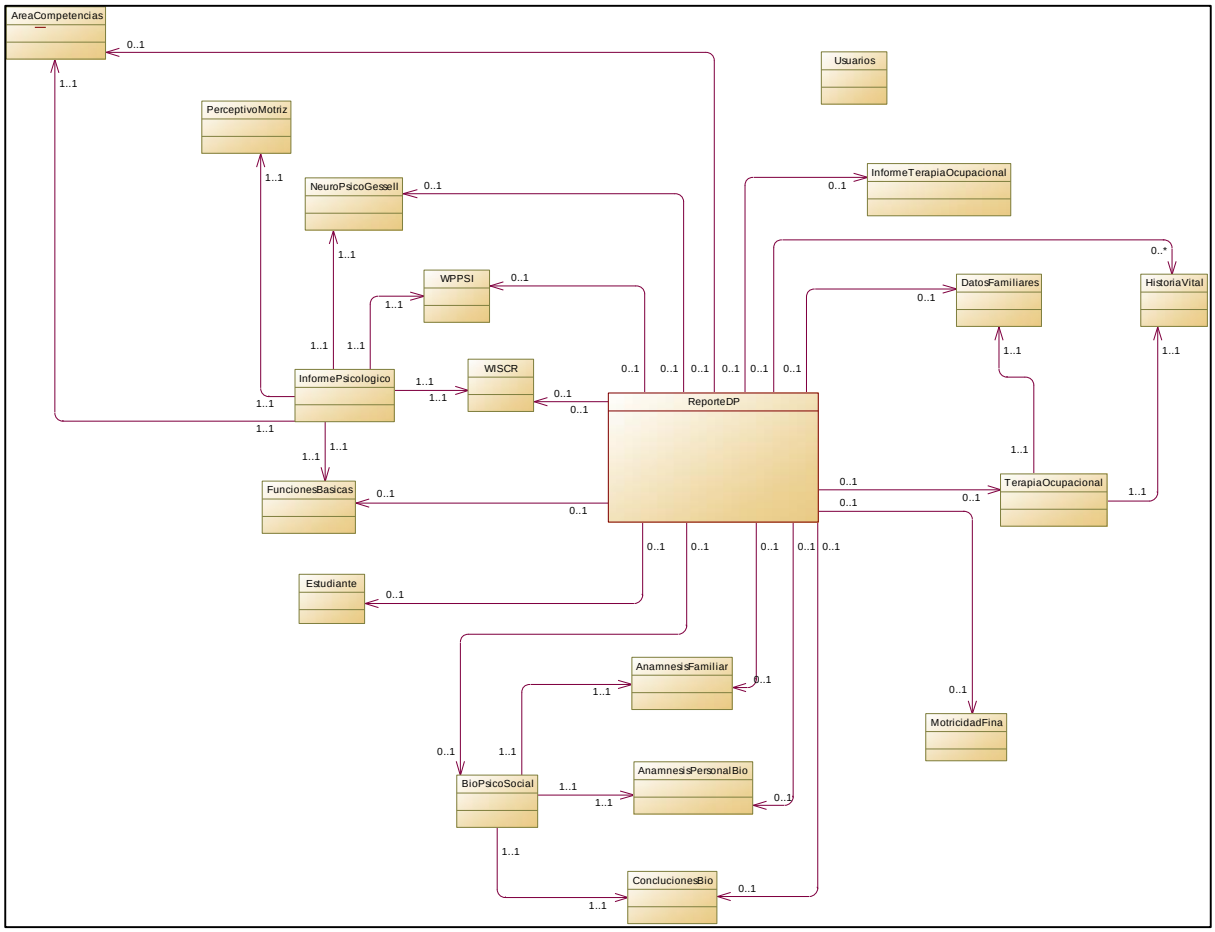

Figura 19: Diagrama de Clases del Dominio del Problema Conceptual (Sánchez, 2017)

4.1.3. Diagrama de Clases del Dominio del Problema

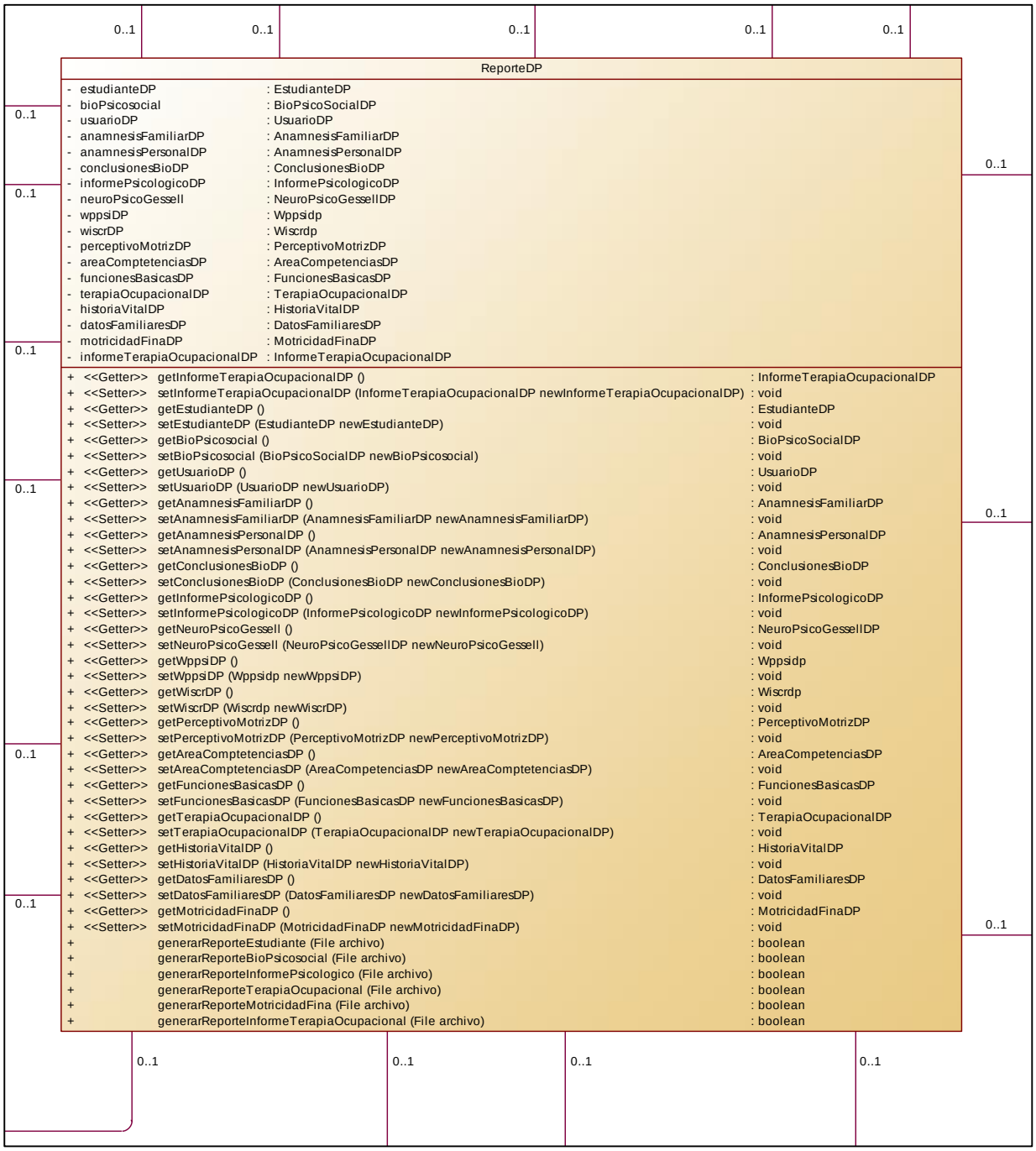

Figura 20: Diagrama de Clases del Dominio del Problema (Sánchez, 2017)

Para mayor detalle por favor revisar el anexo 2. Diagrama de clases y secuencia, que se encuentra en el CD.

4.1.4. Diagrama de Clases de Manejo de Datos Conceptual

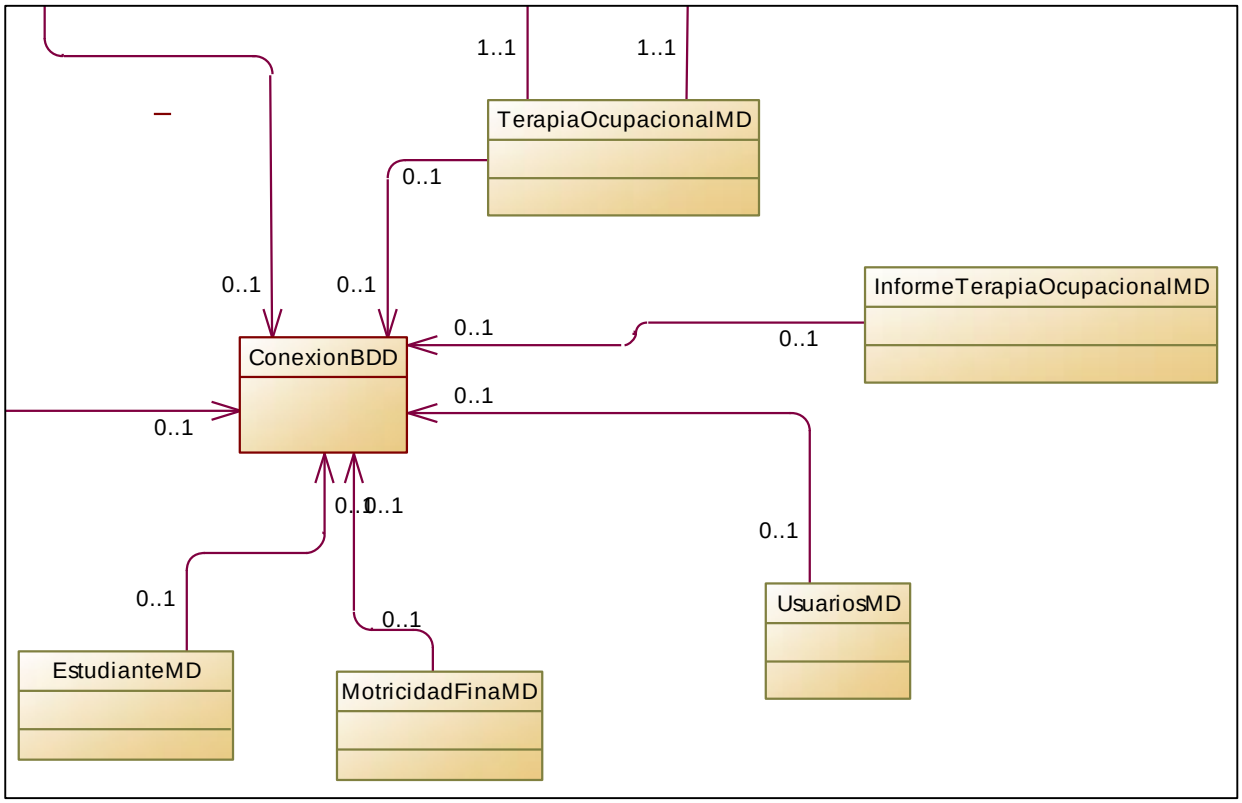

Figura 21: Diagrama de Clases de Manejo de Datos Conceptual (Sánchez, 2017)

Para más detalle por favor revisar el anexo 4. Diagrama conceptual de clases MD que se encuentra en el CD.

4.1.5. Diagrama de Clases de Manejo de Datos

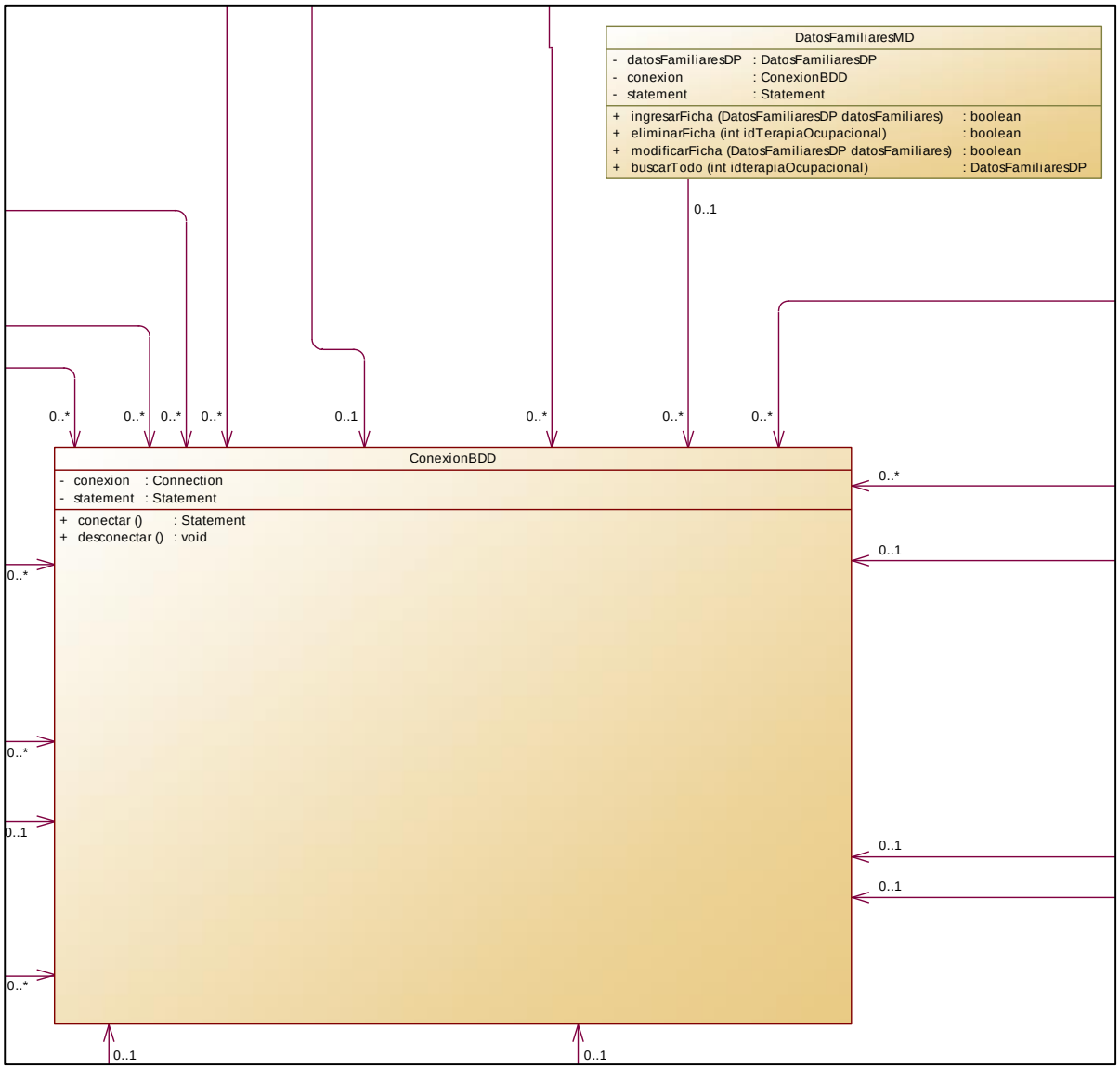

Figura 22: Diagrama de Clases de Manejo de Datos (Sánchez, 2017)

Para mayor detalle por favor revisar el anexo 2. Diagrama de clases y secuencia, que se encuentra en el CD.

4.2. Diagrama Entidad Relación de la Base de Datos

Los siguientes diagramas muestran cómo va a estar estructurada nuestra base de datos, tanto a nivel conceptual como nivel físico.

4.2.1. Diagrama E/R de la Base de Datos a Nivel Conceptual

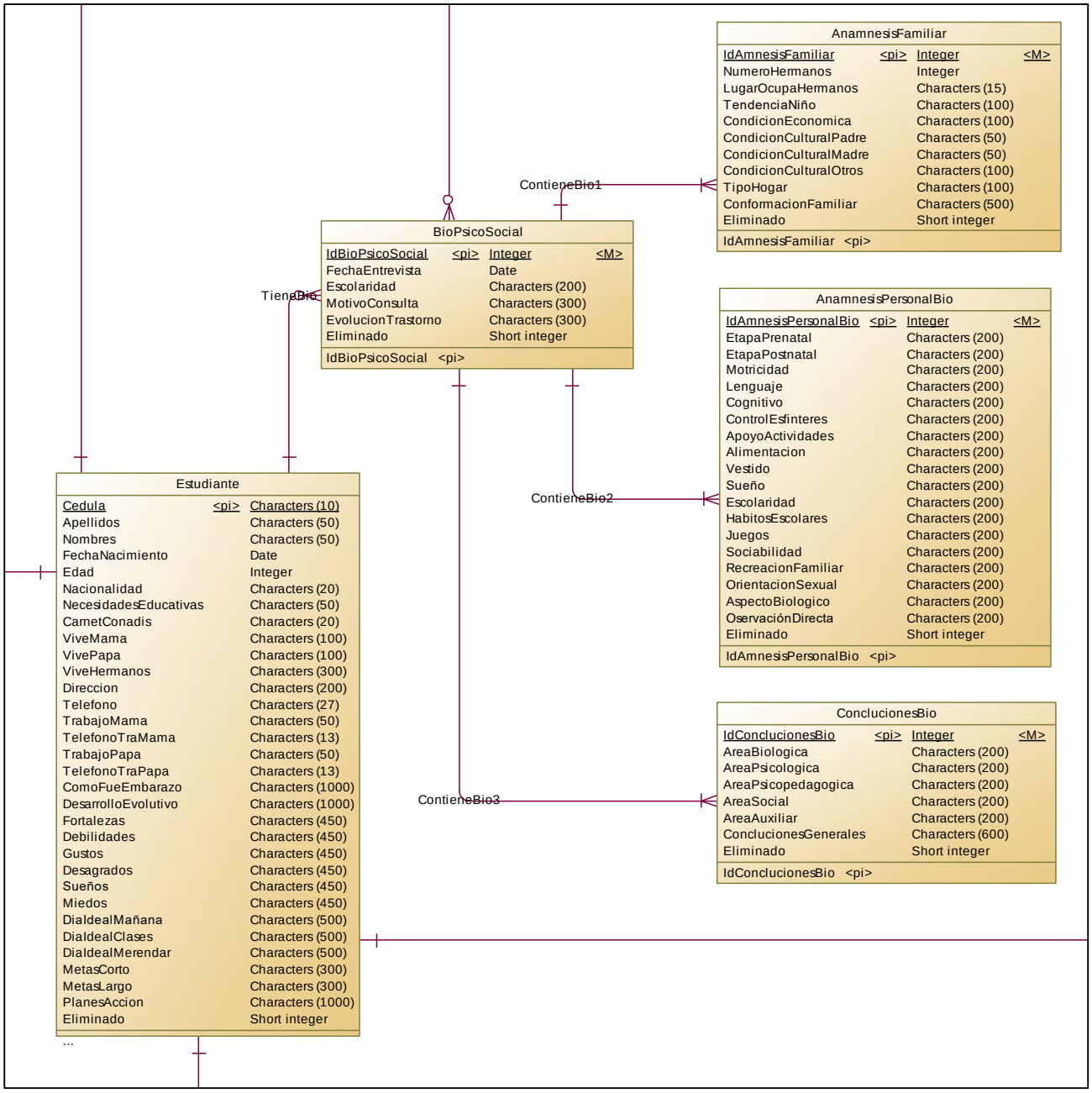

Figura 23: Diagrama E/R de la Base de Datos a Nivel Conceptual (Sánchez, 2017)

Para mayor detalle por favor revisar el anexo 5. Modelo Entidad Relación Conceptual que se encuentra en el CD.

4.2.2. Diagrama E/R de la Base de Datos a Nivel Físico

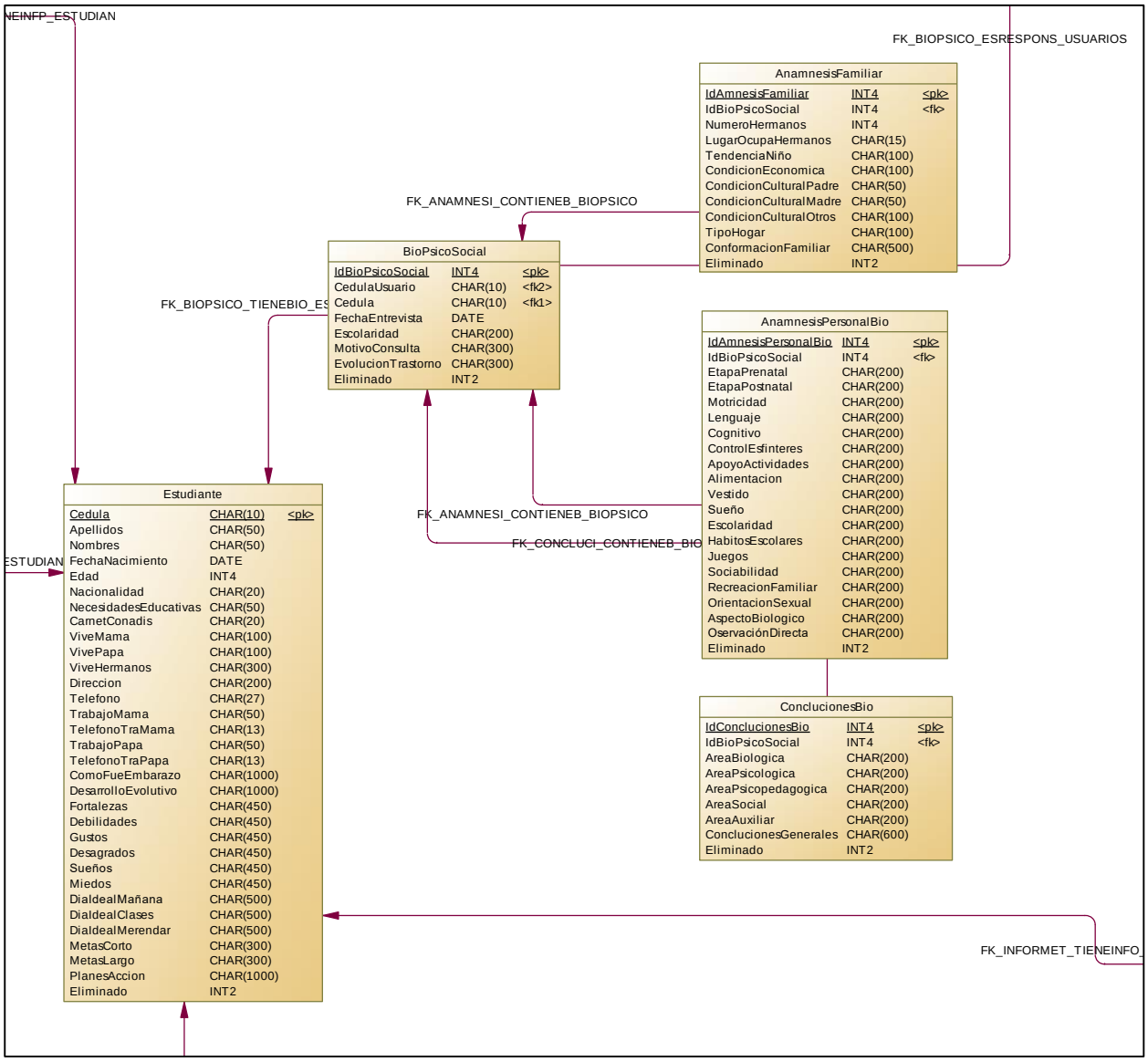

Figura 24: Diagrama E/R de la Base de Datos a Nivel Físico (Sánchez, 2017)

Para más detalle por favor revisar el anexo 6. Modelo Entidad Relación Físico que se encuentra en el CD.

4.3. Diagramas de secuencia

Los siguientes diagramas muestran como están estructurados los tres principales procesos del sistema, es decir, como el usuario interactúa con el sistema y como el sistema procesa y responde.

4.3.1. Diagrama de Secuencia de Gestión de Usuarios

En el diagrama mostrado a continuación se puede ver como el usuario y el sistema interactúan para realizar el proceso de inicio de sesión al sistema y como se realiza el CRUD de los usuarios.

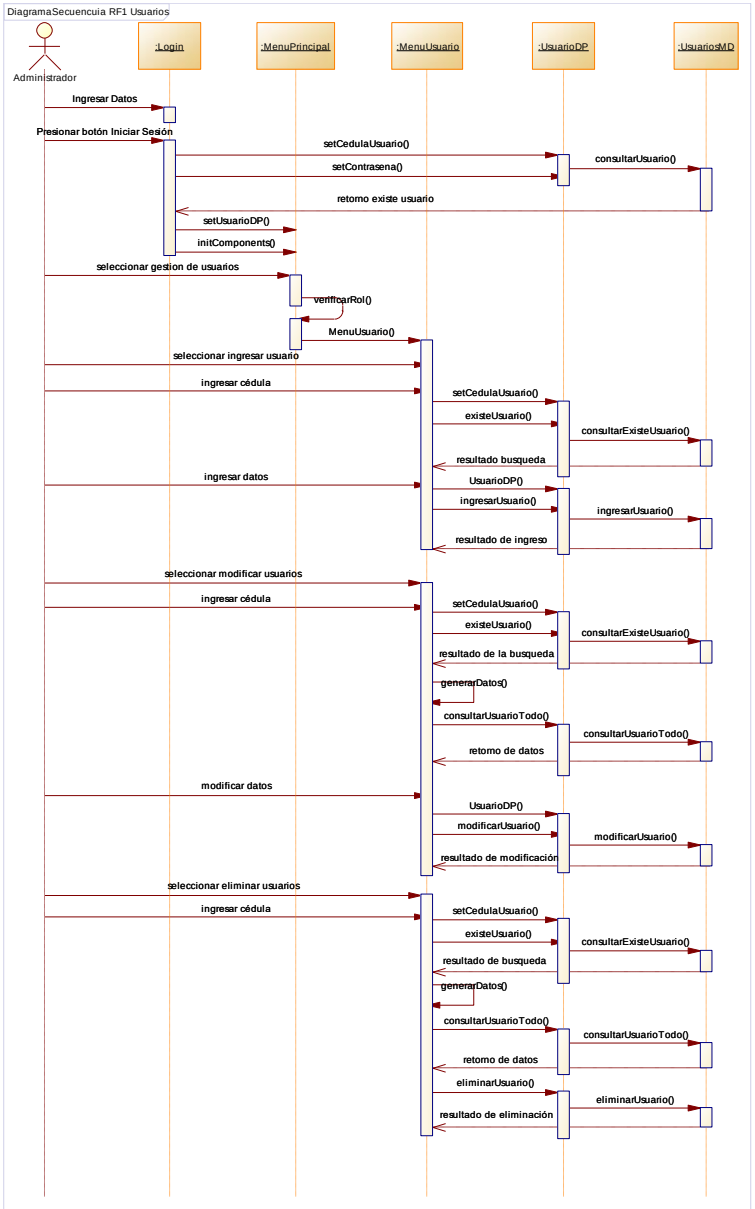

Figura 25: Diagrama de Secuencia de Gestión de Usuarios (Sánchez, 2017)

4.3.2. Diagrama de Secuencia de Gestión de Fichas

El siguiente modelo muestra cómo se manejan las 6 fichas del sistema. Solo se muestra el diagrama de la gestión del informe Psicológico ya que todas las demás fichas fueron programadas en base a este modelo por lo tanto su estructura es idéntica.

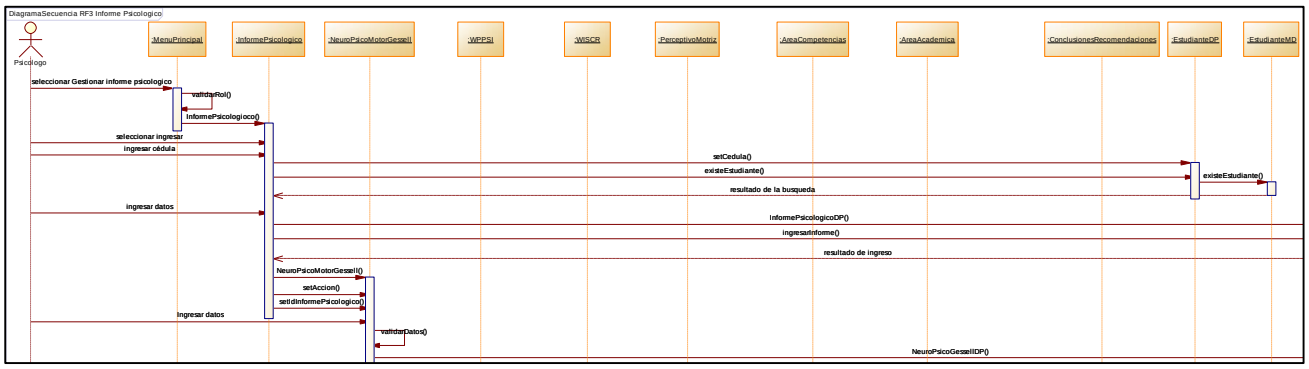

Figura 26: Diagrama de Secuencia de Gestión de Fichas (Sánchez, 2017)

Para mayor detalle por favor revisar el anexo 2. Diagrama de clases y secuencia, que se encuentra en el CD.

4.3.3. Diagrama de Secuencia de Gestión de Reportes

A continuación se muestra el diagrama de gestión de reportes y este muestra cómo se maneja el reporte de la ficha Bio Psicosocial, de igual manera solo se presenta la gestión de reportes bio psicosociales ya que todos los de más reportes fueron realizados basándose en este diagrama.

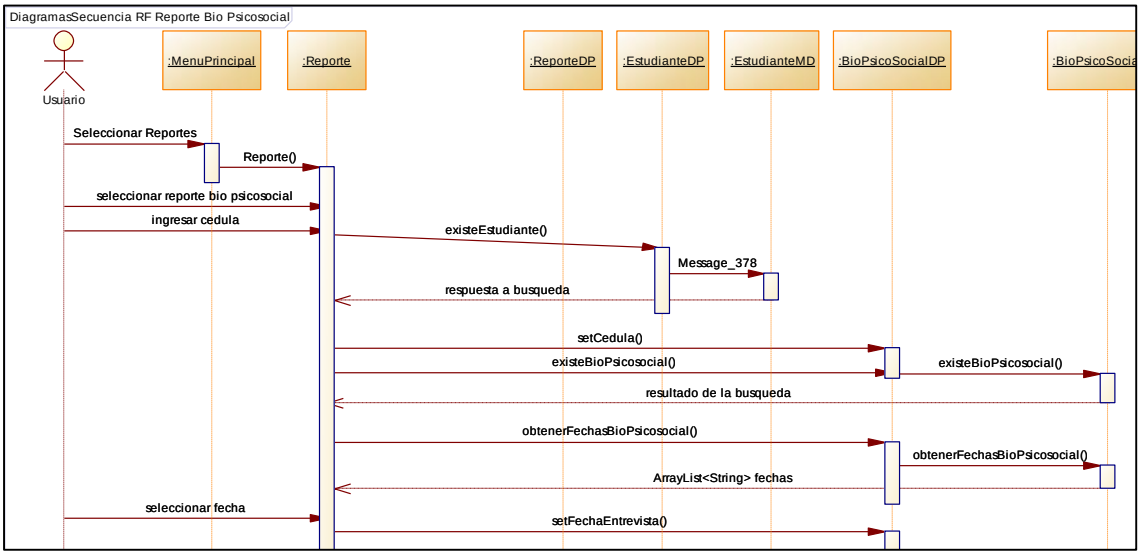

Figura 27: Diagrama de Secuencia de Gestión de Reportes (Sánchez, 2017)

Para mayor detalle por favor revisar el anexo 2. Diagrama de clases y secuencia, que se encuentra en el CD.

4.4. Plan de pruebas del Sistema

4.4.1. RF1. Gestión de Usuarios

Entradas	Resultados Esperados	C.U Estudio	Resultado
Ingreso de cédula	Sistema valida cédula correcta	RF1	
Ingreso de Nombres y apellidos	Sistema controla el ingreso correcto de información	RF1	
Selección de tipo de usuario	El sistema verifica que un tipo de usuario se ha especificado	RF1	
Contraseña	El sistema verifica que las contraseñas ingresadas sean iguales	RF1	

Tabla 3: Pruebas del Sistema RF1. Gestión de Usuarios (Sánchez, 2017)

4.4.2. RF2. Gestión de Fichas en General

Entradas	Resultados Esperados	C.U Estudio	Resultado
Ingreso de cédula	Sistema valida cédula correcta	RF7	
Verificación de existencia de usuario	El sistema busca existencia de fichas	RF7	
verificación de existencia de fichas	El sistema busca la existencia de fichas	RF2,3,4,5,6,7	
Verifica ingreso de datos textuales	El sistema se revisa que la integridad de los datos se mantenga	RF2,3,4,5,6,7	
Verifica ingreso de datos numéricos	El sistema se revisa que la integridad de los datos se mantenga	RF2,3,4,5,6,7	
Verifica ingreso de datos que no sobre pasen el tamaño especificado en la BDD	El sistema se revisa que la integridad de los datos se mantenga	RF2,3,4,5,6,7	
Ingreso de Fichas		RF2,3,4,5,6,7	

	Las fichas se ingresan sin problemas		
Modificación de Fichas	Las fichas se modifican sin problemas	RF2,3,4,5,6,7	
Eliminación de Fichas	Las fichas se eliminan sin problemas	RF2,3,4,5,6,7	

Tabla 4: Pruebas del Sistema RF2. Gestión de Fichas en General (Sánchez, 2017)

4.4.3. RF. Reportes

Entradas	Resultados Esperados	C.U Estudio	Resultado
Ingreso de cédula	Sistema valida cédula correcta	RF7	
Verificación de existencia de usuario	El sistema busca existencia de fichas	RF7	
verificación de existencia de fichas	El sistema busca la existencia de fichas	RF2,3,4,5,6,7	
Generación de reportes	Los reportes se generan adecuadamente	RF2,3,4,5,6,7	

Tabla 5: Pruebas del Sistema RF. Reportes (Sánchez, 2017)

CAPÍTULO 5: CONSTRUCCIÓN

El capítulo siguiente explica cómo fue desarrollado el sistema, tanto en diseño visual como en codificación. No se muestra todo el código de la aplicación, solo se muestran los modelos fundamentales que se utilizan a través de todo el programa.

5.1. Diseño e Implementación de Interfaces de Usuario

5.1.1. Interfaz de Inicio de Sesión

Figura 28: Interfaz de Inicio de Sesión (Sánchez, 2017)

5.1.2. Interfaz del Menú Principal

Figura 29: Interfaz del Menú Principal (Sánchez, 2017)

5.1.3. Interfaz de Gestión de Usuarios

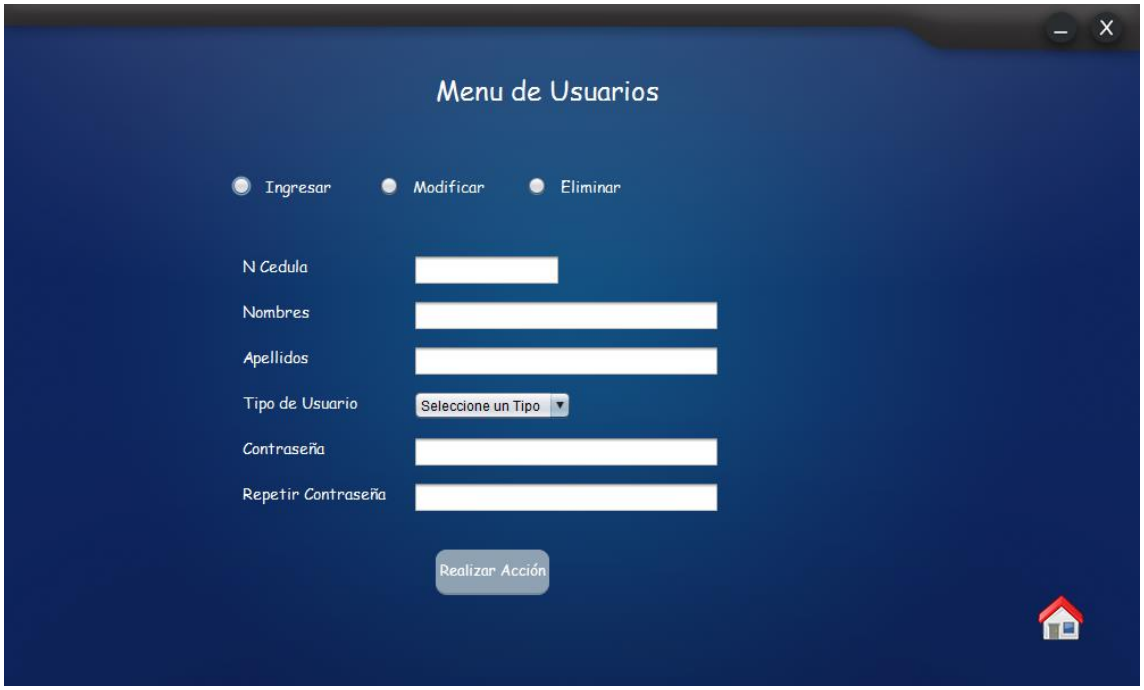

Figura 30: Interfaz de Gestión de Usuarios (Sánchez, 2017)

5.1.4. Interfaz de Gestión de Estudiantes Parte 1

Figura 31: Interfaz de Gestión de Estudiantes Parte 1 (Sánchez, 2017)

5.1.5. Interfaz de Gestión de Estudiantes Parte 2

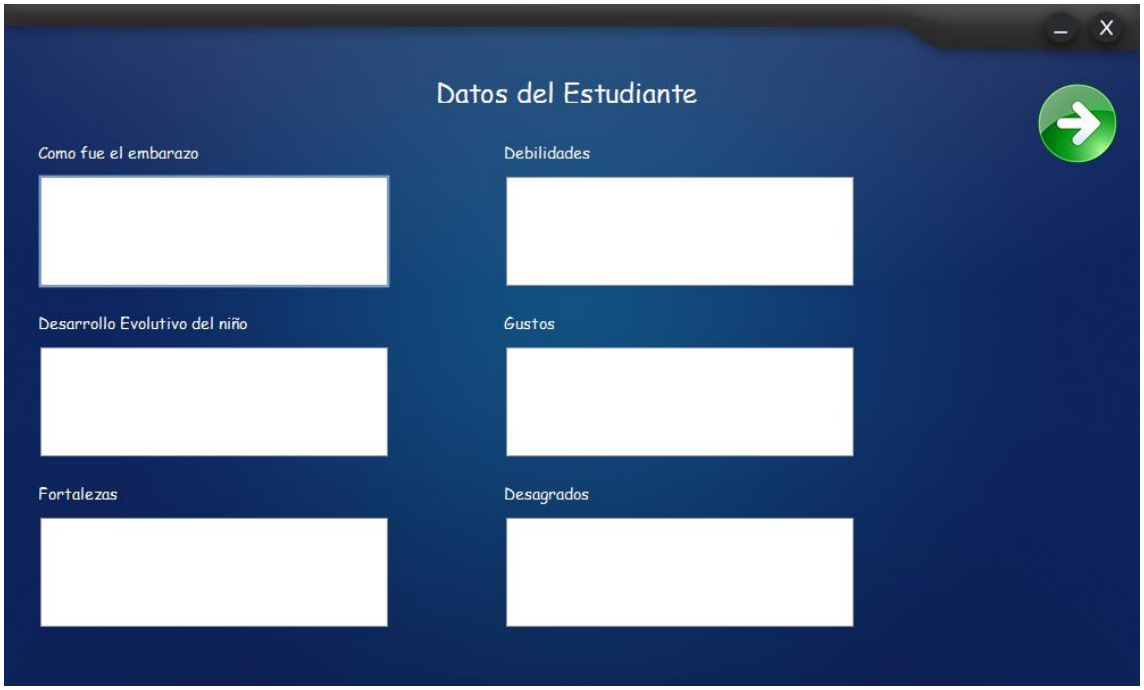

Figura 32: Interfaz de Gestión de Estudiantes Parte 2 (Sánchez, 2017)

5.1.6. Interfaz de Gestión de Estudiantes Parte 3

Figura 33: Interfaz de Gestión de Estudiantes Parte 3 (Sánchez, 2017)

5.1.7. Interfaz de Gestión de Estudiantes Parte 4

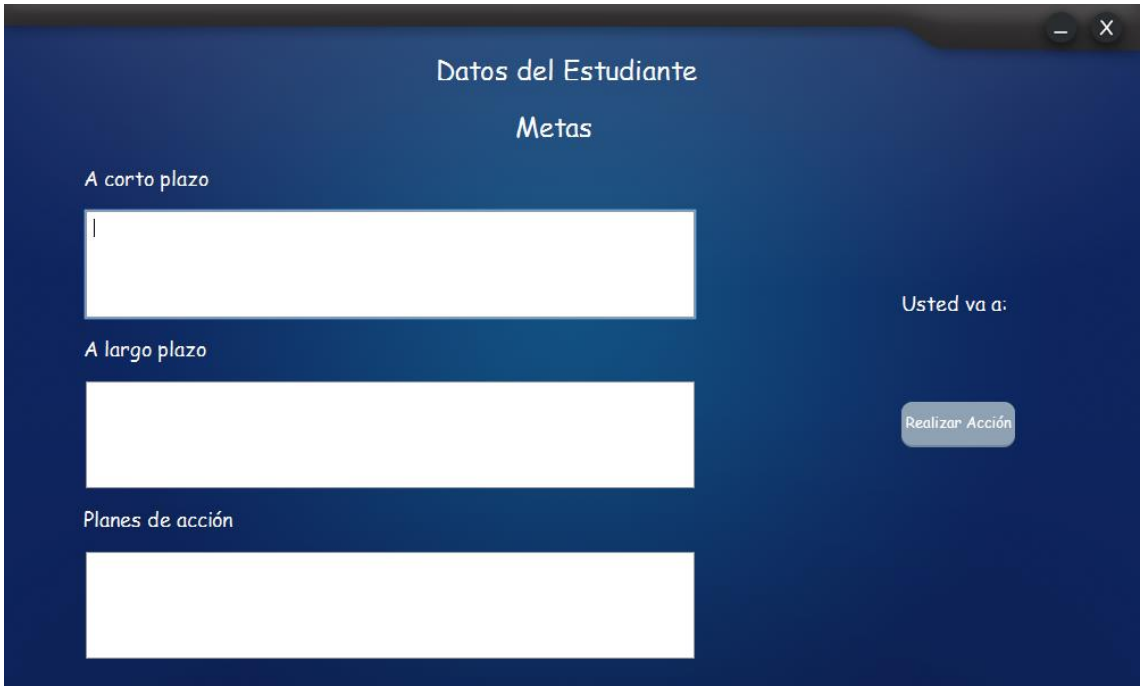

Figura 34: Interfaz de Gestión de Estudiantes Parte 4 (Sánchez, 2017)

5.1.8. Interfaz de Gestión de Ficha Bio Psicosocial

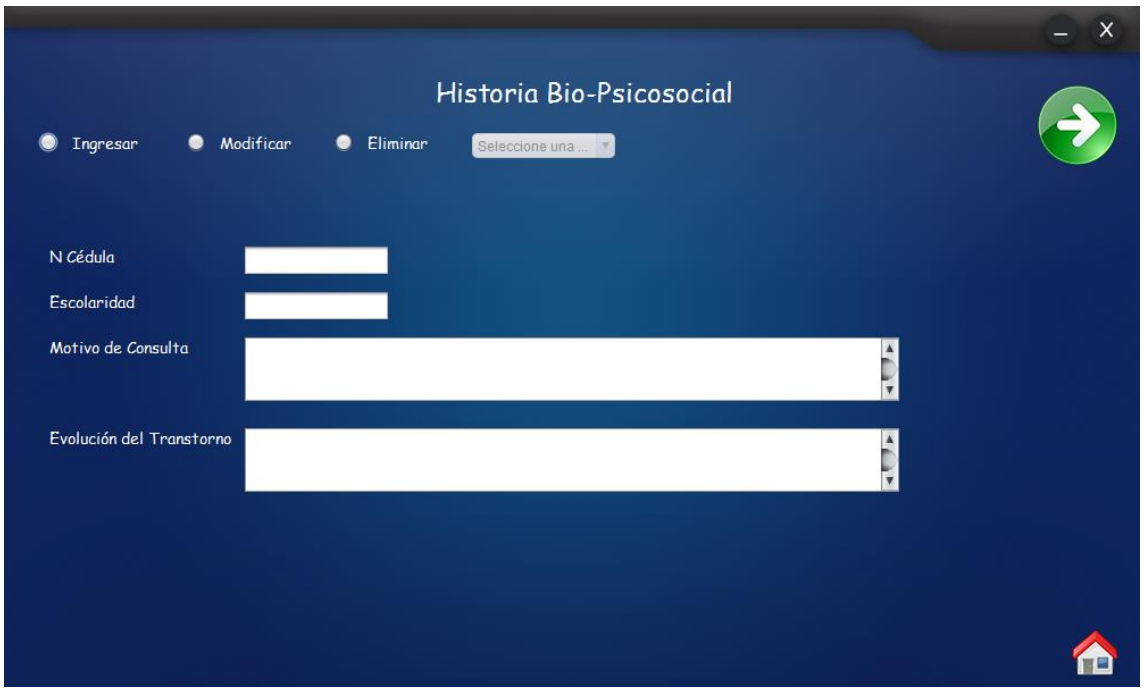

Figura 35: Interfaz de Gestión de Ficha Bio Psicosocial (Sánchez, 2017)

5.1.9. Interfaz de Gestión de Ficha Bio Psicosocial – Anamnesis Familiar

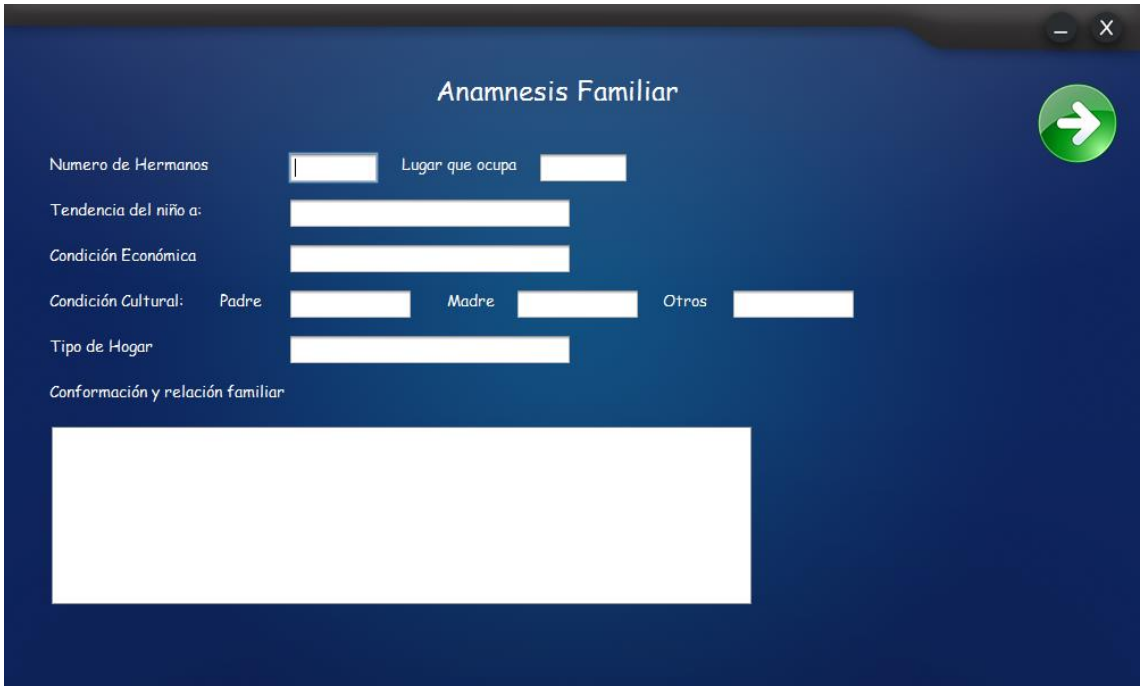

Figura 36: Interfaz de Gestión de Ficha Bio Psicosocial – Anamnesis Familiar (Sánchez, 2017)

5.1.10. Interfaz de Gestión de Ficha Bio Psicosocial – Anamnesis Personal

Figura 37: Interfaz de Gestión de Ficha Bio Psicosocial – Anamnesis Personal (Sánchez, 2017)

5.1.11. Interfaz de Gestión de Ficha Bio Psicosocial – Conclusiones

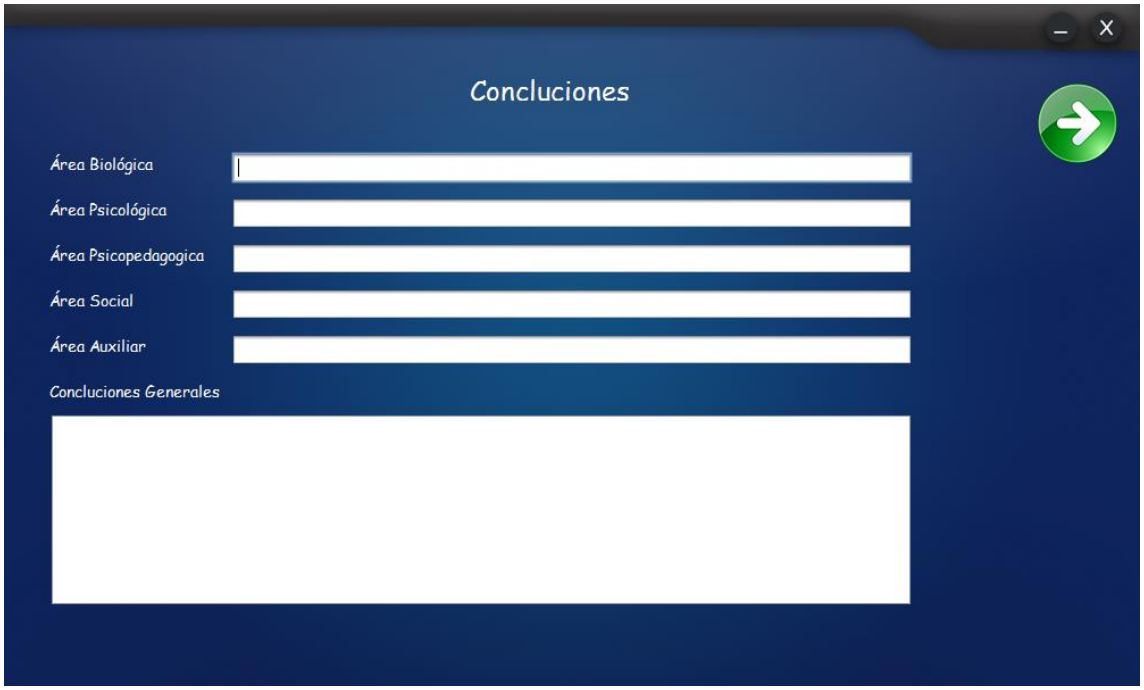

Figura 38: Interfaz de Gestión de Ficha Bio Psicosocial – Conclusiones (Sánchez, 2017)

Todas las demás interfaces se encuentran en el Anexo7. Interfaces de Usuario, por favor revisar en el CD.

5.2. Explicación de Código GUI

5.2.1. Explicación de Código de Movimiento de Botones

Los botones al momento de pasar el mouse por cada uno de ellos realizan un movimiento para señalar al usuario que el botón puede ser seleccionado.

Código:

```
ImageIcon imageIcon = new ImageIcon(getClass().getResource("/Img/btnIniciarSesionU.png")
btnIniciarSesion.setIcon(imageIcon);
```

Figura 39: Código de cambio de imagen (Sánchez, 2017)

5.2.2. Explicación del Código del Inicio de Sesión GUI

Para realizar el inicio de sesión se inicializa un objeto UsuarioDP mediante el cual consultaremos si el usuario realmente existe en la base de datos y su contraseña es la correcta. En caso de no existir el usuario o que la contraseña no coincida con la almacenada en la base de datos se originará el siguiente aviso:

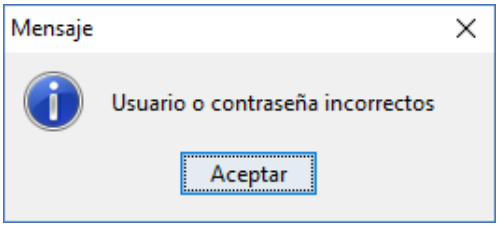

Figura 40: Validación de usuario y contraseña (Sánchez, 2017)

Código:

```
usuarioDP = new UsuarioDP(tfUsuario.getText(),new String(tfContrasena.getPassword()));
        usuarioDP = usuarioDP.consultarUsuario();
        if(usuarioDP.getCedulaUsuario().compareTo("") == 0 && usuarioDP.getContrasena().
        {
            JOptionPane.showMessageDialog(this, "Usuario o contraseña incorrectos");
        }
        else
        {
            MenuPrincipal menuPrincipal = new MenuPrincipal();
            usuarioDP.setContrasena("");
            menuPrincipal.setUsuarioDP(usuarioDP);
            this.dispose();
            menuPrincipal.setVisible(true);
        }
```

Figura 41: Código de validación de usuario y contraseña (Sánchez, 2017)

5.2.3. Validaciones de Datos Numéricos

Todos los campos numéricos del GUI están validados de 5 formas:

- La primera es una validación de ingreso, es decir, el GUI no permite que en los campos numéricos se ingresen caracteres que no sean dígitos. Esta validación se las realiza mediante el evento java KeyTyped.
- En caso de que el valor sea decimal solo se permite ingresar valores numéricos y el '.' para la parte decimal. Esta validación se las realiza mediante el evento java KeyTyped.
- Para respetar la integridad de los datos el sistema valida la longitud máxima permitida de cada campo, para que la información ingresada sea lógica y además evitar así problemas con la base de datos. Esta validación se las realiza mediante el evento java KeyTyped.
- En caso de ser un campo de cédula se valida que la cédula sea correcta y solo tenga números. Esta validación se las realiza mediante el evento java FocusLost.

- En caso de que el campo numérico se deje en blanco el sistema automáticamente lo asignará como 0 o 0.0.

Cada una de las validaciones antes mencionadas generan un aviso al usuario en caso de ser activados.

Avisos:

Aviso de ingreso de caracteres no numéricos

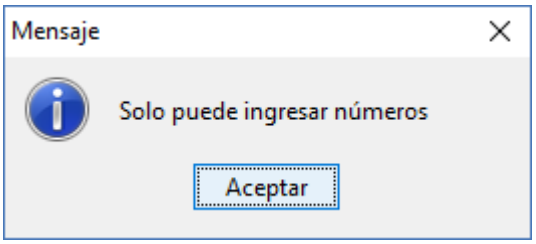

Figura 42: Validación de datos numéricos (Sánchez, 2017)

Aviso de '.' en caso de campos decimales

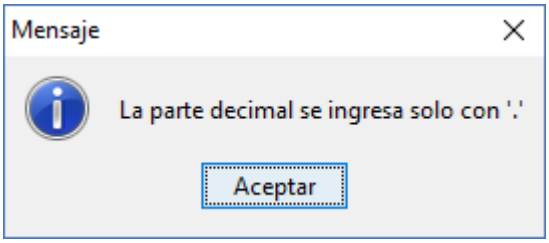

Figura 43: Validación de datos de punto flotante (Sánchez, 2017)

Aviso de longitud máxima permitida

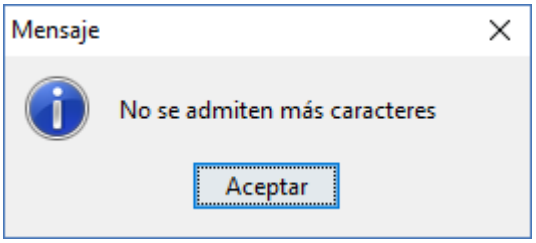

Figura 44: Validación de longitud de caracteres permitida (Sánchez, 2017)

Aviso de cédula incorrecta

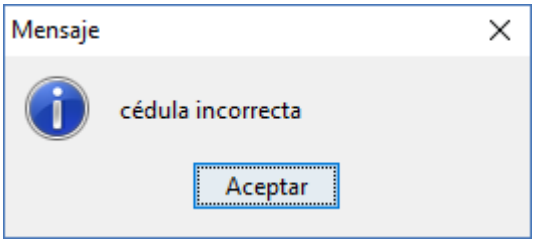

Figura 45: Validación cédula (Sánchez, 2017)

Código de cada una de las validaciones:

Ingreso de caracteres no numéricos

```
if((int) evt.getKeyChar() != 8 && (int) evt.getKeyChar() != 9)
        {
            if(!Character.isDigit(evt.getKeyChar()))
            {
                evt.consume();
                JOptionPane.showMessageDialog(this, "Solo puede ingresar números");
            }
}
```

Figura 46: Código de ingreso de caracteres no numéricos (Sánchez, 2017)

‘.’ En caso de campos decimales

```
if((int) evt.getKeyChar() != 8 && (int) evt.getKeyChar() != 9 && (int) evt.getKeyChar() != 46)
{

            if(!Character.isDigit(evt.getKeyChar()))
            {
                if((int)evt.getKeyChar() == 44)
                {
                    evt.consume();
                    JOptionPane.showMessageDialog(this, "La parte decimal se ingresa solo con '.'");
                }
                else
                {
                    evt.consume();
                    JOptionPane.showMessageDialog(this, "Solo puede ingresar numeros");
                }
            }
}
```

Figura 47: Código de validación de campos decimales (Sánchez, 2017)

Aviso de longitud máxima permitida

```
if(textField.getText().length() >= tamaño)
            {
                evt.consume();
                JOptionPane.showMessageDialog(this, "No se admiten más caracteres");
            }
```

Figura 48: Código de aviso de longitud máxima permitida (Sánchez, 2017)

Aviso de cédula incorrecta

```
if(tfCedula.getText().length() != 10)
      JOptionPane.showMessageDialog(this, "Cédula incorrecta");
```

Figura 49: Código de aviso de cédula incorrecta (Sánchez, 2017)

Validación de campos numéricos en blanco

```
(tfEdadMadre.getText().equals("")) ? 0 : Integer.parseInt(tfEdadMadre.getText())
```

Figura 50: Código de validación de campos numéricos en blancco (Sánchez, 2017)

5.2.4. Validación de Capos de Texto

En estos campos solo se valida que los datos ingresados cumplan con el tamaño máximo permitido en la base de datos, a excepción de campos como nombres y apellidos los cuales no se permite ingresar números. La manera de validación es idéntica a la validación numérica.

5.2.5. Ingreso de Fichas – Parte del GUI

Primero que todo se verifica que para ingresar una ficha exista un estudiante registrado para poderlo asociar a dicha ficha.

Se llama al constructor de la clase correspondiente a la ficha que se está ingresando y se inicializan todos los datos de dicha ficha, luego se llama al método de ingreso de la ficha y en caso de que todavía falte más partes de la ficha por llenar se llama a la ventana GUI respectiva para continuar ingresando datos. Caso contrario se regresa al Menú Principal. Si la ficha ha sido ingresada correctamente el sistema informa a usuario mediante un aviso. En el caso especial de la ficha de Terapia Ocupacional si ya se ha

ingresado una ficha previa al ingresar otra del mismo estudiante se cargarán los datos automáticamente de la ficha anterior para facilitar el ingreso de la nueva ficha.

Aviso:

Aviso de no existencia del estudiante

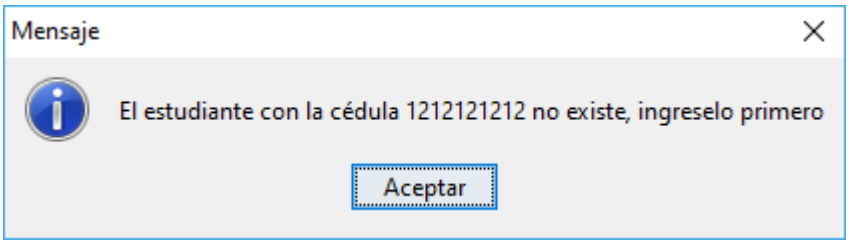

Figura 51: Aviso de no existencia del estudiante (Sánchez, 2017)

Aviso de ingreso correcto a la base de datos

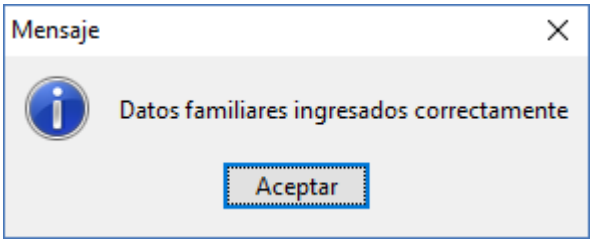

Figura 52: Aviso de ingreso correcto a la base de datos (Sánchez, 2017)

Código:

```
estudianteDP.setCedula(tfCedula.getText());
            if(!estudianteDP.existeEstudiante())
            {
                existeEstudiante = false;
                JOptionPane.showMessageDialog(this, "El estudiante con la cédula "+tfCedula.getText()+" no existe, ingreselo primero");
            }
```

Figura 53: Código de confirmación de existencia de estudiante (Sánchez, 2017)

Verificación de existencia de Usuario

Método de ingreso - GUI

```
datosFamiliaresDP = new DatosFamiliaresDP(0, idTerapiaOcupacional, viveCon, tfNombrePadre.getText(), (tfEdadPadre.getText().equals("")) ? 0 : Integer.parseInt(tfEdadPadre.getText()),
                    tfEstadoCivilPadre.getText(),
                    tfInstruccionPadre.getText(), tfProfesionPadre.getText(), tfCelularPadre.getText(), tfLlegadaPadre.getText(), tfNombreMadre.getText(),
                    (tfEdadMadre.getText().equals(""))? 0 : Integer.parseInt(tfEdadMadre.getText()),
                    tfEstadoCivilMadre.getText(), tfInstruccionMadre.getText(), tfProfesionMadre.getText(), tfCelularMadre.getText(), tfLlegadaMadre.getText(),
                    (tfNumeroHermanos.getText().equals("")) ? 0 : Integer.parseInt(tfNumeroHermanos.getText()),
                    tfLugarOcupa.getText(), tfNombre1.getText(), (tfEdad1.getText().equals("")) ? 0 : Integer.parseInt(tfEdad1.getText()), tfNombre2.getText(),
                    (tfEdad2.getText().equals("")) ? 0 : Integer.parseInt(tfEdad2.getText()), tfNombre3.getText(), (tfEdad3.getText().equals("")) ? 0 : Integer.parseInt(tfEdad3.getText()),
                    tfNombre4.getText(), (tfEdad4.getText().equals("")) ? 0 : Integer.parseInt(tfEdad4.getText() ), (rbSiDiscapacidad.isSelected())?"Si":"No", tfQuienDiscapacidad.getText(),
                    taDatosExtra.getText(), 0);
            int resultado = JOptionPane.showConfirmDialog (this, "¿Los datos son correctos?","Warning",JOptionPane.YES_NO_OPTION);
            if(resultado == JOptionPane.YES_OPTION){
                if(datosFamiliaresDP.ingresarFicha())
                {
                    if(existeFicha)
                    {
                        JOptionPane.showMessageDialog(this, "Datos familiares ingresados correctamente");
                        HistoriaVital historiaVital = new HistoriaVital();
                        historiaVital.setAccion(accion);
                        //areaCompetencias.setUsuarioDP(usuarioDP);
                        historiaVital.setIdTerapiaOcupacional(idTerapiaOcupacional);
                        historiaVital.setIdTerapiaOcupacional2(idTerapiaOcupacional2);
                        historiaVital.setExisteFicha(true);
                        historiaVital.generarDatos();
                        this.dispose();
                        historiaVital.setVisible(true);
                    }
                    else
                    {
                        JOptionPane.showMessageDialog(this, "Datos familiares ingresados correctamente");
                        HistoriaVital historiaVital = new HistoriaVital();
                        historiaVital.setAccion(accion);
                        //areaCompetencias.setUsuarioDP(usuarioDP);
                        historiaVital.setIdTerapiaOcupacional(idTerapiaOcupacional);
                        historiaVital.setExisteFicha(false);
                        this.dispose();
                        historiaVital.setVisible(true);
                    }
```

Figura 54: Código de aviso de ingreso (Sánchez, 2017)

5.2.6. Modificación de Fichas – Parte del GUI

De igual manera que en el ingreso se realizan todas las validaciones y se selecciona la ficha a modificar mediante una fecha y la cédula del estudiante. Para proceder a modificar la ficha.

Avisos:

El sistema advierte que se debe ingresar una fecha para poder modificar

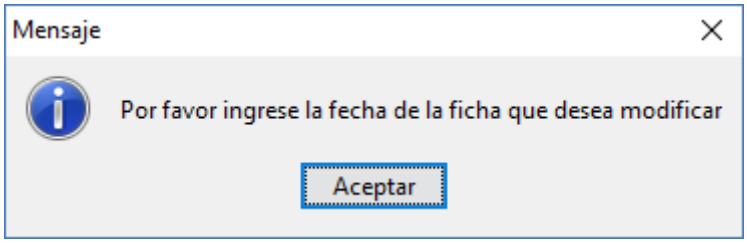

Figura 55: Aviso de confirmación de fecha (Sánchez, 2017)

Código:

```
if(rbModificar.isSelected() && existeEstudiante && existeFicha)
            {
                int resultado = JOptionPane.showConfirmDialog (this, "¿Está seguro de modificar la Ficha?","Warning",JOptionPane.YES_NO_OPTION);
                if(resultado == JOptionPane.YES_OPTION){
                    SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
                    java.util.Date parsed;
                    Date sql = obtenerFechaActual();
                    try {
                        parsed = format.parse(cbFechaFicha.getSelectedItem().toString());
                        sql = new Date(parsed.getTime());
                    } catch (ParseException ex) {
                        Logger.getLogger(BioPsicosocial.class.getName()).log(Level.SEVERE, null, ex);
                    }

                    terapiaOcupacionalDP = new TerapiaOcupacionalDP(0, tfCedula.getText(), usuarioDP.getCedulaUsuario(), tfLugNacimiento.getText(), tfDiagnostico.getText(),
                        tfSector.getText(), tfRepresentate.getText(), sql, 0);

                    resultado = JOptionPane.showConfirmDialog (this, "¿Los datos son correctos?","Warning",JOptionPane.YES_NO_OPTION);
                    if(resultado == JOptionPane.YES_OPTION){
                        accion = "Modificar";
                        if(terapiaOcupacionalDP.modificarFicha())
                        {
                            JOptionPane.showMessageDialog(this, "Ficha Bio-Psicosocial modificada correctamente");
                            DatosFamiliares datosFamiliares = new DatosFamiliares();
                            datosFamiliares.setAccion(accion);
                            datosFamiliares.setIdTerapiaOcupacional(terapiaOcupacionalDP.obtenerID());
                            datosFamiliares.setExisteFicha(false);
                            datosFamiliares.generarDatos();
                            this.dispose();
                            datosFamiliares.setVisible(true);
                        }
                        else
                        JOptionPane.showMessageDialog(this, "Error de la base de datos, reinicie el sistema");
                    }
                }
```

Figura 56: Código de modificaciones (Sánchez, 2017)

5.2.7. Eliminación de Ficha – GUI

La eliminación de una ficha se realiza solamente con la cédula del estudiante y la fecha de ingreso de dicha ficha.

Código:

```
if(rbEliminar.isSelected() && existeEstudiante)
            {
                int resultado = JOptionPane.showConfirmDialog (this, "¿Está seguro de eliminar la ficha?","Warning",JOptionPane.YES_NO_OPTION);
                if(resultado == JOptionPane.YES_OPTION){
                    terapiaOcupacionalDP.setCedula(tfCedula.getText());
                    SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
                    java.util.Date parsed;
                    Date sql = obtenerFechaActual();
                    try {
                        parsed = format.parse(cbFechaFicha.getSelectedItem().toString());
                        sql = new Date(parsed.getTime());
                    } catch (ParseException ex) {
                        Logger.getLogger(BioPsicosocial.class.getName()).log(Level.SEVERE, null, ex);
                    }
                    terapiaOcupacionalDP.setFechaEntrevista(sql);
                    TerapiaOcupacionalDP temp = new TerapiaOcupacionalDP();
                    temp.setCedula(tfCedula.getText());
                    temp.setFechaEntrevista(sql);
                    int id = temp.obtenerID();
                    terapiaOcupacionalDP.setIdTerapiaOcupacional(id);
                    terapiaOcupacionalDP.setEliminado(1);
                    if(terapiaOcupacionalDP.eliminarFicha())
                    {
                        JOptionPane.showMessageDialog(this, "La ficha ha sido eliminada correctamente");
                        this.dispose();
                    }
                    else
                    JOptionPane.showMessageDialog(this, "Ha ocurrido un error por favor reinicie el sistema");
                }
            }
```

Figura 57: Código de aviso de eliminación (Sánchez, 2017)

5.2.8. Llenado automático de campos al modificar una ficha

Si se selecciona una ficha para modificar todos los campos de la misma serán llenados con la información existente en la base de datos.

Código:

```
public void generarDatos()
    {
        historiaVitalDP = new HistoriaVitalDP();
        if(existeFicha)
            historiaVitalDP.setIdTerapiaOcupacional(idTerapiaOcupacional2);
        else
            historiaVitalDP.setIdTerapiaOcupacional(idTerapiaOcupacional);
        historiaVitalDP = historiaVitalDP.buscarTodo();
        tfAccidenteEmbarazo.setText(historiaVitalDP.getAccidentesEmbarazo().trim());
        tfApgar.setText(historiaVitalDP.getApgar()+"");
        tfCausasEmbarazo.setText(historiaVitalDP.getCausasEmbarazo().trim());
        tfCausasParto.setText(historiaVitalDP.getCausasParto().trim());
        tfCualPsicomotricidad.setText(historiaVitalDP.getDificultadPsicomotricidad().trim());
        tfEdadComplementaria.setText(historiaVitalDP.getEdadComplementaria().trim());
        tfEdadLactancia.setText(historiaVitalDP.getEdadLactancia().trim());
        tfEdadMadre.setText(historiaVitalDP.getEdadMadre()+"");
        tfEmpezoCaminar.setText(historiaVitalDP.getEdadCaminar().trim());
        tfGestacion.setText(historiaVitalDP.getSemanasGestacion()+"");
        tfMedicamentosEmbarazo.setText(historiaVitalDP.getMedicamentosEmbarazo().trim());
        tfPeso.setText(historiaVitalDP.getPeso()+"");
        tfTalla.setText(historiaVitalDP.getTalla()+"");
        taDestete.setText(historiaVitalDP.getReaccionDestete().trim());
        if (historiaVitalDP.getTipoEmbarazo().trim().equals("EmbarazoNormal"))
        {
            rbEmbarazoNormal.setSelected(true) ;
            tfCausasEmbarazo.setEnabled(true);
        }
```

Figura 58: Código de llenado de datos (Sánchez, 2017)

5.3. Clases DP

Las clases DP están divididas en dos grupos, el primero son las clases DP de las fichas principales, las cuales tienen más métodos como buscaID(), obtenerFechas() o buscarUltimo(). Mientras que las secundarias solo cuentan con los métodos básicos para realizar el CRUD de la ficha. Esto se diseñó así ya que las clases principales del DP manejan y utilizan más información para el correcto funcionamiento del sistema. Todas las clases principales tienen el nombre de la Ficha a la que están asociados, mientras que las clases secundarias tienen el nombre de cada parte principal de la ficha

5.3.1. Código de Clase Principal

```
public class TerapiaOcupacionalDP {

    private int idTerapiaOcupacional;
    private String cedula;
    private String cedulaUsuario;
    private String lugarNacimiento;
    private String diagnostico;
    private String sector;
    private String representante;
    private Date fechaEntrevista;
    private int eliminado;
    private TerapiaOcupacionalMD terapiaOcupacionalMD;
    private DatosFamiliaresDP datosFamiliaresDP;
    private HistoriaVitalDP historiaVitalDP;
```

Figura 59: Código de instanciación de clase DP (Sánchez, 2017)

```
public TerapiaOcupacionalDP() {
    }

    public TerapiaOcupacionalDP(int idTerapiaOcupacional, String cedula,
        String cedulaUsuario, String lugarNacimiento, String diagnostico,
        String sector, String representante, Date fechaEntrevista, int eliminado) {
        this.idTerapiaOcupacional = idTerapiaOcupacional;
        this.cedula = cedula;
        this.cedulaUsuario = cedulaUsuario;
        this.lugarNacimiento = lugarNacimiento;
        this.diagnostico = diagnostico;
        this.sector = sector;
        this.representante = representante;
        this.fechaEntrevista = fechaEntrevista;
        this.eliminado = eliminado;
    }
```

Figura 60: Código de constructor de una clase DP (Sánchez, 2017)

```
public int getIdTerapiaOcupacional() {
        return idTerapiaOcupacional;
    }

    public void setIdTerapiaOcupacional(int idTerapiaOcupacional) {
        this.idTerapiaOcupacional = idTerapiaOcupacional;
    }
```

Figura 61: Código de ejemplo de getter y setter de clase DP (Sánchez, 2017)

```
public boolean ingresarInforme()
    {
        this.terapiaOcupacionalMD =  new TerapiaOcupacionalMD(this);
        return terapiaOcupacionalMD.ingresarInforme();
    }
```

```
public boolean existeFicha()
{
    this.terapiaOcupacionalMD =  new TerapiaOcupacionalMD(this);
    return terapiaOcupacionalMD.existeFicha();
}
```

```
public ArrayList<String> obtenerFechas()
{
    this.terapiaOcupacionalMD =  new TerapiaOcupacionalMD(this);
    return terapiaOcupacionalMD.obtenerFechas();
}
```

```
public boolean modificarFicha()
{
    this.terapiaOcupacionalMD =  new TerapiaOcupacionalMD(this);
    return terapiaOcupacionalMD.modificarFicha();
}
```

```
public TerapiaOcupacionalDP buscarTodo()
{
    this.terapiaOcupacionalMD =  new TerapiaOcupacionalMD(this);
    return terapiaOcupacionalMD.buscarTodo();
}
```

```
public boolean eliminarFicha()
{

    this.datosFamiliaresDP = new DatosFamiliaresDP();
    datosFamiliaresDP.setIdTerapiaOcupacional(idTerapiaOcupacional);
    datosFamiliaresDP.setEliminado(1);
    datosFamiliaresDP.eliminarFicha();
    this.historiaVitalDP = new HistoriaVitalDP();
    historiaVitalDP.setIdTerapiaOcupacional(idTerapiaOcupacional);
    historiaVitalDP.setEliminado(1);
    historiaVitalDP.eliminarFicha();
    this.terapiaOcupacionalMD =  new TerapiaOcupacionalMD(this);
    return terapiaOcupacionalMD.eliminarFicha();
}
```

```
public int obtenerID()
    {
        this.terapiaOcupacionalMD =  new TerapiaOcupacionalMD(this);
        return terapiaOcupacionalMD.obtenerID();
    }
```

```
public TerapiaOcupacionalDP obtenerUltimo()
{
    this.terapiaOcupacionalMD =  new TerapiaOcupacionalMD(this);
    return terapiaOcupacionalMD.obtenerUltimo();
}
```

Figura 62: Código de CRUD hacia el MD (Sánchez, 2017)

5.4. Clases MD

Todas las clases MD cuentan con un objeto de ConexionBDD en el cual se gestiona la conexión con la base de datos a utilizar. Además cuenta con todos los métodos de realización de CRUD y otros métodos que se han requerido y están especificados en las clases DP.

Código de Clase MD

```
public class TerapiaOcupacionalMD {

    ConexionBDD conexionBDD;
    TerapiaOcupacionalDP terapiaOcupacionalDP;
    Statement statement;
```

Figura 63: Código de instanciación de clase MD (Sánchez, 2017)

```
public TerapiaOcupacionalMD(TerapiaOcupacionalDP terapiaOcupacionalDP) {
    this.terapiaOcupacionalDP = terapiaOcupacionalDP;
    conexionBDD = new ConexionBDD();
}
```

Figura 64: Código de constructor de clase MD (Sánchez, 2017)

```
public boolean ingresarInforme()
    {
        this.statement = conexionBDD.conectar();
        boolean retorno = true;
        try {

            String update = "insert into terapiaocupacional (cedula, cedulaUsuario, lugarNacimiento, diagnostico, sector, "
                      + "representante, fechaEntrevista, eliminado)"
                    + "values ('"+terapiaOcupacionalDP.getCedula()+"', '"+terapiaOcupacionalDP.getCedulaUsuario()+"', "
                    + "'"+terapiaOcupacionalDP.getLugarNacimiento()+"', "
                    + "'"+terapiaOcupacionalDP.getDiagnostico()+"', '"+terapiaOcupacionalDP.getSector()+"', "
                    + "'"+terapiaOcupacionalDP.getRepresentante()+"', "
                    + "'"+terapiaOcupacionalDP.getFechaEntrevista()+"', "+terapiaOcupacionalDP.getEliminado()+")";
            statement.executeUpdate(update);
            conexionBDD.desconectar();
        } catch (SQLException ex) {
            System.out.println(ex);
            retorno = false;
        }
        return retorno;
    }
```

Figura 65: Código de ingreso MD (Sánchez, 2017)

```
public boolean existeFicha()
    {
        this.statement = conexionBDD.conectar();
        boolean retorno = true;
        String cedula = "";
        try {
            String query = "select cedula from terapiaocupacional where Cedula = '"+terapiaOcupacionalDP.getCedula()+"'"
            +"and Eliminado = 0";
            ResultSet resultSet = statement.executeQuery(query);
            while(resultSet.next())
            {
                cedula = resultSet.getString("Cedula");
            }
            conexionBDD.desconectar();
            if(cedula.equals(""))
                return false;
        } catch (SQLException ex) {
            Logger.getLogger(UsuarioMD.class.getName()).log(Level.SEVERE, null, ex);
        }
        return retorno;
    }
```

Figura 66: Código de existencia de ficha MD (Sánchez, 2017)

```
public ArrayList<String> obtenerFechas()
{
    this.statement = conexionBDD.conectar();
    boolean retorno = true;
    ArrayList<String> listaFechas = new ArrayList<>();
    try {
        String query = "select fechaEntrevista from terapiaocupacional where Cedula = '"
        +terapiaOcupacionalDP.getCedula()+"' and Eliminado = 0";
        ResultSet resultSet = statement.executeQuery(query);
        while(resultSet.next())
        {
            listaFechas.add(resultSet.getString("fechaEntrevista"));
        }
        conexionBDD.desconectar();
        return listaFechas;
    } catch (SQLException ex) {
        Logger.getLogger(UsuarioMD.class.getName()).log(Level.SEVERE, null, ex);
    }
    return listaFechas;
```

Figura 67: Código de obtención de fichas MD (Sánchez, 2017)

```
public boolean modificarFicha()
    {
        this.statement = conexionBDD.conectar();
        boolean retorno = true;
        try {
            String update = "update terapiaocupacional set lugarNacimiento = '"+terapiaOcupacionalDP.getLugarNacimiento()+"',+
            "diagnostico = '"+terapiaOcupacionalDP.getDiagnostico()+"', "sector = '"+terapiaOcupacionalDP.getSector()+"',+
            "Representante = '"+terapiaOcupacionalDP.getRepresentante()+"', fechaEntrevista = '"+terapiaOcupacionalDP.getFechaEntrevista()+"',"
                    + "eliminado = 0"
                    + "where Cedula = '"+terapiaOcupacionalDP.getCedula()+"' and fechaEntrevista = '"+terapiaOcupacionalDP.getFechaEntrevista()+"'";
            statement.executeUpdate(update);
            conexionBDD.desconectar();
        } catch (SQLException ex) {
            Logger.getLogger(UsuarioMD.class.getName()).log(Level.SEVERE, null, ex);
            retorno = false;
        }
        return retorno;
    }
```

Figura 68: Código de modificación de ficha MD (Sánchez, 2017)

```
public TerapiaOcupacionalDP buscarTodo()
    {
        this.statement = conexionBDD.conectar();
        TerapiaOcupacionalDP retorno = new TerapiaOcupacionalDP();
        try {
            String query = "select * from terapiaocupacional where Cedula = '"+terapiaOcupacionalDP.getCedula()+"'"
            +"and Eliminado = 0 and fechaEntrevista = '"+terapiaOcupacionalDP.getFechaEntrevista()+"'";
            ResultSet resultSet = statement.executeQuery(query);
            while(resultSet.next())
            {
                retorno.setLugarNacimiento(resultSet.getString("LugarNacimiento"));
                retorno.setDiagnostico(resultSet.getString("Diagnostico"));
                retorno.setSector(resultSet.getString("Sector"));
                retorno.setRepresentante(resultSet.getString("Representante"));
            }
            conexionBDD.desconectar();
            return retorno;
        } catch (SQLException ex) {
            Logger.getLogger(UsuarioMD.class.getName()).log(Level.SEVERE, null, ex);
        }
        return retorno;
    }
```

Figura 69: Código de búsqueda de ficha completa MD (Sánchez, 2017)

```
public boolean eliminarFicha()
{
    this.statement = conexionBDD.conectar();
    boolean retorno = true;
    try {

        String update = "update terapiaocupacional set Eliminado = "+terapiaOcupacionalDP.getEliminado()+""
                + " where Cedula = '"+terapiaOcupacionalDP.getCedula()+"' and fechaEntrevista = '"+
                + "terapiaOcupacionalDP.getFechaEntrevista()+"'";
        statement.executeUpdate(update);
        conexionBDD.desconectar();
    } catch (SQLException ex) {
        Logger.getLogger(EstudianteMD.class.getName()).log(Level.SEVERE, null, ex);
        retorno = false;
    }
    return retorno;
}
```

Figura 70: Código de eliminación de ficha MD (Sánchez, 2017)

```
public int obtenerID()
    {
        this.statement = conexionBDD.conectar();
        int retorno = 0;
        try {
            String query = "select idTerapiaOcupacional from terapiaocupacional "
                    + "where Cedula = '"+terapiaOcupacionalDP.getCedula()+"' and Eliminado = 0 and fechaEntrevista = "
                    +"'"+terapiaOcupacionalDP.getFechaEntrevista()+"'";
            ResultSet resultSet = statement.executeQuery(query);
            while(resultSet.next())
            {
                retorno = resultSet.getInt("idTerapiaOcupacional");
            }
            conexionBDD.desconectar();
            return retorno;
        } catch (SQLException ex) {
            Logger.getLogger(UsuarioMD.class.getName()).log(Level.SEVERE, null, ex);
        }
        return retorno;
    }
```

Figura 71: Código de obtención de ID de ficha MD (Sánchez, 2017)

```
public TerapiaOcupacionalDP obtenerUltimo()
    {
        this.statement = conexionBDD.conectar();
        TerapiaOcupacionalDP retorno = new TerapiaOcupacionalDP();
        try {
            String query = "select * from terapiaocupacional where Cedula = '"+terapiaOcupacionalDP.getCedula()+"' and Eliminado = 0 "
                    + "Order by idTerapiaOcupacional DESC LIMIT 1";
            ResultSet resultSet = statement.executeQuery(query);
            while(resultSet.next())
            {
                retorno.setIdTerapiaOcupacional(resultSet.getInt("IdTerapiaOcupacional"));
                retorno.setLugarNacimiento(resultSet.getString("LugarNacimiento"));
                retorno.setDiagnostico(resultSet.getString("Diagnostico"));
                retorno.setSector(resultSet.getString("Sector"));
                retorno.setRepresentante(resultSet.getString("Representante"));
                retorno.setFechaEntrevista(resultSet.getDate("FechaEntrevista"));
            }
            conexionBDD.desconectar();
            return retorno;
        } catch (SQLException ex) {
            Logger.getLogger(UsuarioMD.class.getName()).log(Level.SEVERE, null, ex);
        }
        return retorno;
    }
}
```

Figura 72: Código de obtención de último registro (Sánchez, 2017)

5.3.1. Clase de Conexión a la base de datos

En esta clase se gestiona la conexión a la base de datos, definiendo el usuario, contraseña, IP del servidor y puerto al que está asociada la base de datos.

Código:

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

Figura 73: Código de instanciación de clase conexión (Sánchez, 2017)

```
public Statement conectar()
{
    try {
        desconectar();
        String url = "jdbc:postgresql://localhost:5432/IEPNI";
        Class.forName("org.postgresql.Driver");
        conexion = DriverManager.getConnection(url, "postgres", "postgres");
        conexion.setAutoCommit(false);
        statement = conexion.createStatement();
    } catch (SQLException | ClassNotFoundException ex) {
        System.out.println("Error al conectar a la base de datos");
    }
    return statement;

}
```

Figura 74: Código de conexión a la base de datos (Sánchez, 2017)

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

Figura 75: Código de desconexión de la base de datos (Sánchez, 2017)

5.5. Reportes

Los reportes se realizaron utilizando la librería Itext.jar mediante la cual se genera un documento PDF de tipo texto, obteniendo como resultado una ficha idéntica a la que se manejaba antes de la automatización.

Para realizar un reporte se debe seleccionar una ficha a través del GUI y el sistema genera un reporte buscando cada uno de los campos de la ficha principal y las fichas secundarias.

El código que se muestra a continuación pertenece a como se realiza un reporte de la ficha BioPsicosocial desde la clase ReporteDP. Todos los demás reportes se manejan de la misma manera que el mostrado a continuación.

Código:

```
public boolean generarReporteBioPsicosocial(File archivo)
    {
        boolean retorno = true;

        Document documento = new Document(PageSize.LETTER, 80, 80, 75, 75);
        PdfWriter writer = null;
        try {
            writer = PdfWriter.getInstance(documento, new FileOutputStream(archivo.getAbsolutePath()+".pdf"));
        } catch (FileNotFoundException | DocumentException ex) {
            ex.getMessage();
            return false;
        }
```

Figura 76: Código de creación de documento (Sánchez, 2017)

```
Date fecha = bioPsicosocialDP.getFechaEntrevista();
anamnesisFamiliarDP = new AnamnesisFamiliarDP();
int idBioPsicosocial = bioPsicosocialDP.obtenerID();
anamnesisFamiliarDP.setIdBioPsicosocial(idBioPsicosocial);
anamnesisFamiliarDP = anamnesisFamiliarDP.buscarTodo();
anamnesisPersonalDP = new AnamnesisPersonalDP();
anamnesisPersonalDP.setIdBioPsicosocial(idBioPsicosocial);
anamnesisPersonalDP = anamnesisPersonalDP.buscarTodo();
conclusionesBioDP = new ConclusionesBioDP();
conclusionesBioDP.setIdPsicosocial(idBioPsicosocial);
conclusionesBioDP = conclusionesBioDP.buscarTodo();
bioPsicosocialDP = bioPsicosocialDP.buscarTodo();
```

Figura 77: Código de búsqueda para creación de ficha (Sánchez, 2017)

```
estudianteDP = estudianteDP.consultarEstudianteTodo();

documento.addAuthor("IEPNI");
documento.open();
documento.addTitle("DATOS DEL ESTUDIANTE");
```

Figura 78: Código de instanciación de documento (Sánchez, 2017)

```
Paragraph parrafo0 = new Paragraph();
parrafo0.setAlignment(Paragraph.ALIGN_CENTER);
parrafo0.setFont(FontFactory.getFont("Times New Roman", 16, Font.BOLD, BaseColor.BLACK));
parrafo0.add("HISTORIA BIO PSICOSOCIAL\n\n");


Paragraph parrafo1 = new Paragraph();
parrafo1.setAlignment(Paragraph.ALIGN_LEFT);
parrafo1.setFont(FontFactory.getFont("Times New Roman", 12, Font.BOLD, BaseColor.BLACK));
parrafo1.add("DATOS DE IDENTIFICACIÓN\n\n");
```

Figura 79: Código de creación de título de documento (Sánchez, 2017)

```
Paragraph parrafo2 = new Paragraph();
parrafo2.setAlignment(Paragraph.ALIGN_LEFT);
parrafo2.setFont(FontFactory.getFont("Times New Roman", 12, Font.NORMAL, BaseColor.BLACK));
parrafo2.add("Nombre: "+estudianteDP.getNombres().trim()+"\nF. Nacimiento:
+"+estudianteDP.getFechaNacimiento()+"\nEdad: "+estudianteDP.getEdad()+
"\nFecha de la Entrevista"+fecha+"\nEscolaridad: "+bioPsicosocialDP.getEscolaridad().trim()+
"\nDirección Dom/Telef: "+estudianteDP.getDireccion().trim()+" / "+estudianteDP.getTelefono().trim());
```

Figura 80: Código de creación de texto del documento (Sánchez, 2017)

```
try {
            documento.add(parrafo0);
            documento.add(parrafo1);
            documento.add(parrafo2);
            documento.add(parrafo3);
            documento.add(parrafo4);
            documento.add(parrafo5);
            documento.add(parrafo6);
            documento.add(parrafo7);
            documento.add(parrafo8);
            documento.add(parrafo9);
            documento.add(parrafo10);
            documento.add(parrafo11);
            documento.add(parrafo12);
        } catch (DocumentException ex) {
            return false;
        }
        documento.close();
        writer.close();
```

Figura 81: Código de creación de documento (Sánchez, 2017)

Las demás partes de código se las puede ver en el anexo10. Todo el código JAVA se encuentra ahí.

CAPÍTULO 6: TRANSICIÓN

El capítulo siguiente muestra las pruebas que se realizarán en el sistema y el informe firmado por el cliente asegurando que el producto funciona de manera correcta.

6.1. Pruebas de Caja Negra

Entradas	Resultados Esperados	C.U Estudio	Resultado
Despliegue de pantallas	Todas las pantallas se muestran en el orden adecuado		OK
Inicio de sesión correcto	Se inicia sesión de acuerdo a los usuarios de la BDD	RF1	OK
Ingreso de Fichas	Las fichas se ingresan de manera correcta	RF 2,3,4,5,6,7	OK
Modificación de Fichas	Las fichas se modifican de manera correcta	RF 2,3,4,5,6,8	OK
Eliminación de Fichas	Las fichas se eliminan de la base de datos adecuadamente	RF 2,3,4,5,6,9	OK
Generación de reportes	Los reportes se generan como se especificó al inicio del desarrollo	RF 2,3,4,5,6,7	OK

Tabla 6: Pruebas de Caja Negra (Sánchez, 2017)

6.2. Pruebas de Caja Blanca

Entradas	Resultados Esperados	C.U Estudio	Resultado
Verificación de Datos numéricos	En los campos numéricos se prohíbe el ingreso de caracteres no numéricos		OK
Verificación de datos Textuales	Se verifica que el tamaño del campo a ingresar no sobrepase a lo especificado en la base de datos		OK
Verificación de integridad de datos	Se verifica que los datos ingresados vayan acorde a las fichas físicas		OK
Verificación de existencia de estudiantes antes de ingresar una ficha	Se verifica que existe un estudiante antes de poder ingresar una ficha	RF 2,3,4,5,6,7	OK
Verificación de existencia de fichas antes de modificar o eliminar	Se verifica que exista una ficha antes de modificarla o eliminarla	RF 2,3,4,5,6,7	OK
Verificación de existencia de fichas antes de generar un reporte	Se verifica que exista una ficha antes de crear un reporte	RF 2,3,4,5,6,7	OK

Tabla 7: Pruebas de Caja Blanca (Sánchez, 2017)

6.3. Informe de acuerdo al plan de pruebas

PRUEBAS DEL SISTEMA

RF1 Gestión de Usuarios

Entradas	Resultados Esperados	C.U Estudio	Resultado
Ingreso de cédula	Sistema valida cédula correcta	RF1	✓
Ingreso de Nombres y apellidos	Sistema controla el ingreso correcto de información	RF1	✓
Selección de tipo de usuario	El sistema verifica que un tipo de usuario se ha especificado	RF1	✓
Contraseña	El sistema verifica que las contraseñas ingresadas sean iguales	RF1	✓

Directora IEPNI

Estudiante PUCE
Sebastián Sánchez

Figura 82: Pruebas de usuario 1 (Sánchez, 2017)

RF Gestión de Fichas General

Entradas	Resultados Esperados	C.U Estudio	Resultado
Ingreso de cédula	Sistema valida cédula correcta	RF7	✓
Verificación de existencia de usuario	El sistema busca existencia de fichas	RF7	✓
verificación de existencia de fichas	El sistema busca la existencia de fichas	RF2,3,4,5,6,7	✓
Verifica ingreso de datos textuales	El sistema se revisa que la integridad de los datos se mantenga	RF2,3,4,5,6,7	✓
Verifica ingreso de datos numéricos	El sistema se revisa que la integridad de los datos se mantenga	RF2,3,4,5,6,7	✓
Verifica ingreso de datos que no sobre pasen el tamaño especificado en la BDD	El sistema se revisa que la integridad de los datos se mantenga	RF2,3,4,5,6,7	✓
Ingreso de Fichas	Las fichas se ingresan sin problemas	RF2,3,4,5,6,7	✓
Modificación de Fichas	Las fichas se modifican sin problemas	RF2,3,4,5,6,7	✓
Eliminación de Fichas	Las fichas se eliminan sin problemas	RF2,3,4,5,6,7	✓

Directora IEPNI

Estudiante PUCE
Sebastián Sánchez

Figura 83: Pruebas de usuario 2 (Sánchez, 2017)

RF Reportes

Entradas	Resultados Esperados	C.U Estudio	Resultado
Ingreso de cédula	Sistema valida cédula correcta	RF7	✓
Verificación de existencia de usuario	El sistema busca existencia de fichas	RF7	✓
verificación de existencia de fichas	El sistema busca la existencia de fichas	RF2,3,4,5,6,7	✓
Generación de reportes	Los reportes se generan adecuadamente	RF2,3,4,5,6,7	✓

Directora IEPNI

Estudiante PUCE
Sebastián Sánchez

Figura 84: Pruebas de usuario 3 (Sánchez, 2017)

CAPÍTULO 7. CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

- Los sistemas computacionales deben ser diseñados y desarrollados de acuerdo a las especificaciones que el usuario otorgue, en el caso del sistema actual los usuarios son profesores y secretarias de la fundación IEPNI.
- La metodología RUP facilitó la organización del desarrollo completo del software, ya que al realizar el sistema en dos ciclos se necesitó seguir una secuencia lógica de pasos para lograr la eficiencia del proceso.
- JAVA al ser un lenguaje multiplataforma gracias a su máquina virtual permite que el sistema pueda ser replicado en cualquier sistema operativo.
- Las librerías .jar como jCalendar o iText permitieron facilitar mucho el desarrollo del sistema ya que otorgó al programador las herramientas necesarias para completar adecuadamente el proyecto.
- La validación de los campos de ingreso de usuario por pantalla es fundamental para poder mantener la integridad de los datos en la base de datos.
- La fundación IEPNI necesita el sistema que se entregó para la búsqueda rápida de las fichas antes mencionadas.

7.2. Recomendaciones

- El sistema está desarrollado en una plataforma cliente servidor, como recomendación para ampliar a futuro el sistema o inclusive que sea usado por otras instituciones, se podría migrarlo a un sistema Web, se debería cambiar todo el paquete de GUI pero los demás paquetes se pueden reutilizar.
- El sistema no cuenta con una opción de respaldar la base de datos desde el sistema actualmente solo se puede respaldar desde el PGAdminIII que es la interfaz de usuario de PostgreSQL.
- Se puede ampliar el software haciendo todo el sistema académico de la institución, para manejar lo que son notas, asistencia y registros académicos en general.
- Debido a los recursos de la fundación el sistema está programado de manera local, para siguientes versiones si la institución tiene la oportunidad de tener un servidor se puede migrar la base de datos al servidor y acceder en red a la base de datos, habría que cambiar la clase de conexionBDD para esto ya que aquí se gestionan todas las conexiones.

- Al ingresar la cédula de un estudiante por comodidad para el usuario en las próximas versiones del programa se podría añadir un campo que indique los nombres del estudiante asociado a esa cédula para poder verificar que es el estudiante correcto.
- Se puede mejorar el sistema incluyendo funcionalidades de software para que se puedan llevar registros estadísticos de los datos que necesite la fundación.

ANEXOS

1. Modelo conceptual de la base de datos
2. Diagrama de clases y secuencia
3. Diagrama de clases DP conceptual
4. Diagrama de clases MD conceptual
5. Modelo entidad relación conceptual
6. Modelo entidad relación físico
7. Interfaces de usuario
8. Diagramas de casos de uso
9. Diagramas de actividades
10. Programa (Código e Instalador)
11. Manual de usuario
12. SRS
13. Certificado de la Fundación IEPNI

BIBLIOGRAFÍA

Arquitectura de tres niveles - EcuRed. (s/f). Recuperado el 29 de agosto de 2016, a partir de http://www.ecured.cu/Arquitectura_de_tres_niveles

Belmonte, O. (2005). *Introducción al Lenguaje de Prograación Java*. Recuperado a partir de http://www3.uji.es/~belfern/pdidoc/IX26/Documentos/introJava.pdf

Computación I: Unidad III. (2017, enero 19). Recuperado el 19 de enero de 2017, a partir de http://www.eplc.umich.mx/salvadorgs/compu1/contenido/U_III/UIII.html

De los Ángeles, J. (2003). Programacion Orientada a Objetos. Universidad Autónoma de Puebla. Recuperado a partir de http://pdi.topografia.upm.es/m.manso/docencia/Informatica_plan92/Curso-2002-2003/poo.pdf

Diagramas UML. (s/f). Recuperado a partir de http://www.teatroabadia.com/es/uploads/documentos/iagramas_del_uml.pdf

Garcia, A. (s/f). *Manual Práctico de SQL*. Recuperado a partir de http://www.lawebdelprogramador.com/cursos/archivos/ManualPracticoSQL.pdf

Rational. (2011). *Rational Unified Process*. Rational Software White Paper. Recuperado a partir de https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Silberschatz, A., Korth, H., & Sudarshan, S. (2002). *Fundamentos de Bases de Datos* (cuarta). McGRAW-HILL/INTERAMERICANA DE ESPAÑA. Recuperado a partir de https://unefazuliasistemas.files.wordpress.com/2011/04/fundamentos-de-bases-de-datos-silberschatz-korth-sudarshan.pdf

TPM | Tutorial de Programación Multiplataforma. (s/f). Recuperado el 29 de agosto de 2016, a partir de http://itslr.edu.mx/archivos2013/TPM/temas/s2u1.html

Cruz, J., Italo, G., & Echeverria, F. (s.f.). • *Analizar las fichas que la institución utiliza para decidir cuál es la mejor forma de automatizarlas mediante un sistema computacional.* Guayaquil.

Sánchez, A. S. (2017). *Diseño y Desarrollo de un Sistema Computacional de Registros Bio-Psicosociales y Terapia Ocupacional para Niños y Jóvenes con Discapacidad Intelectual para la Fundación IEPNI usando la Metodología del Proceso Unificado Racional RUP*. Quito, Ecuador.