PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

ESCUELA DE SISTEMAS

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

"DESARROLLO DE UN SISTEMA ENCARGADO DE LA GESTIÓN DE INFORMACIÓN DE CLIENTES PARA LA FUNDACIÓN FUDRINE USANDO LA METODOLOGÍA DEL PROCESO RACIONAL UNIFICADO RUP"

JUAN CARLOS CONTRERAS PAREDES

DIRECTOR: INGENIERO FABIÁN DE LA CRUZ

CIUDAD: QUITO AÑO: 2017

AGRADECIMIENTOS

Primero que nada quiero agradecer a Dios por darme esta oportunidad de vida donde día a día tengo un constante aprendizaje y preparación tanto de forma intelectual como espiritual.

Agradezco a mi familia por ser una pieza clave en el desarrollo tanto de mi vida personal y estudiantil. Ellos han sido quienes se han encontrado conmigo en todo momento y me han brindado los recursos necesarios para encontrarme como actualmente me encuentro.

Gracias a todo el personal de Ingeniería en Sistemas de la Pontificia Universidad Católica de Ecuador por brindarme conocimiento permitiéndome tener un futuro profesional. Agradezco a mi tutor Fabián de la Cruz y a mis revisores Damián Nicolalde y Alfredo Calderón por ayudarme en el desarrollo de mi proyecto de disertación.

Me siento muy afortunado y orgulloso de ser lo que soy y lograr lo que he logrado, donde sin duda mi familia, amistades, compañeros y profesores han sido de suma importancia para llegar hasta este punto.

ÍNDICE

INTRODUCCIÓN 10
1. CAPÍTULO 1: FUDRINE 12
1.1. HISTORIA 12
1.2. SITUACIÓN ACTUAL 13
1.2.1. MISIÓN 13
1.2.2. VISIÓN 13
1.2.3. VALORES 13
1.2.4. PRINCIPIOS 14
1.2.5. RAZÓN DE SER 14
1.2.6. MÉTODOS DE GESTIÓN DE INFORMACIÓN 14
1.2.7. NECESIDADES 15
1.3. MAPA DE PROCESOS 16
1.4. MAPA JERÁRQUICO 16
2. FUNDAMENTOS TEÓRICOS 17
2.1. GESTIÓN DE INFORMACIÓN 17
2.1.1. APLICACIÓN DE LA GESTIÓN DE INFORMACIÓN EN LA FUNDACIÓN FUDRINE 18
2.2. SELECCIÓN DE LA PLATAFORMA 19
2.2.2. LENGUAJE DE PROGRAMACIÓN 22
2.2.3. ARQUITECTURA DE SOFTWARE 25
2.2.4. PLATAFORMA 27
2.3. TIPO DE APLICACIÓN 28
2.4. METODOLOGÍA RUP 29
2.4.1. PRINCIPIOS 30
2.4.2. CICLO DE VIDA 32
2.4.3. CARACTERÍSTICAS 34
2.4.4. FASES 36
3. INICIACIÓN 40
3.1. DOCUMENTO DE VISIÓN 40
3.1.1. INTRODUCCIÓN 40
3.1.2. POSICIONAMIENTO 41
3.1.3. DESCRIPCIÓN DE LA PARTE INTERESADA Y DEL USUARIO 42
3.1.3.4. PERFILES DE PARTE INTERESADA 44

3.1.4. VISIÓN GENERAL DEL PRODUCTO 47
3.1.6. RESTRICCIONES 51
3.1.7. RANGOS DE CALIDAD 51
3.1.8. PRECEDENCIA Y PRIORIDAD 51
3.1.9. OTROS REQUISITOS DEL PRODUCTO 52
3.1.10. REQUISITOS DE DOCUMENTACIÓN 52
3.2. ESPECIFICACIÓN DE REQUERIMIENTOS 53
3.2.1. DESCRIPCIÓN DE REQUERIMIENTOS 53
3.3. DIAGRAMA DE CASOS DE USO 56
3.3.1. DIAGRAMA GENERAL DE CASO DE USO DEL SISTEMA 56
3.3.2. DIAGRAMAS A DETALLE DE CASOS DE USO DEL SISTEMA 57
4. ELABORACIÓN 79
4.1. DOCUMENTO DE ARQUITECTURA DE SOFTWARE (SAD) 79
4.1.1. VISTA LÓGICA 79
4.1.2. VISTA DE IMPLEMENTACIÓN 82
4.2. DISEÑO DE INTERFACES 99
4.2.1. FUNCIONALIDAD 0 - INGRESO AL SISTEMA 100
4.2.2. FUNCIONALIDAD 1 – GESTIONAR PACIENTES 100
4.2.3. FUNCIONALIDAD 2 – GESTIONAR EMPLEADOS 101
4.2.4. FUNCIONALIDAD 3 – GESTIONAR REPRESENTANTES 101
4.2.5. FUNCIONALIDAD 4 – GESTIONAR ANTECEDENTES 102
4.2.6. FUNCIONALIDAD 5 – GESTIONAR FICHAS 103
4.2.7. FUNCIONALIDAD 6 – GENERAR REPORTES 103
4.2.8. FUNCIONALIDAD 7 – SALIR DEL SISTEMA 104
4.3. DESARROLLO DEL PLAN DE PRUEBAS 104
4.3.1. FUNCIONALIDAD 0 - INGRESO AL SISTEMA 105
4.3.2. FUNCIONALIDAD 1 - GESTIONAR PACIENTES 105
4.3.3. FUNCIONALIDAD 6 - GENERAR REPORTES 106
4.3.4. FUNCIONALIDAD 7 – SALIR DEL SISTEMA 106
5. CONSTRUCCIÓN Y TRANSICIÓN 107
5.1. ESPECIFICACIONES DE REQUISITOS FALTANTES 107
5.2. DESARROLLO DE DIAGRAMAS DE CASOS DE USO 108
5.3. PRUEBAS DEL SISTEMA 116
5.3.1. FUNCIONALIDAD 0 - INGRESO AL SISTEMA 116
5.3.2. FUNCIONALIDAD 1 - GESTIONAR PACIENTES 117

5.3.3. FUNCIONALIDAD 2 - GESTIONAR EMLEADOS 118
5.3.4. FUNCIONALIDAD 3 - GESTIONAR REPRESENTANTES 119
5.3.5. FUNCIONALIDAD 4 - GESTIONAR ANTECEDENTES 120
5.3.6. FUNCIONALIDAD 5 - GESTIONAR FICHAS 121
5.3.7. FUNCIONALIDAD 6 - GENERAR REPORTES 122
5.3.8. FUNCIONALIDAD 7 – SALIR DEL SISTEMA 122
5.4. PRUEBAS DE ACEPTACIÓN 123
6. CONCLUSIONES Y RECOMENDACIONES 123
6.1. CONCLUSIONES 123
6.2. RECOMENDACIONES 124
BIBLIOGRAFÍA 125

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Mapa de procesos de la fundación FUDRINE, 2017 16
Ilustración 2: Mapa jerárquico de la fundación FUDRINE, 2017 16
Ilustración 3: Interacción de un programa con arquitectura de tres capas, 2017 26
Ilustración 4: Ciclo de Vida - Modelo en Espiral, 2017 33
Ilustración 5: Fases e iteraciones, 2016 36
Ilustración 6: Perspectiva del Producto, 2017 48
Ilustración 7: Diagrama General de Sistema de Gestión de Información (FUDRINE), 2017 57
Ilustración 8: Diagrama Funcionalidad 0 - Ingreso al Sistema del Sistema de Gestión de Información (FUDRINE), 2017 58
Ilustración 9: Diagrama Funcionalidad 1 – Gestionar Pacientes del Sistema de Gestión de Información (FUDRINE), 2017 59
Ilustración 10: Diagrama Funcionalidad 1.1 - Ingresar Paciente del Sistema de Gestión de Información (FUDRINE), 2017 60
Ilustración 11: Diagrama Funcionalidad 1.2 – Editar Paciente del Sistema de Gestión de Información (FUDRINE), 2017 61
Ilustración 12: Diagrama Funcionalidad 1.3 – Eliminar Paciente del Sistema de Gestión de Información (FUDRINE), 2017 63
Ilustración 13: Diagrama Funcionalidad 1.4 – Consultar Paciente del Sistema de Gestión de Información (FUDRINE), 2017 64
Ilustración 14: Diagrama Funcionalidad 4 – Gestionar Antecedentes del Sistema de Gestión de Información (FUDRINE), 2017 65
Ilustración 15: Diagrama Funcionalidad 4.1 - Ingresar Representante del Sistema de Gestión de Información (FUDRINE), 2017 66
Ilustración 16: Diagrama Funcionalidad 4.2 – Editar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017 67
Ilustración 17: Diagrama Funcionalidad 4.3 – Eliminar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017 69
Ilustración 18: Diagrama Funcionalidad 4.4 – Consultar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017 70
Ilustración 19: Diagrama Funcionalidad 5 – Gestionar Fichas del Sistema de Gestión de Información (FUDRINE), 2017 71
Ilustración 20: Diagrama Funcionalidad 5.1 - Ingresar Ficha del Sistema de Gestión de Información (FUDRINE), 2017 72
Ilustración 21: Diagrama Funcionalidad 5.2 – Editar Ficha del Sistema de Gestión de Información (FUDRINE), 2017 73
Ilustración 22: Diagrama Funcionalidad 5.3 – Eliminar Ficha del Sistema de Gestión de Información (FUDRINE), 2017 74
Ilustración 23: Diagrama Funcionalidad 5.4 – Consultar Ficha del Sistema de Gestión de Información (FUDRINE), 2017 76
Ilustración 24: Diagrama Funcionalidad 6 – Generar Reportes del Sistema de Gestión de Información (FUDRINE), 2017 77

Ilustración 25: Diagrama Funcionalidad 7 – Salir del Sistema de Gestión de Información (FUDRINE), 2017 78
Ilustración 26: Diagrama de Clases (Dominio del Problema) – Sistema de Gestión de Información (FUDRINE), 2017 81
Ilustración 27: Diagrama de Modelo (E-R) –Sistema de Gestión de Información (FUDRINE), 2017 82
Ilustración 28: Diagrama de Secuencia - Funcionalidad 0: Ingreso al Sistema del Sistema de Gestión de Información (FUDRINE), 2017 83
Ilustración 29: Diagrama de Secuencia - Funcionalidad 1.1: Ingresar Paciente del Sistema de Gestión de Información (FUDRINE), 2017 84
Ilustración 30: Diagrama de Secuencia - Funcionalidad 1.1: Editar Paciente del Sistema de Gestión de Información (FUDRINE), 2017 85
Ilustración 31: Diagrama de Secuencia - Funcionalidad 1.1: Eliminar Paciente del Sistema de Gestión de Información (FUDRINE), 2017 85
Ilustración 32: Diagrama de Secuencia - Funcionalidad 1.1: Consultar Paciente del Sistema de Gestión de Información (FUDRINE), 2017 86
Ilustración 33: Diagrama de Secuencia - Funcionalidad 4.1: Ingresar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017 86
Ilustración 34: Diagrama de Secuencia - Funcionalidad 4.2: Editar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017 87
Ilustración 35: Diagrama de Secuencia - Funcionalidad 4.2: Editar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017 87
Ilustración 36: Diagrama de Secuencia - Funcionalidad 4.3: Consultar Antecedentes del Sistema de Gestión de Información (FUDRINE), 2017 88
Ilustración 37: Diagrama de Secuencia - Funcionalidad 5.1: Ingresar Ficha del Sistema de Gestión de Información (FUDRINE), 2017 88
Ilustración 38: Diagrama de Secuencia - Funcionalidad 5.1: Ingresar Ficha del Sistema de Gestión de Información (FUDRINE), 2017 89
Ilustración 39: Diagrama de Secuencia - Funcionalidad 5.3: Eliminar Ficha del Sistema de Gestión de Información (FUDRINE), 2017 89
Ilustración 40: Diagrama de Secuencia - Funcionalidad 5.4: Consultar Fichas del Sistema de Gestión de Información (FUDRINE), 2017 90
Ilustración 41: Diagrama de Secuencia - Funcionalidad 6: Generar Reportes en el Sistema de Gestión de Información (FUDRINE), 2017 90
Ilustración 42: Diagrama de Secuencia - Funcionalidad 7: Salir del Sistema de Gestión de Información (FUDRINE), 2017 91
Ilustración 43: Diagrama de Colaboración - Funcionalidad 0: Ingreso al Sistema del Sistema de Gestión de Información (FUDRINE), 2017 92
Ilustración 44: Diagrama de Colaboración - Funcionalidad 1.1: Ingresar Paciente del Sistema de Gestión de Información (FUDRINE), 2017 93
Ilustración 45: Diagrama de Colaboración - Funcionalidad 1.1: Editar Paciente del Sistema de Gestión de Información (FUDRINE), 2017 94
Ilustración 46: Diagrama de Colaboración - Funcionalidad 1.1: Eliminar Paciente del Sistema de Gestión de Información (FUDRINE), 2017 94
Ilustración 47: Diagrama de Colaboración - Funcionalidad 1.1: Consultar Paciente del Sistema de Gestión de Información (FUDRINE), 2017 94
Ilustración 48: Diagrama de Colaboración - Funcionalidad 4.1: Ingresar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017 95

Ilustración 49: Diagrama de Colaboración - Funcionalidad 4.2: Editar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017 95
Ilustración 50: Diagrama de Colaboración - Funcionalidad 4.2: Editar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017 96
Ilustración 51: Diagrama de Colaboración - Funcionalidad 4.3: Consultar Antecedentes del Sistema de Gestión de Información (FUDRINE), 2017 96
Ilustración 52: Diagrama de Colaboración - Funcionalidad 5.1: Ingresar Ficha del Sistema de Gestión de Información (FUDRINE), 2017 97
Ilustración 53: Diagrama de Colaboración - Funcionalidad 5.1: Ingresar Ficha del Sistema de Gestión de Información (FUDRINE), 2017 97
Ilustración 54: Diagrama de Colaboración - Funcionalidad 5.3: Eliminar Ficha del Sistema de Gestión de Información (FUDRINE), 2017 98
Ilustración 55: Diagrama de Colaboración - Funcionalidad 5.4: Consultar Fichas del Sistema de Gestión de Información (FUDRINE), 2017 98
Ilustración 56: Diagrama de Colaboración - Funcionalidad 6: Generar Reportes en el Sistema de Gestión de Información (FUDRINE), 2017 99
Ilustración 57: Diagrama de Secuencia - Funcionalidad 7: Salir del Sistema de Gestión de Información (FUDRINE), 2017 99
Ilustración 58: Diseño de Interfaz - Funcionalidad 0: Ingreso al Sistema de Gestión de Información (FUDRINE), 2017 100
Ilustración 59: Diseño de Interfaz - Funcionalidad 1: Gestión de Pacientes en el Sistema de Gestión de Información (FUDRINE), 2017 101
Ilustración 60: Diseño de Interfaz - Funcionalidad 2: Gestión de Empleados en el Sistema de Gestión de Información (FUDRINE), 2017 101
Ilustración 61: Diseño de Interfaz - Funcionalidad 3: Gestión de Representantes en el Sistema de Gestión de Información (FUDRINE), 2017 102
Ilustración 62: Diseño de Interfaz - Funcionalidad 4: Gestión de Antecedentes en el Sistema de Gestión de Información (FUDRINE), 2017 102
Ilustración 63: Diseño de Interfaz - Funcionalidad 5: Gestión de Fichas en el Sistema de Gestión de Información (FUDRINE), 2017 103
Ilustración 64: Diseño de Interfaz - Funcionalidad 6: Generar Reportes en el Sistema de Gestión de Información (FUDRINE), 2017 103
Ilustración 65: Diseño de Interfaz - Funcionalidad 7: Salir del Sistema de Gestión de Información (FUDRINE), 2017 104
Ilustración 66: Estructura del Sistema (FUDRINE), 2017 108
Ilustración 67: Clases de los Paquetes del Sistema (FUDRINE), 2017 109
Ilustración 68: Interfaz – Funcionalidad 2 Gestionar Empleados (FUDRINE), 2017 109
Ilustración 69: Interfaz – Funcionalidad 2 Gestionar Empleados - Consulta (FUDRINE), 2017 110

ÍNDICE DE TABLAS

Tabla 1: Comparación de Bases de Datos, 2017 21
Tabla 2: Resumen de la Parte Interesada, 2017 43
Tabla 3: Oportunidad de Negocio, 2017 43
Tabla 4: Perfiles de Parte Interesada, 2017 45
Tabla 5: Responsable Funcional, 2017 45
Tabla 6: Perfiles de Usuario - Administrador del Sistema, 2017 46
Tabla 7: Perfiles de Usuario – Usuario del Sistema, 2017 46
Tabla 8: Necesidades Clave de la Parte Interesada, 2017 47
Tabla 9: Resumen de Capacidades, 2017 48
Tabla 10: Precedencia y Prioridad, 2017 51
Tabla 11: Requisitos - Atributos de Característica, 2017 53

INTRODUCCIÓN

Para un correcto funcionamiento, las organizaciones requieren almacenar y poseer información de sus clientes lo cual permita generar y llevar un control de sus actividades. Junto con las nuevas tecnologías relacionadas a la información y comunicación es posible el manejo adecuado de dicha información con el objetivo de contar con la misma de forma exacta y al instante. La información es considerada como un activo muy importante para las organizaciones, donde el disponer de la misma en el momento preciso llega a ser de gran ayuda. Gracias a su importancia varias entidades se ven en la necesidad de obtener sistemas que permitan facilitar la Gestión de Información (GI).

FUDRINE es una fundación que busca promover la plena inclusión y el mejoramiento de la calidad de vida de los niños y niñas con Parálisis Cerebral y Síndrome de Down y sus familias. En la actualidad la fundación gestiona la información mediante un proceso físico que no emplea un correcto manejo de la misma, ocasionando falta de organización y varios inconvenientes a la fundación.

En el presente proyecto de disertación se desarrollará un sistema que permita a la fundación FUDRINE gestionar información relacionada a sus clientes (pacientes y estudiantes), por medio de la cual sea posible llevar un control y disposición total de dicha información.

El desarrollo del software se llevará a cabo mediante ciertas técnicas y principios con suficientes detalles como para permitir su interpretación y realización. Ambos procesos serán documentados con la ayuda de distintas formas y herramientas.

El presente proyecto se encuentra estructurado por un conjunto de 6 capítulos brevemente resumidos a continuación.

El capítulo uno presenta la fundación partícipe en la cual se desea desarrollar el proyecto. Recolectando un conjunto de datos relevantes que nos permite conocer punto por punto el inicio, desarrollo y situación actual de la fundación.

El capítulo dos adjunta un grupo de fundamentos teóricos de gran importancia que han sido utilizados de soporte para el desarrollo del proyecto. Aquí se mostrará a detalle las herramientas y metodologías seleccionadas para la elaboración del mismo.

El capítulo tres reúne un grupo de datos recolectados en la fundación que permitirán realizar una planificación, guía y organización de la forma en la que se desarrollará nuestro proyecto. Se presentan los requerimientos de parte del usuario y las funcionalidades que el sistema tendrá en base a los requisitos presentados por la fundación.

El capítulo cuatro recolecta un conjunto de diagramas y diseños clave elaborados con el fin de tener una visión certera de lo que se planea construir. Este capítulo permite al desarrollador de software obtener un gran avance en la logística del mismo.

El capítulo número cinco refleja la construcción del sistema, adjuntando información adicional que será de importancia para finalizar el proyecto correctamente. Es aquí donde las pruebas inician, observamos que el proceso cumpla con lo definido y sea aceptado en su totalidad.

El sexto y último capítulo presenta las conclusiones y recomendaciones obtenidas en relación a los resultados obtenidos durante y después del desarrollo de la presente disertación.

1. CAPÍTULO 1: FUDRINE

En este capítulo se presenta una descripción amplia de la fundación. Conoceremos un poco de su historia, situación actual (misión, visión, valores, principios, razón de ser), a su vez encontramos su método de gestión de información actual y necesidades adicionales que presenta la fundación FUDRINE en la actualidad.

1.1. HISTORIA

FUDRINE es una fundación encargada del diagnóstico, rehabilitación e integración de niños y jóvenes con discapacidades mentales y físicas. La fundación es una entidad privada sin fines de lucro. Inició hace aproximadamente 20 años, exactamente desde el día 25 de julio de 1996 cuando la fundación fue legalmente aprobada y creada mediante el Acuerdo Ministerial No. 1109.

La historia inicia con cuatro muy buenas amigas: Silvia de Grijalva, Mónica Salgado, Liliam Acosta y Alina Navarro; las cuales fueron unidas gracias a un mismo objetivo. Ellas buscaban tratar, atender y ayudar de la mejor manera, con mucha dedicación, amor, respeto, solidaridad, transparencia, honestidad y unión a niños y jóvenes con discapacidades motrices severas, síndrome de Down y necesidades educativas.

FUDRINE se financia a través de autogestión, donaciones y del pago diferenciado de los servicios que ofrece según la evaluación de la capacidad económica familiar. De este modo FUDRINE se ha puesto al servicio de niños y jóvenes de altos, medianos y bajos recursos con una atención individual, integral y especializada. De esta forma se busca la inclusión de los niños y jóvenes potenciando sus capacidades y preparándolos para la vida adulta.

La fundación cuenta con el apoyo técnico permanente de un grupo de médicos franceses tanto de la Asociación María José Solidaridad Hándicap Francia – Ecuador y de la Asociación Notre Dame en Neuilly - Seine, París con gran experiencia. Suelen visitar la fundación al menos dos veces al año. Sus principales aportaciones dentro de la fundación son: evaluar permanentemente el estado de niños y jóvenes, brindar

conocimiento y capacitación profesional al personal de la fundación y el apoyo con material terapéutico. [12]

1.2. SITUACIÓN ACTUAL

La fundación FUDRINE basa su funcionamiento en los siguientes elementos:

1.2.1. MISIÓN

Promover la plena inclusión y el mejoramiento de la calidad de vida de los niños y niñas con Parálisis Cerebral (IMC) y Síndrome de Down y sus familias, mediante la atención integral y especializada, de calidad y calidez.

1.2.2. VISIÓN

Ser una organización autosustentable que ofrece un servicio terapéutico entregado con excelencia, compromiso, vocación de servicio y de capacitación integral.

1.2.3. VALORES

Los valores son aquellos que proveen una base estable sobre la cual se toman decisiones y se ejecutan las acciones. La fundación FUDRINE tienen definidos los siguientes valores que son de guía para su desempeño diario.

- **Excelencia:** Eficiencia y eficacia, profesionalismo, servicio de calidad, ayuda especializada, responsabilidad.
- **Compromiso:** Dedicación, entrega, perseverancia
- **Vocación de Servicio:** Amor, pasión, entusiasmo, alegría, carisma, amabilidad, respeto, unidad, seguridad, tranquilidad, confianza, familia, hogar, amistad.

[1] (FUDRINE, 2017)
[2] (SALVATIERRA, 2014)

1.2.4. PRINCIPIOS

Los principios que orientan y determinan como los miembros de la fundación perciben e interpretan los problemas son los siguientes:

- Solidaridad
- Equidad
- Respeto
- Honradez
- Responsabilidad
- Compromiso

1.2.5. RAZÓN DE SER

La razón de ser son los motivos trascendentes de FUDRINE, es decir, las razones por las cuales la organización hace lo que hace, de la manera que lo hace y no hace otra cosa. La razón de ser de la fundación es: “Somos inversionistas de la Inclusión”.

1.2.6. MÉTODOS DE GESTIÓN DE INFORMACIÓN

Como en toda organización la gestión de información es un factor clave para llevar un proceso o ciclo de vida organizado de nuestros datos. La fundación FUDRINE mantiene un ambiente de datos poco organizado dado que los métodos de regulación y ordenamiento de información son anticuados. La fundación almacena sus datos en varias carpetas de portafolio. Estas carpetas se encuentran apiladas entre ellas. Para adquirir cualquier dato o información de un chico o chica es necesario revisar entre todas estas carpetas. A continuación se presentan algunos de los inconvenientes más frecuentes que tiene la fundación en su proceso o método actual que utiliza para gestionar su información:

- Datos e información duplicada.
- Pérdida de información.

- Dificultad a la hora de adquirir información específica.
- Problemas para realizar reportes o estadísticas.
- Falta de copias de seguridad por cualquier posible suceso.
- Dificultad a la hora de compartir datos.

Como podemos observar el método empleado por la fundación genera varias desventajas hacia la misma. Varios de estos inconvenientes afectan a la evolución, organización y control de los datos que maneja la fundación.

El objetivo del presente proyecto es construir una aplicación que permita a la fundación eliminar todos estos inconvenientes o desventajas que presenta al manejar una forma defectuosa de su gestión de información. Se considera necesario mantener un ambiente o entorno organizado para lograr obtener resultados coherentes y efectivos. Del mismo modo esto facilitará el proceso de culminación hacia objetivos previamente planteados, gracias al conocimiento que nos proporciona al saber cómo y dónde nos encontramos.

1.2.7. NECESIDADES

En la actualidad la fundación FUDRINE presenta varias necesidades tanto materiales como tecnológicas. Estas son algunas necesidades primordiales para la evolución como organización. A continuación se presentan las mismas:

- Gestión de información de forma automatizada mediante un sistema computacional.
- Innovación en sistemas que faciliten la comunicación entre el chico o chica con una discapacidad mental o física y el personal encargado de la fundación.
- Artefactos o elementos innovadores que faciliten el control, enseñanza y rehabilitación de los chicos y chicas.
- Equipos tecnológicos como son: 3 computadores, 1 laptop, 1 proyector, software educativos.
- Material didáctico específico para trabajo con niños con dificultad motriz.

1.3. MAPA DE PROCESOS

El mapa de procesos proporciona una perspectiva global de cada proceso respecto a la cadena de valor de la organización. A continuación se presenta el mapa de procesos de la fundación FUDRINE.

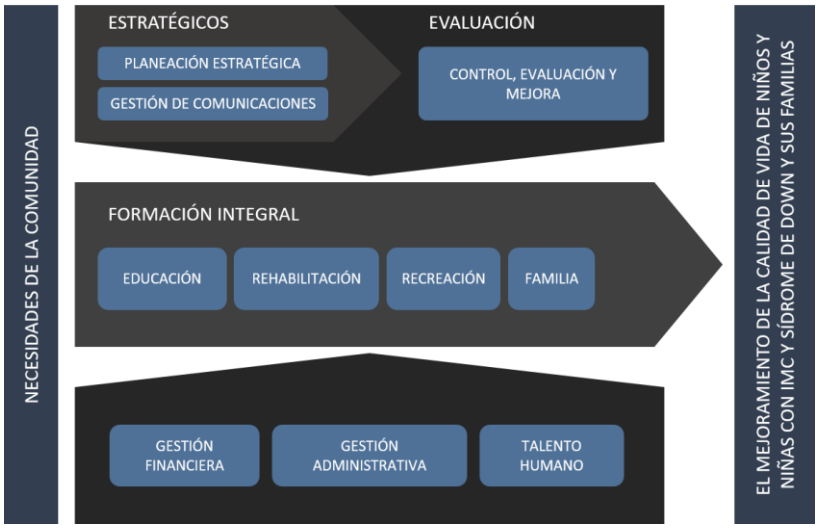

Ilustración 1: Mapa de procesos de la fundación FUDRINE, 2017

1.4. MAPA JERÁRQUICO

Un mapa jerárquico ilustra la estructura organizativa de la fundación, donde cada entidad en la organización, excepto una, está subordinada a una entidad única.

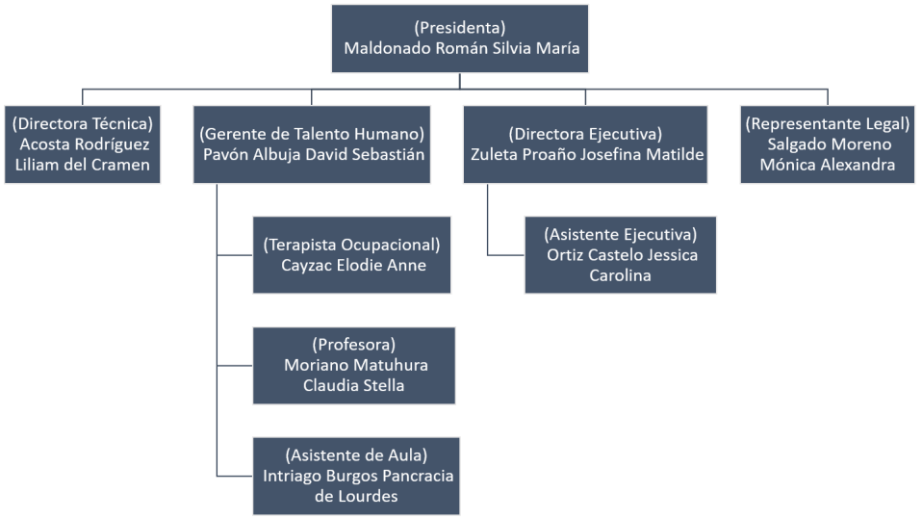

Ilustración 2: Mapa jerárquico de la fundación FUDRINE, 2017

2. FUNDAMENTOS TEÓRICOS

En el capítulo número dos se expone de forma lógica y detallada las ideas que están relacionadas con el tema en particular. Este es el encargado de ser una guía desde el inicio hasta la conclusión del proyecto.

2.1. GESTIÓN DE INFORMACIÓN

Durante miles de años la humanidad se ha encargado de almacenar información que ha sido primordial a la hora de nuestra evolución gracias al conocimiento que nos ha brindado la misma. Por este motivo, ha sido extremadamente necesaria para el desarrollo de la humanidad.

La gestión de la información incluye a varios factores y procedimientos que pueden ser de gran ayuda en todos los campos. Factores como la selección de la información que se precisa, la recolección y análisis de dicha información, el registro y posibilidad de obtener la misma en el momento necesario, el utilizarla y la acción de divulgarla o publicarla. Gracias a estas características que incluye la gestión de información es posible que existan grandes innovaciones necesarias de una forma rápida y oportuna a la hora de conseguir información precisa en el momento que la misma sea demandada.

Podemos encontrar distintas definiciones de la gestión de información, por ejemplo: Lynda Woodman la define como: “Todo lo que se refiere a conseguir la información adecuada, en la forma adecuada, para la persona adecuada, al costo adecuado, en el momento adecuado, en el lugar adecuado, para tomar la acción adecuada”, por otro lado Elizabeth Adams la define como: “Una función de alta dirección para desarrollar una serie de políticas, programas y procedimientos para planificar, gestionar y controlar eficaz y efectivamente las necesidades de información y los recursos de soporte del manejo de la información”.

En un marco general orientando la gestión de la información a organizaciones es posible detallar estos 3 componentes fundamentales en la misma:

- Lograr identificar, valorar y usar tanto los recursos informativos internos como externos.
- Poseer medios necesarios para la recolección, almacenamiento, recuperación y distribución de la información.
- Dar un uso correcto acertando en la mejor forma de gestionar la misma.

El objetivo principal de la gestión de la información se encuentra asociado con la integración, disponibilidad y confidencialidad de la misma.[34]

2.1.1. APLICACIÓN DE LA GESTIÓN DE INFORMACIÓN EN LA FUNDACIÓN FUDRINE

En el caso de la fundación FUDRINE, la gestión de la información sería completamente de gran ayuda para la recolección de datos importantes de cada uno de los chicos y chicas que reciben su servicio, siendo posible llevar un registro y control completo de cada uno de ellos. Con la ayuda de estos registros es posible obtener nuevo conocimiento, estadísticas, organización, avance e innovación en el proceso con el cual maneja la fundación su actividad diaria.

Por los motivos mencionados se requiere una aplicación computacional que permita brindar este proceso de control del ciclo de vida de la información. La aplicación dispondrá de la capacidad de capturar datos esenciales para la fundación, que mediante el tiempo serán de gran ayuda para lograr realizar un análisis y observación de la evolución de cada chico y chica gracias al perfil generado por la misma. Con esta ayuda será posible para los encargados de la fundación el diagnosticar, conocer y aprender de cada uno de sus estudiantes y pacientes.

[3] (Arévalo, 2007)
[4] (Fernández Valdés & Ponjuán Dante, 2017)

Varias estadísticas, cifras y documentación de gran importancia serán posibles obtener como resultado con la ayuda de la aplicación.

2.2. SELECCIÓN DE LA PLATAFORMA

Para la selección de la plataforma se han tomado en cuenta varios aspectos como son: la base de datos que se utilizará para el registro de información que requiera la fundación FUDRINE, la arquitectura que se desea manejar en la aplicación y el lenguaje de programación que sea compatible con estas características. A continuación se detalla de mejor forma cada una de estas características.

2.2.1. BASE DE DATOS

Para generar una aplicación la cual tenga como objetivo almacenar información de forma organizada es necesario hacer uso de una base de datos. Estas ofrecen un gran rango de soluciones a problemas de almacenamiento de información. Existen varios Sistemas Gestores de Base de Datos (SGBD) que nos permiten registrar y acceder de forma rápida, eficaz y estructurada a nuestra información.[5]

Gracias a varias características de las bases de datos, es posible obtener una gran cantidad de ventajas. Por ejemplo:

- Independencia entre los datos y los programas o procesos.
- Menor cantidad de redundancia en los datos.
- Obtener mayor valor informativo.
- Mayor integridad de los datos.
- Mayor seguridad de la información.
- Coherencia en los resultados.
- Reducción en el espacio de almacenamiento.
- Acceso organizado y eficaz hacia los datos.
- Mayor estandarización.

[5] (Codina, 2015)

Estas y muchas más ventajas nos puede proporcionar una base de datos.

Para la presente disertación se ha realizado un análisis de que gestor de base de datos sería el más conveniente para la fundación FUDRINE. En este análisis se considera un enfoque en búsqueda de distintas alternativas de sistemas de gestión de base de datos, realizando una comparación de sus características, ventajas y desventajas. Entre los sistemas de gestión de base de datos más compatibles y apropiados para el caso se obtuvo PostgreSQL y MySQL. Ambos son proyectos de base de datos de código abierto. A continuación presentamos una tabla breve que compara aspectos de importancia para el resultado del análisis:

	ORACLE	**MySQL**	**SQL Server**	**PostgreSQL**
Empresa	Oracle Corporation	Sun Microsystem	Microsoft	PostgreSQL Global Development Group
Licencia	Privada	Libre a nivel de usuario	Privada	Libre
Ventajas	• Motor de base de datos más usado a nivel mundial. • Compatible con cualquier plataforma. • Soporta transacciones. • Estabilidad.	• Permite ser ejecutado en computadores de escasos recursos. • Sencillo al configurar e instalar. • Agrupación de transacciones. • Baja probabilidad de corromper datos. • Conectividad segura.	• Fácil de instalar y configurar. • Utiliza TransactSQL. • Permite administrar permisos a todo. Consultas jerárquicas.	• Conexión estable. • Permisos a nivel de columna. • Consultas complejas Integridad transaccional. • Control de concurrencia.

	ORACLE	MySQL	SQL Server	PostgreSQL
Desventajas	• Precio elevado del producto y de su licencia. • Una mala configuración convierte a Oracle potencialmente lento.	• No tiene soporte • No sincroniza datos con otras bases de datos réplicas.	• Requiere una enorme cantidad de memoria RAM.	• Requiere administradores capacitados. • Lento en comparación a MySQL o SQL Server.
Experiencia Personal	Media	Alta	Media	Baja

Tabla 1: Comparación de Bases de Datos, 2017

Los puntos que resaltaron para el proyecto fueron la búsqueda por un gestor de base de datos que no tenga costo, sea estable, seguro, no requiera de exagerados recursos de hardware, sea multiplataforma y en la cual exista una experiencia y trabajo previo en la misma.

En conclusión, se ha decidido utilizar el sistema de gestión de base de datos MySQL, las razones se detallan en el capítulo subsiguiente.

2.2.1.1. MYSQL

MySQL es un sistema de gestión de base de datos relacional que inicialmente fue desarrollada por MySQL AB (aproximadamente en la época de los 90s) para luego ser adquirida por Sun Microsystems, para a su vez ser comprada por Oracle Corporation. El lenguaje de programación empleado por MySQL es Structured Query Language (SQL) el cual fue desarrollado por la compañía IBM en 1981.

La selección de MySQL como gestor de base de datos frente a otras alternativas ha sido debido a las siguientes características:

- Dispone de una versión gratuita.
- Tiene un excelente rendimiento y una mayor velocidad en conexión.
- No existe límite en el tamaño de sus registros.
- Excelente control de acceso.

- Requiere de bajos recursos tanto del CPU como de la memoria en comparación a otros sistemas de gestión de base de datos.
- Fácil de personalizar, aproximación minimalista.
- Considerada como la base de datos de código abierto más popular del mundo, teniendo una gran cantidad de librerías y herramientas que permiten su uso a través de una gran cantidad de lenguajes de programación.
- Es fácil de instalar y configurar.
- Aprovecha la potencia de sistemas multiproceso dado que posee una implementación multihilo.
- Multiplataforma.
- Posibilidad de acceso desde una gran variedad de lenguajes de programación como: C, C++, C#, Pascal, Delphi, Eiffel, Smaltalk, java, Lisp, Perl, PHP, Python, Ruby, Gambas, REALbasic, FreeBASIC y Tcl.[6]

Estas son características que contribuyen grandes ventajas tanto para la aplicación como para la fundación FUDRINE.

2.2.2. LENGUAJE DE PROGRAMACIÓN

Dentro del campo de la programación es posible encontrar una gran variedad de lenguajes. Estos lenguajes formales son aquellos que nos permiten realizar distintos procesos en computadoras.

Para el presente proyecto se realizó un análisis en base a los requerimientos de la fundación, el desempeño y campo en el que funcionará la aplicación para conocer que lenguaje de programación sería el indicado para el desarrollo de la misma. El análisis inició con una elección de lenguajes de programación de los cuales el programador se encuentre familiarizado, evitando que exista un proceso de

[6] (MySQL, 2017)

aprendizaje del lenguaje lo cual retrasaría el desarrollo del producto. Los lenguajes de programación con los que el programador se encuentra familiarizado son:

- Java
- C#
- PHP
- JavaScript
- C++
- LUA

De la lista señalada se descartó los lenguajes de programación orientados hacia el desarrollo de una aplicación web, en nuestro caso no requerimos de estas herramientas dado que se planea implantar el sistema en un solo sitio, con lo cual una aplicación de escritorio podrá encontrarse siempre funcionado sin la necesidad de una conexión a Internet, teniendo una mayor velocidad en el funcionamiento de la misma y en el manejo de datos. Es así como de la lista de lenguajes de programación inicial llegamos a la lista que se presenta a continuación:

- Java
- C#
- C++
- LUA

Para continuar con el análisis se consideró la importancia de que la aplicación tenga una interfaz amigable con el usuario dado que este ha sido un requerimiento solicitado por la fundación, es por este motivo que entre la lista de lenguajes de programación se seleccionó aquellos que faciliten el desarrollo de una interfaz amigable. Como resultado se obtienen únicamente dos lenguajes de programación:

- Java
- C#

Al final del análisis se mantuvo a flote la importancia de que la aplicación tenga compatibilidad con varios sistemas operativos. En este caso C# permite crear aplicaciones de escritorio que son compatibles con Windows a diferencia de Java el cual mantiene compatibilidad con distintos sistemas operativos gracias a su máquina virtual.

Luego de analizar estos puntos se decide que Java es el seleccionado como lenguaje para el código fuente del programa. A continuación se presenta información relacionada a dicho lenguaje.

2.2.2.1. JAVA

Java es un lenguaje de programación de propósito general, orientado a objetos, a su vez es una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. [7]

Java dispone de un gran grupo de cualidades, características y ventajas frente a otros lenguajes de programación.

- **Universalidad:** los bytecodes hacen de Java el lenguaje ideal para el desarrollo de aplicaciones para Internet. Del mismo modo es posible referirnos a Java como un lenguaje adaptable o independiente de plataforma, dado que para ejecutar un programa únicamente debemos compilarlo una vez ya que puede funcionar sobre cualquier máquina que tenga implementado un intérprete de Java.

  Otro factor clave son la gran cantidad de bibliotecas estándar de funciones y métodos de Java (definidas en su API, Application Programming Interface). Estas y muchas más bibliotecas facilitan la programación para un grupo de acciones complejas.
- **Sencillez:** es un lenguajes muy fácil de aprender y mucho más legible que otros como C o C++.

  En Java el programador no debe preocuparse de muchos detalles que pueden tomar varias horas en programación, este es el caso de los punteros. La ausencia de los mismos evita que haya un comportamiento imprevisto del mismo, siendo esto una de las principales fuentes de errores en la ejecución de un programa. Del mismo modo no es necesario preocuparnos minuciosamente por el uso de memoria, Java nos ayuda a optimizar el uso y manejo de recursos gracias a su característica propia

[7] (Java, 2017)

conocida como "Garbage Collector" (reciclador dinámico de memoria). Este es el proceso en el cual se liberan los objetos no utilizados, siendo posible reutilizar porciones del almacenamiento dinámico de la máquina virtual Java (JVM).

- **Orientado a objetos:** Java implementa la tecnología básica de lenguajes orientados a objetos con varias mejoras, a su vez descarta ciertos elementos para mantener la simplicidad del lenguaje. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo.
- **Es robusto:** Java realiza pruebas y revisiones examinando problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java (clases o interfaces) ayuda a detectar errores, lo antes posible en el ciclo de desarrollo.

  Java proporciona: comprobación de punteros, comprobación de límites de arrays, excepciones y verificación de bytecodes.
- **Es seguro:** en el lenguaje, características como los punteros o el casting implícito que hacen los compiladores de los lenguajes tradicionales se eliminan para prevenir el acceso ilegal a la memoria. Java no posee una semántica específica para modificar la pila de programa, la memoria libre o utilizar los objetos y métodos de un programa sin los privilegios del Kernel del sistema operativo. Java elimina cualquier posible amenaza de forma instantánea al suprimir por completo los apuntadores de lenguaje.
- **Es Multihilo:** Java permite que se realicen varias actividades simultáneas en un mismo programa. [8]

2.2.3. ARQUITECTURA DE SOFTWARE

Para el presente proyecto se ha decidido llevar a cabo la construcción de nuestro software mediante un conjunto de patrones los cuales sean capaces de suministrar un marco de referencia que nos facilite la construcción de dicha aplicación. De esta

[8] (Castillo, 2010)

manera es posible mantener una misma forma de trabajo, cubriendo los objetivos y restricciones planteados para el programa.

Dentro del desarrollo de software es posible encontrar distintos componentes principales. La arquitectura de software señala los siguientes: clientes y servidores, base de datos, hilos, nivel en sistema jerárquico. Entre estos componentes es posible encontrar interacciones como la llamada de procedimientos, comportamiento de variables, protocolos cliente servidor y transmisión de eventos. [9]

Si bien es posible encontrar distintos tipos de arquitectura de software para nuestra aplicación se ha decidido utilizar una arquitectura en capas. Este tipo de arquitectura ha sido seleccionado enfocándonos en el modelo y funcionamiento del software.

La aplicación tendrá una base de datos para guardarlos, una estructura encargada de procesar dichos datos y gestionar lo que se realice con los mismos y el acompañamiento de una interfaz de usuario con la cual es posible interactuar. La arquitectura en tres capas se encarga de separar nuestro sistema en tres partes diferenciadas, de esta forma cada capa solo se comunica con la adyacente.

2.2.3.1. ARQUITECTURA EN TRES CAPAS

La arquitectura de tres capas es una especialización de arquitectura cliente-servidor en la cual podemos encontrar la carga dividida en tres partes diferenciadas. Cada parte tiene su función definida. Es posible encontrar un diseño mayormente utilizado en la actualidad relacionado a las tres capas.

Ilustración 3: Interacción de un programa con arquitectura de tres capas, 2017

[9] (EcuRed, 2017)

De acuerdo a la Ilustración 3, a continuación se detalla cada capa junto con su función principal:

- **Capa de interfaz de usuario:** el objetivo principal de esta capa es interactuar con el usuario. Esta capa será la encargada de transferir las acciones realizadas por el usuario a la capa de negocio.
- **Capa de negocio:** en esta capa encontraremos la lógica principal de la aplicación, es decir aquí el programa reconoce que función realiza con los datos. Se encuentra formada por entidades representadas como objetos. Se encuentra conectada con la capa de acceso a datos.
- **Capa de acceso a datos:** esta capa es la encargada de gestionar los datos, es decir se encarga de su creación, edición y borrado de los mismos. La capa contiene clases encargadas de interactuar con los datos que se encuentran alojados en nuestra base de datos.

Esta forma de distribución de capas nos permite mantener un entorno organizado, de esta manera es completamente sencillo realizar cualquier posible cambio dentro de nuestro programa.

2.2.4. PLATAFORMA

Como se mencionó anteriormente la aplicación se encuentra diseñada en el lenguaje de programación Java. Para su funcionamiento es necesario el uso de una plataforma Java. La plataforma Java es un entorno computacional creado por la empresa informática Sun Microsystems. Es una máquina virtual la cual se encuentra a cargo de ejecutar aplicaciones y un gran grupo de bibliotecas estándar que permiten una funcionalidad común. Consta de tres componentes principales, estos son: el lenguaje Java, los paquetes Java y la máquina virtual Java.

La principal ventaja de esta plataforma es que ejecuta código que se obtiene de la compilación del código fuente (bytecode). La máquina virtual de Java o mejor conocida como JVM (Java Virtual Machine) es la encargada de interpretar el bytecode en instrucciones nativas de la plataforma destino. Es por este motivo que

una sola aplicación puede funcionar en más de un sistema operativo, convirtiéndose ésta en una aplicación multiplataforma.

Al conjunto de la máquina virtual de Java, junto con las bibliotecas y paquetes del mismo se lo conoce como JRE (Java Runtime Environment). Este es el requisito mínimo que necesitará un computador para permitir que la aplicación funcione de forma adecuada en un computador.

Gracias a esta plataforma la fundación FUDRINE tendrá la posibilidad de ocupar la aplicación sin importar el sistema operativo que utilice, ya sea Windows, MacOS, Linux o Solaris.[10]

2.3. TIPO DE APLICACIÓN

Para realizar nuestro presente proyecto de forma correcta es indispensable el conocer hacia qué tipo de software y a su vez de programa está orientada la misma. La clasificación de software puede ser realmente extensa sin embargo es posible dividirla en 3 grandes grupos:

- **Software de sistema:** este tipo de software se encarga fundamentalmente de mantener una interacción directa con el hardware de nuestro computador, administrando los recursos y presentando una interfaz al usuario. Este es el caso de los sistemas operativos.
- **Software de programación:** este software nos permite desarrollar nuestras propias aplicaciones informáticas mediante conocimientos lógicos y lenguajes de programación. Este es el caso de editores de texto, compiladores, intérpretes, entornos de desarrollo integrados (IDEs), entre otros.
- **Software de aplicación:** estos son programas los cuales nos permiten realizar actividades o procesos específicos en nuestro sistema.
  Es posible clasificar este tipo de software en varios tipos de programas como por ejemplo:
    - Aplicaciones educativas

[10] (Oracle, 2017)

- Aplicaciones médicas
- Aplicaciones de cálculo numérico
- Aplicaciones de diseño asistido
- Aplicaciones de control de procesos
- Aplicaciones de gestión de información

Entre otras.

Nuestra aplicación se encontrará dentro de la categoría de software de aplicación dado que tiene como objetivo una utilidad específica que es el facilitar el proceso o ciclo de vida de nuestra información, permitiendo un correcto y organizado método de gestión de la información en la fundación FUDRINE.

2.4. METODOLOGÍA RUP

Para este proyecto se ha buscado una metodología de software que sea compatible con el tamaño y tipo de software, el equipo que trabajará en la construcción de la aplicación (mi persona), la relación con el cliente y el tiempo requerido para construir el programa.

Una metodología de software es un marco referencial de trabajo que se utiliza con la ayuda de herramientas, modelos y métodos para poder administrar un proyecto con altas probabilidades de éxito mediante la estructuración, planificación y control del proceso.

Luego de un análisis de las distintas metodologías se ha decidido que la metodología del Proceso Racional Unificado (RUP) es una gran alternativa para nuestro proyecto.

La metodología RUP es un proceso de desarrollo de software desarrollado y mantenido por Rational® Software, una compañía de IBM. El Proceso Racional Unificado es una guía para conocer cómo utilizar de forma eficaz el Lenguaje Unificado de Modelado (UML). El UML es un lenguaje que nos permite comunicar claramente los requisitos, arquitectura y diseños.

La metodología se apoya de varias herramientas que sirven para automatizar el proceso de modelado, programación, pruebas, entre otros. Del mismo modo se lo conoce como

un proceso configurable dado que puede ser utilizado tanto para pequeños como para grandes proyectos.[11]

2.4.1. PRINCIPIOS

El Proceso Racional Unificado (RUP) consta de 6 principios básicos, a estos principios se los conoce como mejores prácticas. Son denominadas así ya que se encuentran en las organizaciones con mayor éxito. A continuación se detallan cada una de estas prácticas o principios:

- **Desarrollo de software y demostración de valor de forma iterativa:** aunque el resultado final es uno solo, durante el proceso existen distintas etapas iterativas. Se requiere un enfoque iterativo el cual permita entender el problema a resolver con la ayuda de correcciones y mejoras realizadas incrementalmente, logrando así una solución efectiva luego de una serie de procesos o iteraciones. El RUP apoya estas iteraciones principalmente para los elementos que puedan presentar mayor riesgo durante la etapa de ciclo de vida. De este modo es posible reducir considerablemente el nivel de riesgo o fracaso para nuestro proyecto.

  Un factor clave dentro del proceso es que el mismo presenta en cada iteración un avance del proyecto, lo cual tiene como objetivo el desarrollo de éste, presentando frecuentemente resultados y evitando que sea desechado.

  Otro gran beneficio es la recolección de requerimientos, durante el proceso de desarrollo el cliente suele decidir realizar pequeños cambios y modificaciones en el proyecto, en cada iteración es posible modelar el sistema hasta obtener los resultados esperados.
- **Administrar requerimientos:** es muy importante tener en cuenta los requisitos de cada uno de los participantes del proyecto. En varios casos existen distintas contradicciones o disputas por recursos, es por este motivo que lo mejor es encontrar un punto de equilibrio que satisfaga las necesidades de todos, principalmente entre estabilidad y calidad. Para poder

[11] (IBM, 2017)

producir, documentar y organizar requisitos funcionales y limitaciones, RUP nos presenta una guía y descripción de este proceso, facilitando la captura y comunicación de los mismos. Para la metodología RUP, se considera los Casos de Uso una excelente forma de capturar requisitos funcionales, siendo una base sólida para continuar con el diseño, implementación y pruebas de software. Mediante este método el sistema final cumplirá con las necesidades del usuario de una forma coherente.

- **Uso de arquitecturas basadas en componentes:** la metodología RUP estipula que para un correcto desarrollo del proyecto lo mejor es dividir el software en componentes, módulos o subsistemas que cumplen una función clara. Para nuestro caso podemos observar la compatibilidad que existe entre la metodología seleccionada con la arquitectura en tres capas. Esta característica es muy útil para el proceso de desarrollo de software dado que permite que el sistema vaya construyéndose a medida que se crean sus componentes, siendo fácilmente modificable. Es por esto que el proceso busca crear una línea base sólida con la ayuda de una buena arquitectura. Se espera que la arquitectura sea flexible, resistente, apta para cambios y que promueva la reutilización de código.
- **Modelado visual:** este proceso se encarga de utilizar la abstracción ocultando detalles, capturando la estructura y comportamiento de arquitecturas y componentes usando distintos modelos gráficos. Estas abstracciones visuales permiten ver como un elemento del sistema interactúa con otro, junto con su comportamiento**.** De esta forma es posible ver que el código se encuentre acorde a los componentes y de la forma más coherente entre el diseño y la aplicación evitando así las ambigüedades.

  Unified Modeling Language (UML) es un lenguaje de Rational® Software que nos permite visualizar, especificar, construir y documentar software. El uso de esta herramienta facilita la gestión de modelos, mejorando la capacidad del desarrollador para manejar la complejidad del software.
- **Verificar la calidad del software:** dentro de la creación de todo producto existe un factor clave que todo programador debería tener en cuenta durante la creación del mismo, la calidad. La calidad es un factor que debe ser revisado constantemente, estando siempre enfocada en la fiabilidad,

funcionamiento, rendimiento de las aplicaciones y sistema. El RUP es una metodología que ayuda a mantener un control durante la planificación, diseño, implementación, ejecución y evaluación de este tipo de pruebas.

- **Control de cambios del software:** es importante tener en cuenta que siempre existirán cambios dentro de un desarrollo de software. La capacidad de gestionar dichos cambios asegurando que cada cambio es aceptable, controlando y siguiendo cada uno de éstos en cada iteración.

  Los cambios son factores que pueden llegar a tener un alto riesgo para la evolución correcta de un proyecto de software. Los cambios se dan tanto en el desarrollo por cambio de requisitos como en el final del software por posible mantenimiento.

  RUP nos proporciona una guía de cómo establecer espacios de trabajo seguros para los desarrolladores, controlando los posibles cambios en el artefacto de software.[12]

2.4.2. CICLO DE VIDA

Toda metodología consta de un ciclo de vida para el desarrollo de software. El ciclo de vida es una estructura que define las distintas fases intermedias que son necesarias para realizar una validación y supervisión adecuada del desarrollo de la aplicación, de esta forma es posible asegurar que el software cumple con requisitos y se asegura de que los métodos utilizados son apropiados.

Uno de sus principales objetivos se enfoca en disminuir la cantidad de posibles errores que se detecten, de esta forma es posible ahorrar todo tipo de recursos. Algunas de las actividades más comunes que encontramos dentro de todo proceso de desarrollo de software son: planificación, implementación, pruebas, documentación, despliegue y mantenimiento. Todas estas actividades son primordiales para obtener resultados coherentes y de calidad.

[12] (Martínez, 2017)

Si bien existen varios modelos de ciclo de vida con distintas fases y enfoques el Proceso Racional Unificado (RUP) utiliza un modelo de desarrollo de software tipo espiral.

2.4.2.1. MODELO DE ESPIRAL

El modelo espiral de ciclo de vida de software se esfuerza en el desarrollo iterativo y evolutivo. Su creador es Barry Bohem, en el año 1986 el autor ideó y promulgó un modelo distinto al tradicional el cual tiene como principal objetivo tener presente de manera primordial los riesgos que aparecen durante el proceso de desarrollo de software. Utiliza una combinación y aspectos clave del modelo en cascada y del Desarrollo Rápido de Aplicaciones (RAD).

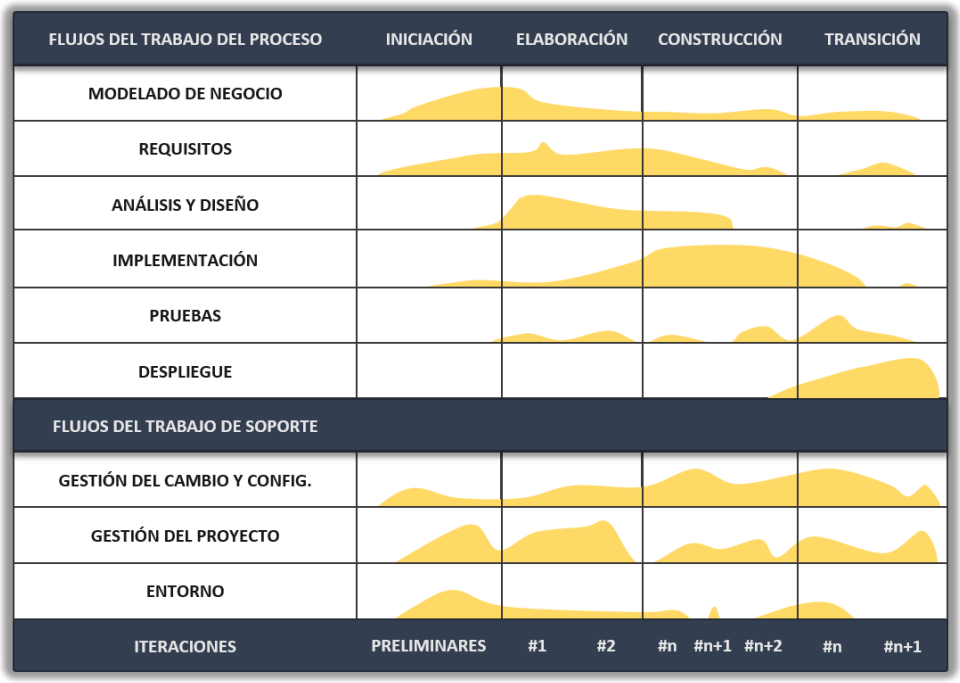

Ilustración 4: Ciclo de Vida - Modelo en Espiral, 2017

El modelo consta de cuatro fases principales en las cuales se desarrollan distintos pasos. Las fases conforme la Ilustración 4, son:

- Iniciación
- Elaboración

- Construcción
- Transición

Las actividades que se realizan durante el tiempo de ejecución o flujos de trabajo del proceso de dichas fase son similares a las que se utilizan en un modelo tipo cascada. Estas son:

- Modelado del negocio
- Requerimientos
- Análisis y diseño
- Implementación
- Pruebas
- Despliegue

Del mismo modo existen flujos de trabajo de soporte durante estas fases. Estas son:

- Gestión de cambio y configuraciones
- Gestión del proyecto
- Entorno

Más adelante se presenta de forma detallada cada una de estas fases y flujos de trabajo del modelo en espiral empleado en la metodología RUP.[13]

2.4.3. CARACTERÍSTICAS

La metodología RUP presenta varias características, entre las más importantes y representativas encontramos las siguientes:

- Maneja un desarrollo iterativo e incremental.
- Se encuentra orientado al manejo de casos de uso.
- Permite determinar y dividir responsabilidades y tareas de una forma sumisa y disciplinada.
    - ¿Quién realiza una acción?

[13] (Martínez, 2017)

  - ¿Cuándo se lleva a cabo la misma?
  - ¿Cómo se implementa dicha acción?
- Es un proceso configurable, es decir satisface las necesidades específicas del proyecto, incluyendo todos sus posibles modificaciones. Maneja un control de cambios de calidad.
- Lleva consigo una gestión y atención frecuente con el usuario a la hora de obtener los requisitos que dicho usuario presenta.
- Se orienta hacia un uso de una arquitectura basada en componentes con el objetivo de que el software sea fácil de complementar, quitar o realizar cualquier posible cambio. De esta forma se logra reducir el riesgo del proyecto.
- Maneja un modelado visual de software coherente y sencillo de comprender para el programador. De esta manera se simplifica la forma de entender el funcionamiento que debe tener la aplicación tanto para el programador como para el usuario final. El modelo puede ser sencillo y abstracto hasta ingresar a los detalles más profundos del funcionamiento esperado.
- Se maneja un constante análisis y dirección hacia la calidad de software, buscando siempre concordancia de los requerimientos funcionales y de rendimiento con los estándares de desarrollo y las características implícitas esperadas. Para cumplir con un control de calidad de software es importante tener presentes los siguientes elementos:
  - Herramientas y métodos para el análisis, diseño, codificación y pruebas del sistema.
  - Revisión constante aplicada en cada fase e iteración del proceso de ingeniería de software.
  - Utilizar un minucioso control de la documentación del software y de cada cambio realizado en el mismo.
  - Llevar un método que nos permita y asegure un ajuste de los estándares de desarrollo previamente determinados.[14]

[14] (Martínez, 2017)

2.4.4. FASES

El ciclo de vida del RUP está comprendido en cuatro fases, donde cada una de estas finaliza con un hito bien definido. Cada una de las fases requiere de recursos y tiempo para manejar dichos recursos. En la Ilustración 5 es posible visualizar de manera aproximada el tiempo de duración de cada fase junto con la cantidad de recursos que requiere la misma.

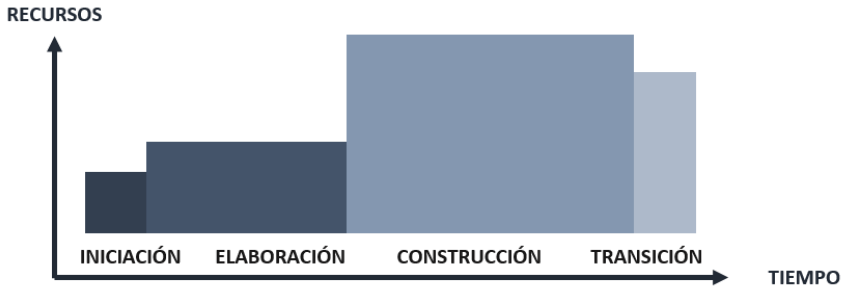

Ilustración 5: Fases e iteraciones, 2016

2.4.4.1. FASE DE INICIO

El objetivo principal de la fase inicial es lograr identificar las entidades externas con las cuales el sistema interactuará. A su vez implica el identificar casos de uso y describir los más significativos.

En el caso de negocio se incluyen distintos criterios como son el de éxito, evaluación de riesgo, estimación de recursos requeridos y un plan en el que se organiza cada uno de los hitos.

Entre los resultados de la fase de inicio se obtienen los siguientes resultados:

- Un documento de visión general donde encontramos los requisitos, características y limitaciones principales del proyecto.
- Modelo de casos de uso completado entre un 10% y 20%.
- Estimaciones de recursos (tiempo, costo, evaluación de riesgos).
- Pequeño prototipo.

Esta fase nos permite determinar si el proyecto tendrá los resultados esperados o será necesario trabajar en cambios o iniciar la cancelación del mismo.[15]

2.4.4.2. FASE DE ELABORACIÓN

La intención de esta fase es analizar el dominio del problema e implementar una base arquitectónica sólida donde se elabora un plan de proyecto y se eliminan todos los riesgos posibles. Las decisiones arquitectónicas deben realizarse comprendiendo por completo el sistema. Debemos tener en cuenta el alcance, funcionalidad principal y requisitos no funcionales.

En la fase de elaboración un prototipo se construye en una o más iteraciones dependiendo de su alcance, tamaño, riesgo y novedad del proyecto.

Dentro de la fase es necesario obtener los siguientes resultados:

- Modelo de casos desarrollado a un 80%.
- Descripción de una arquitectura de software. Creación del documento de arquitectura de software (SAD).
- Prototipo ejecutable de la arquitectura.
- Infraestructura de desarrollo (diagrama de clases, modelo entidad-relación, diagrama de secuencia, diagrama de estados, diagrama de colaboración, diseño de interfaces, desarrollo de plan de pruebas del sistema).

2.4.4.3. FASE DE CONSTRUCCIÓN

Durante la fase de construcción los componentes y las características de aplicación restantes se desarrollan e integran en el producto final, a su vez se realizan. En esta fase nos centramos en la gestión de recursos y el control de las operaciones para optimizar costos, plazos y calidad.

[15] (IBM, 2017)

Con la ayuda de una arquitectura robusta y un plan comprensible se lograrán obtener grandes resultados. El resultado de la fase de construcción debe ser un producto listo para el fácil uso de usuarios finales.

Las actividades a realizar en esta fase son:

- Desarrollo de componentes por completo.
- Optimización y control de procesos.
- Administración de los recursos.
- Integración del sistema.
- Verificación de versión del producto con la visión.

Artefactos necesarios:

- El sistema.
- Plan de publicación.
- Plan de pruebas.
- Material de soporte al usuario final (manual de uso).

La siguiente fase (transición) puede tener que ser aplazada en caso de no completar por completo este hito.[16]

2.4.4.4. FASE DE TRANSICIÓN

El objetivo para la fase es realizar la transición del producto de software hacia la comunidad (usuarios finales). Al entregar el producto al usuario final se suelen presentar ciertos conflictos e inconvenientes donde es necesario corregir algunos problemas o finalizar las funciones que se propusieron.

La fase incluye:

- Pruebas beta.
- Capacitación de usuarios.

[16] (IBM, 2017)

Esta fase incluye varias iteraciones con versiones beta, versiones generales de disponibilidad, así como correcciones de errores y mejoras. Se invierte considerablemente en el desarrollo de documentación orientada al usuario.

Sus metas son:

- Lograr la autosuficiencia del usuario.
- Alcanzar las expectativas de las partes interesadas.
- Lograr la línea base del producto final de manera rápida y rentable.

La fase de transición puede variar de muy simple a extremadamente compleja dependiendo del tipo de producto.[17]

[17] (IBM, 2017)

3. INICIACIÓN

En el capítulo número tres se inicia con el desarrollo de nuestro sistema gestor de información. Identificamos las entidades externas con las cuales el sistema interactuará, sus requisitos, funcionamiento, costo, características y estándares.

3.1. DOCUMENTO DE VISIÓN

El documento de visión establece los siguientes puntos:

3.1.1. INTRODUCCIÓN

En el presente documento se define el alcance, objetivo del proyecto, una declaración clara del problema y la propuesta de solución a la misma.[18]

3.1.1.1. OBJETIVO

El objetivo es adquirir, analizar y definir necesidades de alto nivel así como las características del sistema de gestión de información para la fundación FUDRINE. El documento se centra en la funcionalidad requerida por los participantes en el proyecto y los usuarios finales.

3.1.1.2. ALCANCE

El documento, como ya se ha mencionado, se ocupa del sistema de gestión de información para la fundación FUDRINE. El sistema permitirá a los administradores, profesores y terapeutas llevar un control total de la información de cada estudiante o paciente.[19]

[18] (IBM, 2017)
[19] (IBM, 2017)

3.1.1.3. DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS

El RUP (Proceso Racional Unificado) es un proceso de ingeniería de software que proporciona un enfoque disciplinado para la asignación de tareas y responsabilidades dentro del desarrollo de una organización. Su objetivo es asegurar la producción de software alta calidad que satisfaga las necesidades de sus usuarios finales, dentro de un horario predecible y presupuesto. (IBM, 2016)

3.1.1.4. REFERENCIAS

- Glosario
- Plan de desarrollo de software
- RUP (Proceso Racional Unificado)

3.1.2. POSICIONAMIENTO

En el presente documento se define el alcance, objetivo del proyecto, una declaración clara del problema y la propuesta de solución a la misma.

3.1.2.1. OPORTUNIDAD DE NEGOCIO

Este sistema permitirá a la fundación almacenar y tener acceso instantáneo a la información de cada uno de los pacientes y estudiantes de FUDRINE, lo cual supondrá un acceso rápido y sencillo a los datos, con la ayuda de interfaces gráficas sencillas y amigables.

3.1.2.2. DECLARACIÓN DEL PROBLEMA

El problema de llevar una administración de información de cada paciente y estudiante de forma indebida afecta directamente a los terapeutas, profesores, personal administrativo, estudiantes y pacientes de la fundación FUDRINE. El impacto asociado es mantener una gestión de información completamente digital, organizada, actualizada y que esté al instante accesible. Una solución adecuada sería informatizar el proceso, por medio de un sistema que permita conectar a toda la información que se encuentra en una base de datos, generando interfaces amigables y sencillas para todo usuario.

3.1.2.3. DECLARACIÓN DE POSICIÓN DE PRODUCTO

Para terapeutas, profesores, personal administrativo de la fundación FUDRINE que controlan y manejan constantemente información de cada estudiante y paciente. La herramienta de software que almacena y maneja la información necesaria para el control completo de información asociada a cada estudiante y paciente de la fundación. A diferencia del método anticuado que se lleva a cabo por medio de una gestión de información física, desordenada y ambigua, el producto contará con información accesible, organizada y confiable.

3.1.3. DESCRIPCIÓN DE LA PARTE INTERESADA Y DEL USUARIO

Para proporcionar productos y servicios que satisfagan las necesidades de partes interesadas y usuarios, es necesario identificar e implicar a todos las partes interesadas como parte del proceso de definición de requerimientos. También se debe identificar a los usuarios del sistema y asegurarse que en conjunto las partes interesadas los representa adecuadamente.

Esta sección ofrece un perfil de las partes interesadas y los usuarios que están implicados en el proyecto, también identifica los problemas clave que las partes interesadas y los usuarios consideran que la solución propuesta debe abordar. Esta sección no describe solicitudes ni requisitos específicos.

3.1.3.1. RESUMEN DE LA PARTE INTERESADA

Nombre	Descripción	Responsabilidad
Coordinador y responsable del proyecto	Responsable a nivel directivo y del área de sistemas del proyecto.	- Establecer los lineamientos generales para el desarrollo del proyecto. Coordinar a nivel directivo los distintos requerimientos que se presenten en el desarrollo del sistema. - Responsable del análisis y diseño del proyecto. Administra el correcto desarrollo del proyecto en cuanto a la construcción e implantación.
Responsable funcional	Representante de toda el área usuaria de la fundación FUDRINE.	Responsable de coordinar con los diferentes usuarios la correcta determinación de los requerimientos y la correcta concepción del sistema.

Tabla 2: Resumen de la Parte Interesada, 2017

3.1.3.2. OPORTUNIDAD DE NEGOCIO

Nombre	Descripción	Responsabilidad
Administrador del sistema	Persona de la fundación encargada de administrar el sistema.	Administrar funcionalmente el sistema (gestionar acceso a usuarios, dar mantenimiento al sistema frente a nuevos requerimientos).
Usuario del sistema	Persona de la fundación FUDRINE que hará uso del sistema.	Ingresar, editar y adquirir información de los pacientes y estudiantes de la fundación FUDRINE.

Tabla 3: Oportunidad de Negocio, 2017

3.1.3.3. RESUMEN DE LA PARTE INTERESADA

- Aproximadamente 9 usuarios entre administradores, terapeutas, asistentes y demás serán los beneficiarios del uso del proyecto. Es posible que el número de usuarios varía dependiendo de los empleados de la fundación FUDRINE.

- Los usuarios tendrán acceso mediante una autentificación inicial en el sistema sobre un ordenador y tras este paso entrarán a la parte de aplicación diseñada para cada uno según su rol en la empresa.

- El sistema será multiplataforma dado que será desarrollado en lenguaje JAVA.

3.1.3.4. PERFILES DE PARTE INTERESADA

3.1.3.4.1. COORDINADOR Y RESPONSABLE DEL PROYECTO

Representante	Juan Carlos Contreras Paredes
Descripción	Responsable a nivel directivo y del área de sistemas del proyecto.
Tipo	Director y Analista de Sistemas
Responsabilidades	- Establecer los lineamientos generales para el desarrollo del proyecto. Coordinar a nivel directivo los distintos requerimientos que se presenten en el desarrollo del sistema. - Responsable del análisis y diseño del proyecto. Administra el correcto desarrollo del proyecto en cuanto a la construcción e implantación.
Criterio de éxito	- Mantener una funcionalidad integral en los sistemas. - Cumplir con el cronograma determinado. - Obtener un sistema de calidad que cumpla con los requerimientos funcionales establecidos.

Implicación	Jefe del proyecto
Entregables	- Documento de visión - Resumen del modelo de casos de uso - Especificaciones del modelo de casos de uso

Tabla 4: Perfiles de Parte Interesada, 2017

3.1.3.4.2. RESPONSABLE FUNCIONAL

Representante	Acosta Rodriguez Liliam del Cramen
Descripción	Representante de toda el área usuaria de la fundación FUDRINE.
Tipo	Directora Técnica / Terapista de Lenguaje
Responsabilidades	Responsable de coordinar con los diferentes usuarios la correcta determinación de los requerimientos y concepción del sistema.
Criterio de éxito	Obtener un sistema de calidad que cumpla con los requerimientos funcionales requeridos.
Implicación	Aprueba las especificaciones funcionales y las pruebas realizadas.
Entregables	- Documento de revisión de las especificaciones funcionales. - Documento de revisión de las pruebas funcionales

Tabla 5: Responsable Funcional, 2017

3.1.3.5. PERFILES DE USUARIO

3.1.3.5.1. ADMINISTRADOR DEL SISTEMA

Representante	Maldonado Román Silvia María
Descripción	Persona de la fundación encargada de administrar el sistema.
Tipo	Presidenta / Terapista Física

Responsabilidades	Administrar funcionalmente el sistema (gestionar acceso a usuarios, dar mantenimiento al sistema frente a nuevos requerimientos).
Criterio de éxito	Obtener un sistema de calidad que cumpla con los requerimientos funcionales requeridos.
Implicación	Aprueba las especificaciones funcionales.

Tabla 6: Perfiles de Usuario - Administrador del Sistema, 2017

3.1.3.5.2. USUARIO DEL SISTEMA

Representante	Cayzac Elodie Anne-Claire
Descripción	Persona de la fundación FUDRINE que hará uso del sistema.
Tipo	Terapista Ocupacional
Responsabilidades	Ingresar, editar y adquirir información de los pacientes y estudiantes de la fundación FUDRINE.
Criterio de éxito	Obtener un sistema de calidad que cumpla con los requerimientos funcionales requeridos.
Implicación	Aprueba las especificaciones funcionales.

Tabla 7: Perfiles de Usuario – Usuario del Sistema, 2017

3.1.3.6. NECESIDADES CLAVE DE LA PARTE INTERESADA O DEL USUARIO

Necesidades	Prioridad	Inquietudes	Solución Actual	Solución Propuesta
Sistema integrado que automatice la gestión de información en la fundación FUDRINE.	Alta	El sistema debe permitir tener un control total en la administración de la información.	No existe	Crear un sistema integrado para la gestión de la información de la fundación FUDRINE.

Necesidades	Prioridad	Inquietudes	Solución Actual	Solución Propuesta
Desarrollo del sistema en el menor tiempo posible.	Alta		En la actualidad el proceso se realiza de forma manual.	Finalizar el desarrollo e implantación del sistema en el 2017.
Construcción del sistema mediante herramientas que simplifiquen el proceso.	Alta	Hacer uso de herramientas existentes de software	No aplicable en el caso	Construcción del sistema mediante con la ayuda de las siguientes herramientas: Netbeans (Java) y MySQL.

Tabla 8: Necesidades Clave de la Parte Interesada, 2017

3.1.4. VISIÓN GENERAL DEL PRODUCTO

Esta sección ofrece una vista de alto nivel de las capacidades del producto, interfaces de otras aplicaciones y configuraciones de sistema.

3.1.4.1. PERSPECTIVA DEL PRODUCTO

El producto a desarrollar es un sistema informático que tiene como objetivo mejorar la gestión de la información de la fundación FUDRINE.

Las áreas a tratar son: gestión de usuarios (pacientes, estudiantes, administrativo), gestión de representantes por estudiante, gestión de fichas médicas, gestión de antecedentes.

En la Ilustración 6 es posible observar la interacción que existirá entre el sistema y los distintos elementos.

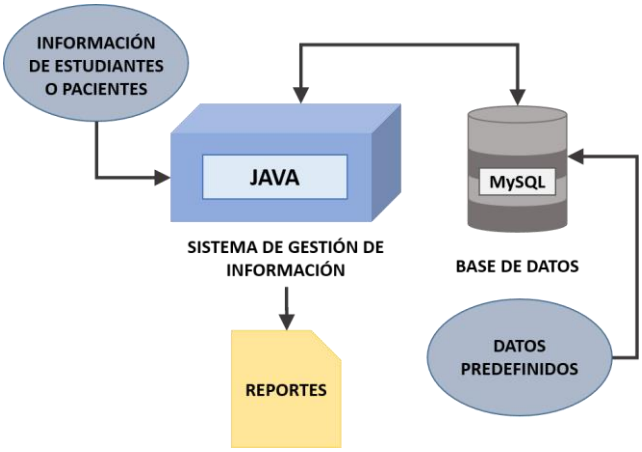

Ilustración 6: Perspectiva del Producto, 2017

3.1.4.2. RESUMEN DE CAPACIDADES

A continuación se resumen los principales beneficios y características que proporcionará el producto.

Beneficio del Cliente	Características de Soporte
Mejor organización en el almacenamiento de datos.	Los datos se almacenan en una base de datos diseñada en función a los requisitos de la fundación.
Obtención inmediata de información	Es posible obtener los datos requeridos de manera inmediata gracias a la organización que existe dentro de la base de datos y la velocidad del procesamiento informático.
Gran cantidad de información.	El sistema almacena grandes cantidades de información capaz de generar reportes y siendo posible el análisis en la misma.
Validación de información	El sistema maneja condiciones que permitan ingresar información válida, de esta forma se evita información basura.

Tabla 9: Resumen de Capacidades, 2017

3.1.4.3. SUPOSICIONES Y DEPENDENCIAS

Son posibles factores que afectan a las características que incluyen el documento de visión. El documento de visión podría variar únicamente en base a grandes cambios de requerimientos realizado por el cliente.

Se ha intentado cubrir todos los aspectos posibles para que el sistema no sea dependiente de un único factor. Un ejemplo claro es la posibilidad de funcionar en más de un sistema operativo.

3.1.4.4. COSTO Y PRECIO

El proyecto será completamente gratuito para la fundación FUDRINE, sin embargo esto no significa que no se requiera recursos para poder desarrollar el mismo.

3.1.4.5. CONCESIÓN DE LICENCIA E INSTALACIÓN

- La instalación del producto será realizada por parte nuestra, con la colaboración del equipo administrativo de la fundación.
- La configuración inicial del sistema será predefinida y posible de alterar por parte del equipo administrativo.
- La capacitación será realizada a los miembros administrativos de la fundación acompañado con un manual que permita comprender el funcionamiento del sistema.

3.1.5. CARACTERÍSTICAS DEL PRODUCTO

Esta sección ofrece una vista de alto nivel de las capacidades del producto, interfaces de otras aplicaciones y configuraciones de sistema.

3.1.5.1. INFORMACIÓN VALIDADA

Los datos son previamente revisados por el sistema antes de ingresarlos para de esta forma evitar cualquier tipo de dato inválido, convirtiendo a la información en confiable y segura.

3.1.5.2. CAPACIDAD DE OBTENER INFORMACIÓN DE FORMA INMEDIATA

Una característica muy importante del sistema es proveer de información al equipo de la fundación en el momento que requieran de la misma.

3.1.5.3. USO SENCILLO

El sistema será desarrollado mediante parámetros generales de usabilidad, convirtiéndose en un sistema intuitivo y fácil de usar.

3.1.5.4. DISPONIBILIDAD DE LOG

Permite al equipo administrativo conocer las acciones que hayan realizado los usuarios en el sistema dado que registra todo movimiento en el mismo.

3.1.5.5. INFORMACIÓN ACTUALIZADA

La información se actualiza de forma inmediata, manteniendo al día a todos los usuarios con los últimos cambios realizados.

3.1.5.6. REPORTES DE INTERÉS

El sistema contará con un sistema de reportes, capaz de generar y desplegar información de importancia para llevar un posible análisis por parte de los miembros de la fundación.

3.1.5.7. GRAN CANTIDAD DE INFORMACIÓN

La base de datos enlazada al sistema permitirá almacenar grandes cantidades de información de forma organizada.

3.1.6. RESTRICCIONES

En el sistema se ha solicitado por parte del equipo administrativo:

- Restricciones de diseño. Se busca un diseño sencillo y fácil de utilizar
- Restricción de sistema operativo. Los miembros de la fundación se encuentran enseñados al uso del S.O. Windows.
- Validación de datos. Se busca restringir que se ingresen datos erróneos al sistema para evitar cualquier tipo de información inválida.

3.1.7. RANGOS DE CALIDAD

El desarrollo del sistema encargado de la gestión de información para la fundación FUDRINE se ajustará a la metodología de desarrollo de software RUP (Proceso Racional Unificado).

3.1.8. PRECEDENCIA Y PRIORIDAD

Prioridad	Característica
1	Ingreso y carga de datos
2	Edición de datos
3	Verificación de datos ingresados
4	Emisión de reportes
5	Interfaces con parámetros de usabilidad
6	Creación de Log de acciones

Tabla 10: Precedencia y Prioridad, 2017

3.1.9. OTROS REQUISITOS DEL PRODUCTO

- Se requiere un computador que funcione como servidor en el cual se almacene la base de datos. El computador debe tener un procesador mayor o igual a Intel Core i3 para un funcionamiento adecuado.
- Cumplimiento de plataforma (Windows)
- Conexión de red local.

3.1.10. REQUISITOS DE DOCUMENTACIÓN

- Guías de instalación: documento que incluye instrucciones relativas a la instalación y configuración inicial del sistema.
- Guías de uso: documento instructivo sobre el uso de la aplicación, y las funcionalidades de la misma.

3.1.11. ATRIBUTOS DE CARACTERÍSTICA

En la siguiente tabla presentamos las características que abarcará el sistema junto a su estado, beneficio hacia la fundación, el esfuerzo requerido, los riesgos, la estabilidad y el encargado de desarrollo.

Característica	Estado	Beneficio	Esfuerzo	Riesgo	Estabilidad	Asignación
1. Gestión de Usuario	Propuesta: Si Aprobada: Si Incorporada: No	Importante	Alto	-	-	Juan Carlos Contreras
2. Gestión de Representante	Propuesta: Si Aprobada: Si Incorporada: No	Importante	Alto	-	-	Juan Carlos Contreras
3. Gestión de ficha médica	Propuesta: Si Aprobada: Si Incorporada: No	Importante	Alto	-	-	Juan Carlos Contreras

Característica	Estado	Beneficio	Esfuerzo	Riesgo	Estabilidad	Asignación
4. Gestión de antecedentes	Propuesta: Si Aprobada: Si Incorporada: No	Importante	Alto	-	-	Juan Carlos Contreras
5. Información validada	Propuesta: Si Aprobada: Si Incorporada: No	Útil	Bajo	-	-	Juan Carlos Contreras
6. Interfaces bajo parámetros de usabilidad	Propuesta: Si Aprobada: Si Incorporada: No	Útil	Medio	-	-	Juan Carlos Contreras
7. Registro de LOG	Propuesta: Si Aprobada: Si Incorporada: No	Útil	Bajo	-	-	Juan Carlos Contreras
8. Reportes	Propuesta: Si Aprobada: Si Incorporada: No	Útil	Alto	-	-	Juan Carlos Contreras

Tabla 11: Requisitos - Atributos de Característica, 2017

3.2. ESPECIFICACIÓN DE REQUERIMIENTOS

La especificación de requerimientos de software (ERS) es un conjunto de información que describe todos los requisitos y requerimientos que el cliente desea. Este permite a los desarrolladores de software realizar un análisis que funcionará a futuro como guía para llevar a cabo una correcta codificación del mismo.

3.2.1. DESCRIPCIÓN DE REQUERIMIENTOS

3.2.1.1. ACTORES

Los actores son entidades externas al sistema que mantienen una relación y demandan una función dentro del mismo.

- **Administrador**: Persona de la fundación encargada de administrar el sistema.
  - Acosta Rodriguez Liliam del Cramen
  - Maldonado Román Silvia María
  - Zuleta Proaño Josefina Matilde
  - Pavón Albuja David Sebastian

- **Empleado**: Persona de la fundación FUDRINE que hará uso de las funcionalidades del sistema.
  - Cayzac Elodie Anne Claire
  - Intriago Burgos Pancracia de Lourdes
  - Moriano Matuhura Claudia Stella
  - Ortiz Castelo Jessica Carolina
  - Salgado Moreno Mónica Alexandra

3.2.1.2. REQUERIMIENTOS NO FUNCIONALES

Los requerimientos no funcionales de un sistema son aquellos que no se relacionan directamente a las funciones específicas que proporciona un sistema, estos se refieren a las propiedades emergentes del mismo. Muy rara vez los requerimientos no funcionales se asocian a características particulares del sistema. A continuación detallamos los requerimientos no funcionales del sistema.[20]

3.2.1.2.1. DEL PRODUCTO

- **Usabilidad:**
  - **RN-1** El sistema está preparado para ser operado a través de ratón y teclado.
  - **RN-2** Presentar mensajes de información, error y advertencia intuitivos.

[20] (Sommerville, 2006)

- **RN-3** Validar tipos de datos y límites de campos de las pantallas en relación al modelo de datos.
- **RN-4** Intuitivo y comprensible.

- **Confiabilidad:**
  - **RN-5** Implementación de métodos que aseguren la integridad de los datos.

- **Documentación:**
  - **RN-6** Correcta redacción y ortografía en las pantallas.
  - **RN-7** Uso estandarizado de pantallas, mensajes y estilos.

3.2.1.2.2. DEL AMBIENTE

- **Ético:**
  - **RN-8** El sistema debe garantizar la confidencialidad de la información del cliente.

- **Legales:**
  - **RN-9** Se debe cumplir lo establecido en los contratos.

3.2.1.3. REQUERIMIENTOS FUNCIONALES

Los requerimientos funcionales de un sistema representan lo que el sistema debe hacer. Los requerimientos funcionales del sistema detallan su función, entradas, salidas, excepciones y más. [21]

3.2.1.3.1. FUNCIONALIDADES DEL SISTEMA

Las funcionalidades con las que contará el sistema de gestión de información para la fundación FUDRINE son las siguientes:

[21] (Sommerville, 2006)

- F0: Ingreso al Sistema
- F1: Gestionar Paciente
- F2: Gestionar Empleado
- F3: Gestionar Representante
- F4: Gestionar Antecedentes
- F5: Gestionar Ficha
- F6: Generar Reporte
- F7: Salir del Sistema

Las funcionalidades se presentan a detalle a continuación en la sección 3.3 donde se explican con la ayuda de diagramas de caso de uso.

3.3. DIAGRAMA DE CASOS DE USO

El diagrama de casos de uso es una representación visual y sencilla de interpretar el cual resume quién usa la aplicación o sistema, y qué pueden hacer en el mismo. En varios casos por la gran cantidad de casos de uso que suelen aparecer, el desarrollador de un sistema crea varios diagramas de caso de uso donde cada diagrama puede representar uno de los subsistemas de negocio.

3.3.1. DIAGRAMA GENERAL DE CASO DE USO DEL SISTEMA

El diagrama general es una representación sin detalle de las funciones que tendrá el sistema junto con los actores que interactúan en las mismas. A continuación se presenta el diagrama general del sistema de gestión de información de la fundación FUDRINE.[22] [23]

[22] (Microsoft, 2017)
[23] (Alarcón, 2008)

Diagrama general:

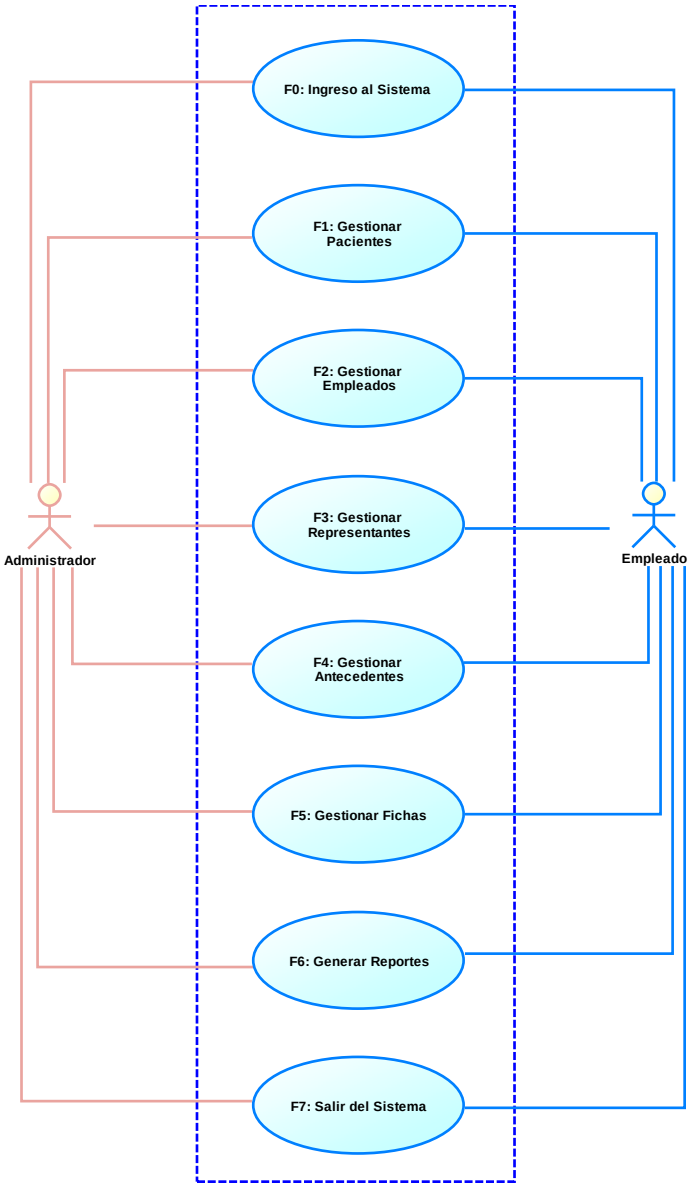

Ilustración 7: Diagrama General de Sistema de Gestión de Información (FUDRINE), 2017

3.3.2. DIAGRAMAS A DETALLE DE CASOS DE USO DEL SISTEMA

El programa consta de 8 funcionalidades las cuales han sido enumeradas de la F0 (Funcionalidad 0) a la F7 (Funcionalidad 7). A continuación se da una descripción a detalle en siguiente nivel de estas funcionalidades.

Para el caso de las funcionalidades uno, dos y tres el procedimiento es similar y por este motivo únicamente se explicará a detalle la funcionalidad número uno (Gestión

de Pacientes). Se podrá encontrar el conjunto de diagramas e información completa en el CD presentado.

3.3.2.1. FUNCIONALIDAD 0 - INGRESO AL SISTEMA

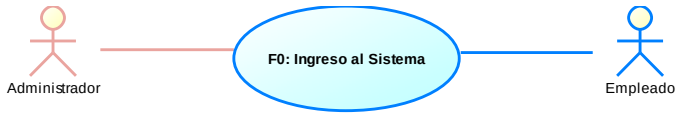

Ilustración 8: Diagrama Funcionalidad 0 - Ingreso al Sistema del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.1.1. DESCRIPCIÓN

Permite ingresar al sistema de la empresa FUDRINE, el cual es autenticado y autorizado para la utilización de este.

3.3.2.1.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.1.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El usuario ingresa al sistema mediante el acceso directo al programa.
2. El sistema presenta la pantalla de ingreso al sistema.
3. El usuario digita el nombre de usuario y contraseña en las casillas correspondientes al formulario.
4. El usuario selecciona la opción de ingresar al sistema.
5. El sistema verifica el nombre de usuario y la contraseña. **(E1)**
6. El sistema autentica al usuario para utilizar el sistema.

7. El sistema autoriza al usuario proporcionándole su perfil de entrada para utilizar el sistema.
8. El sistema redirecciona a la ventana principal del sistema.
9. El sistema almacena la acción realizada en el historial de acciones del sistema.

3.3.2.1.4. FLUJO ALTERNO

5. En el paso 5 del flujo principal, el sistema valida el nombre de usuario y contraseña. En caso de ser erróneos los datos ingresados el sistema presenta un mensaje indicando que los mismos no son correctos. Direccionándonos al paso número dos del flujo principal.

3.3.2.1.5. EXCEPCIONES

- E1: El sistema no permitirá ingresar si los datos son incorrectos. Mensaje: "Error: El nombre de usuario o contraseña son incorrectos."

3.3.2.2. FUNCIONALIDAD 1 – GESTIONAR PACIENTES

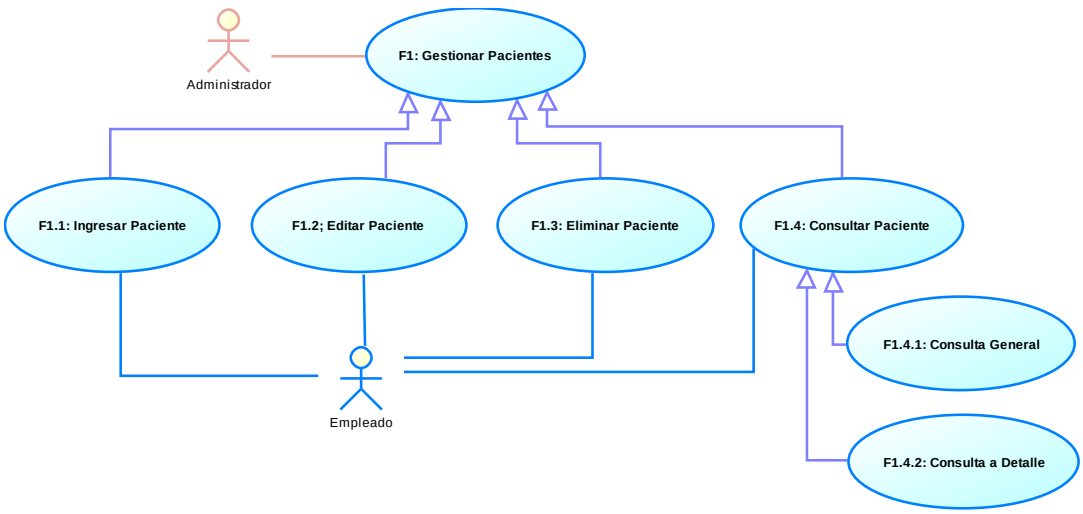

Ilustración 9: Diagrama Funcionalidad 1 – Gestionar Pacientes del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.2.1. DESCRIPCIÓN

El usuario ingresará al sistema y a través de esta opción tendrá la posibilidad del manejo completo de CRUD, es decir de: ingresar, modificar, eliminar un paciente, así como consultar los pacientes existentes.

3.3.2.2.2. FUNCIONALIDAD 1.1 – INGRESAR PACIENTE

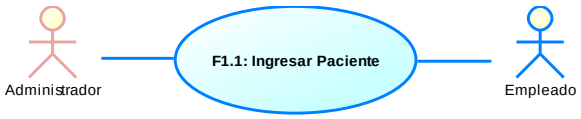

Ilustración 10: Diagrama Funcionalidad 1.1 - Ingresar Paciente del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.2.2.1. DESCRIPCIÓN

El administrador y el empleado podrán ingresar un nuevo paciente completando un formulario.

3.3.2.2.2.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.2.2.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El actor selecciona la opción de gestión de pacientes.
2. El sistema despliega una interfaz para gestionar pacientes.
3. El actor ingresa el código de identificación del paciente.

4. El sistema verifica el código de identificación.
5. El actor ingresa los datos del formulario restantes requeridos para crear un paciente.
6. El actor pulsa el botón guardar. **(E1)**
7. El sistema almacena la información del paciente.
8. El sistema almacena la acción realizada en el historial de acciones del sistema.

3.3.2.2.2.4. FLUJO ALTERNO

4. En el paso 4 del flujo principal el sistema valida el código de identificación. En caso de ya encontrarse registrado en la base de datos se dirige al caso de uso F1.2 o F1.3.

3.3.2.2.2.5. EXCEPCIONES

- E1: El sistema no permitirá ingresar un nuevo paciente si no se han completado todos los campos de manera correcta. Mensaje: "Error: Verifique que los datos ingresados sean correctos".

3.3.2.2.3. FUNCIONALIDAD 1.2 – EDITAR PACIENTE

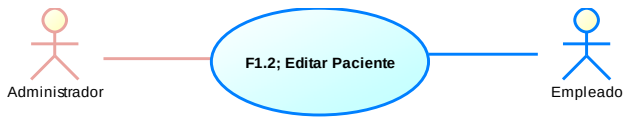

Ilustración 11: Diagrama Funcionalidad 1.2 – Editar Paciente del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.2.3.1. DESCRIPCIÓN

El administrador y el empleado podrán editar la información de un paciente con excepción de su código de identificación.

3.3.2.2.3.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.2.3.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El actor selecciona la opción de gestión de pacientes.
2. El sistema despliega una interfaz para gestionar pacientes.
3. El actor ingresa el código de identificación del paciente.
4. El sistema verifica el código de identificación.
5. El sistema despliega los datos del paciente.
6. El actor modifica los campos deseados.
7. El actor pulsa el botón modificar. **(E1)**
8. El sistema almacena la información del paciente.
9. El sistema almacena la acción realizada en el historial de acciones del sistema.

3.3.2.2.3.4. FLUJO ALTERNO

5. En el paso 4 del flujo principal el sistema valida el código de identificación. En caso de no encontrarse registrado en la base de datos se dirige al caso de uso F1.1.

3.3.2.2.3.5. EXCEPCIONES

- E1: El sistema no permitirá modificar un paciente existente si no se han completado todos los campos de manera correcta. Mensaje: “Error: Verifique que los datos ingresados sean correctos”.

3.3.2.2.4. FUNCIONALIDAD 1.3 – ELIMINAR PACIENTE

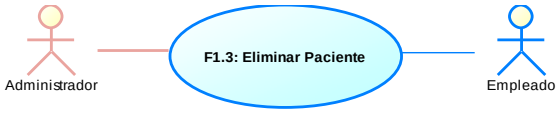

Ilustración 12: Diagrama Funcionalidad 1.3 – Eliminar Paciente del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.2.4.1. DESCRIPCIÓN

El administrador y el empleado podrán eliminar a un paciente mediante el código de identificación.

3.3.2.2.4.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.2.4.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El actor selecciona la opción de gestión de pacientes.
2. El sistema despliega una interfaz para gestionar pacientes.
3. El actor ingresa el código de identificación del paciente.
4. El sistema verifica el código de identificación.
5. El sistema despliega los datos del paciente.
6. El actor pulsa el botón eliminar. **(E1)**
7. El sistema elimina la información del paciente.
8. El sistema almacena la acción realizada en el historial de acciones del sistema.

3.3.2.2.4.4. FLUJO ALTERNO

6. En el paso 4 del flujo principal el sistema valida el código de identificación. En caso de no encontrarse registrado en la base de datos se dirige al caso de uso F1.1.

3.3.2.2.4.5. EXCEPCIONES

- E1: El sistema no permitirá eliminar un paciente si el usuario no confirma la acción. Mensaje: “Atención: Está seguro que desea eliminar al paciente”.

3.3.2.2.5. FUNCIONALIDAD 1.4 – CONSULTAR PACIENTE

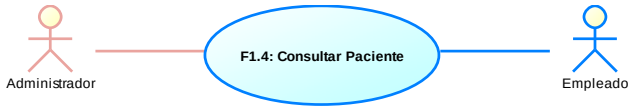

Ilustración 13: Diagrama Funcionalidad 1.4 – Consultar Paciente del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.2.5.1. DESCRIPCIÓN

El administrador y el empleado podrán consultar una lista de todos los pacientes ingresados o de forma detallada mediante el código de identificación.

3.3.2.2.5.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.2.5.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El actor selecciona la opción de gestión de pacientes.
2. El sistema despliega una interfaz para gestionar pacientes.
3. El actor selecciona la opción de consultas.
4. El sistema despliega una ventana para consultar pacientes.
5. El sistema presenta una consulta general de los pacientes con la posibilidad a filtrar la información.
6. El actor selecciona o coloca el dato mediante el cual desea filtrar los datos.
7. El sistema valida los datos. **(E1)**
8. En base al filtro seleccionado o colocado el sistema presentará los datos encontrados.

3.3.2.2.5.4. EXCEPCIONES

- E1: El sistema informará al actor en caso de no existir coincidencias en los datos. Mensaje: "Atención: No hay datos registrados con esas características".

3.3.2.3. FUNCIONALIDAD 4 – GESTIONAR ANTECEDENTES

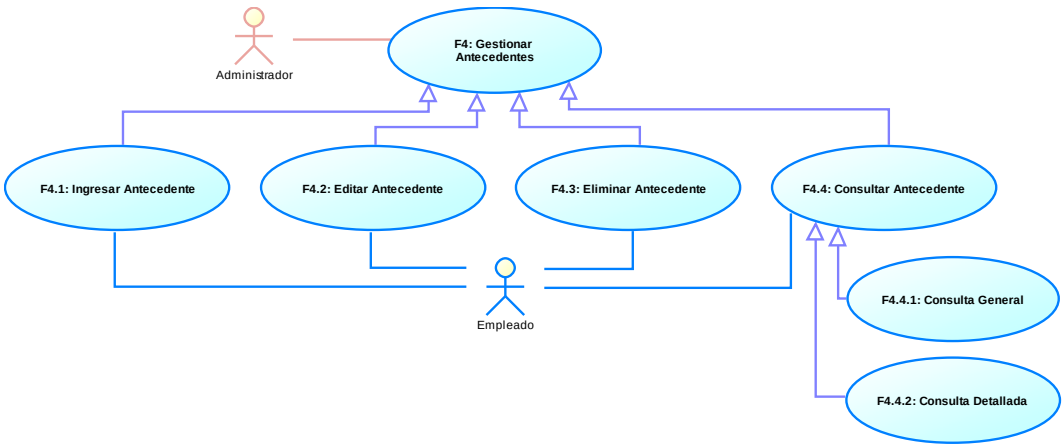

Ilustración 14: Diagrama Funcionalidad 4 – Gestionar Antecedentes del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.3.1. DESCRIPCIÓN

El usuario ingresará al sistema y a través de esta opción tendrá la posibilidad del manejo completo de CRUD, es decir de: ingresar, modificar, eliminar un antecedente, así como consultar los antecedentes existentes de cada paciente.

3.3.2.3.2. FUNCIONALIDAD 4.1 – INGRESAR ANTECEDENTE

Ilustración 15: Diagrama Funcionalidad 4.1 - Ingresar Representante del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.3.2.1. DESCRIPCIÓN

El administrador y el empleado podrán ingresar un nuevo antecedente completando un formulario.

3.3.2.3.2.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.3.2.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El actor selecciona la opción de gestión de antecedentes.
2. El sistema despliega una interfaz para gestionar antecedentes.
3. El actor selecciona un paciente ingresado o coloca su código de identificación.

4. El sistema verifica el código de identificación.
5. El actor ingresa los datos del formulario restantes requeridos para crear un antecedente.
6. El actor pulsa el botón guardar. **(E1)**
7. El sistema almacena la información del antecedente.
8. El sistema almacena la acción realizada en el historial de acciones del sistema.

3.3.2.3.2.4. FLUJO ALTERNO

7. En el paso 4 del flujo principal el sistema valida el código de identificación. En caso de no encontrarse registrado en la base de datos se dirige al caso de uso F1.1 o regresar al paso 3 del caso de uso F5.1.

3.3.2.3.2.5. EXCEPCIONES

- E1: El sistema no permitirá ingresar un nuevo antecedente si no se han completado todos los campos de manera correcta. Mensaje: “Error: Verifique que los datos ingresados sean correctos”.

3.3.2.3.3. FUNCIONALIDAD 4.2 – EDITAR ANTECEDENTE

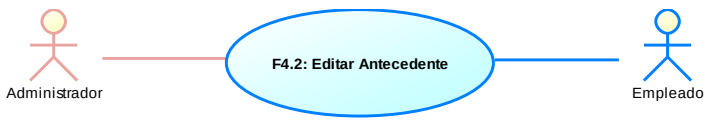

Ilustración 16: Diagrama Funcionalidad 4.2 – Editar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.3.3.1. DESCRIPCIÓN

El administrador y el empleado podrán editar la información de un antecedente de un paciente.

3.3.2.3.3.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.3.3.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El actor selecciona la opción de gestión de antecedentes.
2. El sistema despliega una interfaz para gestionar antecedentes.
3. El actor selecciona el antecedente que desea editar.
4. El sistema despliega los datos del antecedente.
5. El actor modifica los campos deseados.
6. El actor pulsa el botón modificar. **(E1)**
7. El sistema almacena la información del antecedente.
8. El sistema almacena la acción realizada en el historial de acciones del sistema.

3.3.2.3.3.4. EXCEPCIONES

- E1: El sistema no permitirá modificar un antecedente existente si no se han completado todos los campos de manera correcta. Mensaje: "Error: Verifique que los datos ingresados sean correctos".

3.3.2.3.4. FUNCIONALIDAD 4.3 – ELIMINAR ANTECEDENTE

Ilustración 17: Diagrama Funcionalidad 4.3 – Eliminar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.3.4.1. DESCRIPCIÓN

El administrador y el empleado podrán eliminar los antecedentes de un paciente.

3.3.2.3.4.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.3.4.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El actor selecciona la opción de gestión de antecedentes.
2. El sistema despliega una interfaz para gestionar antecedentes.
3. El actor selecciona el antecedente que desea eliminar.
4. El sistema despliega los datos del antecedente.
5. El actor pulsa el botón eliminar. **(E1)**
6. El sistema elimina la información del antecedente.
7. El sistema almacena la acción realizada en el historial de acciones del sistema.

3.3.2.3.4.4. EXCEPCIONES

- E1: El sistema no permitirá eliminar un antecedente si el usuario no confirma la acción. Mensaje: "Atención: Está seguro que desea eliminar el antecedente".

3.3.2.3.5. FUNCIONALIDAD 4.4 – CONSULTAR ANTECEDENTE

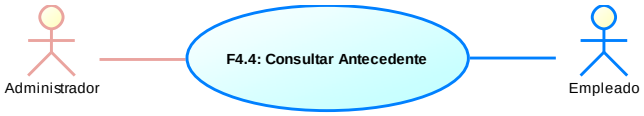

Ilustración 18: Diagrama Funcionalidad 4.4 – Consultar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.3.5.1. DESCRIPCIÓN

El administrador y el empleado podrán consultar una lista de todos los antecedentes ingresados o de forma detallada mediante el código de identificación del paciente.

3.3.2.3.5.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.3.5.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El actor selecciona la opción de gestión de antecedentes.
2. El sistema despliega una interfaz para gestionar antecedentes.
3. El actor selecciona la opción consultar antecedente.
4. El sistema despliega una ventana para consultar antecedentes.
5. El sistema presenta una consulta general de los antecedentes con la posibilidad a filtrar la información.
6. El actor selecciona o coloca el dato mediante el cual desea filtrar los datos.

7. El sistema valida los datos. **(E1)**
8. En base al filtro seleccionado o colocado el sistema presentará los datos encontrados.

3.3.2.3.5.4. EXCEPCIONES

- E1: El sistema informará al actor en caso de no existir coincidencias en los datos. Mensaje: "Atención: No hay datos registrados con esas características".

3.3.2.4. FUNCIONALIDAD 5 – GESTIONAR FICHAS

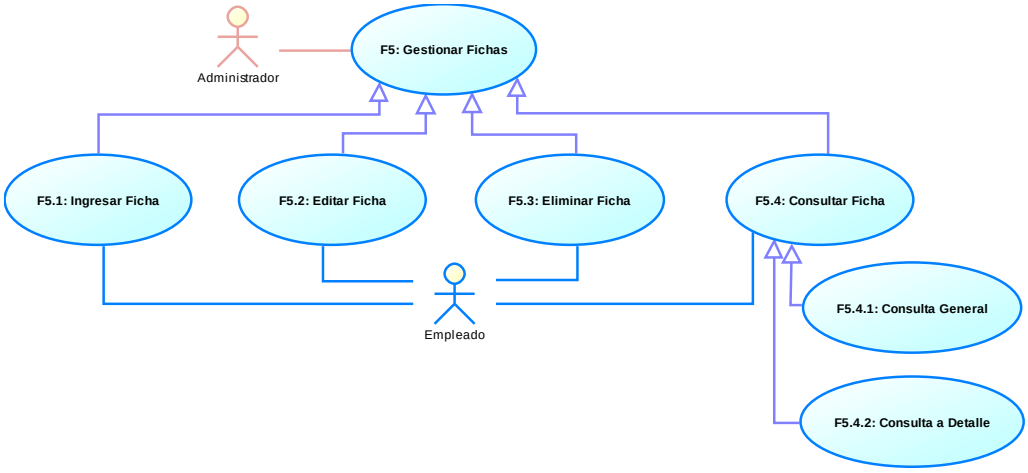

Ilustración 19: Diagrama Funcionalidad 5 – Gestionar Fichas del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.4.1. DESCRIPCIÓN

El usuario ingresará al sistema y a través de esta opción tendrá la posibilidad del manejo completo de CRUD, es decir de: ingresar, modificar, eliminar fichas, así como consultar fichas existentes de cada paciente.

3.3.2.4.2. FUNCIONALIDAD 5.1 – INGRESAR FICHA

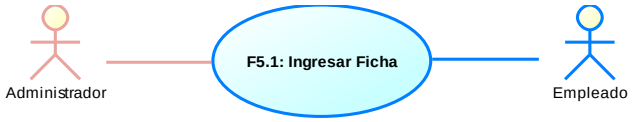

Ilustración 20: Diagrama Funcionalidad 5.1 - Ingresar Ficha del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.4.2.1. DESCRIPCIÓN

El administrador y el empleado podrán ingresar una nueva ficha completando un formulario.

3.3.2.4.2.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.4.2.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El actor selecciona la opción de gestión de fichas.
2. El sistema despliega una interfaz para gestionar fichas.
3. El actor selecciona un paciente ingresado o coloca su código de identificación.
4. El sistema verifica el código de identificación.
5. El actor selecciona el tipo de ficha que desea ingresar.
6. El sistema carga un formulario asociado al tipo de ficha seleccionado
7. El actor ingresa los datos del formulario restantes requeridos para crear una ficha.
8. El actor pulsa el botón guardar. **(E1)**

9. El sistema almacena la información de la ficha.

10. El sistema almacena la acción realizada en el historial de acciones del sistema.

3.3.2.4.2.4. FLUJO ALTERNO

8. En el paso 4 del flujo principal el sistema valida el código de identificación. En caso de no encontrarse registrado en la base de datos se dirige al caso de uso F1.1 o regresar al paso 3 del caso de uso F5.1.

3.3.2.4.2.5. EXCEPCIONES

- E1: El sistema no permitirá ingresar un nuevo antecedente si no se han completado todos los campos de manera correcta. Mensaje: “Error: Verifique que los datos ingresados sean correctos”.

3.3.2.4.3. FUNCIONALIDAD 5.2 – EDITAR FICHA

Ilustración 21: Diagrama Funcionalidad 5.2 – Editar Ficha del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.4.3.1. DESCRIPCIÓN

El administrador y el empleado podrán editar la información de una ficha de un paciente.

3.3.2.4.3.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.4.3.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El actor selecciona la opción de gestión de fichas.
2. El sistema despliega una interfaz para gestionar fichas.
3. El actor selecciona un paciente ingresado o coloca su código de identificación.
4. El sistema verifica el código de identificación.
5. El actor selecciona la ficha que desea editar.
6. El sistema despliega los datos de la ficha.
7. El actor modifica los campos deseados.
8. El actor pulsa el botón modificar. **(E1)**
9. El sistema almacena la información de la ficha.
10. El sistema almacena la acción realizada en el historial de acciones del sistema.

3.3.2.4.3.4. EXCEPCIONES

- E1: El sistema no permitirá modificar una ficha existente si no se han completado todos los campos de manera correcta. Mensaje: “Error: Verifique que los datos ingresados sean correctos”.

3.3.2.4.4. FUNCIONALIDAD 5.3 – ELIMINAR FICHA

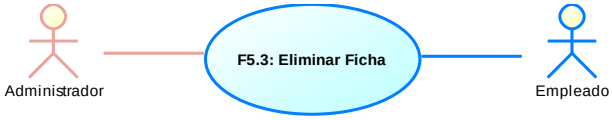

Ilustración 22: Diagrama Funcionalidad 5.3 – Eliminar Ficha del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.4.4.1. DESCRIPCIÓN

El administrador y el empleado podrán eliminar fichas de un paciente.

3.3.2.4.4.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.4.4.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El actor selecciona la opción de gestión de fichas.
2. El sistema despliega una interfaz para gestionar fichas.
3. El actor selecciona un paciente ingresado o coloca su código de identificación.
4. El sistema verifica el código de identificación.
5. El actor selecciona la ficha que desea eliminar.
6. El sistema despliega los datos de la ficha.
7. El actor pulsa el botón eliminar. **(E1)**
8. El sistema elimina la información de la ficha.
9. El sistema almacena la acción realizada en el historial de acciones del sistema.

3.3.2.4.4.4. EXCEPCIONES

- E1: El sistema no permitirá eliminar una ficha si el usuario no confirma la acción. Mensaje: "Atención: Está seguro que desea eliminar la ficha".

3.3.2.4.5. FUNCIONALIDAD 5.4 – CONSULTAR FICHA

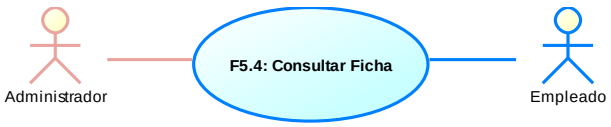

Ilustración 23: Diagrama Funcionalidad 5.4 – Consultar Ficha del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.4.5.1. DESCRIPCIÓN

El administrador y el empleado podrán consultar una lista de todas las fichas ingresadas o de forma detallada mediante el código de identificación del paciente.

3.3.2.4.5.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.4.5.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El actor selecciona la opción de gestión de fichas.
2. El sistema despliega una interfaz para gestionar fichas.
3. El actor selecciona la opción consultar fichas.
4. El sistema despliega una ventana para consultar fichas.
5. El sistema presenta una consulta general de las fichas con la posibilidad a filtrar la información.
6. El actor selecciona o coloca el dato mediante el cual desea filtrar los datos.
7. El sistema valida los datos. **(E1)**

8. En base al filtro seleccionado o colocado el sistema presentará los datos encontrados.
9. El sistema almacena la acción realizada en el historial de acciones del sistema.

3.3.2.4.5.4. EXCEPCIONES

- E1: El sistema informará al actor en caso de no existir coincidencias en los datos. Mensaje: "Atención: No hay datos registrados con esas características".

3.3.2.5. FUNCIONALIDAD 6 – GENERAR REPORTES

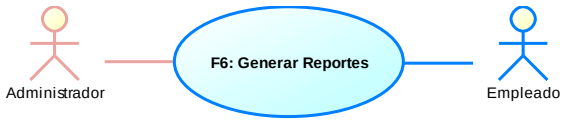

Ilustración 24: Diagrama Funcionalidad 6 – Generar Reportes del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.5.1. DESCRIPCIÓN

Permite generar reportes en base a la recopilación de datos almacenados con la ayuda de nuestro sistema.

3.3.2.5.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.5.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

1. El actor selecciona la opción de reportes.
2. El sistema despliega una interfaz para lograr observar reportes.
3. El actor selecciona el tipo de reporte que desea observar.
4. El sistema presenta la información del reporte seleccionado. **(E1)**

3.3.2.5.4. EXCEPCIONES

- E1: El sistema levantará un mensaje en caso de no poseer los suficientes datos para realizar el reporte. Mensaje: "Error: El sistema no tiene los suficientes datos para desplegar el reporte."

3.3.2.6. FUNCIONALIDAD 7 – SALIR DEL SISTEMA

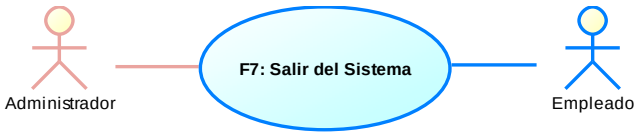

Ilustración 25: Diagrama Funcionalidad 7 – Salir del Sistema de Gestión de Información (FUDRINE), 2017

3.3.2.6.1. DESCRIPCIÓN

Permite cerrar el sistema.

3.3.2.6.2. ACTORES

Los actores que operan en esta funcionalidad son:

- Administrador
- Empleado

3.3.2.6.3. FLUJO PRINCIPAL

El flujo de la funcionalidad consiste en los siguientes pasos:

5. El actor selecciona la opción salir.
6. El sistema finaliza.

4. ELABORACIÓN

En el capítulo número cuatro analizamos el dominio del problema e implementamos una base arquitectónica sólida la cual servirá como plan de proyecto, eliminando algunos posibles riesgos.

4.1. DOCUMENTO DE ARQUITECTURA DE SOFTWARE (SAD)

El documento de arquitectura de software es aquel que se encarga de recopilar representaciones de alto nivel de la estructura de un sistema o aplicación. Describe las partes que lo integran, interacciones entre ellas, patrones que supervisan su composición y restricciones a la hora de aplicar dichos patrones.

La arquitectura de software se determina en base a las instancias de cada tipo de componentes y conectores que lo componen, así como sus enlaces que forma la unión de todas éstas logrando generar una estructura.[24] [25]

4.1.1. VISTA LÓGICA

En la vista lógica encontraremos los datos como deberían percibir los usuarios finales o los especialistas de la empresa.[26]

[24] (Isidro Ramos Salavert, 2000)
[25] (IBM, 2006)
[26] (Kenneth C. Laudon, 2004)

4.1.1.1. DIAGRAMA DE CLASES

El diagrama de clases muestra una estructura estática del sistema modelado junto a un conjunto de clases y relaciones entre estas clases, mostrando los atributos y operaciones que caracterizan cada clase de objetos. Un diagrama de clases dado corresponde a un conjunto infinito de diagramas de objetos.[27] [28] En la Ilustración 26 se presenta el diagrama de clases del dominio del problema para el programa.

[27] (Ledesma, 2005)
[28] (Gutierrez, 2011)

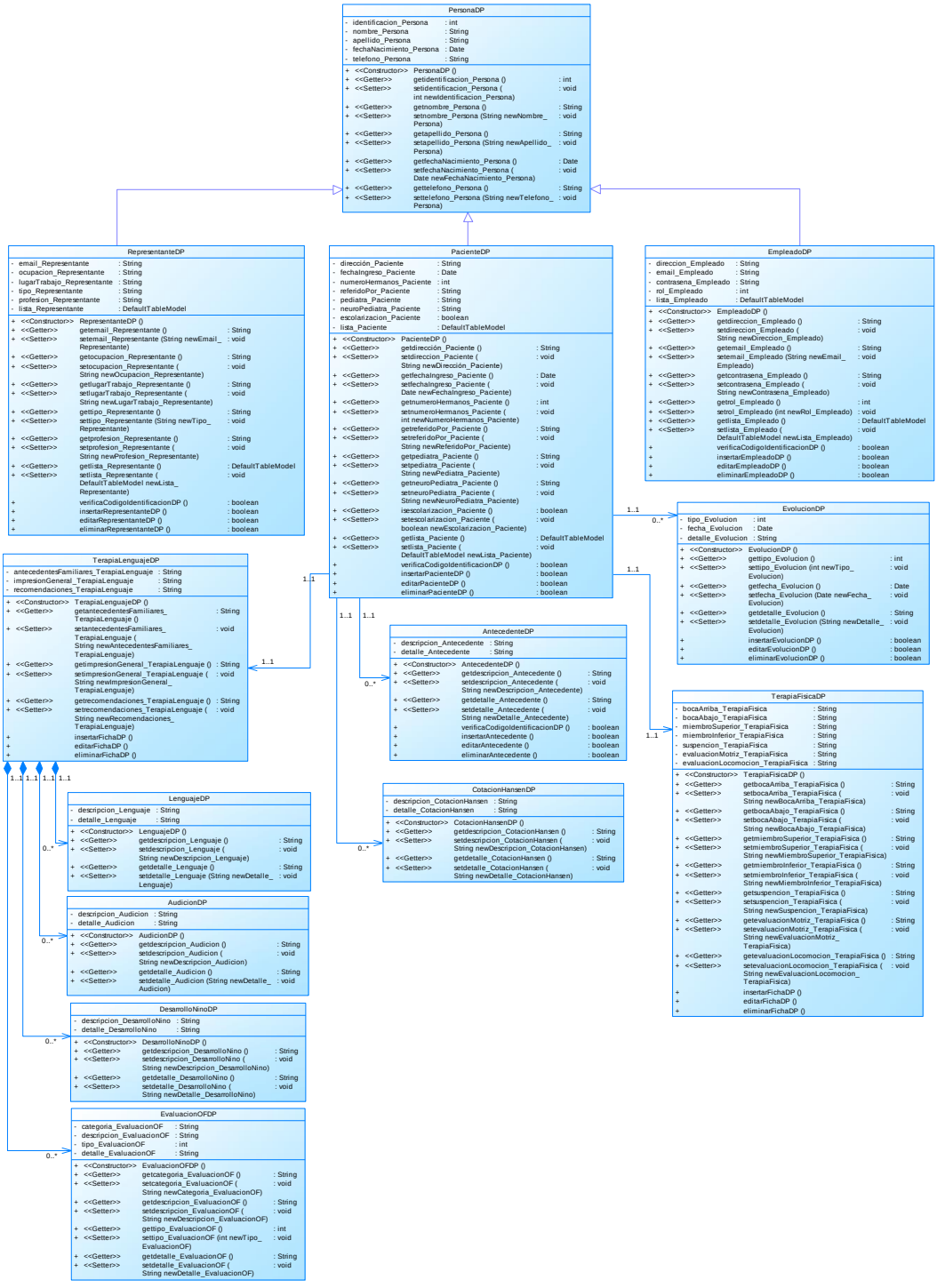

Ilustración 26: Diagrama de Clases (Dominio del Problema) – Sistema de Gestión de Información (FUDRINE), 2017

4.1.1.2. MODELO E-R

El Modelo de Entidad Relación (E-R) se basa en una percepción del mundo real el cual consiste en un conjunto de objetos básicos conocidos como entidades los

cuales se encuentran relaciones entre sí. Su función es obtener un esquema lógico general de la estructura de nuestra base de datos.[29][30][31]

En la Ilustración 27 podemos observar el modelo entidad relación que se empleará como estructura para nuestra base de datos, sin embargo por motivos de espacio no es posible mostrarla completa. Ésta se adjunta completa en el CD de entrega.

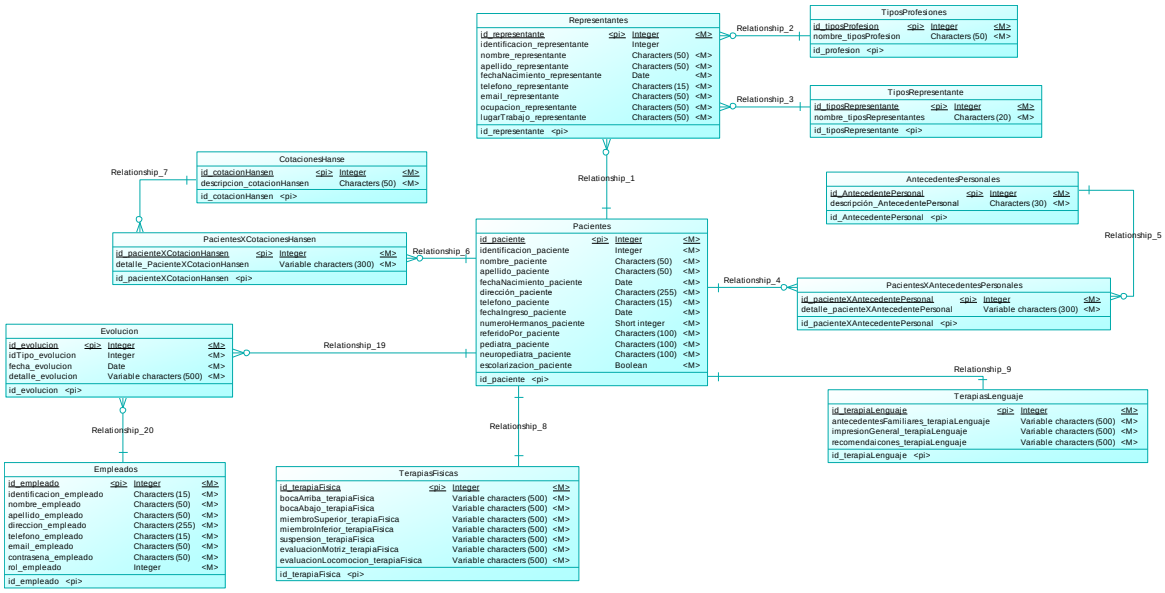

Ilustración 27: Diagrama de Modelo (E-R) –Sistema de Gestión de Información (FUDRINE), 2017

4.1.2. VISTA DE IMPLEMENTACIÓN

La vista de implementación o también conocida como vista de desarrollo es aquella en la cual se reúne las decisiones arquitectónicas tomadas para la implementación. Ésta incluye una enumeración de todos los subsistemas del modelo de implementación junto con dependencias de importación entre subsistemas.[32]

[29] (Rivera, 2008)
[30] (Guillermo Storti, 2007)
[31] (José A. Taboada Gonzáles, 2005)
[32] (Cordero, 2017)

4.1.2.1. DIAGRAMA DE SECUENCIA

Los diagramas de secuencia nos permiten visualizar la interacción que existe entre un grupo de objetos a lo largo del tiempo. Éstos nos permiten modelar escenarios en el sistema. A diferencia de los casos de uso, los diagramas de secuencia nos ofrecen una vista de análisis de bajo nivel de un sistema, donde se incluye información de diseño sobre las interacciones de objetos. Cada caso de uso se descompone en diagramas de secuencia para modelar los objetos que hacen que el escenario funcione.[33] [34]

Para el presente proyecto se ha creado diagramas de secuencia específicos para cada funcionalidad y de ser el caso a detalle para cada CRUD (Create – Remove – Update – Delete) de la misma.

4.1.2.1.1. FUNCIONALIDAD 0 - INGRESO AL SISTEMA

Descripción: Ingreso al sistema de la empresa FUDRINE, el cual es autenticado y autorizado para la utilización de este. En la Ilustración 28 podemos observar su diagrama de secuencia.

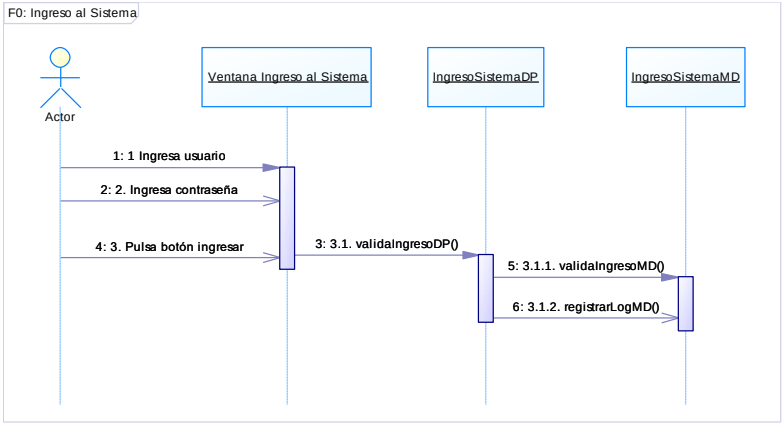

Ilustración 28: Diagrama de Secuencia - Funcionalidad 0: Ingreso al Sistema del Sistema de Gestión de Información (FUDRINE), 2017

[33] (Gutierrez, UML, 2011)
[34] (IBM, 2017)

4.1.2.1.2. FUNCIONALIDAD 1.1 – INGRESAR PACIENTE

En el sistema presentamos un grupo de 3 funcionalidades que son similares: “F1: Gestionar Paciente”, “F2: Gestionar Empleado” y “F3: Gestionar Representante”. Por este motivo se presentará una de estas funcionalidades para ilustrar el procedimiento, sin embargo será posible encontrar todos los diagramas dentro del CD presentado.

Descripción: El actor podrá ingresar un nuevo paciente completando un formulario.

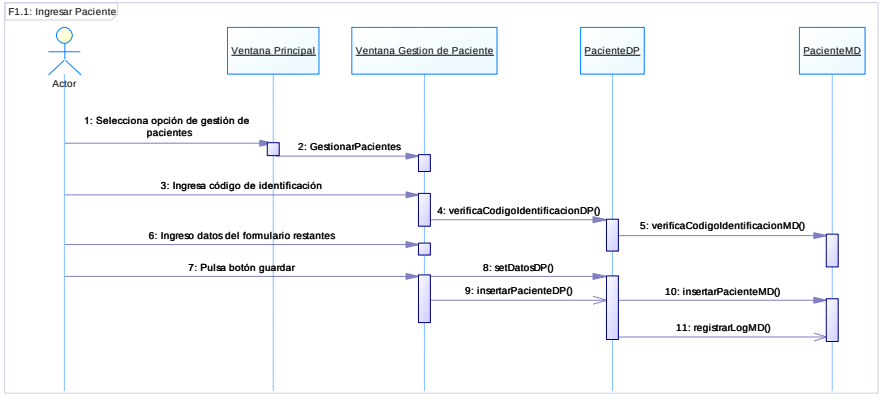

Ilustración 29: Diagrama de Secuencia - Funcionalidad 1.1: Ingresar Paciente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.1.3. FUNCIONALIDAD 1.2 – EDITAR PACIENTE

Descripción: El actor podrá editar la información de un paciente con excepción de su código de identificación.

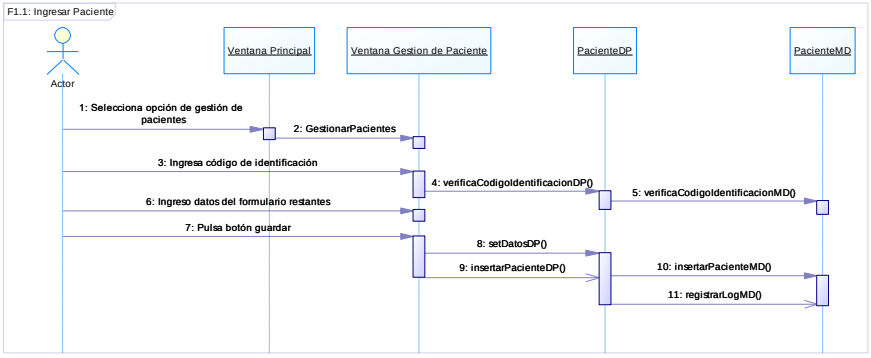

Ilustración 30: Diagrama de Secuencia - Funcionalidad 1.1: Editar Paciente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.1.4. FUNCIONALIDAD 1.3 – ELIMINAR PACIENTE

Descripción: El actor podrá eliminar a un paciente mediante el código de identificación.

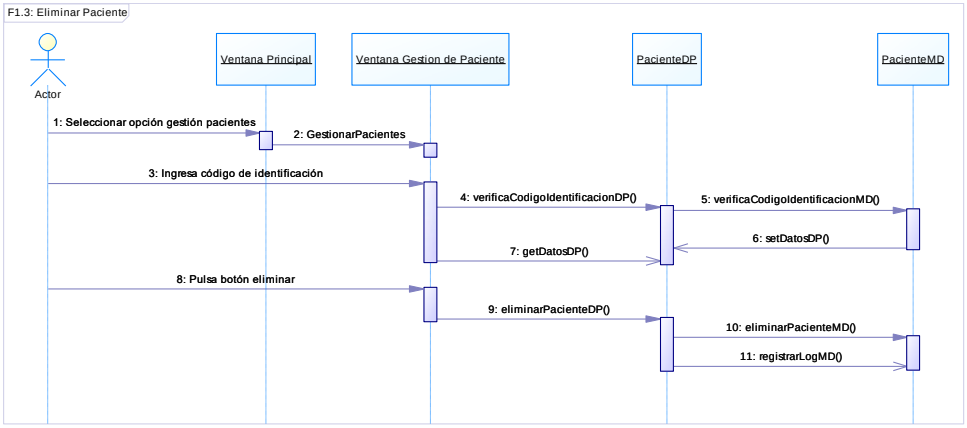

Ilustración 31: Diagrama de Secuencia - Funcionalidad 1.1: Eliminar Paciente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.1.5. FUNCIONALIDAD 1.4.1 – CONSULTAR PACIENTE

Descripción: El actor podrá consultar a los pacientes registrados.

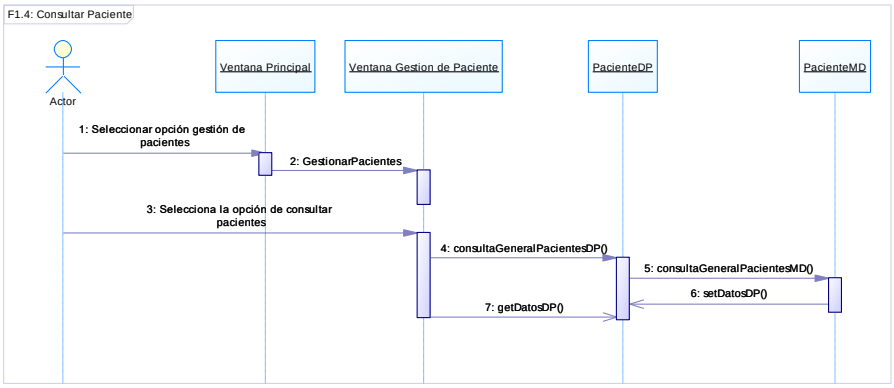

Ilustración 32: Diagrama de Secuencia - Funcionalidad 1.1: Consultar Paciente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.1.6. FUNCIONALIDAD 4.1 – INGRESAR ANTECEDENTE

A continuación se adjuntan los diagramas de secuencia a seguir para el proceso de CRUD de los antecedentes de un paciente.

Descripción: El actor podrá ingresar un nuevo antecedente completando un formulario.

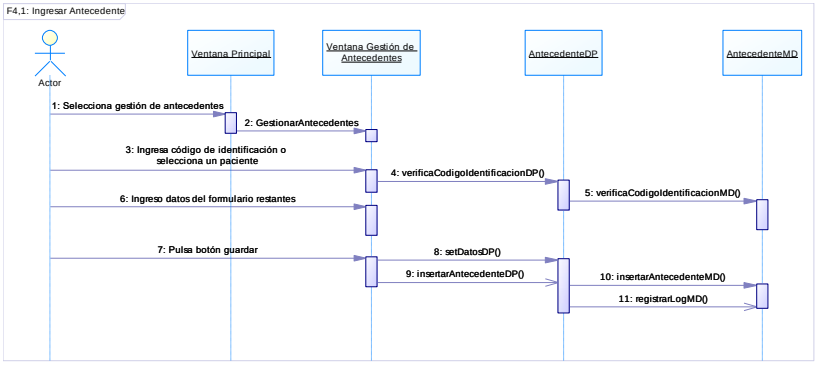

Ilustración 33: Diagrama de Secuencia - Funcionalidad 4.1: Ingresar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.1.7. FUNCIONALIDAD 4.2 – EDITAR ANTECEDENTE

Descripción: El actor podrá editar la información de un antecedente de un paciente.

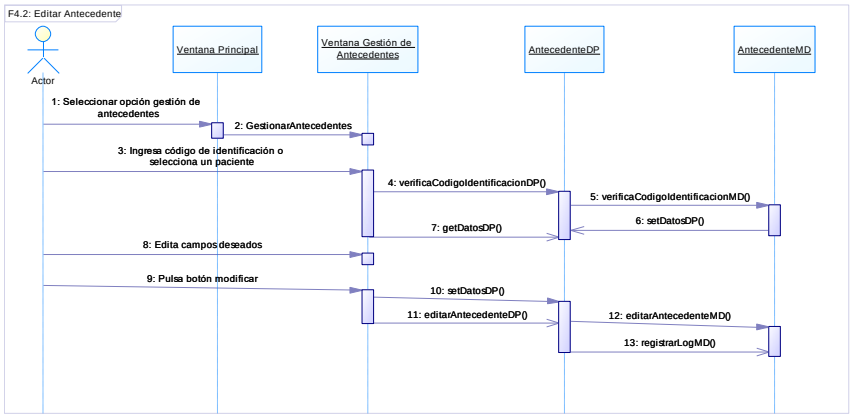

Ilustración 34: Diagrama de Secuencia - Funcionalidad 4.2: Editar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.1.8. FUNCIONALIDAD 4.3 – CONSULTAR ANTECEDENTE

Descripción: El actor podrá consultar una lista de todos los antecedentes ingresados mediante el código de identificación del paciente.

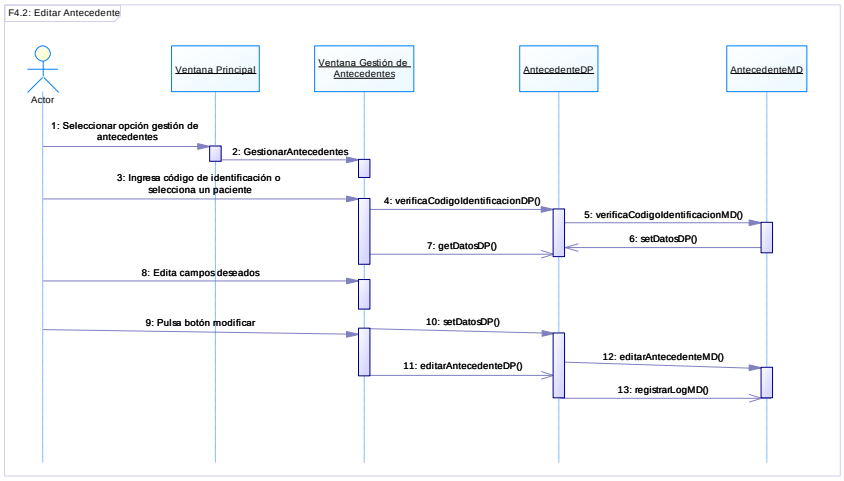

Ilustración 35: Diagrama de Secuencia - Funcionalidad 4.2: Editar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.1.9. FUNCIONALIDAD 4.4.1 – CONSULTAR ANTECEDENTE

Descripción: El actor podrá consultar una lista de todos los antecedentes ingresados mediante el código de identificación del paciente.

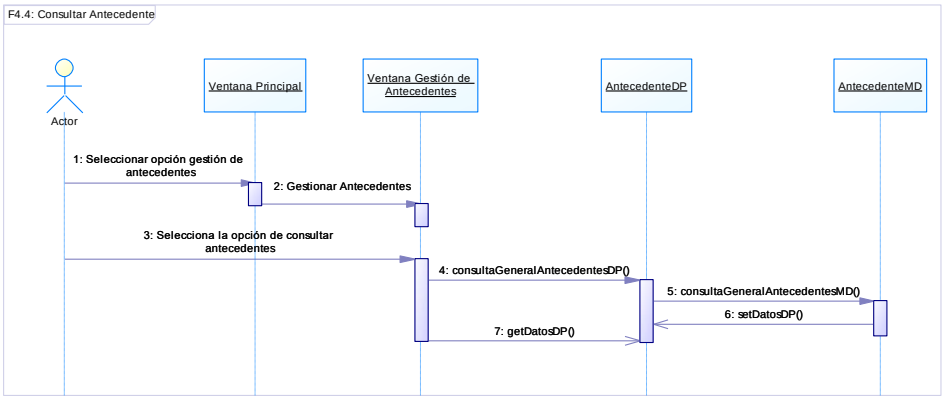

Ilustración 36: Diagrama de Secuencia - Funcionalidad 4.3: Consultar Antecedentes del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.1.10. FUNCIONALIDAD 5.1 – INGRESAR FICHA

El CRUD para gestionar una ficha varía dependiendo del tipo de ficha a ingresar, sin embargo ambas son similares por lo que se expondrá únicamente una de estas y la otra se adjuntará en el CD entregable.

Descripción: El actor podrá ingresar una nueva ficha completando un formulario.

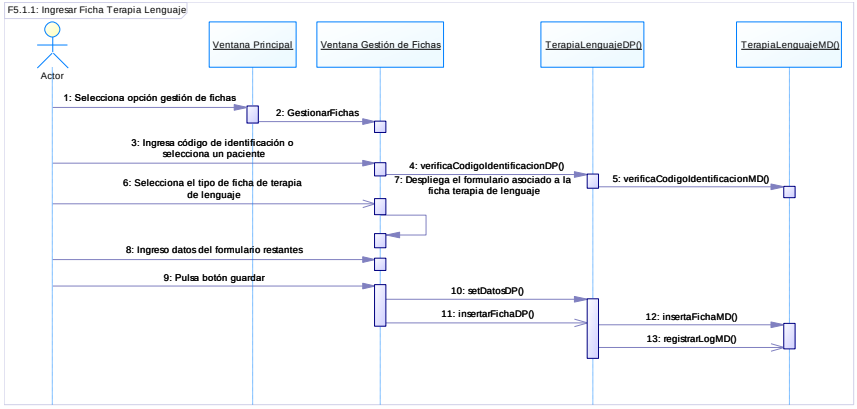

Ilustración 37: Diagrama de Secuencia - Funcionalidad 5.1: Ingresar Ficha del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.1.11. FUNCIONALIDAD 5.2 – EDITAR FICHA

Descripción: El actor podrá editar la información de una ficha de un paciente.

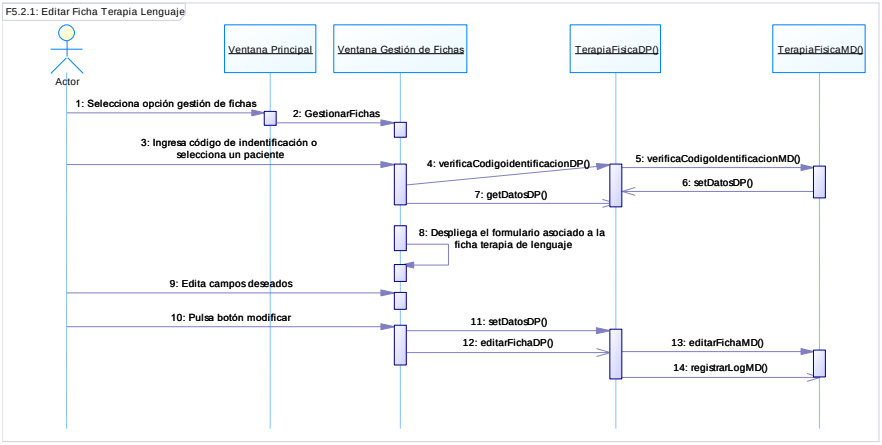

Ilustración 38: Diagrama de Secuencia - Funcionalidad 5.1: Ingresar Ficha del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.1.12. FUNCIONALIDAD 5.3 – ELIMINAR FICHA

Descripción: El actor podrá eliminar fichas de un paciente.

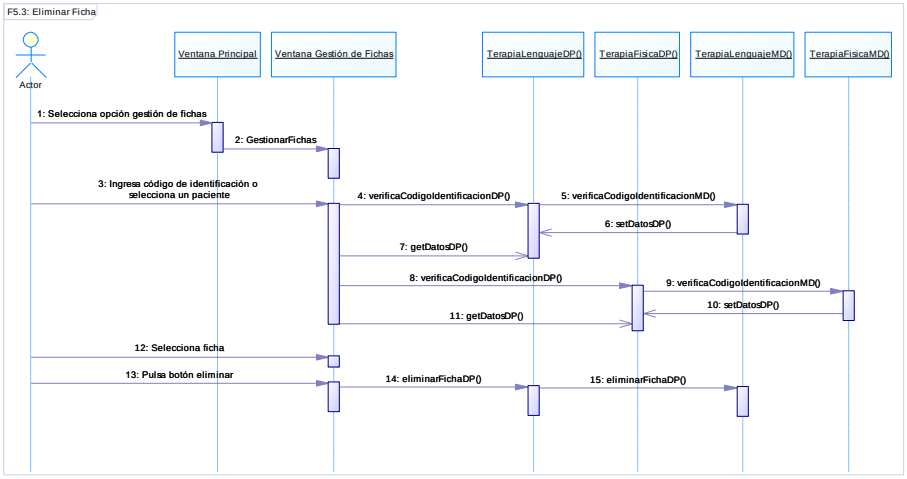

Ilustración 39: Diagrama de Secuencia - Funcionalidad 5.3: Eliminar Ficha del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.1.13. FUNCIONALIDAD 5.4 – CONSULTAR FICHA

Descripción: El actor podrá consultar una lista de todas las fichas ingresadas o de forma detallada mediante el código de identificación del paciente.

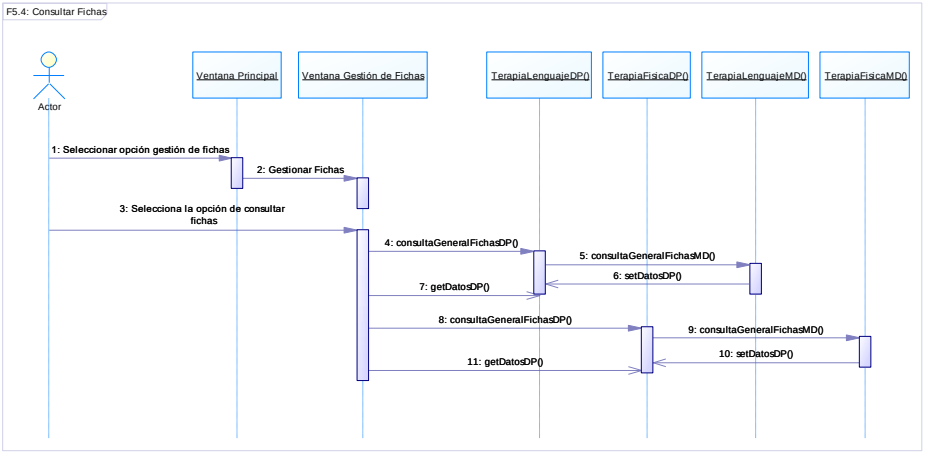

Ilustración 40: Diagrama de Secuencia - Funcionalidad 5.4: Consultar Fichas del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.1.14. FUNCIONALIDAD 6 – GENERAR REPORTES

En el siguiente diagrama se observa la secuencia que se seguirá para obtener un reporte del sistema.

Descripción: Permite generar reportes en base a la recopilación de datos almacenados con la ayuda de nuestro sistema.

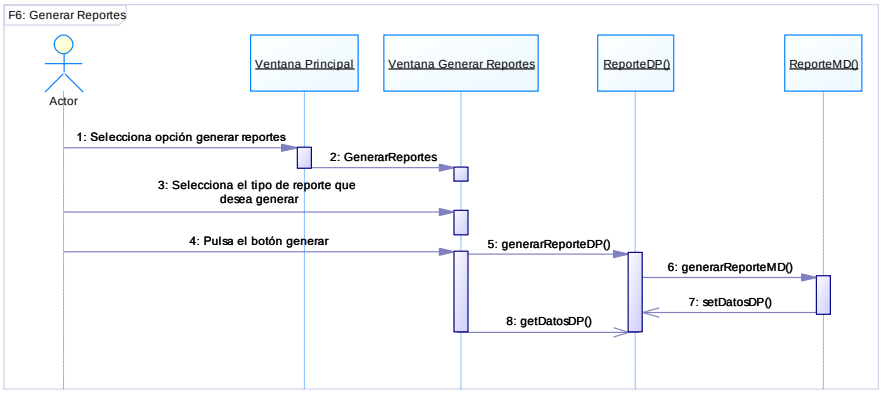

Ilustración 41: Diagrama de Secuencia - Funcionalidad 6: Generar Reportes en el Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.1.15. FUNCIONALIDAD 7 – SALIR DEL SISTEMA

Descripción: Permite cerrar el sistema.

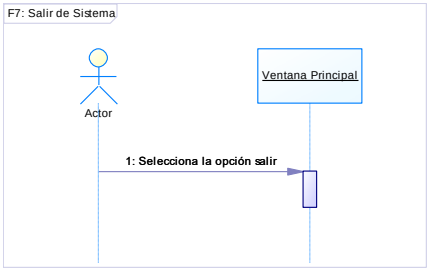

Ilustración 42: Diagrama de Secuencia - Funcionalidad 7: Salir del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.2. DIAGRAMA DE ESTADOS

Ya se han presentado los diagramas de casos de uso y de secuencia los cuales presentan la interacción con el actor y modelan secuencias en acciones entre objetos de un sistema. El diagrama de estado modela el comportamiento dinámico que puede existir por parte de un objeto o una clase de objetos.
Se modela un diagrama de estado para todas las clases de las que se presuma que tengan un comportamiento dinámico. Este tipo de diagrama modela la secuencia de estados por los que pasa un objeto de una clase durante su vida. El diagrama de estado capta los estímulos recibidos por el objeto, las respuestas y las acciones.[35] [36]

Para el caso de nuestro proyecto no es necesario realizar uno o más diagramas de estado dado que el mismo se basa principalmente en la gestión de información mediante CRUDs básicos. Al no existir procesos en los cuales un objeto varíe de estado durante su ciclo de vida entonces no es posible representarlo a detalle mediante un diagrama de estados.

4.1.2.3. DIAGRAMA DE COLABORACIÓN

[35] (IBM, 2017)
[36] (Gutierrez, UML, 2011)

Los diagramas de colaboración o también conocidos como diagramas de comunicación nos permiten observar una perspectiva similar a la que nos brinda el diagrama de secuencia ya que observamos el procedimiento e interacción de los elementos que lo conforman, sin embargo éste se enfoca en las responsabilidades de cada objeto dejando de lado el tiempo en el cual los mensajes se envían entre elementos.[37] [38]

A continuación se presentan los respectivos diagramas de colaboración basados en los diagramas de secuencia previamente presentados.

4.1.2.3.1. FUNCIONALIDAD 0 - INGRESO AL SISTEMA

Descripción: Ingreso al sistema de la empresa FUDRINE, el cual es autenticado y autorizado para la utilización de este. La Ilustración 43 presenta el diagrama de colaboración.

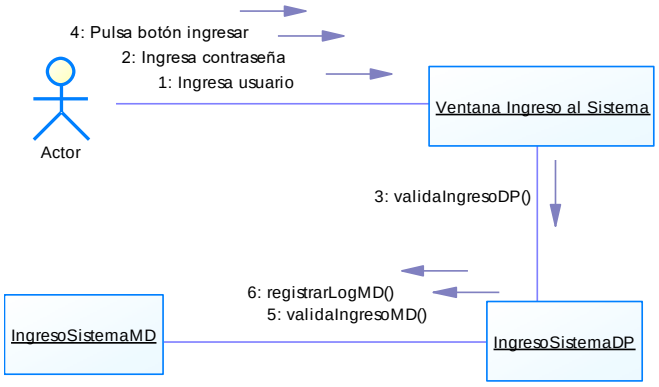

Ilustración 43: Diagrama de Colaboración - Funcionalidad 0: Ingreso al Sistema del Sistema de Gestión de Información (FUDRINE), 2017

[37] (IBM, 2017)
[38] (Mindiola, 2012)

4.1.2.3.2. FUNCIONALIDAD 1.1 – INGRESAR PACIENTE

Descripción: El actor podrá ingresar un nuevo paciente completando un formulario.

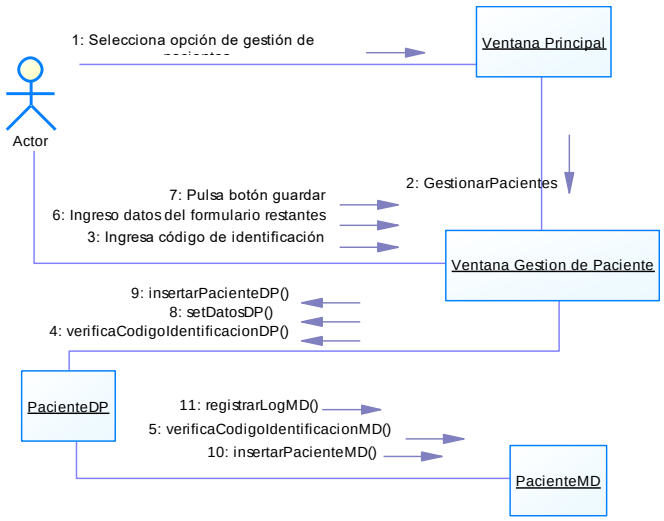

Ilustración 44: Diagrama de Colaboración - Funcionalidad 1.1: Ingresar Paciente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.3.3. FUNCIONALIDAD 1.2 – EDITAR PACIENTE

Descripción: El actor podrá editar la información de un paciente con excepción de su código de identificación.

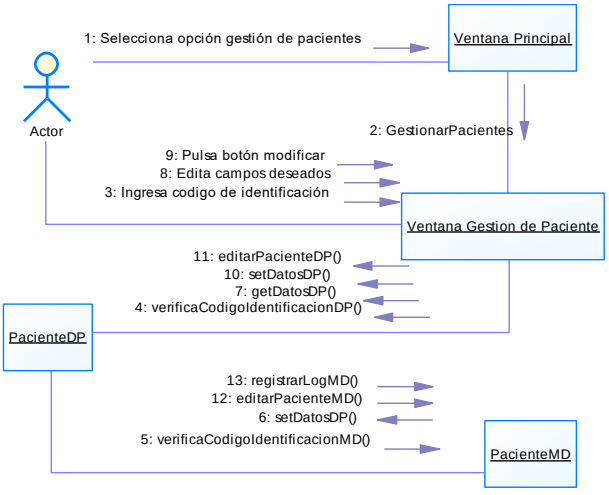

Ilustración 45: Diagrama de Colaboración - Funcionalidad 1.1: Editar Paciente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.3.4. FUNCIONALIDAD 1.3 – ELIMINAR PACIENTE

Descripción: El actor podrá eliminar a un paciente mediante el código de identificación.

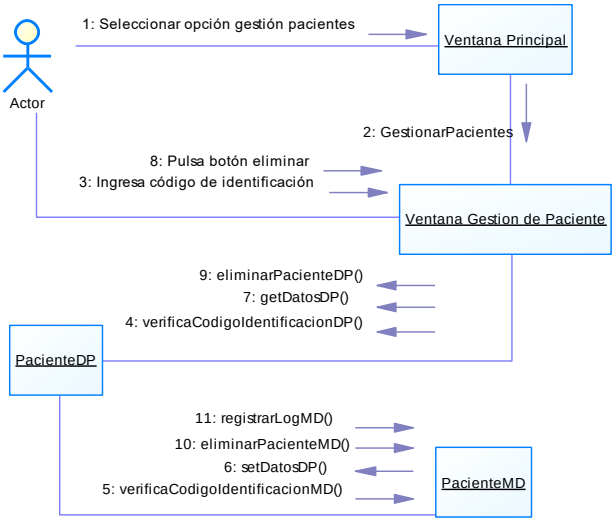

Ilustración 46: Diagrama de Colaboración - Funcionalidad 1.1: Eliminar Paciente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.3.5. FUNCIONALIDAD 1.4.1 – CONSULTAR PACIENTE

Descripción: El actor podrá consultar a los pacientes registrados.

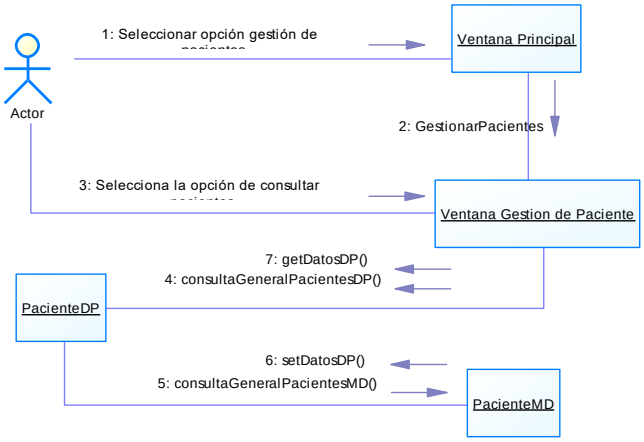

Ilustración 47: Diagrama de Colaboración - Funcionalidad 1.1: Consultar Paciente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.3.6. FUNCIONALIDAD 4.1 – INGRESAR ANTECEDENTE

Descripción: El actor podrá ingresar un nuevo antecedente completando un formulario.

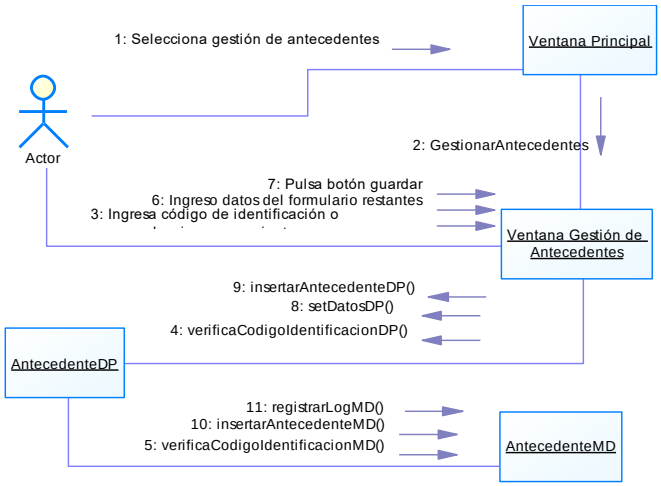

Ilustración 48: Diagrama de Colaboración - Funcionalidad 4.1: Ingresar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.3.7. FUNCIONALIDAD 4.2 – EDITAR ANTECEDENTE

Descripción: El actor podrá editar la información de un antecedente de un paciente.

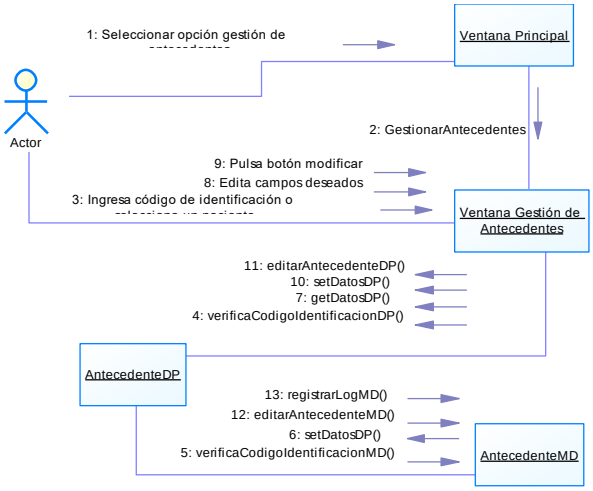

Ilustración 49: Diagrama de Colaboración - Funcionalidad 4.2: Editar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.3.8. FUNCIONALIDAD 4.3 – ELIMINAR ANTECEDENTE

Descripción: El actor podrá consultar una lista de todos los antecedentes ingresados mediante el código de identificación del paciente.

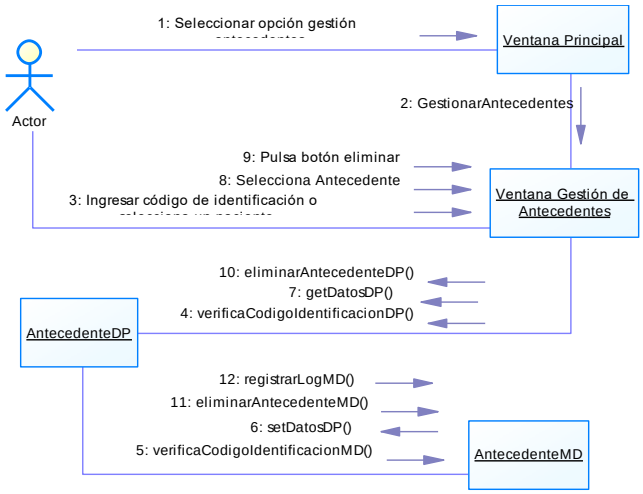

Ilustración 50: Diagrama de Colaboración - Funcionalidad 4.2: Editar Antecedente del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.3.9. FUNCIONALIDAD 4.4.1 – CONSULTAR ANTECEDENTE

Descripción: El actor podrá consultar una lista de todos los antecedentes ingresados mediante el código de identificación del paciente.

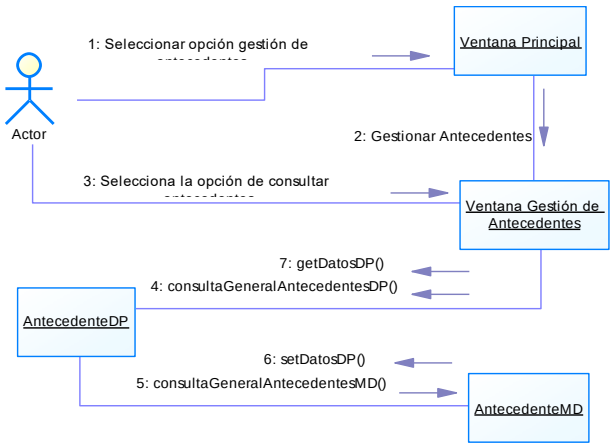

Ilustración 51: Diagrama de Colaboración - Funcionalidad 4.3: Consultar Antecedentes del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.3.10. FUNCIONALIDAD 5.1 – INGRESAR FICHA

Descripción: El actor podrá ingresar una nueva ficha completando un formulario.

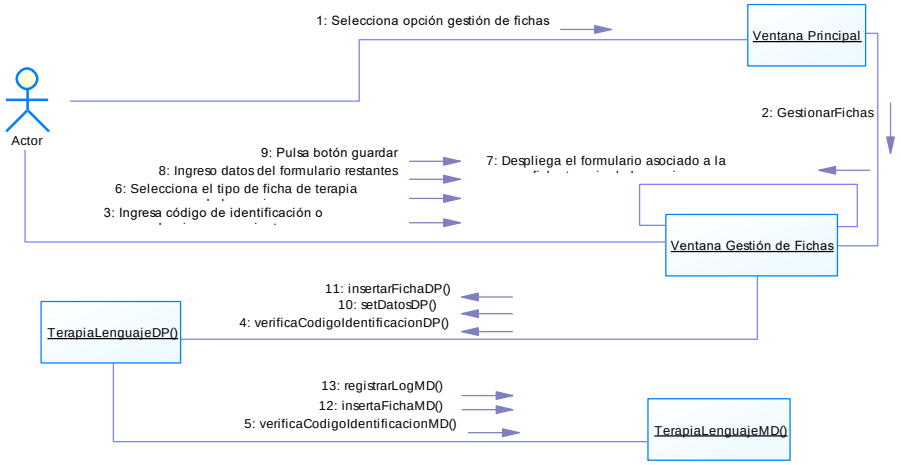

Ilustración 52: Diagrama de Colaboración - Funcionalidad 5.1: Ingresar Ficha del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.3.11. FUNCIONALIDAD 5.2 – EDITAR FICHA

Descripción: El actor podrá editar la información de una ficha de un paciente.

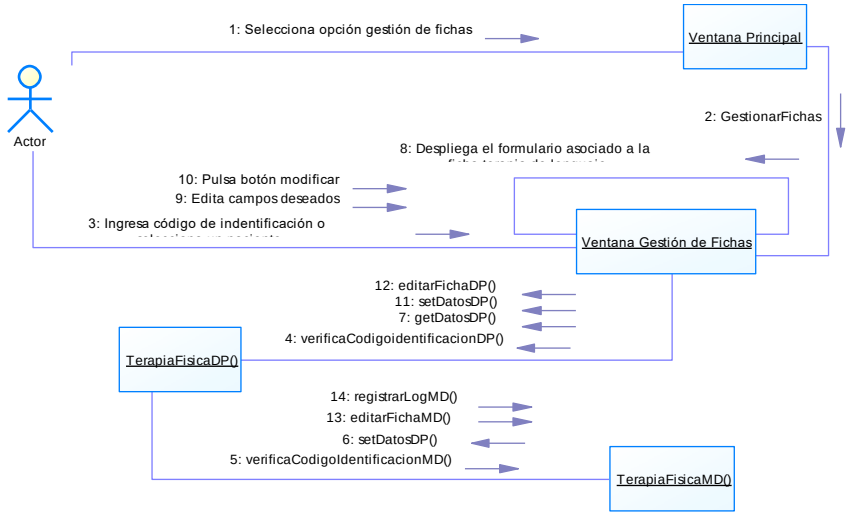

Ilustración 53: Diagrama de Colaboración - Funcionalidad 5.1: Ingresar Ficha del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.3.12. FUNCIONALIDAD 5.3 – ELIMINAR FICHA

Descripción: El actor podrá eliminar fichas de un paciente.

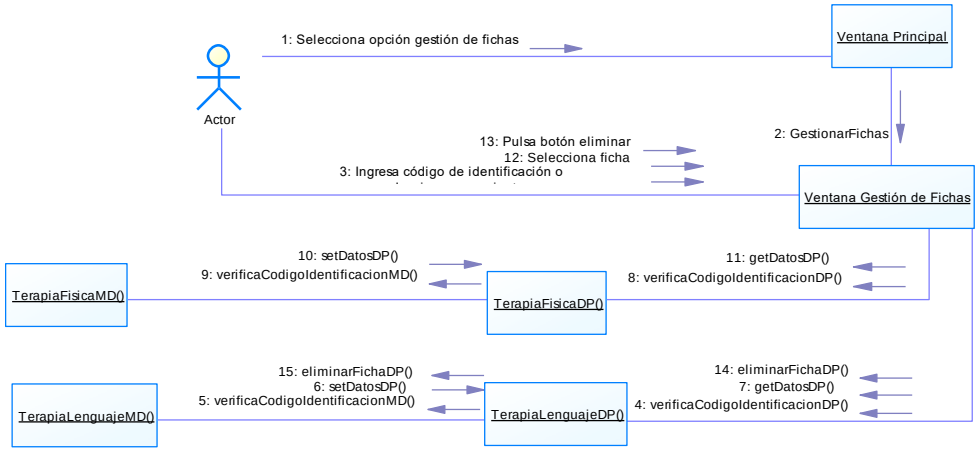

Ilustración 54: Diagrama de Colaboración - Funcionalidad 5.3: Eliminar Ficha del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.3.13. FUNCIONALIDAD 5.4 – CONSULTAR FICHA

Descripción: El actor podrá consultar una lista de todas las fichas ingresadas o de forma detallada mediante el código de identificación del paciente.

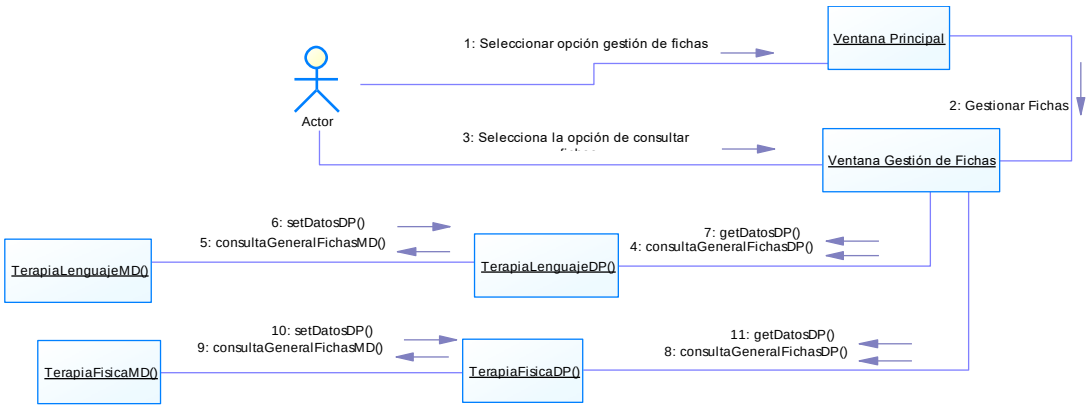

Ilustración 55: Diagrama de Colaboración - Funcionalidad 5.4: Consultar Fichas del Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.3.14. FUNCIONALIDAD 6 – GENERAR REPORTES

Descripción: Permite generar reportes en base a la recopilación de datos almacenados con la ayuda de nuestro sistema.

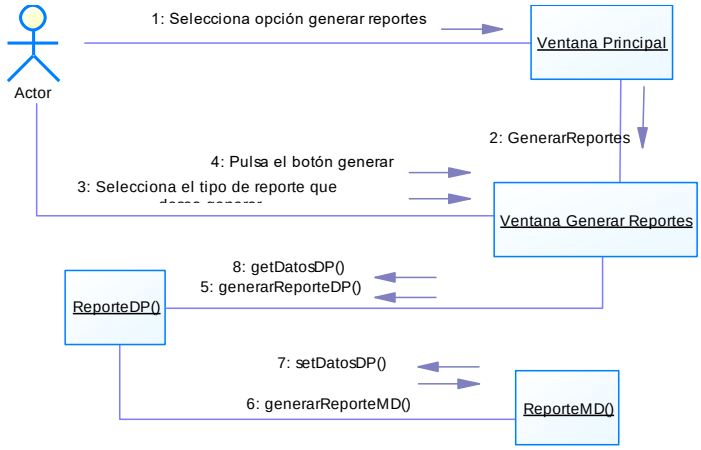

Ilustración 56: Diagrama de Colaboración - Funcionalidad 6: Generar Reportes en el Sistema de Gestión de Información (FUDRINE), 2017

4.1.2.3.15. FUNCIONALIDAD 7 – SALIR DEL SISTEMA

Descripción: Permite cerrar el sistema.

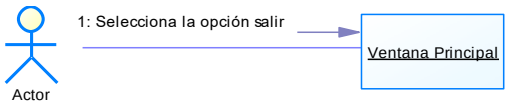

Ilustración 57: Diagrama de Secuencia - Funcionalidad 7: Salir del Sistema de Gestión de Información (FUDRINE), 2017

4.2. DISEÑO DE INTERFACES

Para desarrollar nuestro proyecto es necesario partir de una estructura base, es por este motivo que es de suma importancia crear prototipos para nuestra interfaz. Mediante los prototipos el usuario interactúa con el desarrollador y coordinan la mejor organización y funcionamiento. De esta forma al momento en el que el usuario utilice la aplicación le

será sencillo familiarizarse a la misma y su vez al desarrollador le facilitará el trabajo el tener un plano guía para construir la interfaz de la aplicación.[39][40]

A continuación se presenta los prototipos de la interfaz diseñada para cada uno de las ventanas del software.

4.2.1. FUNCIONALIDAD 0 - INGRESO AL SISTEMA

La Ilustración 58 representa un prototipo de la ventana que el usuario observará al ingresar al programa.

Ilustración 58: Diseño de Interfaz - Funcionalidad 0: Ingreso al Sistema de Gestión de Información (FUDRINE), 2017

4.2.2. FUNCIONALIDAD 1 – GESTIONAR PACIENTES

La Ilustración 59 representa un prototipo de la ventana en la cual el usuario podrá gestionar pacientes.

[39] (Sommerville, Ingeniería del Software, 2006)
[40] (IBM, 2017)

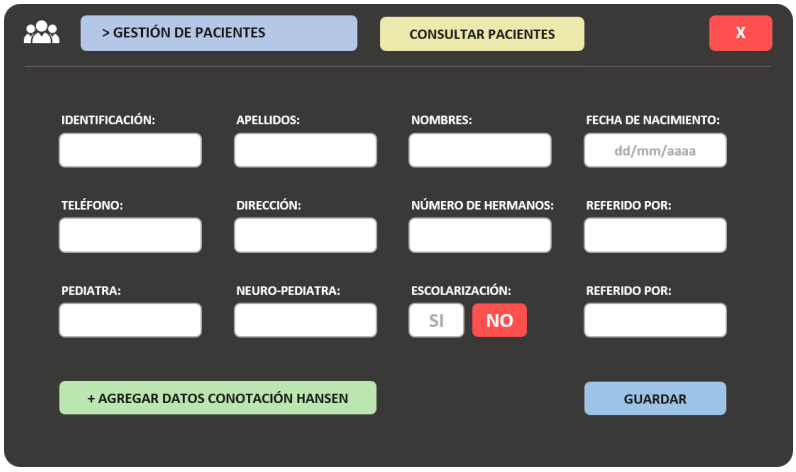

Ilustración 59: Diseño de Interfaz - Funcionalidad 1: Gestión de Pacientes en el Sistema de Gestión de Información (FUDRINE), 2017

4.2.3. FUNCIONALIDAD 2 – GESTIONAR EMPLEADOS

La Ilustración 60 representa un prototipo de la ventana en la cual el usuario podrá gestionar empleados.

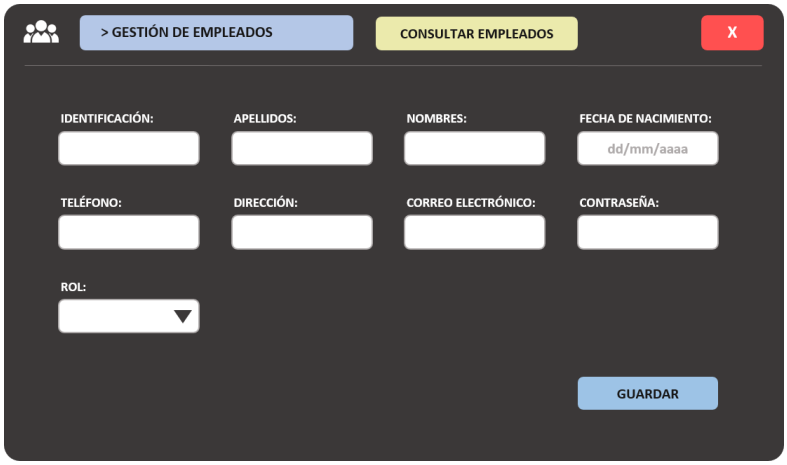

Ilustración 60: Diseño de Interfaz - Funcionalidad 2: Gestión de Empleados en el Sistema de Gestión de Información (FUDRINE), 2017

4.2.4. FUNCIONALIDAD 3 – GESTIONAR REPRESENTANTES

La Ilustración 61 representa un prototipo de la ventana en la cual el usuario podrá gestionar representantes de pacientes.

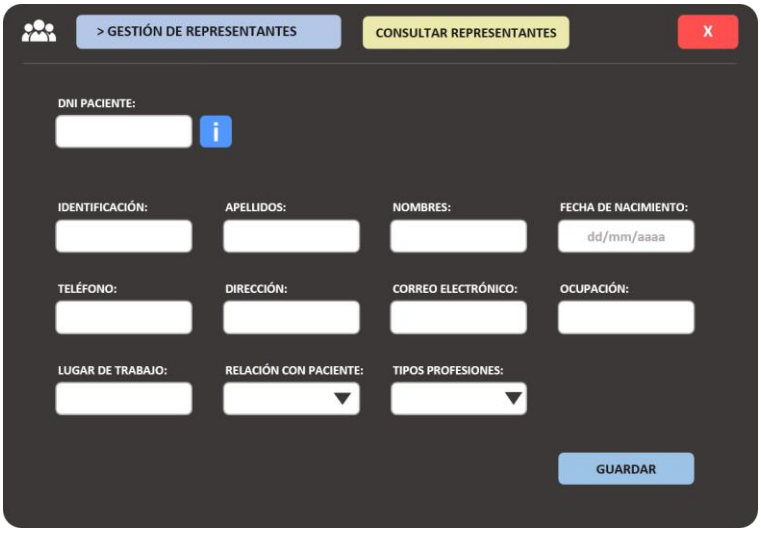

Ilustración 61: Diseño de Interfaz - Funcionalidad 3: Gestión de Representantes en el Sistema de Gestión de Información (FUDRINE), 2017

4.2.5. FUNCIONALIDAD 4 – GESTIONAR ANTECEDENTES

La Ilustración 62 representa un prototipo de la ventana en la cual el usuario podrá gestionar antecedentes de pacientes.

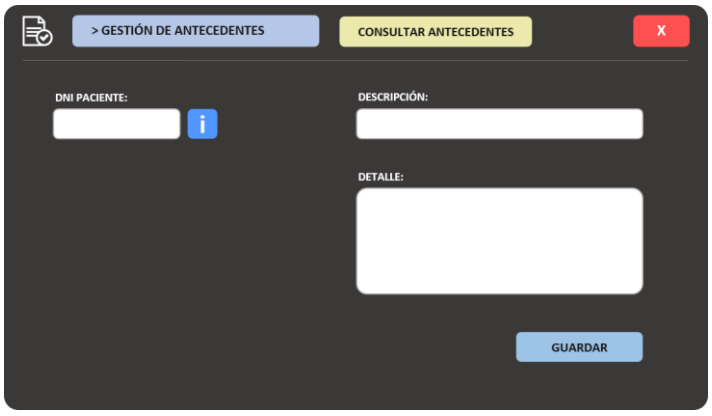

Ilustración 62: Diseño de Interfaz - Funcionalidad 4: Gestión de Antecedentes en el Sistema de Gestión de Información (FUDRINE), 2017

4.2.6. FUNCIONALIDAD 5 – GESTIONAR FICHAS

La Ilustración 63 representa un prototipo de la ventana en la cual el usuario podrá gestionar fichas de pacientes.

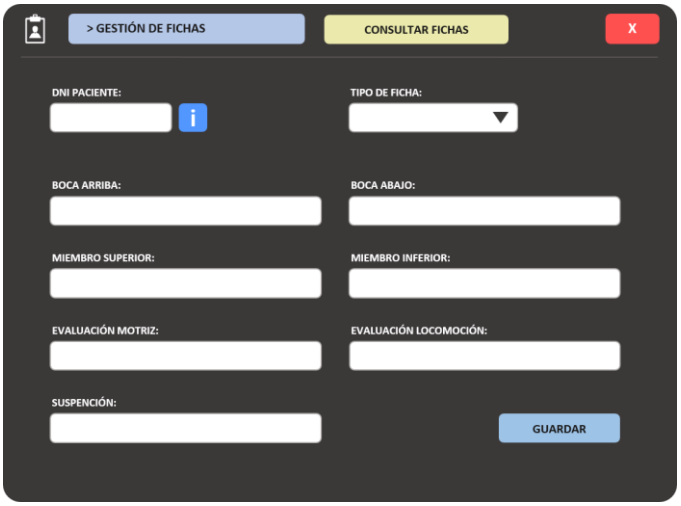

Ilustración 63: Diseño de Interfaz - Funcionalidad 5: Gestión de Fichas en el Sistema de Gestión de Información (FUDRINE), 2017

4.2.7. FUNCIONALIDAD 6 – GENERAR REPORTES

La Ilustración 64 representa un prototipo de la ventana en la cual el usuario podrá generar reportes.

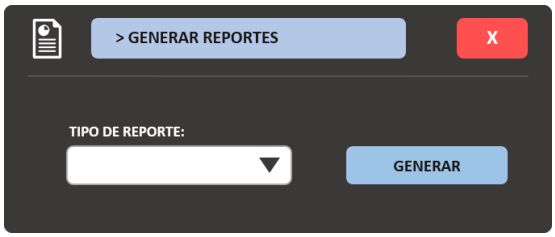

Ilustración 64: Diseño de Interfaz - Funcionalidad 6: Generar Reportes en el Sistema de Gestión de Información (FUDRINE), 2017

4.2.8. FUNCIONALIDAD 7 – SALIR DEL SISTEMA

La Ilustración 65 representa un prototipo de la ventana en la cual el usuario podrá acceder a cualquier ventana y a su vez encontramos el botón por el cual puede salir del programa.

Ilustración 65: Diseño de Interfaz - Funcionalidad 7: Salir del Sistema de Gestión de Información (FUDRINE), 2017

4.3. DESARROLLO DEL PLAN DE PRUEBAS

El objetivo principal de un plan de pruebas para nuestro software a desarrollar es verificar que nos encontramos cubriendo con los objetivos de calidad en un desarrollo de sistemas, cumpliendo con módulos o funcionalidades, verificaciones, entornos, entre otros aspectos. De esta forma se realiza el proceso de validación y verificación de los requerimientos funcionales y no funcionales del sistema.[41]

A continuación se presentan tablas las cuales servirán de plantillas para realizar nuestro plan de pruebas del sistema. Encontraremos una tabla por funcionalidad del sistema, completando un total de 7 tablas. Las mismas serán completadas en el siguiente capítulo, una vez el software se encuentre listo para salir a producción.

[41] (Vanesa Carolina, 2010)

Dado que para todos los CRUDs se verifican que cumplan con las mismas funcionalidades se presentará únicamente tablas de plan de pruebas de las funcionalidades que contengan campos con bastante diferencia, sin embargo en el CD adjunto será posible encontrar todas estas de forma completa.

4.3.1. FUNCIONALIDAD 0 - INGRESO AL SISTEMA

Entradas	Resultados Esperados	C.U Estudio	Resultado
Presionar acceso directo del programa	Inicia el programa y despliega ventana	F0	
Ingresar datos	Ocultar contraseña al escribir la misma	F0	
Presiona botón ingresar	Valida datos y despliega ventana principal y opciones según el rol del usuario	F0	

4.3.2. FUNCIONALIDAD 1 - GESTIONAR PACIENTES

Entradas	Resultados Esperados	C.U Estudio	Resultado
Seleccionar opción gestión de pacientes	Desplegar ventana para la gestión de pacientes	F1	
Ingresa código de identificación	Valida código de identificación y despliega posibles opciones	F1	
Ingresa datos de formulario	Verifica que los datos sean permitidos para cada campo	F1	

Presiona botón guardar	Almacena datos en la base de datos	F1.1	
Presiona botón modificar	Edita registro de la base de datos con la identificación ingresada	F1.2	
Presiona botón eliminar	Elimina registro de la base de datos con la identificación ingresada	F1.3	
Presiona consultar pacientes	Despliega pantalla para consultar pacientes	F1.4	
Selecciona el método por el que desea consultar	Despliega resultados encontrados	F1.4	

4.3.3. FUNCIONALIDAD 6 - GENERAR REPORTES

Entradas	Resultados Esperados	C.U Estudio	Resultado
Seleccionar opción generar reportes	Desplegar ventana para generar reportes	F6	
Seleccionar tipo de reporte e ingresar datos según el tipo	Consultar en base de datos y desplegar campos	F6	

4.3.4. FUNCIONALIDAD 7 – SALIR DEL SISTEMA

Entradas	Resultados Esperados	C.U Estudio	Resultado
Seleccionar opción salir	Cierra el sistema y termina el proceso.	F7	

5. CONSTRUCCIÓN Y TRANSICIÓN

En el capítulo número cinco se inicia con la construcción del sistema, a su vez se presentan requisitos necesarios para finalizar el proyecto de forma correcta. Una vez finalizado el desarrollo se inicia el proceso de pruebas donde será posible comprobar que el sistema cumpla con los requerimientos señalados.

5.1. ESPECIFICACIONES DE REQUISITOS FALTANTES

Durante el tiempo de desarrollo de software es probable que aparezcan nuevos requisitos por parte del cliente. De esta forma el sistema puede ir variando constantemente hasta lograr obtener los resultados deseados por el mismo.

Al iniciar el sistema de gestión de información de la fundación FUDRINE se construyó en base al plan previamente establecido en el diagrama de casos de uso, diagrama de secuencia y modelo entidad-relación. Con el transcurso del proceso de desarrollo fue requerido agregar, modificar o eliminar ciertas características, como son:

- CRUDs complementarios para las funcionalidades principales, entre estos están: la evolución del paciente, inclusión de datos relacionados a la Cotación de Hansen, entre otros.
- Por otra parte por motivos de tiempo la fundación se encontró de acuerdo en retirar el historial de registro de acciones momentáneamente, sin dejar de lado la posibilidad de agregarlo a futuro.
- Las interfaces tuvieron únicamente pequeñas diferencias frente a los diseños establecidos en los prototipos, en cuanto a este aspecto no hubo mayores cambios.
- La base de datos sufrió de pequeños cambios en cuanto a condiciones para ciertos campos. Por ejemplo: campos que permitan mantenerlos

en null, campos denominados como únicos retirando la posibilidad de que se repitan los mismos en varios registros, entre otros.

- Posibilidad de imprimir registros.

Estos cambios se realizaron con el único propósito de complacer al cliente, brindado software de calidad apegado a los requerimientos del cliente.

5.2. DESARROLLO DE DIAGRAMAS DE CASOS DE USO

A continuación se expone el procedimiento que se empleó para la construcción del sistema en base a los casos de uso previamente mencionados en el diagrama del mismo.

Cada funcionalidad fue desarrollada en tres capas como se estableció en un inicio. En la Ilustración 66 podemos observar la estructura de nuestro proyecto agrupado en tres paquetes principales: DP, GUI y MD.

Ilustración 66: Estructura del Sistema (FUDRINE), 2017

Dentro de cada paquete se encuentran clases que cumplen la función asignada de acuerdo a la capa en la que están. En la Ilustración 67 es posible observar las distintas clases de cada paquete.

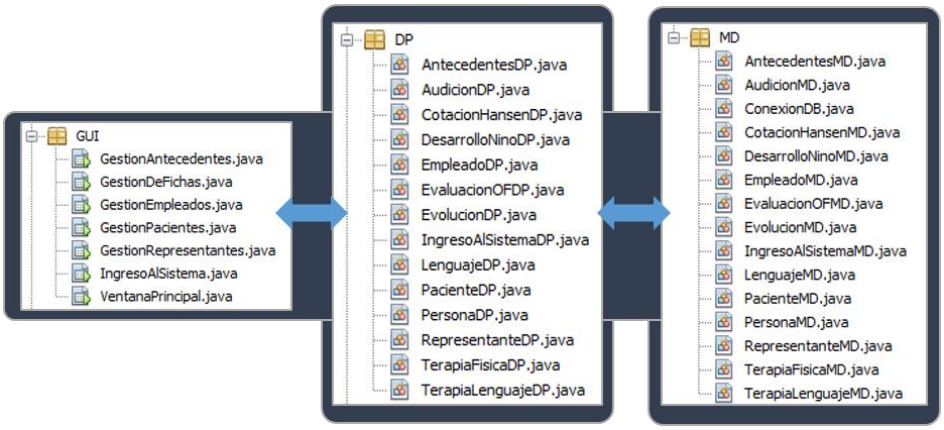

Ilustración 67: Clases de los Paquetes del Sistema (FUDRINE), 2017

La interacción de las clases podremos observarla mediante un pequeño ejemplo con el CRUD de Gestión de Empleado.

En las Ilustraciones 68 y 69 se muestra la interfaz construida en el paquete GUI. Esta se encarga de tener relación con el usuario o actor asociado a dicha funcionalidad. El resultado en este caso es el siguiente:

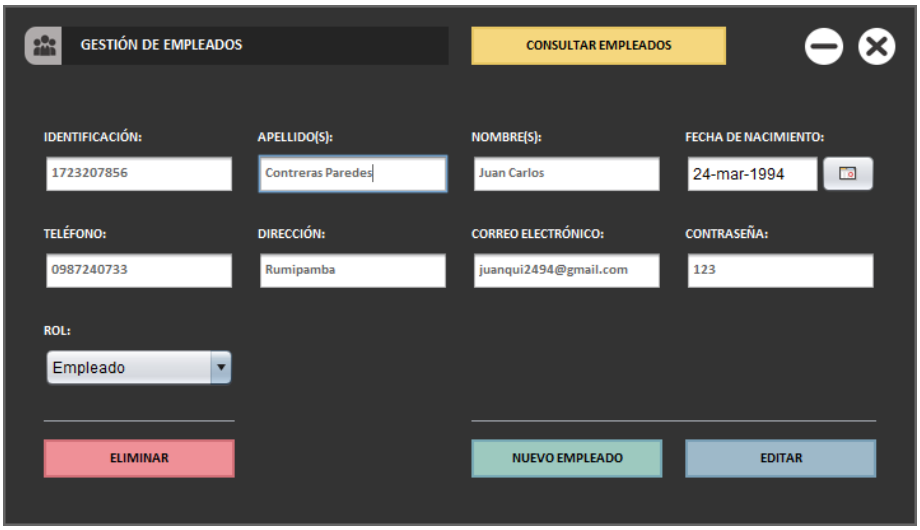

Ilustración 68: Interfaz – Funcionalidad 2 Gestionar Empleados (FUDRINE), 2017

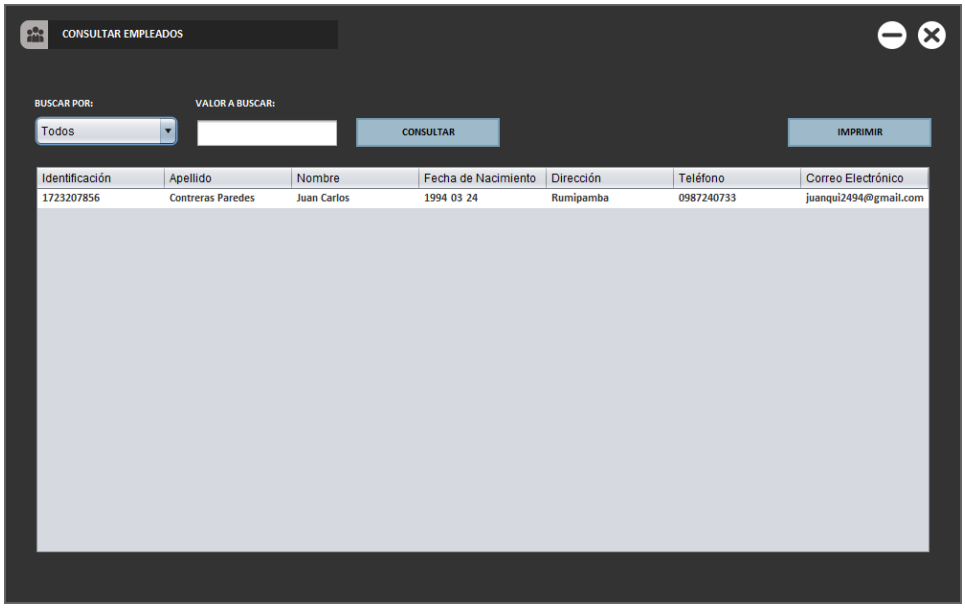

Ilustración 69: Interfaz – Funcionalidad 2 Gestionar Empleados - Consulta (FUDRINE), 2017

A continuación se presentan ciertos fragmentos de código que son muy utilizados en la mayoría de clases del paquete GUI:

- Con la ayuda de la función “setParametrosIniciales” presentada a continuación es posible ubicar nuestra interfaz en el centro de la pantalla en base a la resolución de la misma.

```
public void setParametrosIniciales(){
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        this.setLocation(screenSize.width/2-this.getSize().width/2,
screenSize.height/2-this.getSize().height/2);
    }
```

- La función “imprimirEmpleados” nos permite desplegar una interfaz para realizar la impresión de los registros obtenidos en una determinada consulta.

```
public void imprimirEmpleados(JTable jTable, String header, String footer, boolean showPrintDialog){
      boolean fitWidth = true;
      boolean interactive = true;
      JTable.PrintMode mode = fitWidth ? JTable.PrintMode.FIT_WIDTH : JTable.PrintMode.NORMAL;
      try {
         boolean complete = jTable.print(mode, new MessageFormat(header), new MessageFormat(footer),
showPrintDialog, null, interactive);
         if (complete) {
            JOptionPane.showMessageDialog(jTable,"Impresión Finalizada", "Información",
JOptionPane.INFORMATION_MESSAGE);
         } else {
            JOptionPane.showMessageDialog(jTable, "Impresión Cancelada", "Advertencia",
JOptionPane.WARNING_MESSAGE);
         }
      } catch (PrinterException pe) {
         JOptionPane.showMessageDialog(jTable, "Fallo de Impresión: " + pe.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
      }
   }
```

- En la función "b_GuardarMouseReleased(java.awt.event.MouseEvent evt)", que actúa al realizar el evento de presionar el botón, nos muestra la interacción que existe entre nuestra capa de interfaz de usuario y la capa de negocio para realizar una inserción de datos en la base de datos.

```
    private void b_GuardarMouseReleased(java.awt.event.MouseEvent evt) {
    empleadoDP.setIdentificacion_Persona(tf_Identificacion.getText());
            empleadoDP.setNombre_Persona(tf_Nombres.getText());
            empleadoDP.setApellido_Persona(tf_Apellidos.getText());
            empleadoDP.setFechaNacimiento_Persona(new
    java.sql.Date(dc_FechaNacimiento.getDate().getTime()));
            empleadoDP.setDireccion_Empleado(tf_Direccion.getText());
            empleadoDP.setTelefono_Persona(tf_Telefono.getText());
            empleadoDP.setEmail_Empleado(tf_CorreoElectronico.getText());
            empleadoDP.setContrasena_Empleado(tf_Pw.getText());
            empleadoDP.setRol_Empleado(cb_Rol.getSelectedIndex());

            if(b_Guardar.getText().compareTo("GUARDAR")==0){
                if(empleadoDP.ingresarEmpleado()){
                    JOptionPane.showMessageDialog(null, "El empleado
    "+tf_Apellidos.getText()+" "+tf_Nombres.getText()+" ha sido ingresado
    correctamente.", " Información",JOptionPane.INFORMATION_MESSAGE);
                }
                else{
                    JOptionPane.showMessageDialog(null, "El empleado no se ha guardado
    correctamente.", " Error",JOptionPane.ERROR_MESSAGE);
                }
            }
            else{
                if(empleadoDP.editarEmpleado()){
                    JOptionPane.showMessageDialog(null, "El empleado
    "+tf_Apellidos.getText()+" "+tf_Nombres.getText()+" ha sido editado
    correctamente.", " Información",JOptionPane.INFORMATION_MESSAGE);
                }
                else{
                    JOptionPane.showMessageDialog(null, "El empleado no se ha guardado
    correctamente.", " Error",JOptionPane.ERROR_MESSAGE);
                }
            }
        }
```

En nuestra capa de negocio que se encuentra en el paquete DP del proyecto almacenamos la estructura de nuestras clases que nos sirven para crear objetos con atributos y funciones deseados. Esta capa nos sirve de puente entre el usuario (GUI) y la manipulación de datos (MD).

A continuación se presenta la clase destinada para construir un objeto de tipo EmpleadoDP. La mayoría de clases se encuentran construidas de forma muy parecida, de esta forma con la ayuda del ejemplo a continuación es posible formar una idea de la construcción de las distintas clases. El código por completo se encuentra adjuntado al presente proyecto de disertación.

```
package DP;

import MD.EmpleadoMD;
import java.sql.Date;
import javax.swing.table.DefaultTableModel;

public class EmpleadoDP extends PersonaDP {

    private String direccion_Empleado;
    private String email_Empleado;
    private String contrasena_Empleado;
    private int rol_Empleado;

    private DefaultTableModel lista_Empleado;

    private EmpleadoMD empleadoMD;

    public EmpleadoDP(String direccion_Empleado, String email_Empleado, String
contrasena_Empleado, int rol_Empleado, String identificacion_Persona, String
nombre_Persona, String apellido_Persona, Date fechaNacimiento_Persona, String
telefono_Persona) {
        super(identificacion_Persona, nombre_Persona, apellido_Persona,
fechaNacimiento_Persona, telefono_Persona);
        this.direccion_Empleado = direccion_Empleado;
        this.email_Empleado = email_Empleado;
        this.contrasena_Empleado = contrasena_Empleado;
        this.rol_Empleado = rol_Empleado;

        this.empleadoMD = new EmpleadoMD(this);
    }

    public String getDireccion_Empleado() {
        return direccion_Empleado;
    }

    public void setDireccion_Empleado(String direccion_Empleado) {
        this.direccion_Empleado = direccion_Empleado;
    }

    public String getEmail_Empleado() {
        return email_Empleado;
    }

    public void setEmail_Empleado(String email_Empleado) {
        this.email_Empleado = email_Empleado;
    }
```

```
        public String getContrasena_Empleado() {
            return contrasena_Empleado;
        }

        public void setContrasena_Empleado(String contrasena_Empleado) {
            this.contrasena_Empleado = contrasena_Empleado;
        }

        public int getRol_Empleado() {
            return rol_Empleado;
        }

        public void setRol_Empleado(int rol_Empleado) {
            this.rol_Empleado = rol_Empleado;
        }

        public DefaultTableModel getLista_Empleado() {
            return lista_Empleado;
        }

        public void setLista_Empleado(DefaultTableModel lista_Empleado) {
            this.lista_Empleado = lista_Empleado;
        }

        public boolean getDatosPorIdentificacion(){
            return empleadoMD.getDatosPorIdentificacion();
        }

        public boolean ingresarEmpleado(){
            return empleadoMD.ingresarEmpleado();
        }

        public boolean editarEmpleado(){
            return empleadoMD.editarEmpleado();
        }

        public boolean eliminarEmpleado(){
            return empleadoMD.eliminarEmpleado();
        }

        public boolean consultarEmpleado(int tipoDeBusqueda, String
    parametroABuscar){
            return empleadoMD.consultarEmpleado(tipoDeBusqueda, parametroABuscar);
        }
    }
```

Como se observa, la clase EmpleadoDP hereda atributos, funciones y una estructura de su clase padre “PersonaDP”. Entre sus atributos encontramos a un objeto de tipo EmpleadoMD el cual permite realizar la conexión entre la capa de negocio y la capa de acceso a datos. A su vez contamos con algunas funciones encargadas de dicha relación.

Ahora se presenta la capa de acceso a datos “EmpleadoMD” en la cual observaremos algunas funciones de un CRUD básico.

- Ingreso de empleados:

```
    public boolean ingresarEmpleado(){
            try {
                PreparedStatement script =
    this.conexionDB.getConnection().prepareStatement("INSERT INTO empleados
```

```
    (IDENTIFICACION_EMPLEADO,NOMBRE_EMPLEADO,APELLIDO_EMPLEADO,FECHANACIMIENTO_EMPLEADO
    ,DIRECCION_EMPLEADO,TELEFONO_EMPLEADO,EMAIL_EMPLEADO,CONTRASENA_EMPLEADO,ROL_EMPLEA
    DO) VALUES(?,?,?,?,?,?,?,?,?)");
                script.setString(1, this.empleadoDP.getIdentificacion_Persona());
                script.setString(2, this.empleadoDP.getNombre_Persona());
                script.setString(3, this.empleadoDP.getApellido_Persona());
                script.setDate(4, this.empleadoDP.getFechaNacimiento_Persona() );
                script.setString(5, this.empleadoDP.getDireccion_Empleado());
                script.setString(6, this.empleadoDP.getTelefono_Persona());
                script.setString(7, this.empleadoDP.getEmail_Empleado());
                script.setString(8, this.empleadoDP.getContrasena_Empleado());
                script.setInt(9, this.empleadoDP.getRol_Empleado());

                script.executeUpdate();
                script.close();
                return true;

            } catch (SQLException ex) {
                Logger.getLogger(EmpleadoMD.class.getName()).log(Level.SEVERE, null,
    ex);
                return false;
            }
        }
```

- Edición de empleados:

```
    public boolean editarEmpleado(){
            try {
                PreparedStatement script =
    this.conexionDB.getConnection().prepareStatement("UPDATE empleados SET
    IDENTIFICACION_EMPLEADO=?,NOMBRE_EMPLEADO=?,APELLIDO_EMPLEADO=?,FECHANACIMIENTO_EMP
    LEADO=?,DIRECCION_EMPLEADO=?,TELEFONO_EMPLEADO=?,EMAIL_EMPLEADO=?,CONTRASENA_EMPLEA
    DO=?,ROL_EMPLEADO=?");

                script.setString(1, this.empleadoDP.getIdentificacion_Persona());
                script.setString(2, this.empleadoDP.getNombre_Persona());
                script.setString(3, this.empleadoDP.getApellido_Persona());
                script.setDate(4, this.empleadoDP.getFechaNacimiento_Persona() );
                script.setString(5, this.empleadoDP.getDireccion_Empleado());
                script.setString(6, this.empleadoDP.getTelefono_Persona());
                script.setString(7, this.empleadoDP.getEmail_Empleado());
                script.setString(8, this.empleadoDP.getContrasena_Empleado());
                script.setInt(9, this.empleadoDP.getRol_Empleado());

                script.executeUpdate();
                script.close();
                return true;
            } catch (SQLException ex) {
                Logger.getLogger(EmpleadoMD.class.getName()).log(Level.SEVERE, null,
    ex);
                return false;
            }
        }
```

- Eliminar de empleados:

```
public boolean eliminarEmpleado(){
        try {
            PreparedStatement script =
this.conexionDB.getConnection().prepareStatement("DELETE FROM empleados WHERE
IDENTIFICACION_EMPLEADO = '"+this.empleadoDP.getIdentificacion_Persona()+"'");
            script.executeUpdate();
            script.close();
            return true;

        } catch (SQLException ex) {
            Logger.getLogger(EmpleadoMD.class.getName()).log(Level.SEVERE,
null, ex);
            return false;
        }
    }
```

- Consulta de empleados:

```
public boolean consultarEmpleado(int tipoDeBusqueda, String parametroABuscar){
        try {
            DefaultTableModel resultadoEmpleados = new DefaultTableModel();
            Object[] filas;

            resultadoEmpleados.addColumn("Identificación");
            resultadoEmpleados.addColumn("Apellido");
            resultadoEmpleados.addColumn("Nombre");
            resultadoEmpleados.addColumn("Fecha de Nacimiento");
            resultadoEmpleados.addColumn("Dirección");
            resultadoEmpleados.addColumn("Teléfono");
            resultadoEmpleados.addColumn("Correo Electrónico");

            filas = new Object[resultadoEmpleados.getColumnCount()];

            PreparedStatement script = null;

            if(tipoDeBusqueda == 0)
                script = this.conexionDB.getConnection().prepareStatement("SELECT
IDENTIFICACION_EMPLEADO,APELLIDO_EMPLEADO,NOMBRE_EMPLEADO,FECHANACIMIENTO_EMPLEADO,
DIRECCION_EMPLEADO,TELEFONO_EMPLEADO,EMAIL_EMPLEADO FROM empleados");
            else if(tipoDeBusqueda == 1)
                script = this.conexionDB.getConnection().prepareStatement("SELECT
IDENTIFICACION_EMPLEADO,APELLIDO_EMPLEADO,NOMBRE_EMPLEADO,FECHANACIMIENTO_EMPLEADO,
DIRECCION_EMPLEADO,TELEFONO_EMPLEADO,EMAIL_EMPLEADO FROM empleados WHERE
APELLIDO_EMPLEADO LIKE '%"+parametroABuscar+"%'");
            else if(tipoDeBusqueda == 2)
                script = this.conexionDB.getConnection().prepareStatement("SELECT
IDENTIFICACION_EMPLEADO,APELLIDO_EMPLEADO,NOMBRE_EMPLEADO,FECHANACIMIENTO_EMPLEADO,
DIRECCION_EMPLEADO,TELEFONO_EMPLEADO,EMAIL_EMPLEADO FROM empleados WHERE
NOMBRE_EMPLEADO LIKE '%"+parametroABuscar+"%'");
            else if(tipoDeBusqueda == 3)
                script = this.conexionDB.getConnection().prepareStatement("SELECT
IDENTIFICACION_EMPLEADO,APELLIDO_EMPLEADO,NOMBRE_EMPLEADO,FECHANACIMIENTO_EMPLEADO,
DIRECCION_EMPLEADO,TELEFONO_EMPLEADO,EMAIL_EMPLEADO FROM empleados WHERE
DIRECCION_EMPLEADO LIKE '%"+parametroABuscar+"%'");

            script.execute();
```

```
            ResultSet resultado = script.getResultSet();

            if (resultado.isBeforeFirst() ) {
                while (resultado.next()) {
                    for(int i=0;i<resultadoEmpleados.getColumnCount();i++){
                        filas[i] = resultado.getObject(i+1);
                    }
                    resultadoEmpleados.addRow(filas);
                }
                script.close();
            }
            else{
                script.close();
                return false;
            }

            empleadoDP.setLista_Empleado(resultadoEmpleados);
            return true;

        } catch (SQLException ex) {
            Logger.getLogger(EmpleadoMD.class.getName()).log(Level.SEVERE, null,
ex);
            return false;
        }
    }
```

En los fragmentos de código presentados está la estructura en la que se encuentran construidos los CRUDs y funcionalidades de nuestro sistema. Mediante este método se desarrolló las funcionalidades que fueron previamente enseñadas en el diagrama de casos de uso.

5.3. PRUEBAS DEL SISTEMA

Luego de finalizar el desarrollo del sistema se verifica que las funcionalidades se encuentren desarrolladas en su totalidad en base al plan de pruebas previamente planificado.

A continuación se presentan los resultados obtenidos en las pruebas del sistema:

5.3.1. FUNCIONALIDAD 0 - INGRESO AL SISTEMA

Entradas	Resultados Esperados	C.U Estudio	Resultado
Presionar acceso directo del programa	Inicia el programa y despliega ventana	F0	✓

Entradas	Resultados Esperados	C.U Estudio	Resultado
Ingresar datos	Ocultar contraseña al escribir la misma	F0	✓
Presiona botón ingresar	Valida datos y despliega ventana principal y opciones según el rol del usuario	F0	✓

5.3.2. FUNCIONALIDAD 1 - GESTIONAR PACIENTES

Entradas	Resultados Esperados	C.U Estudio	Resultado
Seleccionar opción gestión de pacientes	Desplegar ventana para la gestión de pacientes	F1	✓
Ingresa código de identificación	Valida código de identificación y despliega posibles opciones	F1	✓
Ingresa datos de formulario	Verifica que los datos sean permitidos para cada campo	F1	✓
Presiona botón guardar	Almacena datos en la base de datos	F1.1	✓
Presiona botón modificar	Edita registro de la base de datos con la identificación ingresada	F1.2	✓
Presiona botón eliminar	Elimina registro de la base de datos con la	F1.3	✓

Entradas	Resultados Esperados	C.U Estudio	Resultado
	identificación ingresada		
Presiona consultar pacientes	Despliega pantalla para consultar pacientes	F1.4	✓
Selecciona el método por el que desea consultar	Despliega resultados encontrados	F1.4	✓

5.3.3. FUNCIONALIDAD 2 - GESTIONAR EMLEADOS

Entradas	Resultados Esperados	C.U Estudio	Resultado
Seleccionar opción gestión de empleados	Desplegar ventana para la gestión de empleados	F2	✓
Ingresa código de identificación	Valida código de identificación y despliega posibles opciones	F2	✓
Ingresa datos de formulario	Verifica que los datos sean permitidos para cada campo	F2	✓
Presiona botón guardar	Almacena datos en la base de datos	F2.1	✓
Presiona botón modificar	Edita registro de la base de datos con la identificación ingresada	F2.2	✓
Presiona botón eliminar	Elimina registro de la base de datos con la	F2.3	✓

Entradas	Resultados Esperados	C.U Estudio	Resultado
	identificación ingresada		
Presiona consultar empleados	Despliega pantalla para consultar empleados	F2.4	✓
Selecciona el método por el que desea consultar	Despliega resultados encontrados	F2.4	✓

5.3.4. FUNCIONALIDAD 3 - GESTIONAR REPRESENTANTES

Entradas	Resultados Esperados	C.U Estudio	Resultado
Seleccionar opción gestión de representantes	Desplegar ventana para la gestión de representantes	F3	✓
Ingresa código de identificación	Valida código de identificación y despliega posibles opciones	F3	✓
Ingresa datos de formulario	Verifica que los datos sean permitidos para cada campo	F3	✓
Presiona botón guardar	Almacena datos en la base de datos	F3.1	✓
Presiona botón modificar	Edita registro de la base de datos con la identificación ingresada	F3.2	✓
Presiona botón eliminar	Elimina registro de la base de datos con la	F3.3	✓

Entradas	Resultados Esperados	C.U Estudio	Resultado
	identificación ingresada		
Presiona consultar representantes	Despliega pantalla para consultar representantes	F3.4	✓
Selecciona el método por el que desea consultar	Despliega resultados encontrados	F3.4	✓

5.3.5. FUNCIONALIDAD 4 - GESTIONAR ANTECEDENTES

Entradas	Resultados Esperados	C.U Estudio	Resultado
Seleccionar opción gestión de antecedentes	Desplegar ventana para la gestión de antecedentes	F4	✓
Ingresa código de identificación	Valida código de identificación y despliega posibles opciones	F4	✓
Ingresa datos de formulario	Verifica que los datos sean permitidos para cada campo	F4	✓
Presiona botón guardar	Almacena datos en la base de datos	F4.1	✓
Presiona botón modificar	Edita registro de la base de datos con la identificación ingresada	F4.2	✓
Presiona botón eliminar	Elimina registro de la base de datos con la	F4.3	✓

Entradas	Resultados Esperados	C.U Estudio	Resultado
	identificación ingresada		
Presiona consultar antecedentes	Despliega pantalla para consultar antecedentes	F4.4	✓
Selecciona el método por el que desea consultar	Despliega resultados encontrados	F4.4	✓

5.3.6. FUNCIONALIDAD 5 - GESTIONAR FICHAS

Entradas	Resultados Esperados	C.U Estudio	Resultado
Seleccionar opción gestión de fichas	Desplegar ventana para la gestión de fichas	F5	✓
Ingresa código de identificación	Valida código de identificación y despliega posibles opciones	F5	✓
Selección tipo de ficha	Carga formulario asociado al tipo de ficha.	F5	✓
Ingresa datos de formulario	Verifica que los datos sean permitidos para cada campo	F5	✓
Presiona botón guardar	Almacena datos en la base de datos	F5.1	✓
Presiona botón modificar	Edita registro de la base de datos con la identificación ingresada	F5.2	✓

Entradas	Resultados Esperados	C.U Estudio	Resultado
Presiona botón eliminar	Elimina registro de la base de datos con la identificación ingresada	F5.3	✓
Presiona consultar fichas	Despliega pantalla para consultar fichas	F5.4	✓
Selecciona el método por el que desea consultar	Despliega resultados encontrados	F5.4	✓

5.3.7. FUNCIONALIDAD 6 - GENERAR REPORTES

Entradas	Resultados Esperados	C.U Estudio	Resultado
Seleccionar opción generar reportes	Desplegar ventana para generar reportes	F6	✓
Seleccionar tipo de reporte e ingresar datos según el tipo	Consultar en base de datos y desplegar campos	F6	✓

5.3.8. FUNCIONALIDAD 7 – SALIR DEL SISTEMA

Entradas	Resultados Esperados	C.U Estudio	Resultado
Seleccionar opción salir	Cierra el sistema y termina el proceso.	F7	✓

5.4. PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación son aquellas realizadas por el cliente, en este caso la fundación FUDRINE es la encargada de realizar dichas pruebas y aceptar que el software es funcional y se encuentra construido en base a los requerimientos especificados.

Estas pruebas se realizaron por parte de los miembros de la fundación FUDRINE. La fundación aprobó el sistema después de hacer uso del mismo y verificar sus funcionalidades.

Dado que el proceso de desarrollo de software es iterativo, no hubo mayor problema en cuanto a disgustos o reclamos frente a falta de requerimientos o mal funcionamiento del software.

En el CD se adjunta el documento en el cual la fundación FUDRINE acepta que el software es funcional y cumple con los requisitos solicitados.

6. CONCLUSIONES Y RECOMENDACIONES

Una vez terminado el presente proyecto junto con el desarrollo del sistema de gestión de información para la fundación FUDRINE fue posible obtener las siguientes conclusiones y recomendaciones.

6.1. CONCLUSIONES

- La metodología RUP es un conjunto de excelentes prácticas que facilitan el proceso de desarrollo de un proyecto, esto ocurre gracias a factores como son su proceso iterativo y la documentación del mismo.
- Gracias al proceso iterativo de la metodología RUP es posible reducir el riesgo en forma temprana y continua. A su vez es posible obtener un progreso demostrable y frecuentes versiones ejecutables.

- La fase de iniciación y elaboración son fases primordiales a la hora de seguir documentos estandarizados por parte de la metodología que a su vez son de excelente ayuda al momento de encontrarnos en la fase de construcción del sistema.
- La metodología RUP se puede emplear tanto en proyectos personales como en proyectos en grupo, así como también es posible aplicarlo en proyectos que sean de corta o larga duración.
- IBM nos proporciona una serie de documentos que se encuentran muy bien estructurados que sirven de soporte para iniciar nuestro proyecto con la metodología RUP.
- El sistema de gestión de información permitirá mantener mayor organización en la fundación, así como adquirir la misma en el momento deseado y sin mayor dificultad.
- Las pruebas de aceptación son de suma importancia al momento de evaluar el resultado final del sistema dado que las mismas reflejan si se llegó a los resultados deseados.

6.2. RECOMENDACIONES

- Se recomienda realizar reuniones constantes con el personal que hará uso del sistema durante la fase de desarrollo para de esta forma observar que todo se proceda de acuerdo a los requerimientos.
- Es recomendable tener contacto con el cliente una vez entregado el producto para así observar el desenvolvimiento del mismo en el ambiente de trabajo.
- Es muy importante organizar tu tiempo correctamente para obtener resultados en el tiempo deseado.
- Es de suma importancia el definir correctamente los requisitos en la fase de iniciación del proceso, de este modo tendremos una mejor visión de cómo construir nuestro sistema.

BIBLIOGRAFÍA

Alarcón, V. F. (2008). *Desarrollo de sistemas de información.* UPC.

Arévalo, J. A. (9 de 11 de 2007). *Gestión de la Información, gestión de contenidos y conocimiento* . Obtenido de http://elis.da.ulcc.ac.uk/11273/1/Jornadas_GRUPO_SIOU.pdf

Castillo, E. G. (2010). *Lenguaje de programación JAVA.* Lima: Macro E.I.R.L.

Codina, L. (6 de 2015). *Sistemas de Gestión de Bases.* Obtenido de Características Principales: http://softwaredocumental.org/repositorio/Texto-completo/2015%20-%20Codina%20-%20Sistemas%20de%20Gestion%20de%20Bases%20de%20Datos%20Documentales%20caracteristicas%20principales%20y%20metodologia%20de%20dise%C3%B1o.pdf

Cordero, R. N. (16 de 06 de 2017). *Las vistas arquitectónicas de software y sus correspondencias mediante la gestión de modelos.* Obtenido de http://www.dsic.upv.es/docs/bib-dig/tesis/etd-10132009-094823/borrador-tesis-rogelio-2.pdf

EcuRed. (19 de 01 de 2017). *EcuRed*. Obtenido de Arquitectura de software: https://www.ecured.cu/Arquitectura_de_software

Fernández Valdés, M., & Ponjuán Dante, G. (19 de 01 de 2017). *Análisis conceptual de las principales interacciones.* Obtenido de http://scielo.sld.cu/pdf/aci/v18n1/aci07708.pdf

FUDRINE. (18 de 01 de 2017). *FUDRINE*. Obtenido de Nosotros: http://fudrine.com/nosotros

Guillermo Storti, G. R. (2007). *Base de datos.* Obtenido de Modelo Entidad Relación: http://www.belgrano.esc.edu.ar/matestudio/carpeta_de_access_introduccion.pdf

Gutierrez, D. (03 de 2011). *UML.* Obtenido de Diagramas de Clases: http://www.codecompiling.net/files/slides/UML_clase_04_UML_clases.pdf

Gutierrez, D. (05 de 2011). *UML.* Obtenido de Diagrama de Secuencia: http://www.codecompiling.net/files/slides/UML_clase_06_UML_secuencia.pdf

Gutierrez, D. (05 de 2011). *UML.* Obtenido de Diagramas de Estados Diagrama de Actividades: http://www.codecompiling.net/files/slides/UML_clase_03_UML_actividades_estados.pdf

IBM. (2006). *Documento de arquitectura de software*. Obtenido de http://betaniatech.com/SmallProjects/core.base_rup/workproducts/rup_software_architecture_document_C367485C.html

IBM. (02 de 06 de 2017). *Documento de visión*. Obtenido de https://www.ibm.com/support/knowledgecenter/es/SSR27Q_4.0.5/com.ibm.rational.rrm.help.doc/topics/r_vision_doc.html

IBM. (15 de 06 de 2017). *IBM Knowledge Center.* Obtenido de Introducción: Modelado de UML para la arquitectura de sistema de información: https://www.ibm.com/support/knowledgecenter/es/SS6RBX_11.4.2/com.ibm.sa.tutorial.doc/topics/intro_builddiagramsusinguml.html

IBM. (16 de 06 de 2017). *IBM Knowledge Center*. Obtenido de Cómo comenzar con un prototipo para una aplicación:

https://www.ibm.com/support/knowledgecenter/es/SSUMNA_6.1.0/com.ibm.dashboard.doc/wdf/wdf_gs_proto_ov.html

IBM. (16 de 06 de 2017). *IBM Knowledge Center*. Obtenido de Lección 2.1: Crear diagramas de secuencia: https://www.ibm.com/support/knowledgecenter/es/SS6RBX_11.4.3/com.ibm.sa.tutorial.doc/topics/Less2.1_BuildASequenceDiagram.html

IBM. (16 de 06 de 2017). *IBM Knowledge Center*. Obtenido de Lección 2.2: Crear diagramas de colaboración: https://www.ibm.com/support/knowledgecenter/es/SS6RBX_11.4.2/com.ibm.sa.tutorial.doc/topics/Less2.2_BuildACollaborationDiagram.html

IBM. (19 de 04 de 2017). *Rational Unified Process.* Obtenido de https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Isidro Ramos Salavert, M. D. (2000). *Ingeniería del software y bases de datos, tendencias actuales.* Real: Servicio de Publicaciones de la Universidad de Castilla - La Mancha.

Java. (19 de 01 de 2017). *Java*. Obtenido de https://www.java.com/es/download/faq/whatis_java.xml

José A. Taboada Gonzáles, J. M. (2005). *SIstemas de Información Medio Ambiental.* Madrid: Gesbiblo.

Kenneth C. Laudon, J. P. (2004). *Sistemas de Información Gerencial.* México: Pearson Prentice Hall.

Ledesma, J. D. (2005). *Sistemas Organizacionales Teoría y Práctica.* Bogotá: Teoría del Color.

Martínez, A. M. (19 de 04 de 2017). *Guía a Rational Unified Process.* Obtenido de https://anaylenlopez.files.wordpress.com/2011/03/trabajo-guia20rup.pdf

Microsoft. (02 de 06 de 2017). *Diagramas de casos de uso de UML: Instrucciones*. Obtenido de https://msdn.microsoft.com/es-ec/library/dd409432.aspx

Mindiola, J. (13 de 09 de 2012). *Ingenieria del Software II* . Obtenido de Diagrama de Secuencia y Colaboracion: http://jams001.blogspot.com/2012/09/diagrama-de-secuencia-y-colaboracion.html

MySQL. (19 de 01 de 2017). *MySQL.* Obtenido de About MySQL: http://www.mysql.com/about/

Oracle. (19 de 01 de 2017). *Oracle*. Obtenido de Java SE 6 Documentation: http://docs.oracle.com/javase/6/docs/index.html

Rivera, F. L. (2008). *Bases de Datos Relacionales Teoría y Práctica.* Valencia: ITM.

SALVATIERRA, A. C. (2014). *ESTUDIO DE FACTIBILIDAD PARA LA CREACIÓN DE UN CENTRO.* Quito.

Sommerville, I. (2006). *Ingeniería de software.* Madrid: Pearson Addison Wasley.

Sommerville, I. (2006). *Ingeniería del Software.* Madrid: PEARSON Asdison Wesley.

Vanesa Carolina, L. C. (2010). *Plan de Pruebas de Software.* Obtenido de Herramienta para la administración de requerimientos de los proyectos de las asignaturas: http://pegasus.javeriana.edu.co/~CIS1010IS05/Documentos/Dise%C3%B1o/STP.pdf