



UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO
INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA
LICENCIATURA EN CIENCIAS COMPUTACIONALES

TESIS
DESARROLLO DE UN SISTEMA
COMPUTACIONAL PARA LA GESTIÓN DE
EXPEDIENTES CLÍNICOS DEL ÁREA DE
PSICOLOGÍA DE LA DSMU – UAEH

Para obtener el título de Licenciada en Ciencias
Computacionales

PRESENTA

Juan Luis Torres Uribe

Directores

Dra. Anilú Franco Árcega

Comité tutorial

Dra. Ma. de Jesús Gutiérrez Sánchez

Dr. Aldo Márquez Grajales

M.I.D. Alberto Suárez Navarrete

Pachuca de Soto, Hidalgo, México abril del 2026



Universidad Autónoma del Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería

School of Engineering and Basic Sciences

Dirección

Office of the Director

Mineral de la Reforma, Hgo., a 14 de mayo de 2026

Número de control: ICBI-D/946/2026

Asunto: Autorización de impresión.

MTRA. OJUKY DEL ROCÍO ISLAS MALDONADO DIRECTORA DE ADMINISTRACIÓN ESCOLAR DE LA UA EH

Con Título Quinto, Capítulo II, Capítulo V, Artículo 51 Fracción IX del Estatuto General de nuestra Institución, por este medio, le comunico que el Jurado asignado al egresado de la Licenciatura en Ciencias Computacionales **Juan Luis Torres Uribe**, quien presenta el trabajo de titulación **“Desarrollo de un sistema computacional para la gestión de expedientes clínicos del área de Psicología de la DSMU - UA EH”**, ha decidido, después de revisar fundamento en lo dispuesto en el Título Tercero, Capítulo I, Artículo 18 Fracción IV; dicho trabajo en la reunión de sinodales, **autorizar la impresión del mismo**, una vez realizadas las correcciones acordadas.

A continuación, firman de conformidad los integrantes del Jurado:

Presidente: Dra. Ma. de Jesús Gutiérrez Sánchez

Secretario: Dr. Aldo Márquez Grajales

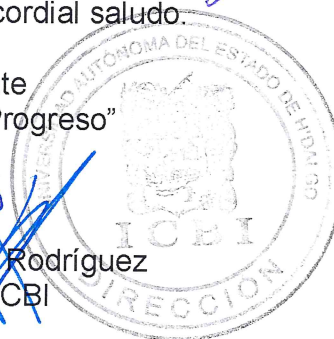
Vocal: Dra. Anilú Franco Árcega

Suplente: M.I.D. Alberto Suárez Navarrete

Sin otro particular por el momento, reciba un cordial saludo.

Atentamente
“Amor, Orden y Progreso”

Dr. Gabriel Vergara Rodríguez
Director del ICBI



GVR/MMM

“Amor, Orden y Progreso”



Ciudad del Conocimiento, Carretera Pachuca-Tulancingo
Km. 4.5 Colonia Carboneras, Mineral de la Reforma, Hidalgo,
México. C.P 42184

Teléfono: 52 (771) 71 720 00 Ext. 40001

direccion_icbi@uaeh.edu.mx, vergara@uaeh.edu.mx

uaeh.edu.mx

Contenido

Índice de figuras	ii
Índice de tablas	iv
Resumen	viii
Introducción	1
1 Marco Teórico	8
1.1 Sistemas de información	8
1.1.1 Ingeniería de software	9
1.2 Herramientas computacionales a usar	23
1.2.1 Bases de datos	23
1.2.2 Modelo Entidad-Relación	27
1.2.3 Modelo relacional	28
1.2.4 Actualizaciones, transacciones y manejo de violaciones de restricciones	32
1.2.5 MySQL	36
1.2.6 Paradigmas de programación	37
1.2.7 Paradigma Orientado a Objetos	38
1.2.8 Java	43
1.3 Área de aplicación	46
1.3.1 Psicología	46
1.3.2 Expedientes clínicos	47
1.4 Estado del arte	50
2 Marco Metodológico	54
2.1 Metodologías tradicionales	54
2.1.1 Cascada	54
2.1.2 Microsoft Solution Framework	56
2.2 Metodologías Ágiles	58
2.2.1 Metodología Scrum	58
2.2.2 Metodología Extreme Programming	60

2.3	Discusión	66
3	Implementación de la metodología	67
3.1	Fase de exploración	67
3.1.1	Requerimientos	68
3.2	Fase de planeación	76
3.3	Fase de iteraciones	78
3.3.1	Primera iteración	78
3.3.2	Segunda iteración	91
3.3.3	Tercera iteración	105
3.3.4	Cuarta Iteración	119
3.4	Fase de producción	134
3.5	Fase de mantenimiento	138
3.5.1	Instrumento de Evaluación: Construcción y Aplicación	138
3.5.2	Metodología de Análisis de Datos	139
3.5.3	Presentación y Análisis de los Resultados	142
	Conclusión	153

Lista de Figuras

2.1	Roles principales en Scrum <i>Fuente: Elaboración propia</i>	59
2.2	Fases de la metodología XP <i>Fuente: Elaboración propia</i>	62
2.3	Historia de Usuario <i>Fuente: Elaboración propia</i>	64
2.4	Tarea de Ingeniería <i>Fuente: Elaboración propia</i>	64
2.5	Prueba de Aceptación <i>Fuente: Elaboración propia</i>	65
2.6	Tarjeta CRC <i>Fuente: Elaboración propia</i>	65
3.1	Interfaz del inicio de sesión <i>Fuente: Elaboración propia</i>	84
3.2	Interfaz del creación de nuevo psicólogo <i>Fuente: Elaboración propia</i>	85
3.3	Interfaz principal de los expedientes <i>Fuente: Elaboración propia</i>	86
3.4	Interfaz del creación de nuevo psicólogo <i>Fuente: Elaboración propia</i>	87
3.5	Interfaz del principal del expediente <i>Fuente: Elaboración propia</i>	98
3.6	Interfaz secundaria en el modulo del historial clínico <i>Fuente: Elaboración propia</i>	99
3.7	Interfaz secundaria en el modulo de las tablas de síntomas <i>Fuente: Elaboración propia</i>	100
3.8	Interfaz de la tabla de síntomas <i>Fuente: Elaboración propia</i>	101
3.9	Interfaz del modulo del Familiograma <i>Fuente: Elaboración propia</i>	112
3.10	Interfaz del modulo de Notas Clínicas <i>Fuente: Elaboración propia</i>	113
3.11	Interfaz de la Nota Clínica <i>Fuente: Elaboración propia</i>	114
3.12	Interfaz del modulo del Plan de Tratamiento <i>Fuente: Elaboración propia</i>	115
3.13	Interfaz del Plan de Tratamiento <i>Fuente: Elaboración propia</i>	116
3.14	Interfaz del módulo de los Test Psicométricos: MMPI <i>Fuente: Elaboración propia</i>	126
3.15	Interfaz del módulo de los Test Psicométricos: Pensamientos automáticos <i>Fuente: Elaboración propia</i>	127
3.16	Interfaz del módulo de los Test Psicométricos: Test Ansiedad y Depresión <i>Fuente: Elaboración propia</i>	128
3.17	Interfaz del administrador: Módulo de psicólogos <i>Fuente: Elaboración propia</i> . .	129
3.18	Interfaz del administrador: Módulo de expedientes <i>Fuente: Elaboración propia</i> .	130
3.19	Interfaz del administrador: Módulo de exportación de los expedientes <i>Fuente: Elaboración propia</i>	131
3.20	Diagrama de la estructura de red <i>Fuente: Elaboración propia</i>	134

3.21	Modelo entidad-relación de la base de datos <i>Fuente: Elaboración propia</i>	135
3.22	Modelo relacional de la base de datos <i>Fuente: Elaboración propia</i>	137
3.23	Distribución de puntajes SUS individuales. <i>Fuente: Elaboración propia</i>	143
3.24	Análisis visual de los ítems del SUS. Los ítems pares han sido invertidos para una interpretación coherente (valor alto = favorable). <i>Fuente: Elaboración propia</i> . .	144
3.25	Perfil de impacto del sistema en las dimensiones evaluadas (valores promediados) <i>Fuente: Elaboración propia</i>	147
3.26	Frecuencia de mención de las funcionalidades más útiles del sistema <i>Fuente: Elaboración propia</i>	149
3.27	Distribución de dificultades reportadas por los usuarios <i>Fuente: Elaboración propia</i>	150
3.28	Tipos de impacto positivo reportados en el trabajo clínico <i>Fuente: Elaboración propia</i>	151

Índice de tablas

1.1	Modelos de obtención de requerimientos <i>Fuente: Elaboración propia.</i>	11
2.1	Roles en la metodología XP <i>Fuente: Elaboración propia</i>	61
3.1	Historias de Usuario - Acceso al Sistema <i>Fuente: Elaboración propia</i>	68
3.2	Historias de Usuario - Registro de nuevos psicólogos en el sistema <i>Fuente: Elaboración propia</i>	69
3.3	Historias de Usuario - Registro de nuevos expedientes <i>Fuente: Elaboración propia</i>	69
3.4	Historias de Usuario - Visualización de los expedientes clínicos <i>Fuente: Elaboración propia</i>	70
3.5	Historias de Usuario - Visualización de la ficha de identificación del paciente adentro del expediente <i>Fuente: Elaboración propia</i>	71
3.6	Historias de Usuario - Tabla de contactos <i>Fuente: Elaboración propia</i>	71
3.7	Historias de Usuario - Historial Clínico <i>Fuente: Elaboración propia</i>	72
3.8	Historias de Usuario - Tabla de Síntomas <i>Fuente: Elaboración propia</i>	72
3.9	Historias de Usuario - Familiograma <i>Fuente: Elaboración propia</i>	73
3.10	Historias de Usuario - Notas clínicas <i>Fuente: Elaboración propia</i>	73
3.11	Historias de Usuario - Plan de tratamiento <i>Fuente: Elaboración propia</i>	74
3.12	Historias de Usuario - Test psicométricos <i>Fuente: Elaboración propia</i>	75
3.13	Historias de Usuario - Apartado de administración <i>Fuente: Elaboración propia</i> .	76
3.14	Asignación de Roles del Proyecto <i>Fuente: Elaboración propia</i>	77
3.15	Detalles de Historias de Usuario <i>Fuente: Elaboración propia</i>	78
3.16	Tareas a realizar para la primera iteración <i>Fuente: Elaboración propia</i>	79
3.17	Tarea de Ingeniería 1 <i>Fuente: Elaboración propia</i>	79
3.18	Tarea de Ingeniería 2 <i>Fuente: Elaboración propia</i>	80
3.19	Tarea de Ingeniería 3 <i>Fuente: Elaboración propia</i>	80
3.20	Tarea de Ingeniería 4 <i>Fuente: Elaboración propia</i>	80
3.21	Tarea de Ingeniería 5 <i>Fuente: Elaboración propia</i>	81
3.22	Tarea de Ingeniería 6 <i>Fuente: Elaboración propia</i>	81
3.23	Tarea de Ingeniería 7 <i>Fuente: Elaboración propia</i>	81
3.24	Tarea de Ingeniería 8 <i>Fuente: Elaboración propia</i>	82
3.25	Tarea de Ingeniería 9 <i>Fuente: Elaboración propia</i>	82

3.26	Tarea de Ingeniería 10	<i>Fuente: Elaboración propia</i>	82
3.27	Tarjeta CRC Psicólogo	<i>Fuente: Elaboración propia</i>	83
3.28	Tarjeta CRC Expediente	<i>Fuente: Elaboración propia</i>	83
3.29	Caso de prueba 1	<i>Fuente: Elaboración propia</i>	88
3.30	Caso de prueba 2	<i>Fuente: Elaboración propia</i>	89
3.31	Caso de prueba 3	<i>Fuente: Elaboración propia</i>	90
3.32	Caso de prueba 4	<i>Fuente: Elaboración propia</i>	91
3.33	Tareas a realizar para la segunda iteración	<i>Fuente: Elaboración propia</i>	92
3.34	Tarea de Ingeniería 11	<i>Fuente: Elaboración propia</i>	93
3.35	Tarea de Ingeniería 12	<i>Fuente: Elaboración propia</i>	93
3.36	Tarea de Ingeniería 13	<i>Fuente: Elaboración propia</i>	93
3.37	Tarea de Ingeniería 14	<i>Fuente: Elaboración propia</i>	94
3.38	Tarea de Ingeniería 15	<i>Fuente: Elaboración propia</i>	94
3.39	Tarea de Ingeniería 16	<i>Fuente: Elaboración propia</i>	94
3.40	Tarea de Ingeniería 17	<i>Fuente: Elaboración propia</i>	95
3.41	Tarea de Ingeniería 18	<i>Fuente: Elaboración propia</i>	95
3.42	Tarea de Ingeniería 19	<i>Fuente: Elaboración propia</i>	95
3.43	Tarea de Ingeniería 20	<i>Fuente: Elaboración propia</i>	96
3.44	Tarea de Ingeniería 21	<i>Fuente: Elaboración propia</i>	96
3.45	Tarea de Ingeniería 22	<i>Fuente: Elaboración propia</i>	96
3.46	Tarjeta CRC Contactos	<i>Fuente: Elaboración propia</i>	97
3.47	Tarjeta CRC historial Clínico	<i>Fuente: Elaboración propia</i>	97
3.48	Tarjeta CRC historial Clínico	<i>Fuente: Elaboración propia</i>	98
3.49	Caso de prueba 5	<i>Fuente: Elaboración propia</i>	102
3.50	Caso de prueba 6	<i>Fuente: Elaboración propia</i>	103
3.51	Caso de prueba 7	<i>Fuente: Elaboración propia</i>	104
3.52	Caso de prueba 8	<i>Fuente: Elaboración propia</i>	105
3.53	Tareas a realizar para la tercera iteración	<i>Fuente: Elaboración propia</i>	106
3.54	Tarea de Ingeniería 23	<i>Fuente: Elaboración propia</i>	106
3.55	Tarea de Ingeniería 24	<i>Fuente: Elaboración propia</i>	107
3.56	Tarea de Ingeniería 25	<i>Fuente: Elaboración propia</i>	107
3.57	Tarea de Ingeniería 26	<i>Fuente: Elaboración propia</i>	107
3.58	Tarea de Ingeniería 27	<i>Fuente: Elaboración propia</i>	108
3.59	Tarea de Ingeniería 28	<i>Fuente: Elaboración propia</i>	109
3.60	Tarea de Ingeniería 29	<i>Fuente: Elaboración propia</i>	109
3.61	Tarea de Ingeniería 30	<i>Fuente: Elaboración propia</i>	109
3.62	Tarea de Ingeniería 31	<i>Fuente: Elaboración propia</i>	110
3.63	Tarea de Ingeniería 32	<i>Fuente: Elaboración propia</i>	110
3.64	Tarea de Ingeniería 33	<i>Fuente: Elaboración propia</i>	110
3.65	Tarjeta CRC Familiograma	<i>Fuente: Elaboración propia</i>	111
3.66	Tarjeta CRC Nota clínica	<i>Fuente: Elaboración propia</i>	111

3.67	Tarjeta CRC Plan de Tratamiento <i>Fuente: Elaboración propia</i>	112
3.68	Caso de prueba 9 <i>Fuente: Elaboración propia</i>	117
3.69	Caso de prueba 10 <i>Fuente: Elaboración propia</i>	118
3.70	Caso de prueba 11 <i>Fuente: Elaboración propia</i>	119
3.71	Tareas a realizar para la cuarta iteración <i>Fuente: Elaboración propia</i>	120
3.72	Tarea de Ingeniería 34 <i>Fuente: Elaboración propia</i>	121
3.73	Tarea de Ingeniería 35 <i>Fuente: Elaboración propia</i>	121
3.74	Tarea de Ingeniería 36 <i>Fuente: Elaboración propia</i>	121
3.75	Tarea de Ingeniería 37 <i>Fuente: Elaboración propia</i>	122
3.76	Tarea de Ingeniería 38 <i>Fuente: Elaboración propia</i>	122
3.77	Tarea de Ingeniería 39 <i>Fuente: Elaboración propia</i>	123
3.78	Tarea de Ingeniería 40 <i>Fuente: Elaboración propia</i>	123
3.79	Tarea de Ingeniería 41	124
3.80	Tarjeta CRC Test Psicométricos <i>Fuente: Elaboración propia</i>	124
3.81	Tarjeta CRC Exportación a Excel <i>Fuente: Elaboración propia</i>	125
3.82	Tarjeta CRC Obtención de datos para el Excel <i>Fuente: Elaboración propia</i> . . .	125
3.83	Caso de prueba 12 <i>Fuente: Elaboración propia</i>	132
3.84	Caso de prueba 13 <i>Fuente: Elaboración propia</i>	133
3.85	Puntajes individuales de la escala SUS <i>Fuente: Elaboración propia</i>	142
3.86	Estadísticas descriptivas de los ítems del SUS (Escala 1-5) <i>Fuente: Elaboración propia</i>	144
3.87	Estadísticas descriptivas de las preguntas de impacto y eficiencia (Escala 1-5) <i>Fuente: Elaboración propia</i>	146
3.88	Temas identificados en la funcionalidad más útil <i>Fuente: Elaboración propia</i> . .	148
3.89	Impactos percibidos en el trabajo clínico <i>Fuente: Elaboración propia</i>	150

Resumen

Desde sus inicios, la computación se ha posicionado como una herramienta fundamental para la recolección, organización y procesamiento de datos, simplificando significativamente los procesos administrativos en diversos ámbitos. En el área de la psicología clínica, persiste una problemática notable: la gestión manual de expedientes clínicos que documentan el progreso terapéutico de los pacientes. Esta práctica tradicional conlleva serias dificultades en la administración de información sensible, entre las que destacan la vulnerabilidad en la seguridad de los datos, la falta de integridad en los registros, la inexistencia de respaldos confiables y la dificultad para acceder a historiales completos de manera eficiente.

Para abordar estas problemáticas, este proyecto de tesis desarrolló e implementó un sistema de gestión de expedientes clínicos especializado para el área de psicología de la Dirección del Servicio Médico Universitario de la Universidad Autónoma del Estado de Hidalgo. La solución tecnológica propuesta se diseñó mediante una metodología ágil basada en Extreme Programming (XP), seleccionada intencionalmente para favorecer un desarrollo estrechamente vinculado con el cliente final y asegurar el cumplimiento exacto de los requerimientos específicos del personal médico y administrativo. Este enfoque permitió iteraciones rápidas y adaptativas durante todo el ciclo de vida del desarrollo del software.

El sistema implementado consta de módulos especializados para el registro seguro de pacientes, la documentación detallada de sesiones terapéuticas, la generación automatizada de reportes clínicos y la administración de usuarios con perfiles diferenciados. Incorpora medidas robustas de seguridad informática, incluyendo autenticación de usuarios, cifrado de información sensible y auditoría detallada de accesos. La arquitectura del sistema se diseñó para garantizar alta disponibilidad y escalabilidad futura.

Los resultados obtenidos demuestran que la solución desarrollada logra fortalecer eficazmente la seguridad, integridad y confidencialidad de los expedientes clínicos psicológicos, al tiempo que optimiza significativamente los tiempos de gestión documental. La implementación de este sistema no sólo moderniza los procesos del servicio de psicología universitaria, sino que también contribuye a elevar la calidad de la atención proporcionada a la comunidad estudiantil mediante un acceso más rápido y confiable a la información clínica relevante. Este proyecto establece las bases para la transformación digital del área de salud mental dentro de la institución universitaria, con potencial de expansión a otras especialidades médicas.

Introducción

En la era digital, la computación ha revolucionado la manera en que se maneja y procesa la información. Uno de los aspectos más relevantes dentro de este campo es el desarrollo y utilización de sistemas de información, que son herramientas diseñadas para recopilar, almacenar, procesar y distribuir datos de manera eficiente. Los sistemas de información se han convertido en un componente esencial para una amplia gama de industrias, ya que permiten transformar grandes volúmenes de datos en información valiosa que puede ser utilizada para tomar decisiones informadas.

Los sistemas de información son fundamentales para la organización y gestión de datos, especialmente en contextos donde la precisión, la seguridad y la accesibilidad de la información son críticas. A través de la automatización de procesos y la digitalización de documentos, estos sistemas no sólo mejoran la eficiencia operativa, sino que también minimizan los errores humanos y optimizan el flujo de trabajo. En particular, permiten una gestión más estructurada de la información, lo que resulta en una mejor toma de decisiones y en la posibilidad de realizar análisis más complejos y detallados.

Además; los sistemas de información ofrecen ventajas significativas en términos de seguridad y protección de datos. Con la implementación de protocolos avanzados, como la autenticación de usuarios, encriptación de datos y control de acceso, se garantiza que la información esté protegida contra accesos no autorizados y manipulaciones indebidas. Esto es particularmente importante en sectores donde la confidencialidad de la información es primordial.

La adopción de sistemas de información en diversas áreas, incluyendo la salud, ha demostrado ser un factor clave para mejorar la organización, el acceso y la seguridad de los datos. En este contexto, la computación no sólo se presenta como una herramienta técnica, sino como un aliado estratégico en la mejora de procesos y en la calidad de los servicios proporcionados.

En el ámbito de la salud, la gestión eficiente y segura de la información de los pacientes es crucial para proporcionar una atención médica de calidad. Es por eso que los expedientes clínicos funcionan como un conjunto de documentos que contienen información detallada y completa sobre la historia médica de un paciente. Este expediente incluye datos personales, antecedentes médicos, diagnósticos, tratamientos, evolución clínica, y cualquier otra información relevante

que los profesionales de la salud necesiten para proporcionar una atención adecuada y continua al paciente.

Los expedientes clínicos han sido utilizados históricamente en el sector salud como la base fundamental para la gestión de la información del paciente. Su uso permite a los médicos y otros profesionales de la salud registrar información detallada sobre los antecedentes médicos y personales del paciente, incluyendo enfermedades pasadas, cirugías, alergias y hábitos de vida. Además, estos documentos permiten llevar un registro cronológico de las consultas, tratamientos y evolución del estado de salud del paciente, lo cual es crucial para proporcionar un cuidado continuo y coherente.

Asimismo, los expedientes clínicos facilitan la comunicación y el intercambio de información entre distintos profesionales de la salud que atienden al mismo paciente, garantizando así una atención coordinada y multidisciplinaria. También proveen la información necesaria para la toma de decisiones clínicas informadas, ayudando a los profesionales a determinar el diagnóstico y el tratamiento más adecuados.

Además, los expedientes clínicos permiten la evaluación de la calidad de la atención proporcionada, facilitando auditorías clínicas y revisiones de casos. Por último, estos documentos cumplen con los requisitos legales y éticos, asegurando que se mantenga un registro preciso y completo de la atención proporcionada a cada paciente.

Planteamiento del problema

Llevar expedientes clínicos de manera manual presenta diversos desafíos que pueden comprometer tanto la eficiencia como la calidad de la atención médica. Uno de los principales problemas es el riesgo constante de extravío o daño de los documentos físicos. Estos expedientes pueden perderse fácilmente debido a errores humanos, deteriorarse con el tiempo, o verse afectados por factores ambientales como incendios, inundaciones o condiciones de almacenamiento inadecuadas. Esta vulnerabilidad pone en riesgo la integridad de la información del paciente, lo que podría tener graves repercusiones en su tratamiento y seguimiento.

Además, el acceso a la información en un expediente físico puede ser sumamente ineficiente. La búsqueda manual de datos específicos resulta ser un proceso lento y engorroso, lo que puede retrasar la toma de decisiones clínicas importantes, especialmente en situaciones donde la inmediatez es crítica. En emergencias, contar con información rápida y precisa es vital para la adecuada atención del paciente, pero los expedientes físicos no siempre están disponibles en el momento que se necesita.

Otro aspecto preocupante es la seguridad y confidencialidad de la información contenida en estos documentos. Los expedientes físicos son vulnerables a accesos no autorizados, lo que puede comprometer la privacidad del paciente. A pesar de que se suelen implementar medidas como el almacenamiento en áreas restringidas o el control de acceso, estas soluciones no siempre resultan ser prácticas o efectivas a largo plazo, exponiendo la información sensible a riesgos considerables.

Por otro lado, la gestión manual de expedientes supone una carga administrativa significativa para el personal de salud. El proceso de documentación, archivo y organización consume una considerable cantidad de tiempo y recursos, los cuales podrían ser mejor empleados en la atención directa al paciente. Este esfuerzo adicional reduce la eficiencia operativa de las instituciones de salud y puede afectar la calidad general del servicio que se brinda.

Justificación

La gestión de expedientes psicológicos es una tarea fundamental en la práctica clínica, ya que proporciona a los profesionales de la salud mental la información necesaria para ofrecer un tratamiento adecuado y continuo a sus pacientes. Sin embargo, el manejo de expedientes físicos presenta múltiples desafíos que pueden afectar negativamente la eficiencia y calidad del servicio ofrecido.

En primer lugar, la organización de expedientes físicos es un proceso que consume una cantidad considerable de tiempo y recursos. El personal de salud debe dedicar horas a la clasificación, archivo y búsqueda de documentos, lo que reduce el tiempo disponible para la atención directa de los pacientes. Además, la falta de un sistema organizado puede resultar en la pérdida o extravío de documentos, comprometiendo la integridad del historial clínico de los pacientes.

La seguridad de la información es otro aspecto crítico que se ve comprometido en el manejo de expedientes físicos. Los documentos en papel son vulnerables a accesos no autorizados, manipulación indebida y riesgos ambientales como incendios o inundaciones. La confidencialidad de los datos de los pacientes es esencial en la práctica psicológica, y la incapacidad de garantizarla puede erosionar la confianza de los pacientes y tener repercusiones legales y éticas para los profesionales de la salud.

La implementación de un sistema digital de expedientes psicológicos ofrece una solución a estos problemas. Un sistema digital permite la gestión eficiente y organizada de la información, facilitando el acceso rápido y seguro a los datos de los pacientes. La digitalización elimina el riesgo de extravío y daño físico de los documentos, y las medidas de seguridad avanzadas, como la autenticación de usuarios y la encriptación de datos, protegen la confidencialidad de la información sensible.

Además, los profesionales de la salud pueden acceder a la información de los pacientes de manera instantánea desde cualquier ubicación autorizada, lo que es especialmente útil en contextos de tele-medicina o cuando se requiere colaboración entre distintos profesionales.

En términos de eficiencia operativa, la automatización de la gestión de expedientes reduce significativamente el tiempo y esfuerzo necesarios para el manejo de la información, permitiendo a los profesionales de la salud dedicar más tiempo a la atención directa de los pacientes. Esto no sólo mejora la calidad del servicio, sino que también aumenta la satisfacción y confianza de los pacientes.

Propuesta de solución

La tecnología emerge como una solución viable para mitigar muchos de estos problemas. Implementar un sistema digital para la gestión de expedientes clínicos no sólo elimina los riesgos de pérdida o daño físico de los documentos, sino que también garantiza que la información esté disponible en cualquier momento y desde cualquier lugar autorizado. Este tipo de sistemas no sólo permite un acceso rápido y eficiente a los datos del paciente, sino que también mejora la capacidad de respuesta en situaciones de emergencia, optimizando el flujo de trabajo.

Además, los sistemas digitales fortalecen la confidencialidad y seguridad de la información a través de medidas avanzadas como la autenticación de usuarios, la encriptación de datos y los permisos de acceso basados en roles. Estas características protegen de manera eficiente la información sensible del paciente, garantizando que sólo el personal autorizado pueda acceder a los datos necesarios.

Finalmente, la automatización de la gestión de expedientes reduce considerablemente la carga administrativa, permitiendo al personal de salud centrarse en lo más importante: la atención al paciente. La digitalización facilita la captura, organización y actualización de los datos clínicos, mejorando la continuidad del tratamiento y la calidad de la atención.

Por lo tanto, este trabajo propone el desarrollo de un sistema integral que permita la creación, modificación y exportación de expedientes clínicos en formato digital. Este sistema ofrecerá a los profesionales de la salud un acceso rápido y seguro a los expedientes de sus pacientes, optimizando la gestión de la información y garantizando su protección. Así, cada profesional podrá crear nuevos expedientes, actualizar la información existente y exportar los datos cuando sea necesario, todo desde una plataforma centralizada y eficiente.

Antecedentes

Dentro de la Dirección de Servicios Médicos Universitarios (DSMU), de la Universidad Autónoma del Estado de Hidalgo los expedientes clínicos del área de psicología son llenados manualmente por los profesionales de la salud. Este proceso comienza con un formato inicial en el cual se registran las características básicas del paciente, tales como su nombre, edad, sexo, religión y el motivo principal de su consulta. Este documento inicial es crucial ya que proporciona la información esencial que permite a los profesionales comprender el contexto y las necesidades inmediatas del paciente.

En la siguiente etapa, se procede a completar el historial clínico del paciente. Durante esta fase, se realiza un interrogatorio detallado en el que se pregunta al paciente sobre su historial médico, incluyendo cualquier enfermedad que haya tenido, visitas recientes al médico, operaciones o procedimientos médicos previos, así como cualquier alergia conocida. Además, se elabora un árbol familiar que ayuda a identificar posibles condiciones hereditarias que puedan influir en la salud mental del paciente.

Una vez completado este primer documento de características básicas e historial clínico, se inicia el seguimiento con las notas clínicas. Por cada consulta, el profesional de la salud redacta una nota clínica que documenta el progreso del paciente, las intervenciones realizadas y las observaciones relevantes. Este proceso es continuo y meticuloso, ya que cada cuatro notas clínicas individuales se sintetizan en una nota clínica general. Esta nota general es anexada al expediente para proporcionar una visión consolidada del progreso del paciente, el tipo de tratamiento que está recibiendo y cómo va evolucionando su condición a lo largo del tiempo.

Adicionalmente, dependiendo de las necesidades específicas de cada paciente, se pueden realizar diversas pruebas psicométricas. Estas pruebas son diseñadas para evaluar diferentes aspectos de la salud mental del paciente y son administradas y calificadas manualmente por los profesionales de la salud. Los resultados de estas pruebas también se anexan al expediente clínico, proporcionando un conjunto de datos más completo y detallado que ayuda en la toma de decisiones clínicas.

Objetivos

Objetivo general

Desarrollar un sistema computacional de expedientes psicológicos que permita a los profesionales de la salud registrar, gestionar y acceder a la información de los pacientes y sus consultas de manera eficiente y segura, mejorando la organización y protegiendo la confidencialidad de los datos empleando el lenguaje de programación Java y el gestor de base de datos MySQL.

Objetivos específicos

- Crear una arquitectura de base de datos con el lenguaje MySQL que permita almacenar de forma organizada y accesible la información de los pacientes, incluyendo datos personales, historiales clínicos y notas de consultas
- Diseñar interfaces de usuario que faciliten la entrada, edición y consulta de datos por parte de los profesionales de la salud, para asegurar una experiencia de usuario eficiente y amigable, utilizando una herramienta de diseño para crearlas de manera rápida y sencilla.
- Considerar funcionalidades para el registro y seguimiento que permitan a los profesionales de la salud registrar y gestionar notas clínicas detalladas durante y después de las consultas, facilitando el seguimiento continuo del estado y progreso de los pacientes, utilizando el paradigma de la programación orientada a objetos.
- Incorporar medidas de seguridad como autenticación de usuarios y control de accesos, para proteger la información sensible de los pacientes contra accesos no autorizados y posibles vulnerabilidades creando perfiles y contraseñas para cada profesional de la salud a los cuales se conectarán a la base de datos ubicada en la computadora de la administración.
- Integrar los elementos propuestos para crear la herramienta de solución empleando una plataforma de desarrollo.
- Conducir pruebas del sistema con usuarios finales para evaluar su usabilidad, funcionalidad e impacto, usando un sistema de escalas y realizar las mejoras necesarias basadas en los resultados obtenidos.

Estructura del documento

Este documento tendrá la siguiente estructuración de sus contenidos para llevar el desarrollo de la investigación de manera correcta.

Capítulo 1 Marco teórico en este apartado se fundamentarán teóricamente todas las funciones dentro del desarrollo del mismo, para poder saber cómo es que se puede desarrollar un software hasta las partes que lo componen y las herramientas que se utilizarán en este proyecto.

Capítulo 2 Marco metodológico en este siguiente capítulo se describirán las metodologías que se pueden usar para trabajar en este proyecto, para que con esta información se escoja la óptima y aplicarla al desarrollo del mismo.

Capítulo 3 Implementación de la metodología En este siguiente apartado se realizará toda la documentación de la creación del proyecto, siguiendo las fases que se llevé en la meto-

dología seleccionada.

Capítulo 1

Marco Teórico

El presente marco teórico está diseñado para ofrecer una base conceptual sólida que sustente el desarrollo del proyecto. En primer lugar, se abordarán los sistemas de información, un componente esencial en la gestión de datos, destacando su relevancia en el ámbito de la ingeniería de software. A continuación, se examinarán las herramientas computacionales que se utilizarán para llevar a cabo el proyecto, específicamente el lenguaje de programación Java y la base de datos MySQL, que brindan el soporte técnico necesario para el desarrollo de aplicaciones eficientes y funcionales.

Posteriormente, se explorará el contexto de aplicación del proyecto, centrándose en la psicología y la importancia de la gestión de expedientes clínicos. Esta sección permitirá entender cómo la tecnología puede mejorar la atención y el seguimiento en el ámbito psicológico. Finalmente, se discutirá sobre las metodologías de desarrollo de software, presentando un análisis de cuatro enfoques diferentes, a partir del cual se seleccionará la metodología adecuada para el proyecto en cuestión.

1.1 Sistemas de información

Según González-Longatt en su artículo “Introducción a los Sistemas de Información: Fundamentos Francisco M. González-Longatt” (2007), el concepto de “Sistemas de Información” (SI) ha sido interpretado y aplicado en numerosos campos a lo largo del tiempo. En términos generales, un SI incluye tanto componentes humanos como tecnológicos que trabajan en conjunto para organizar, procesar y distribuir información relevante para la toma de decisiones.

Por otro lado Molina et al. (2015) explican que los Sistemas de Información (SI) comprenden seis componentes clave. Entre ellos se encuentran:

1. **Base de Datos:** El lugar donde se almacena la información necesaria para tomar decisiones, organizada en registros bien definidos.
2. **Transacciones:** Acciones que los usuarios pueden realizar, como consultar, añadir, modificar o eliminar información.
3. **Informes:** Permiten obtener datos específicos o estadísticas a partir de criterios establecidos por el usuario.
4. **Procesos:** Operaciones que generan nuevos datos siguiendo una lógica predefinida y son manejados por el usuario.
5. **Usuario:** Todas las personas que interactúan con el sistema, desde los operativos hasta los ejecutivos.
6. **Procedimientos Administrativos:** Normas que regulan el comportamiento del usuario frente al sistema, asegurando que el acceso directo a la base de datos no sea posible para ningún usuario.

1.1.1 Ingeniería de software

La ingeniería de software se ha definido por varios autores, de acuerdo con Gómez et al. en su libro “Fundamentos de Ingeniería de Software” de la Universidad Autónoma Metropolitana, citan a Somerville (2004), uno de los padres de la ingeniería de software donde describe la ingeniería de software como “una disciplina de la ingeniería que comprende todos los aspectos de la producción del software”.

Por otro lado, Gómez et al. en (2019) citan a Bohem (1976), quien definió la ingeniería de software como “la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos.” De igual forma menciona que se suele conocer como desarrollo de software o producción de software.

Ciclo de vida software

En la publicación Procesos de Ingeniería de Software por Cubero (2020), citando a Campderich (2013), indica que el ciclo de vida del software está constituido por el conjunto de todas las etapas que preceden y suceden a la producción de software.

De acuerdo con el autor Cubero los métodos y técnicas de la ingeniería del software se estructuran dentro del ciclo de vida del software, que consta de varias etapas fundamentales, cada una con su propio enfoque y objetivos. Asimismo Cubero menciona al autor Berzal (s.f.),

indicando que el ciclo de vida incluye las siguientes fases:

1. **Planificación:** Organiza el proyecto y abarca la definición del ámbito, estudio de viabilidad, análisis de riesgos, estimación de costos, planificación temporal y asignación de recursos.
2. **Análisis:** Busca comprender los requerimientos del sistema, destacando la necesidad de colaboración con el cliente, quien a menudo no tiene claridad sobre sus necesidades.
3. **Diseño:** Representa las características del sistema y estudia alternativas de implementación, con un enfoque en la estructura general. Se deben considerar patrones de diseño para evitar errores comunes.
4. **Implementación:** Inicia la programación del sistema basado en el análisis y diseño previos. Es crucial seleccionar herramientas adecuadas y mantener un código legible y bien documentado.
5. **Pruebas:** Detecta errores en etapas anteriores. Incluye pruebas de unidad, integración, alfa, beta y de aceptación, así como revisiones a lo largo del proceso.
6. **Instalación o despliegue:** Planifica el entorno de operación del sistema, considerando aspectos de hardware y software necesarios.
7. **Uso y mantenimiento:** Involucra tres tipos de mantenimiento: correctivo (eliminar defectos), adaptativo (ajustar a nuevas necesidades) y perfectivo (agregar funcionalidades).

Análisis de software

Por otro lado, Párraga et al. (2023), en su libro Ingeniería del Software I: Requerimientos y Modelado del Software, mencionan que el análisis de los requerimientos da como producto la especificación de las características operativas del software, indica que la interfaz de éste y otros elementos del sistema, establecen restricciones que limitan al software.

Los autores mencionan que el análisis de los requerimientos permite al profesional construir acerca de los requerimientos básicos establecidos durante las tareas de concepción, indagación y negociación, ya que son elementos de la ingeniería de los requerimientos.

De acuerdo con Párraga et al. (2023), la acción de modelar los requerimientos da como resultado uno o más de los siguientes tipos de modelo, como se explica en la Tabla 1.1:

Los autores hablan acerca de los requerimientos funcionales y no funcionales Párraga et al. (2023), mencionan que los requerimientos funcionales son descripciones de los servicios que el

Modelos	Características
Modelos basados en el escenario.	Se basa en los requerimientos desde el punto de vista de distintos “actores” del sistema.
Modelos de datos.	Ilustran el dominio de información del problema.
Modelos orientados a clases.	Representan clases orientadas a objetos (atributos y operaciones) y la manera en la que las clases colaboran para cumplir con los requerimientos del sistema.
Modelos orientados al flujo.	Representan los elementos funcionales del sistema y la manera como transforman los datos a medida que se avanza a través del sistema.
Modelos de comportamiento.	Ilustran el modo en el que se comparte el software como consecuencia de “eventos” externos.

Tabla 1.1: Modelos de obtención de requerimientos *Fuente: Elaboración propia.*

sistema debe proporcionar, cómo debe responder ante entradas específicas y cómo debería comportarse en distintas situaciones. En ocasiones, también especifican lo que el sistema no debe hacer.

Los requerimientos funcionales dependen del tipo de software que se esté desarrollando, los usuarios previstos y el enfoque que sigue la organización al redactarlos. Cuando se formulan como requerimientos de usuario, suelen expresarse de manera abstracta para ser comprensibles para los usuarios. Sin embargo, los requerimientos funcionales más detallados describen con precisión las funciones del sistema, sus entradas, salidas y excepciones.

Por otro lado, los requerimientos no funcionales son restricciones sobre los servicios o funciones del sistema, incluyendo limitaciones de tiempo, desarrollo o normas aplicables. Estos requerimientos suelen aplicarse al sistema en su totalidad, en lugar de a características o funciones individuales.

De igual forma Molina et al. (2015), los requerimientos de usuario para un sistema deben detallar tanto los requerimientos funcionales como los no funcionales, de una manera que sea accesible y entendible para los usuarios, sin necesidad de tener conocimientos técnicos profundos.

No obstante, los autores mencionan que al redactarlos utilizando frases en lenguaje natural dentro de un documento de texto, pueden aparecer diversos inconvenientes, algunos de ellos son:

1. **Falta de claridad:** A veces resulta complicado emplear un lenguaje preciso y sin ambigüedades, sin que esto haga que el documento se vuelva extenso y difícil de entender.
2. **Confusión en los requerimientos:** Los requerimientos funcionales y no funcionales, los

objetivos del sistema y la información de diseño no siempre se diferencian claramente, lo que genera confusión.

3. **Combinación de requerimientos:** Es común que distintos requerimientos se presenten juntos como si fueran uno solo, lo que puede dificultar su comprensión.

Una revisión de requerimientos es un proceso en el que participan tanto el cliente como los desarrolladores del sistema. El objetivo de este proceso es asegurarse de que el documento de requerimientos esté libre de errores, inconsistencias o carencias, verificando que todo esté correctamente planteado. En paralelo, la construcción de prototipos implica mostrar a los clientes y usuarios finales un modelo funcional del sistema. Esto les permite interactuar con el prototipo para comprobar si satisface sus necesidades reales y si se alinea con sus expectativas (Gómez et al., 2019).

Los autores Gómez et al. (2019), mencionan sobre la validación de requerimientos consiste en asegurar que el desarrollo del sistema responde a lo que realmente pidió el cliente, ayudando a responder la pregunta: ¿se está construyendo lo que el cliente necesita?

Por su parte, la verificación de requerimientos se enfoca en garantizar que cada requerimiento de software sea comprobable. Es decir, se debe poder demostrar que el producto final cumple con lo solicitado. Una parte clave de este proceso es planificar cómo se verificará cada requerimiento. La verificación busca asegurar que el sistema no sólo cumple con las expectativas del cliente, sino que también se ajusta a los estándares establecidos mediante pruebas de aceptación. Estas pruebas se diseñan para validar que el sistema cumple con lo requerido y, en caso de que algún requerimiento no pueda ser verificado a través de pruebas, se debe reformular. Este proceso responde a la pregunta: ¿el sistema se está construyendo funciona de acuerdo con lo que solicitó el cliente?

Diseño de software

De igual forma, los autores Gómez et al. (2019), en su libro Fundamentos de Ingeniería de Software mencionan que un modelo es una representación de un concepto, objeto o sistema que permite explicar características específicas, como su forma, comportamiento, estructura o las relaciones entre diferentes entidades.

Cuando un programador construye un modelo mental del sistema, puede resultar complicado transmitirlo con claridad a otros programadores, ya que la comunicación verbal no siempre es precisa. Aunque se pueda dibujar el sistema en un pizarrón o en papel, existe la posibilidad de que los demás no lo comprendan de la misma manera si no se emplea un lenguaje común para describir el modelo mental de cada persona.

Si el creador de un sistema de software no documenta los modelos que concibió al diseñarlo, esa información podría perderse, incluso para él mismo con el tiempo. La modelación permite una mejor comprensión del sistema en desarrollo, ayudando a visualizar lo que se desea que haga, a especificar su estructura y comportamiento, y a guiar su creación. Además, modelar es fundamental para comunicarse eficazmente con los clientes o usuarios, explorar soluciones de diseño, y documentar las decisiones tomadas, lo que resulta crucial para la producción de software de calidad (Gómez et al, 2019).

Los autores Gómez y Moraleda (2020), en su libro teórico “Aproximación a la ingeniería del software” indican que, durante el proceso de diseño de un sistema, se llevan a cabo diversas actividades que buscan organizar y refinar progresivamente su estructura. La cantidad y denominación de estas actividades puede variar según la complejidad del sistema.

De forma que los autores enumeran y describen actividades habituales en el diseño de un sistema:

1. **Diseño arquitectónico:** Es la primera actividad y se encarga de definir la estructura y organización general del sistema, dividiéndolo en subsistemas o módulos, y estableciendo las relaciones e interfaces entre ellos. Este paso es crucial para desarrollar el sistema de manera efectiva.
2. **Diseño detallado:** En esta fase se organiza internamente cada uno de los módulos definidos en el diseño arquitectónico, ajustando su estructura y añadiendo o eliminando módulos si es necesario. En sistemas sencillos, este diseño se integra como un refinamiento del diseño arquitectónico.
3. **Diseño procedimental:** Aquí se diseña cómo se ejecutarán las operaciones o servicios dentro de cada módulo, detallando los algoritmos más relevantes mediante pseudocódigo, que luego se refinan en la codificación.
4. **Diseño de datos:** Esta actividad define cómo se organizarán los datos del sistema, asegurando que el diseño facilite su reutilización y mantenimiento. A menudo, se realiza en paralelo con el diseño detallado y procedimental.
5. **Diseño de la interfaz de usuario:** Dado que la interacción entre el usuario y el sistema es clave, esta actividad se centra en crear interfaces que sean fáciles de usar y aprender, tomando en cuenta aspectos psicológicos y de ergonomía.

El resultado final de estas actividades es un Documento de Diseño de Software (SDD), que describe de forma jerarquizada la estructura del sistema, detallando su evolución hasta la etapa de codificación. Si el sistema es muy complejo, se pueden generar varios documentos para reflejar

diferentes niveles de detalle (Gómez y Moraleda, 2020).

Las herramientas CASE (Computer-Aided Software Engineering) facilitan la comunicación de los aspectos clave de un sistema durante su fase de conceptualización. Proveen un apoyo automatizado que permite visualizar de inmediato los flujos de datos y otros componentes en pantalla, lo que facilita que los clientes sugieran correcciones o cambios de manera rápida y eficiente, algo que antes tomaba más tiempo con herramientas tradicionales (Molina et al., 2015).

Los autores Molina et al. (2015), hablan de estas herramientas, ya que también permiten generar un "programa esqueleto" a partir de un diseño, lo que incluye la implementación de interfaces. El desarrollador sólo debe añadir detalles funcionales para completar el programa. Además, las herramientas CASE ayudan a modelar sistemas mediante UML (Lenguaje de Modelado Unificado), reduciendo costos y tiempo en el ciclo de desarrollo del software, al tiempo que permiten la generación de código a partir de diagramas predefinidos. El UML (Unified Modeling Language) es un lenguaje estándar ampliamente utilizado en metodologías de desarrollo de software. Este lenguaje emplea diagramas y cuadros para construir, visualizar y documentar todos los requisitos del sistema. UML no es una metodología en sí, pero puede adaptarse a diversas metodologías de desarrollo, proporcionando una notación uniforme y semántica clara, lo que facilita su integración en diferentes enfoques.

Para entender UML, es esencial reconocer que es un lenguaje que sigue reglas de sintaxis y semántica, lo que permite modelar sistemas de manera estructurada. UML no sólo representa la parte conceptual de un sistema, sino que también define el contexto para su construcción. Sus principales aplicaciones incluyen el desarrollo de software comercial y no comercial, la captura de requisitos y la documentación de sistemas nuevos o existentes (Molina et al., 2015).

Desarrollo de software

Los autores Gómez et al. (2019), definen que en la ingeniería de software, la codificación es el proceso de convertir la especificación de diseño en un lenguaje de programación. Este paso implica transformar los conceptos del diseño en código fuente en un lenguaje específico, como C++, Java o Python. La codificación es una fase crucial del desarrollo de sistemas de software y, según el modelo de desarrollo adoptado, puede realizarse una vez que todos los componentes del sistema estén diseñados o a medida que cada componente o grupo de componentes relacionados se vaya diseñando.

La codificación, en esencia, implementa el diseño en un lenguaje de programación, y un código bien desarrollado debe cumplir con ciertas características. Según Gómez et al. (2019), citando a Tsui (2014), las metas de una codificación de calidad son:

- **Código legible:** Debe ser comprensible para otros programadores, permitiéndoles entender fácilmente el programa.
- **Código trazable:** Cada elemento o grupo de elementos debe poder relacionarse con un componente del diseño. Esto garantiza que se pueda rastrear desde un requerimiento hasta el código que lo implementa, y viceversa.
- **Código correcto:** El código debe cumplir los requerimientos especificados.
- **Código completo:** Debe abarcar todos los requerimientos establecidos.
- **Buen rendimiento:** El programa debe ejecutarse eficientemente siempre que cumpla con las características previas.

Lograr un código legible y fácil de mantener es tan esencial como asegurar que sea correcto y completo. Generalmente, la legibilidad y mantenibilidad del código son más prioritarias que su velocidad de ejecución.

Los lenguajes de programación

De acuerdo con Gómez y Moraleda (2020), los lenguajes de programación son herramientas que permiten a los humanos comunicarse con las computadoras, modificando su comportamiento y controlando su funcionamiento. Por lo que en la guía Swebok, estos lenguajes se pueden clasificar en:

- **Lenguajes de configuración:** Permiten al ingeniero de software seleccionar parámetros específicos entre opciones predefinidas, como los archivos de configuración de sistemas Unix o Windows.
- **Lenguajes de caja de herramientas o “toolkits”:** Se utilizan para construir aplicaciones combinando componentes preprogramados. También se conocen como lenguajes de programación de aplicaciones.
- **Lenguajes de script o macros:** Incluyen scripts o archivos de procesamiento por lotes
- **Lenguajes de programación propiamente dichos:** Estos permiten desarrollar aplicaciones completas con gran flexibilidad, siendo los que se estudiarán en mayor profundidad.

A pesar de la corta historia de la computación, existen cientos de lenguajes de programación creados con diversos objetivos, muchos de ellos para fines experimentales o de investigación. Sólo un grupo reducido de estos lenguajes ha sido realmente útil a nivel industrial (Gómez y Moraleda, 2020). Analizar su evolución histórica permite obtener una visión general de los mismos, y esta evolución se clasifica habitualmente en cuatro generaciones de lenguajes, aunque

estas generaciones pueden coincidir en el tiempo (Gómez y Moraleda, 2020).

Primera generación de lenguajes

Los primeros computadores, costosos y con limitada capacidad, se programaban directamente mediante códigos binarios, una tarea que exigía conocer a fondo la estructura interna del equipo y que solo unos pocos realizaban. En la década de 1950, con el aumento de los computadores, la programación empezó a profesionalizarse. Para reducir errores y simplificar este proceso, se crearon los lenguajes ensambladores, en los cuales cada instrucción de máquina se asocia con un símbolo nemotécnico que representa su función, haciendo la programación ligeramente más accesible que con código binario puro. Estos lenguajes ensambladores se consideran la primera generación, ya que poseen un nivel de abstracción mínimo (Gómez y Moraleda, 2020).

Segunda generación

Con el avance en la capacidad de memoria y almacenamiento, fue posible desarrollar programas más complejos. Sin embargo, los programas en ensamblador se volvían complicados de depurar y mantener. A finales de los años 50, surgieron los primeros lenguajes de alto nivel, los cuales permitieron la programación simbólica sin depender de un computador específico. Estos lenguajes marcaron un cambio importante al introducir elementos abstractos, como tipos de datos no dependientes de la máquina y variables simbólicas. También aparecieron estructuras de control como bucles y condicionales, que facilitaron el desarrollo de software con un enfoque más abstracto y funcional (Gómez y Moraleda, 2020).

Tercera generación

La mayoría de los lenguajes actuales pertenecen a esta generación. A finales de los años 60 y principios de los 70, se estableció la programación estructurada, transformando la programación en un proceso metódico y productivo. Dado que los equipos de desarrollo crecían, los lenguajes de tercera generación implementaron un fuerte tipado y estructuras de datos organizadas, permitiendo detectar errores en la fase de compilación y facilitando la coherencia en el código (Gómez y Moraleda, 2020).

Cuarta generación

Los lenguajes de cuarta generación (L4G) buscan ofrecer al programador un alto nivel de abstracción, eliminando la necesidad de interactuar directamente con el hardware. Generalmente, no son lenguajes de propósito general, sino herramientas diseñadas para resolver problemas específicos, permitiendo a personas sin gran experiencia en programación crear aplicaciones sencillas. Sin embargo, no son recomendables para desarrollar proyectos complejos debido a la ineficiencia del código que generan. Su utilidad radica, en parte, en la creación de prototipos

que pueden refinarse luego con lenguajes de tercera generación (Gómez y Moraleda, 2020).

Criterios para seleccionar un lenguaje de programación

Gómez y Moraleda (2020), indican que elegir el lenguaje de programación adecuado para un proyecto es complejo, dado que cada lenguaje ofrece distintas capacidades. La elección del lenguaje es fundamental, ya que es la herramienta principal del equipo de desarrollo y afecta considerablemente la depuración y el mantenimiento de la aplicación. Aunque los criterios técnicos suelen tener prioridad, también es importante considerar otros factores operativos. Entre los criterios de selección a evaluar están:

- Imposición del cliente
- Tipo de aplicación
- Disponibilidad y entorno
- Experiencia previa
- Reusabilidad
- Transportabilidad
- Uso de varios lenguajes

Normas y estilo de codificación

Al iniciar la fase de codificación de un proyecto, es esencial establecer normas que todos los programadores deben seguir para garantizar la uniformidad en el resultado final. Las empresas de desarrollo suelen contar con un conjunto de normas generales aplicables a sus proyectos (Gómez y Moraleda, 2020).

Muchos lenguajes permiten formatos flexibles al escribir sentencias y omiten espacios y comentarios adicionales. Así, el mismo programa puede escribirse en varios formatos, que pueden ser significativos para el lector humano, aunque irrelevantes para el compilador. Para facilitar la comprensión y mantenimiento de un programa, se deben fijar normas que establezcan un estilo claro de codificación. Estas normas incluyen:

- **Formato y contenido de las cabeceras de cada módulo:** especificando su identificación, descripción, y registro de revisiones con fechas.
- **Formato y contenido de comentarios:** definiendo su sección, orden, posición en la columna y margen.

- **Indentación y alineación:** como el máximo nivel de indentación y formato para selecciones e iteraciones.

Estas directrices buscan hacer el código comprensible tanto para el autor como para futuros lectores, facilitando la continuidad del proyecto a largo plazo (Gómez y Moraleda, 2020).

Aspectos de eficiencia

Gómez y Moraleda, (2020), mencionan la eficiencia en la codificación no es tan crucial como en los primeros días de la computación. La capacidad de procesamiento y la abundancia de memoria en los computadores actuales permite priorizar la claridad del código sin necesidad de comprometerla por cuestiones de eficiencia. Existen distintos enfoques para evaluar la eficiencia:

- Eficiencia en el uso de memoria
- Eficiencia en el tiempo de ejecución

Dado el bajo costo de la memoria, en caso de requerir ahorro, generalmente basta con utilizar un compilador que incluya funciones de compresión automática. Sin embargo, cuando el volumen de datos es mayor que la memoria disponible, se deben evaluar alternativas en la fase de diseño detallado y elegir el algoritmo que mejor optimice la memoria. La eficiencia en tiempo de ejecución es crítica en sistemas de tiempo real, donde se priorizan plazos muy ajustados. A veces, aumentar la eficiencia temporal implica reducir la eficiencia en memoria, por ejemplo, al almacenar en tablas cálculos complejos para acceder a ellos en lugar de realizarlos repetidamente. En la fase de diseño detallado, analizar diferentes alternativas y algoritmos ayuda a optimizar el tiempo de ejecución, aunque las mejoras que los compiladores ofrecen para optimizar el código pueden ser difíciles de prever (Gómez y Moraleda, 2020).

Transportabilidad de software

Hacer que una aplicación sea transportable exige un esfuerzo adicional, pero este se ve recompensado en muchos casos. La transportabilidad permite utilizar el mismo software en diferentes sistemas actuales y futuros, lo cual es beneficioso no solo a corto plazo, sino también a mediano y largo plazo, ya que facilita la adaptación y el mantenimiento del software en nuevas arquitecturas (Gómez y Moraleda, 2020).

Los factores clave para la transportabilidad incluyen:

- Uso de estándares
- Aislamiento de especificidades

Un software desarrollado exclusivamente con estándares (como lenguaje de programación, bases de datos o bibliotecas gráficas) debería ser transportable sin modificaciones en plataformas compatibles con dichos estándares. La ausencia de estándares dificulta la transportabilidad; por ello, se deben usar herramientas que sean ampliamente aceptadas como estándares “de facto”. Se deben evitar elementos que aún no estén consolidados y que puedan estar sujetos a cambios (Gómez y Moraleda, 2020).

Para aislar las especificidades, se recomienda crear módulos específicos que contengan estas peculiaridades, de modo que el proceso de transporte implique sólo la adaptación de dichos módulos al nuevo sistema. Las principales especificidades suelen estar relacionadas con:

- La arquitectura del computador
- El sistema operativo

Gestión de proyectos de software

Bayona et al. (2016), explican que la gestión de proyectos de software es la base de la ingeniería de software y cubre todo el ciclo de desarrollo. Para lograr que un proyecto de software sea exitoso, es necesario tener en cuenta varios factores: el alcance de trabajo, los posibles riesgos, los recursos necesarios, las tareas a realizar, el esfuerzo y el costo involucrados, además de un plan claro a seguir. Una gestión adecuada no sólo ayuda a canalizar el trabajo de los desarrolladores de manera eficiente y productiva, sino que también orienta el proyecto hacia un resultado que cumpla con las expectativas de todos los interesados, especialmente el cliente.

Elementos clave en la gestión de proyectos de software según los autores:

1. **Gestión:** La gestión en proyectos de software abarca todas las actividades y tareas realizadas por una o varias personas para planificar y controlar las acciones necesarias para alcanzar los objetivos del proyecto, mejorar los resultados o finalizar las tareas de forma eficiente.
2. **Proyectos:** En el ámbito de software, un proyecto se considera una serie de procesos y actividades organizadas mediante una metodología específica para alcanzar metas concretas. Los proyectos deben gestionarse en función de los recursos disponibles, como dinero, tiempo, personal y herramientas, lo que requiere una planificación y seguimiento constante a lo largo de la duración del proyecto.

Aspectos fundamentales en la gestión de proyectos según los autores:

- **Control:** El control en la gestión de proyectos garantiza que el desarrollo se mantenga alineado con los objetivos establecidos desde el inicio. Esto implica supervisar el tiempo, recursos, costos, alcance y otros factores que puedan impactar positiva o negativamente el proyecto. El objetivo es que el líder del proyecto tenga una visión clara de su estado actual y, en caso de identificar desviaciones, pueda aplicar las correcciones necesarias de manera oportuna. Esto permite una adaptación constante a los cambios, preservando el rumbo y la eficacia del proyecto.
- **Dirección:** La dirección de un proyecto de software implica una serie de actividades que el director del proyecto debe gestionar. Estas incluyen:
 - Tomar decisiones clave para guiar el desarrollo.
 - Asignar tareas y coordinar al equipo de trabajo.
 - Liderar grupos y motivar a los participantes.
 - Establecer compromisos claros.
 - Supervisar el cumplimiento de tareas asignadas.
 - Implementar medidas correctivas cuando sea necesario.
 - Programar y planificar actividades.
 - Gestionar la comunicación y el flujo de información.

La dirección eficiente asegura que cada miembro del equipo entiende sus responsabilidades y que las decisiones estratégicas se implementan de acuerdo con los objetivos del proyecto.

- **Reclutamiento:** El reclutamiento es el proceso de selección del personal necesario para llevar a cabo el proyecto. Escoger a las personas adecuadas es crucial, ya que la competencia y cualificación del equipo impactan directamente en el éxito o fracaso del proyecto. Los equipos multidisciplinarios, formados por individuos con habilidades específicas y compatibles, aportan diversidad de perspectivas y conocimientos, fortaleciendo la estructura y capacidad del equipo para resolver problemas complejos.
- **Organización:** La organización de un proyecto de software requiere planificar las actividades y tareas de manera que faciliten el control y ejecución de cada una. Además, la organización ayuda a definir claramente los límites de autoridad y asignar responsabilidades y obligaciones a cada miembro del equipo. Con una estructura organizativa sólida, es posible mejorar la eficiencia y el rendimiento del equipo, garantizando que todos conozcan sus funciones y se mantengan alineados con los objetivos del proyecto.
- **Planificación:** La planificación de un proyecto de software proporciona un marco que permite al gerente o director estimar los recursos, costos y tareas necesarias para completar el proyecto. Además, la planificación debe prever posibles escenarios, tanto favorables como

adversos, para que los resultados puedan ajustarse según las condiciones en el desarrollo del proyecto. La estimación anticipada de recursos y actividades contribuye a una ejecución más estructurada, permitiendo que el proyecto avance con pasos calculados hacia sus metas.

Estos elementos de gestión permiten que los proyectos de software se desarrollen de manera estructurada y eficiente, minimizando riesgos y maximizando los recursos para lograr resultados de calidad (Bayona et al., 2016).

Pressman y Contreras en su libro Ingeniería del software un enfoque práctico publicado en el 2010, indican acerca de la administración efectiva de un proyecto de software se organiza en torno a cuatro elementos fundamentales: personal, producto, proceso y proyecto. Este enfoque secuencial no es casual, ya que el éxito del proyecto depende en gran medida de reconocer que el desarrollo de software es una actividad humana que requiere comunicación comprensiva, una planificación sólida y el uso de métodos y herramientas adecuadas.

1. **El Personal:** Desde los años 60, se ha reconocido la importancia de contar con un equipo motivado y altamente capacitado en desarrollo de software. Este factor humano es tan crucial que el Software Engineering Institute desarrolló el Modelo de Madurez de Capacidades del Personal (People-CMM), cuyo objetivo es mejorar la capacidad de una organización para atraer, formar, organizar y retener el talento necesario para alcanzar sus objetivos estratégicos. Este modelo incluye áreas prácticas clave como gestión de plantilla, comunicación y coordinación, ambiente laboral, desempeño, capacitación, desarrollo de competencias y equipo, entre otros. Las organizaciones que alcanzan altos niveles en este modelo tienen una probabilidad elevada de implementar prácticas de gestión efectivas en sus proyectos.
2. **El Producto:** Antes de planificar un proyecto, es fundamental establecer los objetivos y el ámbito del producto, explorar alternativas y definir restricciones técnicas y administrativas. Esta información es necesaria para realizar estimaciones precisas de costos, valorar riesgos, desglosar tareas y crear un cronograma manejable. Este proceso comienza con la definición de objetivos y el alcance del producto, incluyendo datos, funciones y comportamientos. Una vez comprendidos estos aspectos, se deben considerar soluciones alternativas para identificar el mejor enfoque, teniendo en cuenta limitaciones de tiempo, presupuesto y disponibilidad de recursos.
3. **El Proceso:** El proceso de software proporciona una estructura que permite desarrollar un plan completo. Ciertas actividades clave, como las tareas, hitos y aseguramiento de calidad, se aplican a todos los proyectos de software, sin importar su tamaño. Las actividades “sombrija”, como el aseguramiento de calidad, la administración de configuración y las mediciones, cubren todo el proceso, apoyando el cumplimiento de los objetivos de calidad.

4. **El Proyecto:** La planificación y control del proyecto son fundamentales para gestionar la complejidad inherente al desarrollo de software. Aun con estos controles, muchos proyectos de software enfrentan dificultades. En un estudio de grandes proyectos, solo el 10 % cumplió sus objetivos de tiempo, costo y calidad, mientras que la mayoría tuvo retrasos significativos o se abandonaron sin completar. Para evitar estos problemas, los gerentes de proyectos y el equipo de desarrollo deben estar atentos a señales de advertencia comunes, conocer los factores de éxito y aplicar enfoques racionales para la planificación y supervisión continua del proyecto.

En resumen, los autores Pressman y Contreras (2010), buscan explicar que una gestión efectiva de proyectos de software depende de entender y aplicar estos cuatro elementos con una estrategia bien definida, lo cual aumenta las posibilidades de éxito al mitigar riesgos y responder adecuadamente a las complejidades del desarrollo.

La planeación de proyectos es una tarea esencial para el administrador de proyectos de software, que consiste en dividir el trabajo en tareas, asignarlas a los miembros del equipo y anticipar posibles problemas junto con sus soluciones. El plan inicial no sólo sirve para comunicar cómo se llevará a cabo el proyecto y medir el progreso, sino que también se ajusta a medida que se avanza y se obtiene nueva información (Sommerville, 2011). Según el autor la planeación se da en tres etapas clave a lo largo del ciclo de vida de un proyecto:

1. **Etapas de propuestas:** En esta fase, se realiza un plan general para evaluar si existen los recursos necesarios para llevar a cabo el proyecto y se calcula una cotización para el cliente.
2. **Fase de inicio:** Con más detalles disponibles, se asignan recursos específicos, se definen los incrementos del proyecto y se afinan las estimaciones de tiempo y costo.
3. **Revisiones periódicas:** El plan se actualiza según el progreso del equipo, los requerimientos cambiantes y las nuevas estimaciones de tiempo y esfuerzo.

Para enfoques ágiles, los planes son más flexibles y se adaptan continuamente conforme el software evoluciona. En la fase de propuesta, los planes suelen ser especulativos debido a la falta de requerimientos específicos. Se requiere un estimado de costos que permita calcular el precio de cotización. Factores adicionales también afectan la fijación del precio final del proyecto, más allá de sumar costos y ganancias (Sommerville, 2011).

Sommerville (2011) indica que, en la planeación de proyectos de software, tres parámetros clave determinan los costos:

1. **Costos de esfuerzo:** Incluyen el pago a ingenieros y administradores.

2. **Costos de hardware y software:** Abarca tanto el equipo necesario como el mantenimiento.
3. **Costos de viajes y capacitación:** Se asocian principalmente a los traslados y formación de equipo.

En la mayoría de los casos, los costos de esfuerzo representan la mayor parte. Para esto, se calcula el tiempo de trabajo necesario en meses-hombre, añadiendo una contingencia en caso de que la estimación inicial resulte demasiado optimista. Los sistemas comerciales pueden usar hardware de bajo costo, pero los gastos de software pueden aumentar si es necesario licenciar middleware o plataformas adicionales. Si el equipo de desarrollo está en varias ubicaciones, los viajes pueden sumar gastos, aunque el tiempo en traslado suele ser improductivo. Utilizar herramientas de colaboración en línea puede reducir la necesidad de viajes y así optimizar el uso del tiempo en tareas productivas del proyecto (Sommerville, 2011).

Al adjudicar un contrato, el plan inicial debe afinarse para crear un plan de inicio, que ayuda a definir las necesidades de personal y presupuesto, especialmente en entornos de desarrollo ágil. El plan establece la base para asignar recursos dentro de la organización, definir la contratación de nuevo personal y estructurar un sistema de seguimiento, permitiendo comparar el avance y los costos reales con lo planeado (Sommerville, 2011).

1.2 Herramientas computacionales a usar

1.2.1 Bases de datos

Según Elmasri et al. (2007), una base de datos es un conjunto organizado de datos interrelacionados que contienen información significativa. Los datos se refieren a hechos o detalles específicos que se pueden registrar y que poseen un sentido definido, como nombres, números de teléfono y direcciones. Estos datos pueden almacenarse en herramientas como un libro de direcciones o en aplicaciones digitales como Microsoft Access o Excel. Una base de datos, en esencia, agrupa estos datos significativos de manera estructurada. Los autores indican que una base de datos presenta las siguientes características:

- **Representación de un aspecto real:** Refleja un ámbito del mundo real, también conocido como minimundo o universo de discurso (UoD). Cualquier cambio en este minimundo se actualiza en la base de datos.
- **Coherencia lógica:** Organiza los datos de manera lógica y significativa, por lo que un conjunto aleatorio de información no se considera una base de datos.

- **Diseño con propósito específico:** Es creada, estructurada y poblada con datos para cumplir objetivos particulares, dirigidos a un grupo de usuarios y con aplicaciones previamente definidas que responden a sus necesidades.

DBSM

Elmasri et al. (2007), indican que una base de datos computarizada puede crearse y gestionarse mediante aplicaciones diseñadas para esa función o, de forma más eficiente, a través de un sistema de administración de bases de datos (DBMS). Un DBMS es un conjunto de programas que permite a los usuarios definir, construir, manipular y compartir bases de datos. Este sistema facilita la definición de datos especificando tipos, estructuras y restricciones; además, almacena descripciones de estos datos como metadatos en un catálogo o diccionario. La construcción de una base de datos implica cargar los datos en un medio de almacenamiento que el DBMS controla.

Los autores mencionan que Compartir una base de datos permite el acceso simultáneo de varios usuarios y programas. Las aplicaciones interactúan con la base de datos mediante consultas o solicitudes enviadas al DBMS, lo que permite recuperar, leer o escribir datos. Además, el DBMS asegura la protección y mantenimiento continuo de la base de datos, permitiéndole adaptarse a nuevos requerimientos a lo largo de su ciclo de vida, que puede extenderse durante varios años.

Nivel de abstracción de los datos

Silberschatz et al. (2006), mencionan que, para que un sistema sea práctico, debe recuperar los datos de manera eficaz. Esta necesidad de eficiencia ha llevado a los diseñadores a utilizar estructuras de datos complejas en la representación de la información dentro de la base de datos. Sin embargo, debido a que muchos usuarios no tienen conocimientos avanzados de informática, los desarrolladores implementan distintos niveles de abstracción para simplificar la interacción con el sistema.

Según los autores existen tres niveles de abstracción de los datos:

- **Nivel físico:** Este es el nivel más bajo de abstracción y describe el modo en que los datos se almacenan realmente, detallando las estructuras de datos de bajo nivel.
- **Nivel lógico:** Un nivel superior de abstracción que define qué datos se almacenan y las relaciones entre ellos. Aunque el nivel lógico utiliza estructuras simples, estas pueden estar respaldadas por estructuras más complejas del nivel físico. Los administradores de bases de datos emplean este nivel para gestionar y organizar la información que se guarda.

- **Nivel de vistas:** Este nivel, el más alto de abstracción, muestra sólo una parte de la base de datos, permitiendo a los usuarios acceder únicamente a la información que necesitan sin preocuparse por la complejidad subyacente. El sistema puede ofrecer múltiples vistas de la misma base de datos, simplificando así la interacción.

Lenguajes de la BD

Los sistemas de bases de datos ofrecen un lenguaje de definición de datos, que permite especificar el esquema de la base de datos, y un lenguaje de manipulación de datos, utilizado para formular consultas y realizar modificaciones en la base de datos (Silberschatz et al. 2006).

Lenguajes de manipulación de la BD

Silberschatz et al. (2006), indican que un lenguaje de manipulación de datos (LMD) permite a los usuarios acceder a los datos organizados según el modelo de datos o modificarlos. Los tipos de operaciones de acceso incluyen:

- Recuperar información almacenada en la base de datos.
- Insertar nueva información en la base de datos.
- Eliminar información de la base de datos.
- Modificar la información existente en la base de datos.

Existen dos tipos principales de LMDs:

- LMDs procedimentales: requieren que el usuario indique tanto los datos específicos que necesita como el método para obtenerlos.
- LMDs declarativos (o no procedimentales): solo requieren que el usuario especifique los datos que necesita sin necesidad de indicar cómo obtenerlos.

Lenguaje de definición de datos

Los esquemas de una base de datos se determinan a través de definiciones realizadas en un lenguaje específico llamado lenguaje de definición de datos (LDD). Este lenguaje también permite establecer propiedades adicionales de los datos. La estructura de almacenamiento y los métodos de acceso que emplea el sistema de bases de datos se definen mediante un conjunto de comandos en un tipo especializado de LDD conocido como lenguaje de almacenamiento y definición de

datos (Silberschatz et al., 2006).

Los autores Silberschatz et al. (2006) muestran que los datos almacenados en una base de datos deben cumplir con ciertas restricciones para asegurar su consistencia. El LDD ofrece herramientas para definir este tipo de restricciones. En los sistemas de bases de datos, el enfoque se centra en restricciones de integridad que puedan verificarse con una carga mínima:

- **Restricciones de dominio:** Cada atributo debe tener un dominio de valores posibles, como enteros, caracteres, o fechas y horas.
- **Integridad referencial:** Se establece cuando se desea asegurar que un valor presente en un conjunto de atributos en una relación también aparezca en un conjunto de atributos en otra relación.
- **Asertos:** Los asertos representan cualquier condición que la base de datos debe cumplir siempre. Las restricciones de dominio e integridad referencial son casos específicos de asertos. Sin embargo, algunas restricciones, como la de "cada préstamo debe tener al menos un cliente con una cuenta de saldo mínimo de 1,000 euros," sólo pueden expresarse mediante asertos.
- **Autorización:** Permite definir los permisos de acceso para distintos usuarios, determinando el tipo de acceso permitido a diversos datos en la base de datos.

Bases de datos relacionales

Según Montalbán y Vázquez (2014) una base de datos representa la información sobre un ámbito específico del mundo real. Diseñar una base de datos implica identificar y recopilar todos los datos importantes relacionados con un problema determinado. Por ejemplo, esto puede incluir los datos necesarios para gestionar el proceso de facturación en una empresa de venta de maquinaria agrícola o para registrar el historial veterinario de los animales en un zoológico. Para identificar estos datos, se realiza un análisis detallado del ámbito del problema, determinando qué información es clave para la base de datos y descartando aquella que no sea relevante. Una vez definidos los datos esenciales, comienza la etapa de modelado, donde se construye un esquema con una herramienta de diseño de bases de datos, representando con precisión toda la información que debe almacenarse.

Para el modelo conceptual, se emplea el modelo Entidad/Relación, el cual representa el análisis del problema mediante diagramas de entidad-relación.

1.2.2 Modelo Entidad-Relación

El Modelo Entidad-Relación (ME-R), desarrollado por Peter Chen en la década de 1970, ofrece una visión unificada de los datos, enfocándose en la estructura lógica y abstracta de los mismos para representar el mundo real, sin depender de aspectos físicos específicos según Nevado (2010). Los elementos clave de este modelo, según el autor son:

- Entidad
- Relaciones
- Dominios

Una entidad es cualquier persona, lugar, objeto, concepto o evento, ya sea real o abstracto, relevante en el ámbito del problema. Es el objeto del cual se desea almacenar información en la base de datos. A la estructura general de una entidad se le llama tipo de entidad, y cada instancia concreta de ese tipo se conoce como ocurrencia de la entidad (Nevado, 2010).

Una relación es la conexión entre tipos de entidades. Al hablar de Tipo de relación, se refiere a la estructura general que conecta dos o más tipos de entidad. La Ocurrencia de una Interrelación representa una instancia específica de esta conexión, enlazando ocurrencias concretas de cada tipo de entidad participante. Visualmente, las relaciones se representan con un rombo con su nombre, conectado por líneas a las entidades relacionadas (Nevado, 2010).

Ejemplo y características clave de una relación:

- **Name:** Identifica la interrelación de forma única y se coloca en el rombo.
- **Grado:** Número de entidades en la interrelación. Puede ser binaria (dos entidades), ternaria (tres), o en general, n-aria.
- **Tipo de correspondencia:** Indica el número máximo de ocurrencias de una entidad que pueden asociarse con cada ocurrencia de la otra entidad.
- **Atributo:** Define las propiedades de una entidad o interrelación, como Nombre o Nacionalidad en el tipo de entidad AUTOR, o el atributo Orden-de-Firma en la interrelación AL.

Un Dominio es el conjunto de posibles valores que puede tomar un atributo, con un nombre y existencia propia, independiente de cualquier entidad o atributo. Al asignar un dominio a un atributo, se establece que todos sus valores deben pertenecer al conjunto especificado, actuando como una restricción. El Atributo Identificador Principal (AIP) es el atributo seleccionado en un tipo de entidad que permite identificar de forma única y mínima cada una de sus ocurrencias (Nevado, 2010).

Restricciones en las interrelaciones

Según el autor Nevado (2010), existen cuatro tipos de restricciones en las interrelaciones:

- **Restricción de Exclusividad:** Dos o más interrelaciones son exclusivas cuando cada instancia de un tipo de entidad solo puede vincularse con instancias de otro tipo. Por ejemplo, los empleados, según sus capacidades, pueden ser diseñadores de productos o participar en su fabricación, pero no pueden ocupar ambos roles al mismo tiempo.
- **Restricción de Exclusión:** Esta restricción surge cuando las instancias de una entidad solo pueden combinarse mediante una única interrelación. Un caso sería el de un profesor, que no puede impartir y recibir el mismo curso; aunque puede desempeñar uno de estos roles, no ambos a la vez.
- **Restricción de Inclusividad:** Una interrelación se considera inclusiva cuando cada instancia de una entidad que participa en ella también ha tenido que participar previamente en otra. Por ejemplo, para que un empleado trabaje como diseñador, es requisito que haya asistido a al menos dos cursos específicos.
- **Restricción de Inclusión:** En este caso, una instancia de una entidad debe estar relacionada con otra mediante una segunda interrelación para poder establecer la asociación principal. Un ejemplo es el divorcio: para que un hombre se divorcie de una mujer, primero deben haberse casado.

1.2.3 Modelo relacional

Según Elmasri et al. (2007), El modelo relacional fue introducido por primera vez en 1970 por Ted Codd, investigador de IBM, en un artículo que se volvió un clásico (Codd, 1970). Este modelo rápidamente ganó atención por su simplicidad y sólida base matemática. Su estructura se basa en el concepto de relación matemática, que funciona de manera similar a una tabla de datos, y se fundamenta en la teoría de conjuntos y la lógica de primer orden.

Conceptos del modelo relacional

Según Elmasri et al. (2007), en el modelo relacional, la base de datos se representa como un conjunto de relaciones, que informalmente se asemejan a una tabla de valores o a un archivo plano de registros. Cada fila de una tabla suele reflejar un hecho que se relaciona con una entidad o relación real, y tanto el nombre de la tabla como de sus columnas se emplean para interpretar los valores que se encuentran en las filas. En términos formales, una fila en el modelo relacional se denomina tupla, una cabecera de columna se conoce como atributo, y el nombre de la tabla representa una relación. Cada columna está definida por un dominio, que delimita los posibles

valores que pueden aparecer.

Dominios, atributos, tuplas y relaciones

Un dominio, D , es un conjunto de valores que son indivisibles dentro del modelo relacional. Para especificar un dominio, generalmente se define un tipo de dato que describe los valores permitidos y se le asigna un nombre que ayude a interpretar su contenido. Algunos ejemplos de dominios, según los autores incluyen:

- **NumerosTelefonosFijos:** conjunto de nueve dígitos que conforman los números de teléfono fijo en España.
- **NumerosTelefonosMoviles:** conjunto de nueve dígitos que forman los números de teléfonos móviles en España.
- **DocumentoNacionalIdentidad:** conjunto de documentos nacionales de identidad (DNI) válidos en España.
- **Nombres:** conjunto de caracteres que representan el nombre de una persona.

Estas definiciones de dominio son lógicas, y a cada una de ellas se le asigna un tipo de dato o formato. Cada dominio tiene un nombre, un tipo de dato y un formato; además, puede incluir información adicional para facilitar la interpretación de sus valores. Por ejemplo, un dominio numérico como `PesoPersona` debería especificar la unidad de medida, como kilogramos o libras.

Un ejemplo de esquema de relación para una universidad podría ser el siguiente:

`ESTUDIANTE(Nombre, Dni, TlfParticular, Dirección, TlfTrabajo, Edad, Mnc)`

Utilizando los tipos de datos para cada atributo, el esquema puede verse como:

`ESTUDIANTE(Nombre: cadena, Dni: cadena, TlfParticular: cadena, Dirección: cadena, TlfTrabajo: cadena, Edad: entero, Mnc: real)`

En este esquema, `ESTUDIANTE` es el nombre de la relación y cuenta con siete atributos.

Características de las relaciones

Elmasri et al. (2007), definen que una relación tiene ciertas características que la diferencian de una tabla o archivo común. A continuación se describen algunas de estas particularidades,

según los autores.

- **Orden de las tuplas en una relación:** En una relación, las tuplas se agrupan como un conjunto, y matemáticamente, los elementos de un conjunto no siguen un orden específico. Esto significa que el orden de las tuplas en una relación no es relevante. Sin embargo, en un archivo, los registros se almacenan en disco o en memoria con un orden definido físicamente.
- **Orden de los valores en una tupla y definición alternativa de relación:** Aunque una n-tupla se define como una secuencia ordenada de valores, el orden específico de los atributos y valores no es crucial mientras se conserve la correspondencia entre ellos. A nivel lógico, lo importante es la relación entre los atributos y sus valores, no su posición.
- **Valores y NULL en las tuplas:** En una tupla, cada valor es atómico, es decir, indivisible dentro de la estructura del modelo relacional básico, lo que se conoce también como modelo relacional plano. Gran parte de la teoría subyacente a este modelo se basa en el principio de primera forma normal, lo cual implica que los atributos multivaluados deben ser representados en relaciones distintas, y los compuestos solo mediante atributos simples. También se usa el valor NULL para representar valores desconocidos o no aplicables en una tupla.
- **Interpretación de una relación:** Un esquema de relación se puede entender como una afirmación sobre los datos. Algunas relaciones representan datos sobre entidades, mientras que otras reflejan relaciones entre ellas. Por ejemplo, el esquema de relación ESPECIALIDAD (DniEstudiante, CodDpto) indica que los estudiantes están especializados en determinadas disciplinas. En el modelo relacional, tanto entidades como relaciones se representan uniformemente, lo cual puede dificultar en algunos casos distinguir si se trata de una entidad o de una relación.

Restricciones en el modelo relacional y en los esquemas de bases de datos relacionales

Elmasri et al. (2007), Mencionan que en una base de datos relacional, existen múltiples relaciones y las tuplas se conectan de diferentes formas. El estado de la base de datos en su totalidad refleja el estado de todas sus relaciones en un momento específico. En este contexto, las restricciones sobre los valores en el estado de la base de datos se dividen en tres categorías principales:

1. **Restricciones inherentes al modelo de datos:** Son restricciones implícitas que dependen directamente del modelo.
2. **Restricciones explícitas:** Estas se especifican directamente en los esquemas de datos mediante el DDL (Lenguaje de Definición de Datos).

3. **Restricciones semánticas:** No se pueden expresar directamente en los esquemas, por lo que deben implementarse en los programas y se conocen también como reglas de negocio o restricciones basadas en la aplicación.

Restricciones de dominio

Las restricciones de dominio establecen que, en cada tupla, el valor de un atributo específico debe ser un valor atómico dentro del dominio de dicho atributo. Los tipos de datos usados habitualmente incluyen valores numéricos estándar, como enteros (corto, estándar, y largo) y reales (en formatos de simple o doble precisión). También existen tipos de datos para almacenar caracteres, valores booleanos, cadenas de longitud fija y variable, además de fechas, horas y monedas. Algunos dominios pueden definirse como un subconjunto de un tipo de dato específico o como un tipo enumerado, en el cual se indican todos los valores posibles de manera explícita (Elmasri et al., 2007).

Restricciones de clave y restricciones de valores NULL

Elmasri et al. (2007), indican que una relación es un conjunto de tuplas, todas las tuplas en una relación deben ser únicas. Esto significa que no puede haber dos tuplas con los mismos valores en todos sus atributos.

Cualquier grupo de atributos se denomina superclave del esquema de relación. Una superclave asegura que dos tuplas distintas no compartan el mismo valor en esos atributos. Cada relación tiene al menos una superclave básica: el conjunto de todos sus atributos. Sin embargo, una superclave puede incluir atributos redundantes, lo cual introduce el concepto de clave, que es una superclave sin redundancia.

Una clave K en un esquema de relación R es una superclave que, si se elimina cualquier atributo de K , se convierte en un conjunto que ya no es una superclave de R . Esto significa que una clave cumple dos condiciones:

1. No pueden existir dos tuplas con valores idénticos para todos los atributos de la clave.
2. Es una superclave mínima; es decir, si eliminamos algún atributo, pierde la propiedad de ser una superclave.

Elmasri et al. (2007), mencionan un esquema de relación puede tener múltiples claves, cada una de las cuales se llama clave candidata. Por ejemplo, en la relación COCHE, existen dos claves candidatas:

NúmeroPermisoConducir y NúmeroBastidor. Por convención, se suele escoger una de estas claves candidatas como clave principal para identificar de forma única cada tupla. Cuando una relación tiene varias claves candidatas, la elección de la clave principal es opcional, aunque suele preferirse una clave con un único atributo o un número reducido de ellos.

Integridad de entidad, integridad referencial y claves externas

Las restricciones de integridad de entidad establecen que los valores de cualquier clave primaria no pueden ser NULL, ya que dichas claves se usan para identificar de manera única cada tupla dentro de una relación. Permitir valores NULL en una clave primaria impediría identificar ciertas tuplas. Por ejemplo, si dos o más tuplas tuvieran un valor NULL en sus claves primarias, no sería posible diferenciarlas en el momento de referenciarlas desde otras relaciones (Elmasri et al. 2007).

Elmasri et al. (2007), indican que las restricciones de clave y de integridad de entidad se definen dentro de una sola relación. En cambio, las restricciones de integridad referencial se aplican entre dos relaciones distintas y ayudan a mantener la coherencia entre las tuplas de ambas. De manera informal, estas restricciones indican que una tupla de una relación que hace referencia a otra debe apuntar a una tupla existente en esa relación. Un ejemplo, Es el atributo Dno de la relación EMPLEADO indica el número del departamento en el que cada empleado trabaja; por tanto, su valor en cada tupla de EMPLEADO debe coincidir con el valor de NumeroDpto de alguna tupla en la relación DEPARTAMENTO.

1.2.4 Actualizaciones, transacciones y manejo de violaciones de restricciones

Los autores Elmasri et al. (2007), explican que las operaciones en el modelo relacional se dividen en recuperaciones y actualizaciones. Las operaciones de álgebra relacional permiten especificar las recuperaciones. Una expresión de álgebra relacional produce una nueva relación al aplicar una serie de operadores algebraicos sobre un conjunto existente de relaciones; esta herramienta se usa principalmente para realizar consultas en la base de datos. A través de una consulta, el usuario define la información de su interés, lo que permite generar una nueva relación aplicando operadores relacionales para recuperar los datos deseados. Esta relación se presenta como la respuesta a la consulta del usuario.

Operación de inserción

La operación de inserción (Insert) incluye una lista de valores de atributos para una nueva tupla que se incorporará a una relación específica (R). Sin embargo, la inserción puede infringir una o

más de las restricciones definidas anteriormente. Cuando una inserción resulta en una violación, la opción por defecto es rechazarla. En tales casos, es conveniente que el sistema de gestión de bases de datos (DBMS) proporcione una explicación sobre la razón de este rechazo (Elmasri et al. 2007).

Operación de borrado (Delete)

La operación de borrado (Delete) puede afectar la integridad referencial si la tupla que se va a eliminar está referenciada por foreign keys en otras tuplas de la base de datos. Para especificar un borrado, se define una condición sobre los atributos de la relación que selecciona la(s) tupla(s) a eliminar (Elmasri et al. 2007).

Operación de actualización (Update)

Los autores Elmasri et al. (2007), indica que la operación de actualización (Update o Modify) se utiliza para modificar los valores de uno o varios atributos de una tupla en una relación determinada. Para seleccionar la información que será actualizada, es necesario establecer una condición en los atributos de la relación correspondiente.

Concepto de transacción

En el contexto de una aplicación de base de datos referencial, se ejecuta una serie de transacciones que incluyen tanto lecturas como operaciones de inserción, borrado y actualización. Al final de cada transacción, la base de datos debe quedar en un estado coherente que respete todas las restricciones establecidas. Una transacción puede incluir una o varias operaciones de recuperación y diversas operaciones de actualización. Existen numerosas aplicaciones comerciales que ejecutan transacciones en bases de datos relacionales mediante sistemas de Procesamiento de Transacciones en Línea (OLTP), alcanzando frecuentemente velocidades de procesamiento de cientos de transacciones por segundo (Elmasri et al. 2007).

SQL

Los autores Silberschatz et al. (2006), explican que la versión inicial de SQL fue desarrollada por IBM como parte del proyecto System R a inicios de la década de 1970, cuando se le conocía como Sequel. Desde entonces, este lenguaje ha evolucionado y se le denomina actualmente SQL (Structured Query Language o Lenguaje Estructurado de Consultas). SQL se ha convertido en el estándar para la gestión de bases de datos relacionales, siendo compatible con numerosos productos en la actualidad. Los autores explican que, el SQL está compuesto por varios elementos

principales:

- **Lenguaje de Definición de Datos (LDD):** Este componente incluye comandos para definir los esquemas de relaciones, eliminar relaciones y modificar los esquemas existentes.
- **Lenguaje de Manipulación de Datos (LMD):** El LMD de SQL integra un lenguaje de consultas basado en álgebra relacional y cálculo relacional de tuplas, además de comandos para insertar, eliminar y actualizar tuplas.
- **Integridad:** A través del LDD, SQL permite definir restricciones de integridad que deben cumplir los datos en la base de datos. Cualquier actualización que viole estas restricciones será rechazada.
- **Definición de Vistas:** El LDD de SQL también permite definir vistas dentro de la base de datos.
- **Control de Transacciones:** SQL ofrece comandos para señalar el inicio y el fin de las transacciones.
- **SQL Embebido y SQL Dinámico:** Estos permiten la inclusión de instrucciones SQL dentro de lenguajes de programación generales como C, C++, Java, PL/I, Cobol, Pascal, y Fortran.
- **Autorización:** El LDD incluye comandos para definir derechos de acceso sobre las relaciones y vistas.

Definición de Datos

Cada base de datos debe especificarse mediante un conjunto de relaciones que el sistema define a través de un lenguaje de definición de datos (LDD). En SQL, el LDD no sólo permite la especificación de relaciones, sino también el almacenamiento de información adicional relacionada con esas relaciones incluyendo:

- El esquema de cada relación.
- El dominio de valores asociado a cada atributo.
- Las restricciones de integridad.
- El conjunto de índices que se deben mantener para cada relación.
- La información de seguridad y de autorización de cada relación.
- La estructura de almacenamiento físico de cada relación en el disco.

Definición Básica de Esquemas en SQL

Silberschatz et al. (2006), mencionan que las relaciones en SQL se definen mediante el comando "create table". En esta definición, el nombre de la relación es especificado junto con los atributos y sus tipos de dominio, además de cualquier restricción de integridad relevante.

Cuando se insertan o modifican tuplas en una relación, SQL verifica que no existan valores nulos en los atributos que forman la clave primaria y que no haya duplicados en estos valores. Si alguna de estas condiciones no se cumple, SQL genera un error y detiene la actualización para mantener la consistencia de la base de datos.

Al crear una nueva relación, esta se inicializa vacía, y se pueden añadir datos con el comando "insert". Al especificar los valores, deben respetarse el orden y la correspondencia con los atributos de la relación.

Para borrar tuplas de una relación se utiliza el comando "delete", que permite eliminar todas las tuplas o, mediante variaciones específicas, solo aquellas que cumplan con ciertas condiciones.

Si se desea eliminar toda una relación de la base de datos, se emplea el comando "drop table", lo cual borra tanto el esquema como los datos de la relación. Esto implica que no se podrán insertar más datos en la relación hasta que se cree nuevamente con la instrucción "create table".

Para añadir un nuevo atributo a una relación existente, se utiliza el comando "alter table", que asigna un valor nulo al nuevo atributo para todas las tuplas que ya existen en la relación.

Estructura Fundamental de las Consultas en SQL

Los autores Silberschatz et al. (2006), explican que en las bases de datos relacionales, el conjunto de datos se organiza en múltiples relaciones, cada una con un nombre único que la identifica. Estas relaciones siguen una estructura específica, como la descrita anteriormente. En SQL, es posible emplear valores nulos para señalar que un valor es desconocido o que simplemente no está presente.

La estructura básica de una expresión SQL consta de tres cláusulas: select, from y where.

- La cláusula **select** se corresponde con la operación proyección del álgebra relacional. Se usa para obtener una relación de los atributos deseados en el resultado de una consulta.
- La cláusula **from** se corresponde con la operación producto cartesiano del álgebra relacional. Genera una lista de las relaciones que deben ser analizadas en la evaluación de la expresión.

- La cláusula **where** se corresponde con el predicado selección del álgebra relacional. Es un predicado que engloba a los atributos de las relaciones que aparecen en la cláusula **from**.

Las consultas habituales de SQL tienen la forma:

```
select A1, A2, . . . , An from r1, r2, . . . , rm where P
```

SQL genera el producto cartesiano de las relaciones especificadas en la cláusula **from**, luego aplica el filtro de selección usando las condiciones definidas en la cláusula **where** y, finalmente, selecciona y muestra los atributos indicados en la cláusula **select**. En la práctica, SQL puede optimizar esta expresión convirtiéndola a una forma equivalente que permita un procesamiento más eficiente (Silberschatz et al. 2006).

1.2.5 MySQL

MySQL según Casillas et al. (2010), es un sistema de gestión de bases de datos (SGBD, DBMS en inglés) ampliamente reconocido y utilizado, destacando por su sencillez y rendimiento notable. Aunque no cuenta con algunas de las funciones avanzadas de otros SGBD, su facilidad de uso y rápida implementación lo hacen atractivo tanto para aplicaciones comerciales como de entretenimiento. Además, su distribución gratuita bajo licencia GPL facilita su adopción, sumando estabilidad y un desarrollo acelerado como beneficios clave.

Los autores indican que MySQL ha ganado popularidad gracias a varias características destacadas:

- Está programado en C/C++.
- Se distribuye en versiones para casi diecinueve plataformas diferentes.
- Ofrece una API compatible con C, C++, Eiffel, Java, Perl, PHP, Python, Ruby y TCL.
- Está optimizado para sistemas de múltiples procesadores.
- Su velocidad de respuesta es sobresaliente.
- Puede operar como cliente-servidor o incrustarse en aplicaciones.
- Ofrece una amplia variedad de tipos de datos.
- Soporta múltiples métodos de almacenamiento de tablas, permitiendo optimizar el rendimiento según el caso.

- Su administración se basa en la gestión de usuarios y permisos.
- Es capaz de manejar grandes volúmenes de datos, con casos documentados de cincuenta millones de registros, sesenta mil tablas y cinco millones de columnas.
- Su conectividad incluye TCP/IP, sockets UNIX, sockets NT y soporte completo para ODBC.
- Permite mensajes de error en español y realiza ordenaciones adecuadas para palabras con acentos o la letra ‘ñ’.
- Es altamente confiable en términos de estabilidad.

Para los usuarios familiarizados con UNIX y el lenguaje C/C++, MySQL resulta intuitivo, ya que su diseño sigue la filosofía de UNIX: “crear herramientas que hagan bien una sola cosa”. La meta principal de MySQL es ofrecer una base de datos estable y eficiente, priorizando características que aseguren la mejor velocidad de respuesta sin comprometer la estabilidad. La influencia de C/C++ y UNIX se refleja también en su sintaxis, que incluye el uso de expresiones regulares, diferenciación de funciones por paréntesis, valores lógicos como 0 y 1, y la opción de completar sentencias con el tabulador (Casillas et al., 2010).

1.2.6 Paradigmas de programación

Spigariol (2005) menciona que a lo largo de la evolución de los lenguajes de programación, se han ido incorporando características que no solo contribuyen a crear programas más sólidos y eficientes, sino que también facilitan al programador su desarrollo, mantenimiento y adaptación. Este avance, sin embargo, no ha seguido un camino recto; algunas características básicas introducidas en ciertos lenguajes han sido cuestionadas y criticadas por otros. Esto ha dado lugar a teorías diversas y a distintos grupos de lenguajes que proponen maneras diferentes de abordar la resolución de problemas.

Con el avance de las ciencias de la computación, estos enfoques han derivado en lo que actualmente conocemos como “paradigmas” de programación. Un paradigma de programación representa un modelo fundamental para el diseño y desarrollo de programas, basado en principios o fundamentos aceptados en el ámbito. Es, en esencia, un conjunto de modelos conceptuales que orientan tanto el proceso de diseño como la forma de abordar problemas, influyendo así en la estructura final del programa (Spigariol, 2005).

Clasificación de los paradigmas

Según el autor (Spigariol, 2005), los paradigmas de programación pueden clasificarse de diversas formas, según los criterios que se consideren prioritarios. De acuerdo a los principios básicos de

cada uno en relación con cómo orientan la construcción de soluciones, es posible distinguir entre paradigmas procedimentales y declarativos:

- **Paradigmas procedimentales u operacionales:** Se centran en el método para resolver el problema, especificando paso a paso el proceso para obtener la solución.
- **Paradigmas declarativos:** Enfocados en describir las características que la solución debe cumplir, estos paradigmas definen "qué" se espera lograr, sin necesidad de indicar el "cómo" hacerlo.

Paradigmas procedimentales

También conocidos como operacionales, estos paradigmas se caracterizan por establecer una secuencia ordenada de etapas que permiten resolver un problema. En los lenguajes procedimentales, la codificación incluye instrucciones de control que regulan el flujo de la ejecución mediante decisiones, ciclos y otros mecanismos, formando así diversos "algoritmos" (Spigariol, 2005).

De acuerdo con el autor actúan modificando repetidamente la representación de sus datos, basándose en asignaciones destructivas con efecto de lado. Utilizan un modelo en el que las variables están estrechamente relacionadas con direcciones de la memoria del ordenador. Cuando se ejecuta el programa, el contenido de estas direcciones se actualiza repetidamente, pues las variables reciben múltiples asignaciones, y al finalizar el trabajo, los valores finales de las variables representan el resultado.

"Su mayor dificultad reside en determinar si el valor computado es una solución correcta del problema, por lo que se han desarrollado multitud de técnicas de depuración y verificación para probar la corrección de los problemas desarrollados basándose en este tipo de paradigmas. En otras palabras, se basan en "cómo" lograr la solución" (Spigariol, 2005).

1.2.7 Paradigma Orientado a Objetos

Según el autor Pérez en su publicación "Programación orientada a objetos" (2024), la Programación Orientada a Objetos (POO) es un paradigma en el que un programa se estructura como un conjunto de entidades denominadas "objetos". Estos objetos mantienen su propio estado interno y se comunican entre sí mediante mensajes, con el propósito de:

- modificar su propio estado
- compartir información

- pedir a otros objetos que procesen cierta información

Este enfoque, conocido como Object-Oriented Programming (OOP) en inglés, ofrece una manera de organizar y estructurar programas. Al igual que ocurre con las funciones en la abstracción de datos, los objetos establecen límites de abstracción entre la implementación y el uso de los datos. Similar a diccionarios o funciones de despacho, los objetos responden a solicitudes (mensajes) que reciben de otros objetos para ejecutar ciertos comportamientos. Los objetos disponen de un estado interno que no se puede modificar directamente desde fuera; sólo es accesible mediante las operaciones que el propio objeto permite. En términos prácticos, los objetos pueden considerarse como datos abstractos, Pérez (2024).

Este enfoque sintáctico es común en varios lenguajes orientados a objetos. Sin embargo, su sistema de objetos aporta algo más que facilidad de uso:

- Permite una nueva perspectiva para diseñar programas en los que diversos agentes autónomos interactúan en el sistema.
- Cada objeto encapsula tanto el estado como el comportamiento de una manera que reduce la complejidad.
- Los objetos se comunican entre ellos, y a partir de esa interacción se obtienen resultados útiles.
- No solo intercambian mensajes, sino que también comparten comportamientos con otros objetos del mismo tipo y heredan atributos de otros tipos relacionados.

De acuerdo con el autor Pérez (2024), este paradigma tiene un vocabulario específico que sustenta la metáfora de los "objetos", proporcionando una visión coherente del diseño en la POO.

Conceptos de la Programación Orientada a Objetos

Según Mazón et al. (2015), en su libro "Fundamentos de Programación Orientada a objetos en java" indica que a diferencia de la programación tradicional, la Programación Orientada a Objetos (POO) introduce nuevos conceptos teóricos que se consideran fundamentales, como: objeto, clase, miembros de clase (tanto datos como métodos), mensaje, herencia y otros términos como abstracción, instanciación, reutilización, polimorfismo, encapsulamiento, ocultamiento y especificadores de acceso.

Clase

Una clase es un tipo de dato que el programador define y que agrupa tanto la estructura de datos como las operaciones que se pueden realizar con esos datos. Los miembros de la clase incluyen datos que representan características comunes para todos los objetos de esa clase y métodos que permiten manipular dichos datos. Así, una clase puede verse como un modelo o plantilla que describe uno o más objetos de un tipo específico, tomando como base un conjunto de objetos con atributos y comportamientos similares del mundo real, según Mazón et al. (2015).

Como describen Mazón et al., (2015), la creación de un objeto ocurre cuando se envía un mensaje a la clase para solicitar su generación, lo cual convierte a los objetos en instancias de su clase. El primer método que se activa automáticamente para construir un objeto se llama "constructor".

Un objeto protege ciertos atributos de acceso externo, permitiendo que solo se modifiquen o consulten mediante métodos públicos de lectura ("get") o escritura ("set"). Cada objeto es una entidad encapsulada que, a su vez, interactúa con otros objetos dentro de la aplicación.

De acuerdo con Mazón et al., (2015), el nivel de encapsulamiento en una clase, determinado por el grado de visibilidad de sus miembros, se define mediante especificadores de acceso, que son:

- **Público** ("+"): Los métodos de cualquier clase pueden acceder a estos miembros.
- **Predeterminado** (sin especificador): El acceso se permite solo a métodos de clases que pertenezcan al mismo paquete.
- **Protegido** ("#"): El acceso se restringe a métodos de la misma clase y de sus clases derivadas.
- **Privado** ("-") : El acceso se limita a métodos de la propia clase únicamente.

Objeto

Los autores Mazón et al. (2015), describen que, en un programa orientado a objetos, se trabaja con una colección de entidades o "objetos" que representan elementos del mundo real. Un objeto se define por ciertos atributos o propiedades (datos) y por su comportamiento, es decir, las operaciones que realiza sobre esos datos (métodos). Los datos de un objeto pueden incluir diversos elementos como constantes, variables, arreglos, cadenas, estructuras o incluso otros objetos. Por ejemplo, un auto puede ser un objeto que tiene atributos como el número de puertas, color, tamaño, marca, modelo y año; además, puede ejecutar comportamientos como acelerar, frenar, cambiar de marcha o girar.

De acuerdo con Mazón et al., (2015), en esencia, un objeto puede considerarse como un “encapsulado” que reúne un conjunto de métodos (operaciones) que otros objetos pueden invocar y un estado que retiene los efectos de estas operaciones, es decir, los valores actuales de sus datos. Además de un estado interno, un objeto proporciona una interfaz (métodos públicos) para que otros elementos puedan interactuar con él.

Según los autores Mazón et al. (2015), los componentes de un objeto incluyen:

- **Identidad:** Es el identificador único que distingue a un objeto de otros, independientemente de su estado. Este identificador generalmente deriva del problema que se busca resolver.
- **Tiempo de vida:** La duración de un objeto es limitada dentro del programa; muchos objetos solo existen durante una parte de la ejecución. Se crean mediante un proceso llamado instanciación y se eliminan cuando ya no son necesarios.
- **Estado:** Cada objeto tiene un estado que depende de sus atributos y de los valores que contienen en un momento dado, lo que define sus propiedades actuales.
- **Comportamiento:** La interfaz de un objeto, definida a través de sus métodos, permite que otros objetos en el programa interactúen con él.

La estructura interna de un objeto está protegida y no es accesible desde el resto del programa; la única manera de comunicarse con él es mediante “mensajes”, que son invocaciones a sus métodos.

Mensajes

El autor Joyanes Aguilar en su libro programación orientada a objetos (1996), define que en la Programación Orientada a Objetos, un mensaje es una solicitud de un objeto hacia otro para que ejecute uno de sus métodos. En esta interacción, el objeto que envía la solicitud se denomina emisor, mientras que el que la recibe es el receptor.

De acuerdo con el autor Aguilar (1996), un mensaje se compone de tres elementos:

1. **Identidad del receptor:** el objeto que ejecutará el método.
2. **Método a ejecutar:** la acción específica que se solicita.
3. **Información adicional:** cualquier argumento o parámetro que el método necesite para realizar la operación.

Cuando un objeto, que puede estar en estado inactivo, recibe un mensaje, se activa automáticamente. Este mensaje contiene un método asociado que se ejecuta una vez que el receptor lo recibe. Sin embargo, el emisor no necesita conocer los detalles de cómo el receptor realiza la operación, lo cual permanece oculto (Joyanes, 1996).

La serie de mensajes que un objeto puede responder constituye su comportamiento. No todos los mensajes se pueden responder; deben pertenecer a la interfaz accesible del objeto.

Nombre del Mensaje

Cada mensaje incluye el nombre de la operación a realizar y cualquier argumento que requiera. En ocasiones, es útil hacer referencia a una operación solo por su nombre, sin detallar sus argumentos (Aguilar, 1996).

Métodos

Cuando un objeto recibe un mensaje, realiza la operación solicitada ejecutando un método, que es el algoritmo correspondiente a la operación nombrada en el mensaje. La secuencia es que el emisor envía un mensaje; el receptor, al recibirlo, ejecuta el método correspondiente con los parámetros necesarios y luego responde al emisor, reconociendo el mensaje y devolviendo la información solicitada si corresponde (Aguilar, 1996).

Paso de mensajes

El autor Aguilar (1996), describe que en el paradigma orientado a objetos, la comunicación entre objetos se realiza mediante el envío de mensajes. El mensaje establece una interfaz clara entre el objeto y su entorno. Este protocolo de mensajes requiere que haya dos partes involucradas: el emisor y el receptor. Cuando un objeto (emisor) envía un mensaje a otro (receptor), debe especificar tres elementos:

1. El receptor del mensaje.
2. El nombre del mensaje.
3. Los parámetros o argumentos, si son necesarios.

El receptor debe ser un objeto que pueda procesar el mensaje indicado; aquellos que no sean especificados no responderán. El receptor intenta asociar el nombre del mensaje con los mensajes que tiene definidos en su interfaz. Si el receptor no reconoce el mensaje, no responderá; en

caso contrario, activará el método correspondiente para procesar la solicitud (Aguilar, 1996).

Los parámetros o argumentos pueden incluir:

1. Datos que el método necesita para su ejecución.
2. Otro mensaje en sí mismo.

Herencia

La encapsulación y el ocultamiento de información son características clave en el diseño de objetos, lo que permite que un objeto combine datos con las operaciones que los manipulan. Además, un rasgo importante de la orientación a objetos es la herencia, que permite construir nuevos objetos a partir de otros ya existentes. En otras palabras, los objetos pueden aprovechar las estructuras de datos y los métodos de sus "antepasados" o "clases superiores". Esto facilita la reutilización de código previamente desarrollado (Aguilar, 1996).

De acuerdo con el autor la herencia toma su concepto de la vida cotidiana: las clases principales o básicas se subdividen en subclases. Por ejemplo, "animal" se divide en subclases como mamíferos, anfibios, e insectos. De manera similar, "vehículo" incluye subclases como automóvil, motocicleta, y camión. Esta estructura jerárquica permite compartir características comunes entre clases: todos los vehículos tienen un motor y ruedas, aunque algunos, como los camiones, pueden incluir características adicionales, como una caja para carga, mientras que las motocicletas tienen manillar en lugar de volante.

La herencia se basa en una clase base que da origen a una jerarquía de clases derivadas. Las clases derivadas pueden utilizar tanto el código como los datos de su clase base, además de agregar o modificar estos elementos según sea necesario (Aguilar, 1996).

1.2.8 Java

Martínez en su libro "Fundamentos de programación en JAVA" (2015), describe que Java es un lenguaje de programación creado por Sun Microsystems, y se dio a conocer en la segunda mitad de 1995. Desde entonces, se ha vuelto uno de los lenguajes de programación más populares. Uno de los aspectos más destacados de Java es su capacidad para ejecutarse en múltiples plataformas, ya sea Windows, Mac OS, Linux o Solaris. James Gosling, quien lideró el equipo de desarrollo de Java, logró cumplir la visión de crear un lenguaje que no dependiera de una sola plataforma. La meta era que una aplicación pudiera programarse una vez y después ejecutarse en diversos sistemas operativos y equipos. Esta capacidad de portabilidad se logra mediante el uso del Java Runtime Environment (JRE), un entorno de ejecución para programas compilados,

el cual es gratuito y está disponible para los principales sistemas operativos. Esto permite que el mismo programa Java funcione en diferentes plataformas, como Windows, Mac OS, Linux o Solaris (Martínez, 2015, p. 2).

Martínez (2015), describe el concepto "Escribir una vez, ejecutar en cualquier lugar" resume el propósito del equipo de Java, es decir, crear un código que pueda ser utilizado en distintos sistemas operativos sin modificarlo. Gracias a esta independencia de plataforma, los programas Java pueden ejecutarse en una variedad de dispositivos, ya sea una PC de escritorio, una laptop, una tableta, un teléfono móvil, un reproductor de música o cualquier otro dispositivo con sistema operativo.

Java compila su código a un lenguaje intermedio llamado Bytecode. Este Bytecode es interpretado por la Máquina Virtual de Java (JVM) del JRE, lo cual asegura que el programa funcione en distintas plataformas. El JRE actúa como un puente entre el Bytecode y los diferentes sistemas operativos. Así, un programa Java compilado en Bytecode puede ejecutarse en sistemas como Windows, Linux, Mac OS, Solaris, BlackBerry OS, iOS o Android mediante el JRE correspondiente.

La portabilidad es una característica esencial de los lenguajes de programación modernos. Un programa se considera portable cuando puede ejecutarse en cualquier sistema operativo y dispositivo sin importar la plataforma. Esta característica ha permitido que Java se convierta en una opción popular para el desarrollo de sistemas de comercio electrónico e información en Internet, ya que reduce los costos de desarrollo y mantenimiento y hace que las aplicaciones sean compatibles en varios sistemas operativos (Martínez, 2015).

Java ha evolucionado rápidamente, y su plataforma de desarrollo, conocida como el Java Development Kit (JDK), ha crecido en funcionalidad y en número de desarrolladores en todo el mundo. En realidad, Java es más que un lenguaje de programación; es una plataforma completa que incluye el lenguaje en sí, un entorno de ejecución, y un amplio conjunto de bibliotecas conocidas como Java Application Programming Interface (Java API), para desarrollar aplicaciones complejas (Martínez, 2015).

Pasos para crear y ejecutar un programa en Java

De acuerdo con Deitel et al. (2016), describen las 5 fases para crear y ejecutar un programa de Java, las cuales son:

Fase 1: Creación del programa La primera fase consiste en crear un archivo con el código del programa usando un editor de texto, el cual es comúnmente conocido como editor. En este editor, el programador escribe el código Java, hace las correcciones necesarias y guarda el

archivo en un dispositivo de almacenamiento, como el disco duro. Los archivos de Java llevan la extensión `.java` para indicar que contienen código fuente en este lenguaje. Los entornos de desarrollo integrados (IDE) ayudan en este proceso, ofreciendo herramientas como editores y depuradores para identificar errores lógicos que puedan causar que el programa funcione de manera incorrecta. Algunos IDE populares para Java incluyen Eclipse, NetBeans e IntelliJ IDEA.

Fase 2: Compilación del programa a Bytecode En esta fase, el programador usa el comando `javac`, el compilador de Java, para compilar el programa. Por ejemplo, para compilar un archivo llamado `Bienvenido.java`, se debe escribir `javac Bienvenido.java` en la ventana de comandos del sistema. Si la compilación es exitosa, el compilador generará un archivo `.class` llamado `Bienvenido.class`, que contiene la versión compilada del programa. Generalmente, los IDE incluyen opciones de menú como "Build" o "Make" que ejecutan automáticamente el comando `javac`. En caso de que se detecten errores, se debe volver a la fase anterior para corregirlos.

El compilador de Java transforma el código fuente en Bytecode, un conjunto de instrucciones que la Máquina Virtual de Java (JVM) ejecuta durante la fase de ejecución. La JVM, parte fundamental del JDK y de la plataforma Java, actúa como una computadora virtual, ocultando el sistema operativo y el hardware real al programa que la utiliza. Esto significa que el mismo programa puede ejecutarse en distintas plataformas que implementen la JVM, independientemente del sistema operativo. La JVM es una de las máquinas virtuales más extendidas, y la plataforma `.NET` de Microsoft también usa una arquitectura de máquina virtual similar.

A diferencia del lenguaje máquina, que depende del hardware específico de una computadora, el Bytecode de Java es independiente de la plataforma. Esto hace que los programas Java sean portables: el Bytecode se puede ejecutar en cualquier sistema que tenga una JVM compatible con la versión en la que se compiló el programa. Para ejecutar un programa Java, se utiliza el comando `java`. Por ejemplo, para ejecutar la aplicación `Bienvenido`, se escribe `java Bienvenido` en la línea de comandos, lo cual invoca a la JVM para iniciar el programa. Normalmente, los IDE ofrecen opciones de menú como "Run" para ejecutar automáticamente el programa mediante el comando `java`.

Fase 3: Carga del programa en la memoria En la tercera fase, la JVM carga el programa en la memoria para su ejecución. El cargador de clases de la JVM transfiere los archivos `.class` que contienen el Bytecode del programa a la memoria principal. También carga automáticamente cualquier otro archivo `.class` que el programa requiera y que esté en la biblioteca de Java.

Fase 4: Verificación del Bytecode A medida que se cargan las clases, el verificador de Bytecode revisa las instrucciones para asegurar que cumplen con las restricciones de seguridad de Java. Esto garantiza que los programas Java descargados de internet no puedan dañar los archivos o el sistema del usuario, previniendo ataques similares a los de los virus informáticos y gusanos.

Fase 5: Ejecución del programa Finalmente, en la fase 5, la JVM ejecuta el Bytecode del programa, llevando a cabo las instrucciones que este contiene. En las primeras versiones de Java, la JVM interpretaba el Bytecode una instrucción a la vez, lo cual hacía que la ejecución de los programas fuera relativamente lenta. Sin embargo, las JVM modernas utilizan una combinación de interpretación y "compilación en tiempo de ejecución" (JIT). En este proceso, la JVM analiza el Bytecode en busca de secciones que se ejecutan frecuentemente, conocidas como "puntos activos". Para estas secciones, un compilador JIT, como el HotSpot™ de Oracle, traduce el Bytecode directamente a lenguaje máquina, el cual es más rápido de ejecutar. De esta manera, los programas en Java pasan por dos fases de compilación: una inicial que convierte el código fuente en Bytecode para que sea portable, y una segunda durante la ejecución, en la cual el Bytecode se convierte a lenguaje máquina para la computadora específica en la que se está ejecutando.

1.3 Área de aplicación

1.3.1 Psicología

William James, (1890) considerado uno de los fundadores de la psicología moderna, definió esta disciplina como "la ciencia de la vida mental, tanto de sus fenómenos como de sus condiciones". En su visión, la psicología abarca el estudio de la mente y los procesos mentales, explorando no sólo las experiencias conscientes, como pensamientos y emociones, sino también los mecanismos y condiciones biológicas que las sustentan. James entendía la vida mental como un proceso complejo que conecta aspectos físicos y psicológicos, reconociendo así la relación entre el cuerpo y la mente, y cómo ambos influyen en el comportamiento y la experiencia humana. Su enfoque fue fundamental para el desarrollo del funcionalismo, corriente que estudia las funciones de la mente en la adaptación al entorno.

Psicología clínica

La psicología clínica es una especialidad dentro de la psicología que se centra en el estudio, diagnóstico, tratamiento y prevención de una variedad de problemas psicológicos y emocionales, como los trastornos mentales, conductuales y de ajuste. Los psicólogos clínicos aplican teorías científicas y prácticas basadas en evidencia para comprender y abordar el malestar psicológico, mejorando así la calidad de vida y la salud mental de las personas. Esta disciplina se basa en una combinación de investigación científica y práctica terapéutica, y abarca intervenciones diversas, desde evaluaciones diagnósticas hasta terapias individuales, familiares y grupales (American Psychological Association, 2018).

Además de la intervención directa con los pacientes, la psicología clínica incluye la consultoría, la supervisión de otros profesionales de la salud, y la investigación para mejorar los métodos de tratamiento. Esta especialidad se practica en contextos diversos, como hospitales, clínicas, instituciones de salud mental, y entornos comunitarios, promoviendo la adaptación personal, el bienestar emocional, y la resiliencia en individuos y comunidades (American Psychological Association, 2018).

1.3.2 Expedientes clínicos

De acuerdo con Ornelas (2013) en su libro “El expediente clínico” menciona todos los criterios que debe llevar, definiendo según la NOM- 004-SSA3, (2012) al el expediente clínico como una herramienta clave que garantiza el derecho a la salud al contener la información personal y médica única de cada paciente. Este conjunto de registros puede incluir documentos en formatos diversos, como escritos, gráficos, imágenes y archivos electrónicos, que documentan de manera continua y cronológica las intervenciones realizadas por el personal de salud. Estos registros describen no sólo el estado de salud físico del paciente, sino también su bienestar mental y social, proporcionando una visión integral de su situación.

La normativa vigente establece criterios científicos, éticos y administrativos para la creación, manejo y resguardo del expediente clínico, asegurando su confidencialidad y la propiedad de los datos por parte del paciente. Esta norma busca mejorar la calidad de los servicios médicos y los resultados en la atención al paciente, promoviendo un proceso de atención profesional, respetuoso y adecuado en los sectores de salud público, privado y social. Además, garantiza que el paciente tenga control sobre los tratamientos que recibe mediante el consentimiento informado y protege su derecho a la privacidad sobre su información personal, reforzando el principio ético del secreto profesional en los servicios de salud.

Expediente clínico electrónico

De igual forma Ornelas (2013), el expediente clínico electrónico es un registro digital de la historia médica de un paciente que reúne antecedentes clínicos esenciales, notas de consulta, resultados de laboratorio, datos administrativos y demás información necesaria para construir un perfil médico completo.

Entre las ventajas de un expediente clínico electrónico el autor Ornelas (2013), encuentra:

1. **Rapidez:** En un entorno donde la información se difunde rápidamente, el ámbito médico también se beneficia de la rapidez en la gestión de datos, mejorando la atención al paciente al reducir tiempos y movimientos innecesarios.

2. **Capacidad de almacenamiento:** Este sistema electrónico permite manejar una base de datos extensa, capaz de almacenar mucha más información que los métodos tradicionales. Además, ayuda a optimizar espacio, recursos de personal y papelería, y facilita el acceso simultáneo a los expedientes desde distintas oficinas. También puede ser consultado en cualquier momento y desde ubicaciones remotas.
3. **Seguridad:** Los registros electrónicos garantizan que sólo el personal autorizado pueda acceder a los expedientes, lo cual refuerza la protección y confidencialidad de la información médica.
4. **Accesibilidad:** Los usuarios pueden revisar la información desde dispositivos móviles o computadoras en el consultorio, lo cual facilita el acceso directo.
5. **Optimización de la infraestructura:** Al implementar expedientes clínicos electrónicos, los hospitales pueden mejorar el uso de sus instalaciones y ampliar el acceso a la información, que ya no se limita únicamente al personal médico.
6. **Eficiencia:** Aunque la eficiencia no es lo mismo que la velocidad, este sistema agiliza la gestión de información, lo que reduce los tiempos de manejo y los costos asociados a los archivos tradicionales.
7. **Adaptabilidad:** Esta tecnología destaca por su diseño flexible y capacidad de personalización según las necesidades del usuario o la institución. Facilita nuevas formas de organización y gestión del conocimiento, permitiendo una transición a historiales clínicos estructurados y potenciando su uso en la investigación científica. Además, mejora la organización de horarios, citas, estudios de laboratorio y el transporte de información médica en formatos ecológicos.

Ficha de identificación

La ficha de identificación deberá incluir, como mínimo, los datos personales del paciente, tales como su grupo étnico cuando corresponda, antecedentes familiares de enfermedades, antecedentes personales de salud (incluyendo el consumo de tabaco, alcohol y otras sustancias psicoactivas, de acuerdo con las normas específicas de salud), antecedentes no patológicos, y detalles del padecimiento actual, considerando tratamientos previos convencionales, alternativos o tradicionales. Asimismo, debe incluirse una revisión de síntomas y funcionamiento de los distintos sistemas corporales.

Esta información se registra conforme a la **NOM-004-SSA3-2012**, que regula el contenido mínimo de los expedientes clínicos en México.

Contenido del informe psicológico

De acuerdo con Heredia et al. (2012), un informe psicológico es un documento clínico que presenta de manera estructurada los resultados de la evaluación, detallando los antecedentes de la persona, sus problemas actuales, ventajas, y limitaciones, junto con observaciones conductuales e interpretaciones de las pruebas aplicadas. La relevancia de este informe radica en su capacidad para responder de forma efectiva a la pregunta de canalización o derivación que motivó la evaluación. La organización de este informe depende del enfoque teórico del profesional; algunos psicólogos lo estructuran conforme a un marco teórico específico, mientras que otros lo hacen según los resultados obtenidos o los objetivos de la evaluación. No existe un modelo único de informe, ya que debe adaptarse al contexto y a las características del evaluado (como su edad) y a los objetivos de la evaluación, ya sea para selección, diagnóstico, diseño de intervención o seguimiento.

Heredia et al. (2012), citando a Reyes et al. (1987) proponen, desde el modelo psicodinámico, una estructura para organizar los datos recogidos en la evaluación clínica. Su enfoque integra información obtenida a través de diversas fuentes, como la observación, entrevistas, historia clínica, pruebas psicológicas y técnicas proyectivas. Para la elaboración de un informe psicológico, sugieren incluir elementos como:

- **Ficha de identificación:** Contiene datos personales del paciente (nombre, edad, ocupación, etc.).
- **Objetivos del reporte:** Explica el motivo de la evaluación y las expectativas de la misma.
- **Fecha y técnicas:** Enumera las herramientas empleadas para la evaluación.
- **Descripción del paciente:** Incluye observaciones sobre su apariencia y actitud durante la evaluación.
- **Antecedentes:** Resume aspectos relevantes de su historia de vida (familia, desarrollo, salud, educación).
- **Resultados e interpretación:** Analiza las pruebas aplicadas en función de las áreas evaluadas.
- **Impresión diagnóstica:** Ofrece un diagnóstico integrador, destacando la organización de la personalidad del evaluado.
- **Pronóstico y sugerencias de intervención:** Propone recomendaciones realistas y un pronóstico basado en los hallazgos.
- **Firma del profesional:** Avala la responsabilidad ética del evaluador.

Esta estructura organiza la información en áreas específicas del funcionamiento y la personalidad del paciente, adaptándose a sus necesidades y al objetivo de la evaluación.

1.4 Estado del arte

Las investigaciones sobre el desarrollo de sistemas de gestión de expedientes clínicos son diversas y se encuentran en diferentes fuentes bibliográficas. A continuación se expondrán aquellas investigaciones que ofrecen un aporte significativo al tema en cuestión y que, a la vez, servirán para que se vislumbre la trayectoria que llevará la siguiente investigación.

Inicialmente Rivera et al. (2021) en su artículo realizan un estudio para desarrollar una arquitectura de información el cual tiene como objetivo presentar un prototipo de un historial clínico digital para ser utilizado como consulta provisional en el área de Oftalmología Pediátrica y Estrabismo en Pinar del Río.

Se utilizaron métodos teóricos y empíricos para obtener la evolución de la historia clínica en la Oftalmopediatría, utilizando como herramientas computacionales la plataforma LAMP (Linux, Apache, MySQL, PHP) para cumplir con todo lo que debe de tener una aplicación web: un sistema operativo, un servidor web, una base de datos y un lenguaje de programación.

Dando como resultados un prototipo de 5 pantallas principales en las cuales contiene un inicio de sesión para el doctor que da la consulta, y una general que contiene las otras 4 interfaces para los datos del paciente en la cual se preguntan los antecedentes del paciente con su enfermedad, se añaden la información de los resultados de los test que se aplican al paciente, su diagnóstico y tratamiento que va a llevar el paciente.

A modo de conclusión Rivera et al. (2021) nos indica las características básicas que tiene un historial clínico para ser transportado a digital teniendo en cuenta los antecedentes del paciente, el registro de pruebas que se lleven en la consulta así como el diagnóstico y el tratamiento que se va a llevar consigo para poder tener un correcto seguimiento del paciente.

Así mismo Bajaña (2018) en su tesis realiza un proyecto de investigación el cual tiene como objetivo diseñar un software para la clínica odontológica Albo Dental de la ciudad de Guayaquil en el cual se registre la historia clínica de los pacientes.

Se utilizó una investigación descriptiva correlacional para obtener los requerimientos del sistema utilizando encuestas y cuestionarios creados por el autor que se aplicaron a 48 pacientes de la clínica y los 2 empleados que están a cargo de la misma. Para que al finalizar el análisis de estos datos dar con los diferentes diagramas y tablas para complementar el diseño del software teniendo diagramas de entidad-relación, diagramas de casos de uso, diccionarios de datos

y diagramas de flujo para la conceptualización del sistema.

El resultado de este trabajo es una aplicación la cual cuenta con 6 interfaces, las cuales se componen de un inicio de sesión para el odontólogo, una ficha de identificación del paciente, el tratamiento que se va a llevar sobre el mismo y la posibilidad de agendar citas para tener un óptimo control sobre las consultas que se van a tener en el consultorio.

A modo de conclusión Bazaña (2018) realza lo importante de un cuidadoso y meticuloso diseño de una aplicación de gestión de historiales clínicos teniendo en cuenta las opiniones de los que van a ser integrados por el sistema para tener unos requerimientos completos y en los cuales se pueda basar el sistema para tener un correcto funcionamiento del mismo.

Por otro lado, Fernández (2022) en su tesis presenta un informe de investigación en el cual propone una manera de mejorar la gestión de las consultas ambulatorias en el Hospital “Belén” de Lambayeque que tiene como objetivo desarrollar una aplicación web de Historias Clínicas electrónicas para mejorar este proceso de consultas, utilizando la metodología Extreme Programming (XP).

Se utilizó la correcta estructuración de las fases y estándares ofrecidos por la metodología para obtener el correcto análisis y diseño del sistema, implementándolo con el gestor de base de datos PostgreSQL y el lenguaje de programación PHP con librerías de AJAX y JavaScript para llevar a cabo la codificación web y tener un resultado más dinámico.

Además de que al implementar el sistema web se llevó a cabo una validación por el mismo hospital a base de cuestionarios realizados por el autor que se aplicaron al personal del hospital.

Obteniendo como resultado el 50 % de reducción en el tiempo de espera en la atención de las consultas y un 96 % de reducción de tiempo para sacar citas a los pacientes, reduciendo errores de control y seguimiento de los pacientes así como la pérdida o extravió de estos expedientes.

A modo de conclusión Fernández (2022) indica que resulta eficiente la metodología propuesta para la realización de este tipo de sistemas ya que su correcta utilización da resultados óptimos en el desarrollo y validación del software de gestión de expedientes clínicos.

Por su cuenta Evangelista et al. (2023) en su artículo, a base de una investigación aplicada al desarrollo tecnológico, describen el desarrollo de un prototipo de software basado en la cartilla de salud del adulto mayor usando la evaluación Heurística para analizar la usabilidad del software, teniendo como objetivo optimizar e incentivar el uso de esta herramienta electrónica en las rutinas de los servicios de salud, utilizando una metodología ágil.

Utilizando las 4 fases recurrentes de su metodología se pudo generar un prototipo navegable

utilizando la herramienta MockFlow para el diseño de interfaces amigables para el usuario. Se usaron las 10 heurísticas de Nilsen las cuales describen como debe de ser la correcta interacción entre usuarios, programas y aplicaciones permitiendo identificar los problemas que se crean durante el desarrollo permitiendo que sean revisados y adaptados para obtener una mejor calidad de usabilidad para los clientes finales.

Así es que como resultado se presentó un prototipo de 5 pantallas en el cual tanto los adultos mayores o el personal de salud pueden crear y editar los datos de las cartillas, dando primero un inicio de sesión y después la visualización de la cartilla del paciente o la visualización y búsqueda de todas las cartillas registradas, así cumpliendo con las 10 heurísticas de Nilsen para tener una mejor fiabilidad en la ocupación del software.

A modo de conclusión Evangelista et al. (2023) rescata la importancia de la evaluación de los profesionales de la salud que van a ser quienes ocupen el sistema ya que gracias a estas evaluaciones se puede llevar una gran mejora en no solo la fiabilidad y usabilidad del sistema sino también en estar mas cerca de los requerimientos reales que se usan cuando estos registros se llevan realmente.

Adicionalmente Carvajal y Solano (2020) en su tesis presenta una solución informática a la unidad médica Family Care que tiene como objetivo desarrollar una aplicación web el cual permita el control de citas y expedientes clínicos para garantizar la fiabilidad y seguridad en el manejo de los datos almacenados de los pacientes, utilizando la metodología Extreme Programming (XP) para el desarrollo del mismo.

Se utilizó la correcta estructuración de las fases y pasos de la metodología para obtener un correcto desarrollo del mismo, haciendo diagramas de casos de uso, modelados de entidad relación para el manejo de la base de datos, así como los entregables de la metodología como las historias de usuario y las tarjetas CRC, para así utilizar un proveedor de servidor web que usa Apache y el lenguaje PHP para el obtener ventajas en la usabilidad y estabilidad de la aplicación web.

Se obtuvo como resultados una aplicación la cual cuenta con 6 módulos en los que interactúan un administrador, una secretaria, los médicos y los pacientes. Esta plataforma cuenta con un amplio entorno en el cual no solo se registran los pacientes y su historial clínico, sino que también cuenta con las funciones para poder reportar las tareas siguientes a hacer y citas con los pacientes.

A modo de conclusión Carvajal y Solano (2020) lo importante que estas soluciones informáticas ayudan a esta área de salud, ya que al ser áreas que regularmente no tienen una sistematización en la forma en daré consultas este tipo de proyectos ayudan inmensamente a seguridad, eficiencia y normalización de llevar el seguimiento de los pacientes.

Capítulo 2

Marco Metodológico

En el contexto del desarrollo de la ingeniería de software, Porras (2023) expone que al principio no existía un cuerpo teórico exclusivo para esta disciplina, por lo que fue necesario integrar componentes de otras áreas, como la administración y la ingeniería de sistemas e industrial, adaptándolos a las particularidades del desarrollo de software.

De acuerdo con Shenhar y Dvir (como se citó en Nossa, 2021), la naturaleza amplia de los estándares y normas en la gestión de proyectos ha hecho posible la inclusión de metodologías como fuentes fundamentales de conocimiento. Estas metodologías permiten que los sistemas de ejecución se adapten mejor a las necesidades específicas de cada intervención. Por lo general, la práctica de la dirección de proyectos ha sido orientada por algún cuerpo de conocimiento, mediante la implementación de metodologías y el uso de herramientas y técnicas.

2.1 Metodologías tradicionales

Según Restrepo et al. (2019), estas metodologías aportan una estructura disciplinada al proceso de desarrollo de software, con el fin de alcanzar una mayor eficiencia y previsibilidad en el producto final. Este enfoque es visto como el más tradicional debido a que fue el primero utilizado en el desarrollo de software. Requiere una planificación detallada de todo el trabajo previo al inicio del desarrollo.

2.1.1 Cascada

Según lo indicado por Maida y Pacienza (2015), el modelo en cascada en el desarrollo de software se llama así debido a que las etapas del proceso se desarrollan de forma consecutiva, como si fluyeran por gravedad de una a otra. Este enfoque metodológico organiza el desarrollo de software en fases estrictamente secuenciales, en las cuales cada etapa debe esperar a que la

anterior se haya completado antes de comenzar. Al finalizar cada fase, se realiza una revisión que evalúa si el proyecto está listo para continuar a la siguiente. Este fue el primer modelo de ciclo de vida y la base de todos los demás.

Esta metodología tradicional se compone por seis fases según Maida y Pacienza (2015), las cuales son:

1. Ingeniería y Análisis del Sistema

- **Objetivo:** Establecer los requisitos del sistema global.
- **Proceso:**
 - Identificar y definir los requisitos de todos los elementos del sistema.
 - Asignar un subconjunto de estos requisitos al software específico.

2. Análisis de los Requisitos del Software

- **Objetivo:** Recopilar y detallar los requisitos específicos del software.
- **Proceso:**
 - Intensificar el enfoque en los requisitos del software.
 - El ingeniero de software (analista) debe comprender:
 - * El ámbito de la información del software.
 - * Las funciones, rendimiento y interfaces requeridas.

3. Diseño

- **Objetivo:** Crear una representación detallada del software basada en los requisitos.
- **Atributos del Diseño:**
 - Estructura de los datos.
 - Arquitectura del software.
 - Detalle procedimental.
 - Caracterización de la interfaz.
- **Proceso:** Traducir los requisitos en un diseño que cumpla con la calidad esperada antes de comenzar la codificación.

4. Codificación

- **Objetivo:** Convertir el diseño en código ejecutable por la máquina.
- **Proceso:**
 - Traducir el diseño a una forma legible para la máquina.

- Realizar la codificación de manera mecánica si el diseño es detallado.

5. Prueba

- **Objetivo:** Verificar que el código funcione correctamente.
- **Proceso:**
 - Evaluar la lógica interna del software.
 - Realizar pruebas funcionales para asegurar que las entradas definidas produzcan los resultados esperados.

6. Mantenimiento

- **Objetivo:** Adaptar y corregir el software post-entrega.
- **Proceso:**
 - Realizar cambios debido a errores encontrados.
 - Adaptar el software a cambios en el entorno externo (sistemas operativos, dispositivos periféricos).
 - Implementar ampliaciones funcionales o mejoras en el rendimiento según las necesidades del cliente.

2.1.2 Microsoft Solution Framework

En el trabajo de Ortiz (2013), se expone que Microsoft Solution Framework (MSF) es una metodología que estructura los planes de desarrollo de software con un enfoque claro y específico. A diferencia de metodologías tradicionales que se basan en un solo modelo, MSF fusiona las ventajas del modelo cascada y el modelo espiral, creando así una metodología práctica y adaptable a proyectos tecnológicos.

Es posible aplicar este enfoque en proyectos cuyos requerimientos del usuario son claramente definidos y permanecen constantes al inicio, lo cual permite una mejor planificación de los recursos. No obstante, el proceso se complica cuando dichos requerimientos cambian en etapas posteriores. A diferencia del modelo en cascada, el modelo espiral se caracteriza por la flexibilidad en los requerimientos, permitiendo que estos se ajusten continuamente a lo largo del desarrollo, como señala Ortiz (2013), basándose en la retroalimentación del cliente en cada fase.

El Microsoft Solution Framework (MSF) se compone de 5 fases, según Ortiz (2013), las cuales son:

Primero, se establece la **Estrategia, Visión y Alcance**. En esta fase inicial, se define una visión clara del sistema a desarrollar, involucrando al cliente en la identificación de problemas actuales y las posibles soluciones. Los objetivos se trazan de manera que el equipo y el cliente

comprendan las metas a lograr dentro del tiempo establecido. Entre las tareas principales se encuentran verificar los requisitos y la funcionalidad del sistema, comprobando si las metas se cumplen.

Luego, en la **Planificación del Proyecto**, se detallan aspectos fundamentales como duración, recursos y características. Este proceso se apoya en tres subfases principales: el diseño conceptual, lógico y físico.

- **Diseño Conceptual:** Aquí se esboza una visión general de la solución que se discute con el cliente. Se documentan los requerimientos, se modela una solución inicial y se preparan las bases para la planificación detallada. Este diseño ayuda a definir qué se entregará al final del proyecto.
- **Diseño Lógico:** Este paso toma la documentación del diseño conceptual y la desarrolla con mayor detalle, identificando los modelos necesarios y las tecnologías adecuadas. Se busca simplificar la complejidad del sistema, facilitando la coordinación entre los subsistemas y asegurando que los objetivos del proyecto se cumplan.
- **Diseño Físico:** Finalmente, el diseño físico traduce los modelos lógicos en implementaciones reales. Se crean bocetos de la interfaz de usuario y de los componentes físicos del sistema, seleccionando las tecnologías específicas y preparando el terreno para el desarrollo efectivo del software.

A continuación, en la fase de **Desarrollo del Proyecto**, los programadores transforman los documentos en código, guiados por las especificaciones y cronogramas establecidos en las fases previas. Es crucial que las tareas asignadas sigan un calendario definido, con entregas claras para cada desarrollador.

En la fase de **Estabilización**, el proyecto se prueba en entornos que simulan situaciones reales, corrigiendo errores y optimizando el sistema. Una vez que se minimizan los problemas, se lanza una versión beta para validar el desempeño del sistema antes de su implementación final.

Finalmente, la **Implementación o Despliegue del Proyecto** asegura que el sistema se instale correctamente en el entorno del cliente, verificando su funcionamiento. Se proporciona toda la documentación necesaria, como manuales de usuario y guías de soporte, que permiten evaluar el proyecto y cerrar las actividades relacionadas.

Cada fase incluye una serie de entregables y documentación, desde reportes de análisis hasta guías de usuario, que ayudan a mantener el control y asegurar que el proyecto cumpla con los requisitos establecidos y las expectativas del cliente.

2.2 Metodologías Ágiles

La idea de agilidad en el desarrollo de software fue promovida por varios expertos, quienes intentaron hacer más eficiente un proceso que históricamente era muy rígido y demandaba mucha documentación. Esto se consolidó en lo que se conoce como el manifiesto ágil, un escrito donde se explican los principios de esta filosofía (de Jiménez, 2015).

Como menciona de Jiménez (2015), el manifiesto ágil pone énfasis en la prioridad de las personas y la interacción del equipo sobre los procesos y herramientas, así como en la importancia de desarrollar software que funcione en lugar de enfocarse en la documentación. También se destaca la preferencia por la colaboración con el cliente frente a la negociación de contratos, y la capacidad de adaptarse a los cambios por encima de seguir un plan estricto. Esta metodología se basa en doce principios que ayudan a simplificar el desarrollo y responder de manera ágil a los cambios, siempre con la participación del cliente.

García Rodríguez (2015) sostiene que el ciclo de vida ágil es recomendable para proyectos de software debido a que los cambios en los requisitos son vistos como algo natural y positivo. En lugar de definir todos los requisitos de antemano y luego controlar los cambios, el enfoque ágil permite adaptarse a ellos durante todo el proyecto. Este enfoque, también llamado “paraguas ágil”, se caracteriza por ciclos de desarrollo incrementales, cooperación continua entre el cliente y el equipo, un método sencillo y flexible, y una alta capacidad de adaptación a nuevos requisitos.

2.2.1 Metodología Scrum

Una de las metodologías ágiles más conocidas es Scrum que busca fomentar la colaboración entre los miembros del equipo de desarrollo, promoviendo un enfoque creativo y eficiente para la creación de productos. Este método, conocido como holístico o “rugby”, se inspira en el deporte donde los jugadores avanzan juntos como una unidad, tal como lo describen Nonaka y Takeuchi, (como es citado por de Jiménez 2015).

Esta metodología ofrece varias ventajas como la gestión constante de las expectativas del cliente, resultados anticipados, flexibilidad y adaptación, así como un retorno de inversión eficiente. Según lo explica García Rodríguez (2015), también proporciona un equipo motivado, mitigación de riesgos y un alineamiento claro entre el cliente y el equipo de desarrollo, incrementando la productividad y la calidad.

En la metodología Scrum, los roles fundamentales son representados por tres componentes clave: el Scrum Master, el Product Owner, y el Team. La Figura 2.1 ilustra la interacción y responsabilidad de estos roles dentro de un equipo de desarrollo.

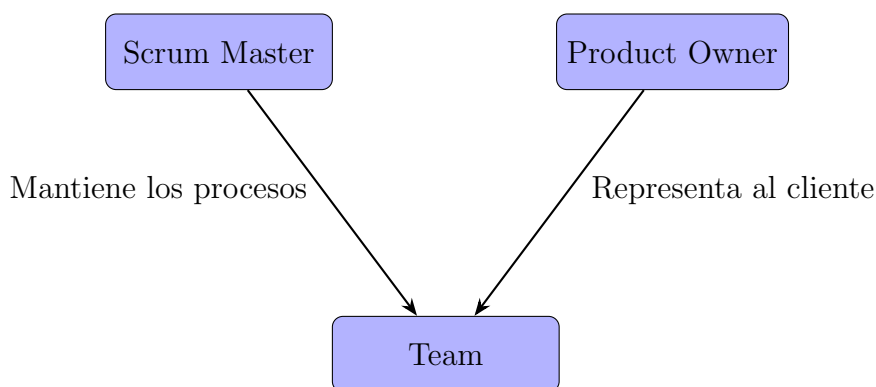


Figura 2.1: Roles principales en Scrum *Fuente: Elaboración propia*

1. **Scrum Master:** Es el responsable de mantener los procesos ágiles funcionando correctamente. Actúa como facilitador y guía del equipo, asegurándose de que todos los miembros sigan las buenas prácticas de Scrum. También trabaja en estrecha colaboración con el jefe de proyecto para eliminar cualquier obstáculo que afecte al progreso del equipo.
2. **Product Owner:** Representa al cliente dentro del equipo Scrum. Su principal función es asegurarse de que el equipo trabaje en las tareas que más valor aportan al producto desde la perspectiva del negocio. El Product Owner se encarga de gestionar el backlog del producto, priorizando las funcionalidades según las necesidades del cliente.
3. **Team:** Este grupo está compuesto por los desarrolladores y otros profesionales que son responsables de construir el producto. El equipo Scrum trabaja de manera auto-organizada y colaborativa, siendo capaz de adaptarse a los cambios y producir incrementos funcionales del producto.

Como señala Rodríguez (2015), el desarrollo en Scrum se organiza en ciclos iterativos llamados sprints, los cuales tienen una duración de entre dos a cuatro semanas, definida y respetada por el equipo. Al final de cada sprint, el equipo entrega un incremento funcional del software. Las funcionalidades que se trabajarán en cada sprint son seleccionadas del product backlog, una lista de requisitos priorizados en función de su valor y el tiempo estimado para completarlos. Esta selección se decide en una reunión de planificación de sprint.

En la reunión de planificación de Scrum, el Product Owner revisa con el equipo los elementos del product backlog que considera prioritarios para la siguiente iteración. El equipo se compromete a completar una cantidad de trabajo específica durante ese periodo, generando así el sprint backlog. Según Rodríguez (2015), durante la iteración, el backlog no puede ser modificado, lo que congela los requisitos. Diariamente, el equipo realiza reuniones breves de unos 15 minutos para compartir avances y ajustar el sprint backlog según corresponda. Al concluir la iteración, el equipo presenta el incremento del software a los stakeholders para su validación, antes de

iniciar una nueva iteración.

2.2.2 Metodología Extreme Programming

Extreme Programming (XP) es una metodología que pone al cliente como un integrante fundamental del equipo de desarrollo, asegurando una colaboración constante y un flujo de retroalimentación continua. Su objetivo principal es reforzar las relaciones interpersonales dentro del equipo de desarrollo. Esta metodología, creada por Kent Beck (como es citado por de Jiménez 2015), resulta adecuada para proyectos con requisitos que tienden a ser inciertos y cambiantes.

Según Rodríguez (2015), los roles que se cumplen en la metodología XP son lo expuesta en la Tabla 2.1:

Rol	Descripción
Programador	Se encarga de escribir el código del sistema y de realizar las pruebas unitarias correspondientes.
Cliente	Es responsable de definir las historias de usuario y de crear las pruebas funcionales para validar su implementación. Además, establece las prioridades de las historias y decide cuáles se desarrollan en cada iteración, buscando maximizar el valor comercial.
Tester	Se especializa en realizar las pruebas y colabora con el cliente en la elaboración de las pruebas funcionales. También comunica los resultados al equipo y gestiona las herramientas de soporte para pruebas.
Tracker	Supervisa el proyecto, controlando las estimaciones realizadas y el tiempo efectivo invertido. Monitorea el progreso de cada iteración y evalúa si los objetivos son alcanzables dentro de los límites de tiempo y recursos, sugiriendo los ajustes necesarios.
Entrenador	Actúa como experto en la metodología XP, guiando al equipo en la aplicación correcta de las prácticas y en la adhesión al proceso.
Consultor	Un especialista externo que aporta conocimientos específicos sobre una materia o tecnología clave para el proyecto, apoyando al equipo en su implementación.
Gestor	Sirve como enlace entre el cliente y los programadores, coordinando el trabajo y facilitando un entorno de trabajo efectivo para que el equipo alcance sus objetivos.

Tabla 2.1: Roles en la metodología XP *Fuente: Elaboración propia*

Como describe de Jiménez (2015) en su libro el ciclo de vida de la metodología XP se estructura en diversas fases interrelacionadas, lo que facilita el desarrollo del proyecto, como se ilustra en la Figura 2.2

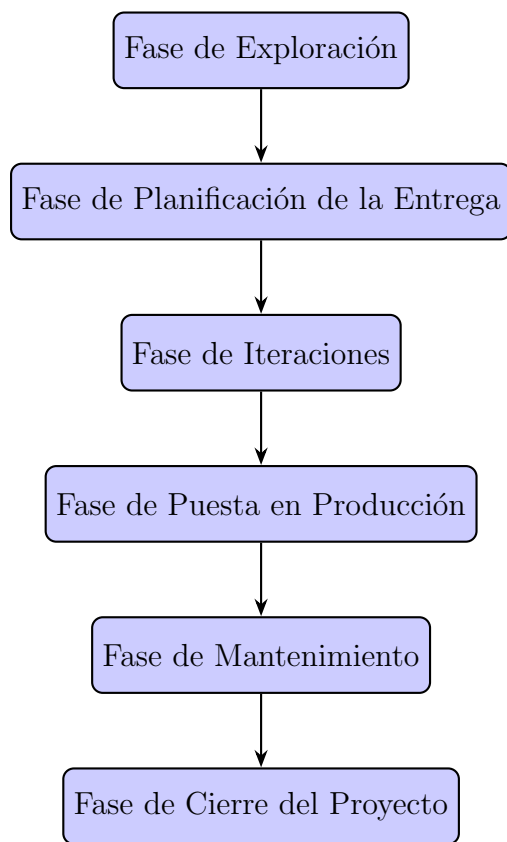


Figura 2.2: Fases de la metodología XP *Fuente: Elaboración propia*

Fase de exploración: Durante esta etapa, los clientes delinean los requisitos, que se expresan como historias de usuario. El equipo de desarrollo comienza a entender la lógica del negocio y a conocer en profundidad el proyecto. La duración de esta fase depende de la cantidad de programadores y su familiaridad con el sistema que se pretende crear.

Fase de planificación de la entrega: En esta etapa, el cliente asigna prioridades a cada historia de usuario y el equipo de desarrollo estima el esfuerzo necesario para implementar cada una en puntos, equivalentes generalmente a una semana de trabajo. La suma de puntos por iteración determina la velocidad de trabajo del equipo, y esta fase establece el cronograma, las entregas y el resultado esperado, conformando así el plan de entregas.

Fase de iteraciones: El plan de entrega se organiza en iteraciones de aproximadamente tres semanas cada una. En la cual se evalúa:

- Historias de usuario no abordadas
- La velocidad del proyecto

- Pruebas de aceptación no aceptadas en la iteración anterior
- Tareas no terminadas en la iteración anterior

Fase de puesta en producción: En esta fase se llevan a cabo pruebas adicionales y verificaciones de rendimiento antes de implementar el software en el entorno del cliente. A menudo, se integran nuevas características sugeridas por el cliente, que deben documentarse para futuras implementaciones.

Fase de mantenimiento: Aquí se revisa el rendimiento y se llevan a cabo pruebas adicionales antes de entregar el software final. Las propuestas o sugerencias deben documentarse para su posible inclusión. Tras la entrega de la primera versión, el proyecto XP continúa con las siguientes iteraciones, donde el soporte al cliente resulta esencial.

Fase de cierre del proyecto: En esta etapa final, ya no existen nuevas historias de usuario, por lo que se procede a generar la documentación definitiva del sistema. Esta fase, conocida como “cierre del proyecto”, puede ocurrir si el sistema no cumple los objetivos o si no hay financiamiento para su mantenimiento.

Para poder tener documentos que nos ayuden a la realización de las fases de estas metodologías de Jiménez (2015) explica que existen 4 entregables esenciales para esta metodología, las historias de usuario, las tareas de ingeniería, las pruebas de aceptación y las tarjetas de CRC cada una de estas describe detalladamente desde los requerimientos de parte del usuario como un diseño del sistema que auxilia a la hora de tener reuniones en el cliente y ayudarlo a comprender como es que funciona el sistema propuesto para que el pueda hacer observaciones o propuesta y así mejorar la integridad del proyecto. En las Figuras 2.3 a 2.6 se explica detalladamente que es lo que debe de tener cada uno de estos entregables.

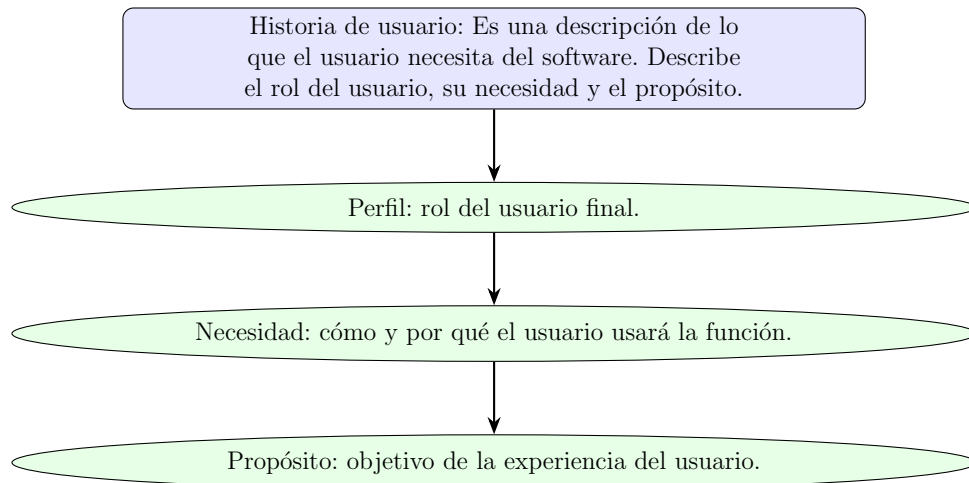


Figura 2.3: Historia de Usuario *Fuente: Elaboración propia*

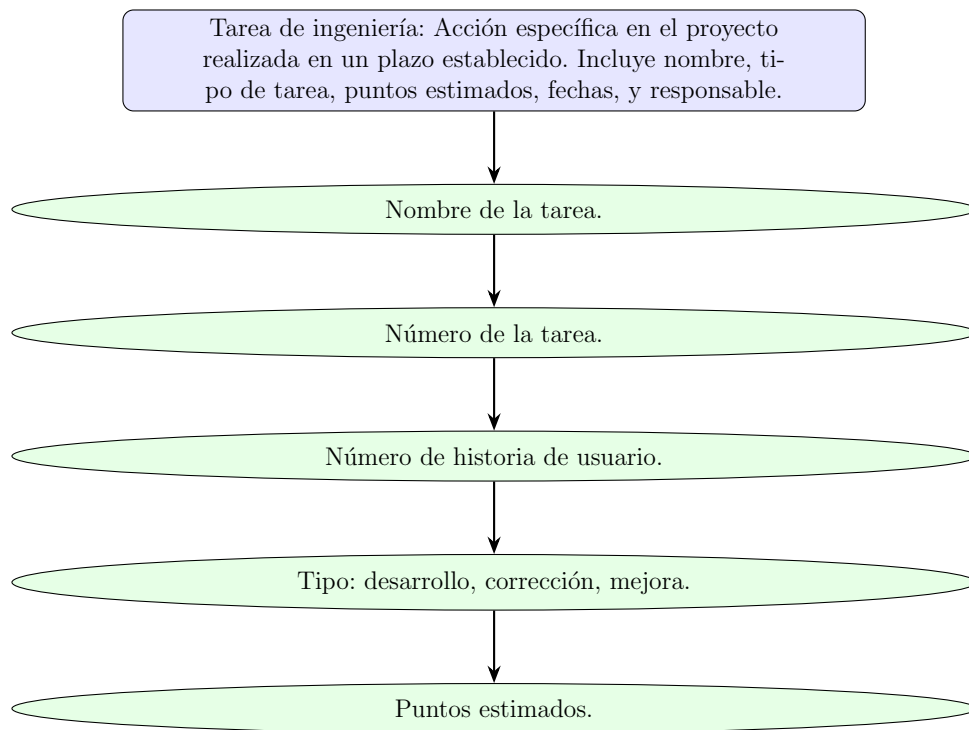


Figura 2.4: Tarea de Ingeniería *Fuente: Elaboración propia*

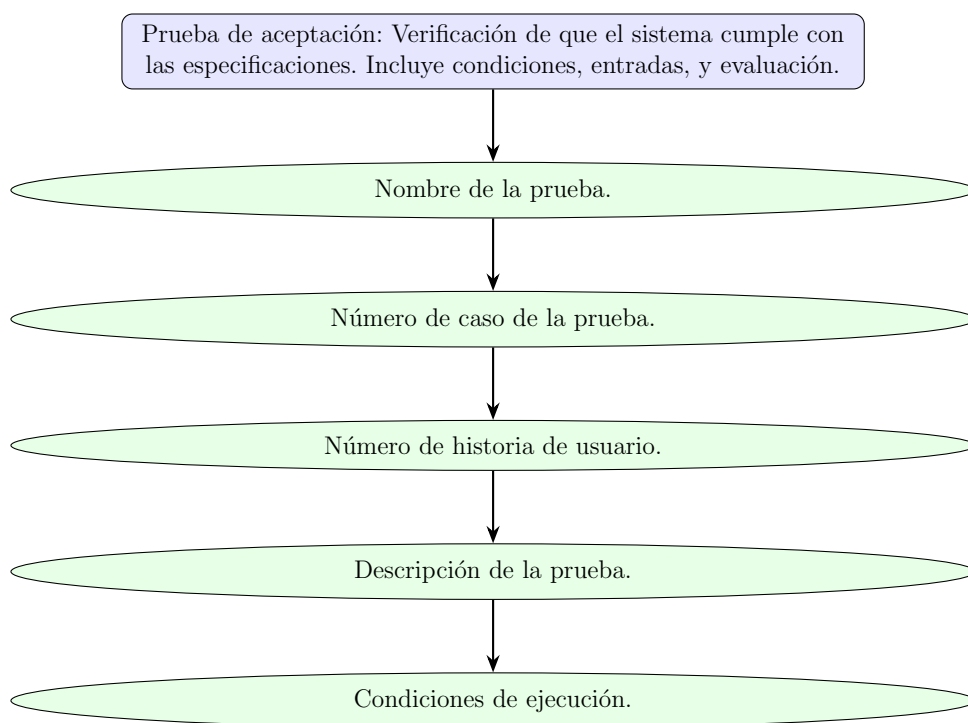


Figura 2.5: Prueba de Aceptación *Fuente: Elaboración propia*

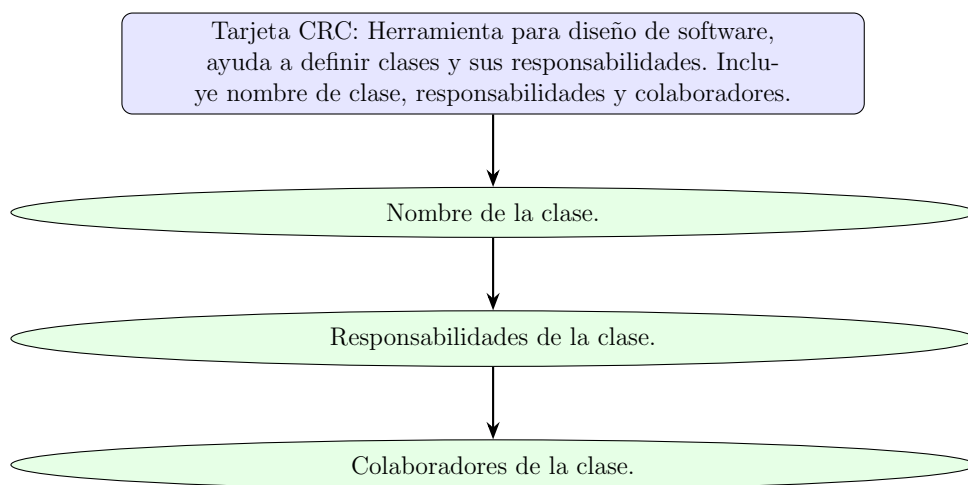


Figura 2.6: Tarjeta CRC *Fuente: Elaboración propia*

2.3 Discusión

Después de la revisión de todas estas metodologías se observa para que es eficiente cada una componiéndose de objetivos diferentes para contextos diferentes.

En las tradicionales se observó como en la de cascada se busca un desarrollo simple en la cual el proyecto pueda iniciar de manera correcta y tener sus fases una por una sin pasar a la siguiente hasta terminar la anterior haciendo de una metodología simple pero bien implementada para proyectos que necesitan cumplir estrictamente con estos pasos.

Con MFS se mostró la utilidad de tener un cuadro de trabajo en el cual todos los involucrados tengan conocimiento de cada una de las fases, aunque también contiene la característica de avanzar fase por fase se busca que tenga una adaptabilidad a las requerimientos del cliente sobre el proyecto sin embargo esta adaptabilidad no siempre se puede alcanzar en etapas mas tardías de la solución del proyecto.

Esto dio paso a investigar las metodologías ágiles las cuales tienen un método mas flexible para resolver los problemas que conlleva realizar el proyecto propuesto, la metodología Scrum promueve un desarrollo incremental a través de iteraciones cortas, conocidas como sprints. Este enfoque permite la incorporación continua de retroalimentación y ajustes en función de la evolución de los requerimientos. No obstante, la implementación efectiva de Scrum requiere un alto nivel de disciplina y colaboración dentro del equipo de desarrollo, así como un compromiso activo de los stakeholders.

Y en la metodología XP se centra en la mejora de la calidad del software y en la satisfacción del cliente. XP integra prácticas clave como la programación en pareja, el desarrollo guiado por pruebas y la integración continua, lo cual es particularmente beneficioso en un contexto donde la calidad y la funcionalidad son fundamentales, como en la gestión de datos clínicos.

Es por eso que se escogió esta última metodología ya que la flexibilidad que ofrece es esencial para poder corregir todos los problemas que van surgiendo en el desarrollo del proyecto, teniendo iteraciones que son más largas que los sprints que permiten que se pueda desarrollar más el proyecto y al final poder corregir todo lo que el cliente haya observado en los prototipos desarrollados hasta el momento, y lo mas importante es la comunicación tan abierta que se tiene con el cliente y lo absorbido que está en el seguimiento del proyecto lo cual es bastante importante para la resolución de este proyecto, ya que es necesario tener en cuenta la opinión de los psicólogos en todo momento para poder tener un sistema correcto y afianzado a las necesidades de la institución.

Capítulo 3

Implementación de la metodología

En la siguiente sección se explicarán las técnicas y métodos utilizados para la realización del sistema de gestión de expedientes clínico del área de psicología para la Dirección de Servicio Médicos Universitarios (DSMU) de la Universidad Autónoma del Estado de Hidalgo (UAEH). Explicando las fases de la metodología ágil que se va a ocupar en este proyecto que es la XP.

3.1 Fase de exploración

El siguiente proyecto es una solución informática vital para esta área del DSMU, ya que es un área dentro de la universidad que no tiene algún sistema digital exclusivamente para este servicio. Actualmente se siguen llenando estos expedientes a mano por lo cual trae muchos problemas humanos, tales como la falta de legibilidad de las letras de los psicólogos que dan las consultas, la falta de normalización en los documentos de tratamiento que se lleva con el paciente, así como el difícil seguimiento que se hace con los pacientes al cambiar de practicantes en el área por la pobreza o falta de organización del expediente realizado con anterioridad. Por lo cual, la implementación de este software hará un gran cambio en la forma en que se dan las consultas y seguimientos de los pacientes que se tiene dentro de la institución, propagando la eficiencia y la integridad de los datos para su uso a futuro.

Dentro del área existe la directora del área de psicología la Psic. Mónica Abigail Aquino Sánchez, ella es la encargada de administrar qué pacientes lleva cada psicólogo en el área y también da consultas a pacientes que llevan ya un tratamiento de bastante tiempo. Por otro lado, también está el subdirector del área el Psic. Julio Cesar Casiano Domínguez el cual se encarga de administrar los recursos del área así como ayudar a las decisiones fundamentales de la oficina junto con al directora, y por último están los practicantes del área que son de la misma universidad de la licenciatura en Psicología. Ellos son administrados por la directora y el subdirector, quienes les asignan pacientes y, ellos llevan su registro el tiempo de su estadía de prácticas profesionales o servicio social. Cuando un paciente ingresa a solicitar una consulta es

asignado a uno de los practicantes y esté lleva su tratamiento, la primera vez que tiene consulta el paciente llena la ficha de identificación y empieza a integrarse el expediente, después varía dependiendo del motivo de consulta y las necesidades de su tratamiento. Sin embargo, durante estas sesiones se empieza a llenar su historial clínico, la tabla de síntomas, el familiograma del paciente, y un apartado para tener resultados de test psicométricos. Adicionalmente cada 4 sesiones se llena una nota clínica como un resumen de las sesiones transcurridas y se empieza a realizar un plan de tratamiento que haga cumplir con los objetivos de consulta de cada paciente.

Es así el proceso como se llevan las consultas y la integración del expediente clínico del paciente. Por tal motivo, para poder cumplir con este proceso, se evaluarán cuáles son los requerimientos necesarios para efectuar las necesidades del sistema, realizando junto con el cliente las historias de usuario necesarias.

3.1.1 Requerimientos

Las Tablas de la 3.1 a la 3.13 son las historias de usuarios que definen cuáles son los requerimientos que debe de tener el sistema para su correcto desarrollo y funcionamiento dentro del sistema.

Número:	1
Usuario:	Psicólogo
Nombre historia:	Acceso al Sistema
Prioridad en la institución:	Alta
Riesgo de desarrollo:	Baja
Puntos Estimados:	2
Iteración Asignada:	1
Programador Responsable:	Juan Luis Torres Uribe
Descripción:	El usuario tiene una clave única y contraseña para poder acceder al sistema
Observaciones:	La clave única que se maneja es la que la universidad provee a todos sus trabajadores y alumnos

Tabla 3.1: Historias de Usuario - Acceso al Sistema *Fuente: Elaboración propia*

Número:	2
Usuario:	Psicólogo
Nombre historia:	Registro de nuevos psicólogos en el sistema
Prioridad en la institución:	Alta
Riesgo de desarrollo:	Baja
Puntos Estimados:	2
Iteración Asignada:	1
Programador Responsable:	Juan Luis Torres Uribe
Descripción:	Si es que aún no se tiene al psicólogo dentro del sistema éste debe de tener la oportunidad de poder registrarse dentro del mismo para poder gestionar los expedientes clínicos de los pacientes.
Observaciones:	Importante tener en cuenta cuantos usuarios pueden estar al mismo tiempo en el sistema

Tabla 3.2: Historias de Usuario - Registro de nuevos psicólogos en el sistema *Fuente: Elaboración propia*

Número:	3
Usuario:	Psicólogo
Nombre historia:	Registro de nuevos expedientes
Prioridad en la institución:	Alta
Riesgo de desarrollo:	Baja
Puntos Estimados:	2
Iteración Asignada:	1
Programador Responsable:	Juan Luis Torres Uribe
Descripción:	El sistema debe de permitir que el usuario registre nuevos expedientes clínicos de los pacientes que tiene a cargo, dándolos de alto con la ficha de identificación del paciente que contiene los datos generales del mismo.
Observaciones:	Es importante que se tengan desde el principio estos datos para poder continuar con la integración de los demás componentes del expediente.

Tabla 3.3: Historias de Usuario - Registro de nuevos expedientes *Fuente: Elaboración propia*

Número:	4
Usuario:	Psicólogo
Nombre historia:	Visualización de los expedientes clínicos
Prioridad en la institución:	Alta
Riesgo de desarrollo:	Media
Puntos Estimados:	2
Iteración Asignada:	1
Programador Responsable:	Juan Luis Torres Uribe
Descripción:	El sistema debe de dar todos los expedientes clínicos a cargo del psicólogo para que él pueda escoger con cuál va a trabajar y al seleccionarlo mostrar la información de este expediente.
Observaciones:	Es importante recalcar que los psicólogos tengan visualización de los expedientes de sus pacientes por la sensibilidad de estos datos.

Tabla 3.4: Historias de Usuario - Visualización de los expedientes clínicos *Fuente: Elaboración propia*

Número:	5
Usuario:	Psicólogo
Nombre historia:	Visualización de la ficha de identificación del paciente adentro del expediente
Prioridad en la institución:	Alta
Riesgo de desarrollo:	Baja
Puntos Estimados:	2
Iteración Asignada:	2
Programador Responsable:	Juan Luis Torres Uribe
Descripción:	Al entrar al expediente lo primero que se debe de observar son los datos básicos del paciente y si no se tenían todos los datos en el momento se deben de poder agregar o modificar los ya ingresados.
Observaciones:	La clave con la cual se identifica el expediente es lo único que no se puede modificar.

Tabla 3.5: Historias de Usuario - Visualización de la ficha de identificación del paciente adentro del expediente *Fuente: Elaboración propia*

Número:	6
Usuario:	Psicólogo
Nombre historia:	Tabla de contactos
Prioridad en la institución:	Media
Riesgo de desarrollo:	Baja
Puntos Estimados:	2
Iteración Asignada:	2
Programador Responsable:	Juan Luis Torres Uribe
Descripción:	Dentro del expediente se deben de poder agregar contactos cercanos del paciente por si existe alguna emergencia
Observaciones:	Por mucho unicamente se registran 3 contactos

Tabla 3.6: Historias de Usuario - Tabla de contactos *Fuente: Elaboración propia*

Número:	7
Usuario:	Psicólogo
Nombre historia:	Historial Clínico
Prioridad en la institución:	Alta
Riesgo de desarrollo:	Media
Puntos Estimados:	3
Iteración Asignada:	2
Programador Responsable:	Juan Luis Torres Uribe
Descripción:	El sistema debe de replicar el formato que contiene la institución sobre el historial clínico del paciente
Observaciones:	Este historial por lo regular no se llena todo la primera vez por lo cual se debe de considerar que se llene poco a poco.

Tabla 3.7: Historias de Usuario - Historial Clínico *Fuente: Elaboración propia*

Número:	8
Usuario:	Psicólogo
Nombre historia:	Tabla de Síntomas
Prioridad en la institución:	Alta
Riesgo de desarrollo:	Alta
Puntos Estimados:	3
Iteración Asignada:	2
Programador Responsable:	Juan Luis Torres Uribe
Descripción:	Se debe de replicar el formato de tabla de síntomas en el cual se debe de registrar más de una tabla de síntomas para compararlos dentro del desarrollo del paciente en su tratamiento
Observaciones:	Es importante considerar la forma en que se modele ya que siempre se comparan los mismos 46 síntomas.

Tabla 3.8: Historias de Usuario - Tabla de Síntomas *Fuente: Elaboración propia*

Número:	9
Usuario:	Psicólogo
Nombre historia:	Familiograma
Prioridad en la institución:	Alta
Riesgo de desarrollo:	Media
Puntos Estimados:	2
Iteración Asignada:	3
Programador Responsable:	Juan Luis Torres Uribe
Descripción:	En el sistema se deben de tener preguntas específicas relacionadas a la relación familiar del paciente así como una tabla en la cual se explique sus relaciones con los integrantes de su familia.
Observaciones:	La forma de funcionar tiene que ser algo específica ya que se ingresa información de sus relaciones familiares y cómo están estructuradas estas relaciones.

Tabla 3.9: Historias de Usuario - Familiograma *Fuente: Elaboración propia*

Número:	10
Usuario:	Psicólogo
Nombre historia:	Notas clínicas
Prioridad en la institución:	Alta
Riesgo de desarrollo:	Baja
Puntos Estimados:	2
Iteración Asignada:	3
Programador Responsable:	Juan Luis Torres Uribe
Descripción:	Se debe de tener un aparatado en el sistema en el cual se haga un resumen de las consultas llevadas por el paciente
Observaciones:	Buscar cómo organizar estas notas para saber cuál es la primera que se creó y así continuamente.

Tabla 3.10: Historias de Usuario - Notas clínicas *Fuente: Elaboración propia*

Número:	11
Usuario:	Psicólogo
Nombre historia:	Plan de tratamiento
Prioridad en la institución:	Alta
Riesgo de desarrollo:	Baja
Puntos Estimados:	2
Iteración Asignada:	3
Programador Responsable:	Juan Luis Torres Uribe
Descripción:	El sistema debe de tener una forma en la cual se puedan generar varios planes de tratamiento dependiendo de los objetivos y generalidades que se busca en el tratamiento del paciente.
Observaciones:	Hay un aparatado de generalidades en el cual se deben de ingresar una vez y luego obtener los datos para cada plan de tratamiento que tenga el paciente.

Tabla 3.11: Historias de Usuario - Plan de tratamiento *Fuente: Elaboración propia*

Número:	12
Usuario:	Psicólogo
Nombre historia:	Resultados de test psicométricos aplicados
Prioridad en la institución:	Alta
Riesgo de desarrollo:	Medias
Puntos Estimados:	3
Iteración Asignada:	4
Programador Responsable:	Juan Luis Torres Uribe
Descripción:	El sistema debe de tener un apartado para poder ingresar los resultados de algunos test psicométricos que se aplican en la institución, detalladamente para los test MMPI, pensamientos automáticos, test de depresión y ansiedad
Observaciones:	Son muchos datos numéricos que se deben de ingresar, tener cuidado en que solo se acepten este tipo de datos y en cómo organizarlo en la interfaz.

Tabla 3.12: Historias de Usuario - Test psicométricos *Fuente: Elaboración propia*

Número:	13
Usuario:	Psicólogo Administrador
Nombre historia:	Apartado de administrador dentro del sistema
Prioridad en la institución:	Alta
Riesgo de desarrollo:	Alta
Puntos Estimados:	4
Iteración Asignada:	4
Programador Responsable:	Juan Luis Torres Uribe
Descripción:	Se debe de tener un apartado para administradores en el sistema en el cual se pueda ver cuáles son los psicólogos registrados poder modificarlos y eliminarlos, también poder ver los expedientes que se tiene y poder traspasarlos a otro psicólogo si se necesitan y por último poder exportar estos expedientes a un Excel para el uso de sus datos así como poder eliminarlos.
Observaciones:	Tener en cuenta cómo hacer para poder llevarlos a Excel ya que son muchos datos en muchas tablas de la base de datos.

Tabla 3.13: Historias de Usuario - Apartado de administración *Fuente: Elaboración propia*

Después de que estas historias fueran evaluadas por los miembros que se encargados del área de psicología, confirmaron que éstos eran los requerimientos necesarios para poder obtener el sistema que ellos necesitan por lo cual se pasó a la siguiente fase de la metodología.

3.2 Fase de planeación

Para el correcto funcionamiento en el cual se puede visualizar la información completa de lo que compone el expediente de cada paciente se llevó a cabo la siguiente formulación de los módulos que deben de integrar el sistema:

- **Perfiles:** cada usuario contará con un perfil en específico donde se muestre la información del usuario en sesión y los expedientes que tienen a su cargo.
- **Expediente:** En él se visualizarán la ficha de identificación del paciente y se integrarán los componentes restantes del expediente.

- **Panel de componentes:** En este panel se contendrán todos los demás componentes que se integran en el expediente como el historial clínico, el familiograma, la tabla de síntomas, las notas clínicas y el plan de tratamiento.
- **Panel administrador:** Este aparatado que solo debe de ser visible para los usuarios distinguidos como administradores debe de contener las funciones para poder manejar los usuarios, los expedientes y poder exportarlos a Excel.

Estos módulos se establecieron en reuniones con la directora del área de psicología del DSMU.

En la Tabla 3.14 se muestran los roles del proyecto que se van a asignar.

Roles	Encargado a
Programador	Juan Luis Torres Uribe
Cliente	DSMU - Área de Psicología
Tester	Juan Luis Torres Uribe
Tracker	Juan Luis Torres Uribe
Coach	Juan Luis Torres Uribe
Consultor	Monica Abigail Aquino Sánchez
Big Boss	Juan Luis Torres Uribe

Tabla 3.14: Asignación de Roles del Proyecto *Fuente: Elaboración propia*

Con base a las historias de usuario que se definieron se creó un plan de entrega en el cual tomando en cuenta las iteraciones, se definen las historias de usuario y la prioridad en el cual se deben de realizar estas historias. En la Tabla 3.15 se observa este plan de entrega.

Historias	Iteración	Prioridad	Dificultad	Fecha de Inicio	Fecha Final
Historia 1	1	Alta	2	13/08/2024	20/08/2024
Historia 2	1	Alta	2	20/08/2024	27/08/2024
Historia 3	1	Alta	2	27/08/2024	03/09/2024
Historia 4	1	Alta	2	03/09/2024	10/08/2024
Historia 5	2	Alta	2	10/09/2024	17/09/2024
Historia 6	2	Alta	2	17/09/2024	24/09/2024
Historia 7	2	Alta	3	24/09/2024	01/10/2024
Historia 8	2	Alta	3	01/10/2024	15/10/2024
Historia 9	3	Alta	2	15/10/2024	22/10/2024
Historia 10	3	Alta	2	22/10/2024	29/10/2024
Historia 11	3	Alta	2	29/10/2024	05/11/2024
Historia 12	4	Alta	2	13/01/2025	03/02/2025
Historia 13	4	Alta	3	03/02/2025	14/03/2025

Tabla 3.15: Detalles de Historias de Usuario *Fuente: Elaboración propia*

3.3 Fase de iteraciones

En esta sección se mostrarán las iteraciones que se necesitaron para crear el sistema cada una de ellas cumpliendo con las historias de usuario correspondientes y así documentando todo el proceso que se llevó para llegar a los requerimientos del sistema.

3.3.1 Primera iteración

En esta primera iteración se crearon las funciones principales para arrancar el sistema, creando las cuentas de usuario de los psicólogos y los expedientes que son las raíz de lo que se va necesitar para poder estructurar todo lo demás, así como, se creo el módulo de perfiles y de expediente. En la Tabla 3.16 se detallan de manera general las tareas que van a corresponder a estas funciones.

Número de tarea	Historia	Nombre de la tarea
1	1	Crear interfaz de inicio de sesión
2	1	Crear la tabla de la base de datos para el registro de psicólogos
3	1	Interacciones de la base de datos con el inicio de sesión
4	2	Crear interfaz del registro de psicólogos
5	2	Interacción de la base de datos para crear nuevos psicólogos
6	3	Crear interfaz del registro de expedientes
7	3	Crear la tabla de la base de datos para almacenar expedientes
8	3	Interacción con la base de datos para crear expedientes
9	4	Crear interfaz para visualizar los expedientes
10	4	Interacción con la base de datos para mostrar los expedientes que tiene cada psicólogo

Tabla 3.16: Tareas a realizar para la primera iteración *Fuente: Elaboración propia*

Para poder observar mejor cómo es que van a llevar estas tareas en las Tablas 3.17 a 3.26 se mostrarán los detalles de cada tarea de ingeniería.

Número de tarea: 1	Número de historia: 1
Nombre de tarea: Crear interfaz de inicio de sesión	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 13/08/24	Fecha fin: 16/08/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Se realizará una interfaz amigable en la cual el usuario ingrese sus número de cuenta y contraseña para entrar al sistema.	

Tabla 3.17: Tarea de Ingeniería 1 *Fuente: Elaboración propia*

Número de tarea: 2	Número de historia: 1
Nombre de tarea: Crear la tabla de la base de datos para el registro de psicólogos	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 16/08/24	Fecha fin: 18/08/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Se creará el espacio en la base de datos para almacenar a los psicólogos	

Tabla 3.18: Tarea de Ingeniería 2 *Fuente: Elaboración propia*

Número de tarea: 3	Número de historia: 1
Nombre de tarea: Interacciones de la base de datos con el inicio de sesión	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 18/08/24	Fecha fin: 20/08/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear los métodos necesarios para interactuar con la base de datos verificar que el usuario que ingresa, exista	

Tabla 3.19: Tarea de Ingeniería 3 *Fuente: Elaboración propia*

Número de tarea: 4	Número de historia: 2
Nombre de tarea: Crear interfaz del registro de psicólogos	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha inicio: 20/08/24	Fecha fin: 24/08/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Realizar una interfaz para que nuevos usuarios se puedan registrar en el sistema	

Tabla 3.20: Tarea de Ingeniería 4 *Fuente: Elaboración propia*

Número de tarea: 5	Número de historia: 2
Nombre de tarea: Interacción de la base de datos para crear nuevos psicólogos	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha inicio: 24/08/24	Fecha fin: 27/08/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear los métodos necesarios para insertar nuevos psicólogos en la base de datos	

Tabla 3.21: Tarea de Ingeniería 5 *Fuente: Elaboración propia*

Número de tarea: 6	Número de historia: 3
Nombre de tarea: Crear interfaz del registro de expedientes	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 27/08/24	Fecha fin: 30/08/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Diseñar una interfaz para que se puedan ingresar nuevos expedientes por lo psicólogos encargados	

Tabla 3.22: Tarea de Ingeniería 6 *Fuente: Elaboración propia*

Número de tarea: 7	Número de historia: 3
Nombre de tarea: Crear la tabla de la base de datos para almacenar expedientes	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 30/08/24	Fecha fin: 01/09/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear la tabla en la base de datos que almacene los expedientes	

Tabla 3.23: Tarea de Ingeniería 7 *Fuente: Elaboración propia*

Número de tarea: 8	Número de historia: 3
Nombre de tarea: Interacción con la base de datos para crear expedientes	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 01/09/24	Fecha fin: 03/09/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear los métodos necesarios para poder insertar nuevos expedientes a la base de datos	

Tabla 3.24: Tarea de Ingeniería 8 *Fuente: Elaboración propia*

Número de tarea: 9	Número de historia: 4
Nombre de tarea: Crear interfaz para visualizar los expedientes	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha inicio: 03/09/24	Fecha fin: 07/09/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Diseñar la interfaz en la que se ingrese la ficha de identificación del paciente para crear su expediente	

Tabla 3.25: Tarea de Ingeniería 9 *Fuente: Elaboración propia*

Número de tarea: 10	Número de historia: 4
Nombre de tarea: Interacción con la base de datos para mostrar los expedientes que tiene cada psicólogo	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha inicio: 07/09/24	Fecha fin: 10/09/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear los métodos necesarios para que se muestren todos los expedientes que tiene a cargo el psicólogo que ingresó al sistema	

Tabla 3.26: Tarea de Ingeniería 10 *Fuente: Elaboración propia*

Diseño y codificación

En la siguiente sección se mostrarán las tarjetas CRC que se utilizaron para crear las clases de esta primera iteración así como las interfaces que terminaron dentro del proyecto.

En las Tablas 3.27 y 3.28 se explican las tarjetas CRC de esta primera iteración.

Psicólogo	
Responsabilidades	Colaboradores
Crear psicólogo	Usuario
Editar psicólogo	
Dar de baja psicólogo	
Validar campos	
Validar campos obligatorios	

Tabla 3.27: Tarjeta CRC Psicólogo *Fuente: Elaboración propia*

Expediente	
Responsabilidades	Colaboradores
Crear expedientes	Psicólogo
Editar expedientes	
Dar de baja expedientes	
Obtener expedientes por psicólogo	
Validar campos	
Validar campos obligatorios	

Tabla 3.28: Tarjeta CRC Expediente *Fuente: Elaboración propia*

Después de realizar estos diseños y ser aceptados se pasó a la creación y codificación de las interfaces las cuales se muestran los siguientes resultados.

The image shows a web application window with a standard Windows-style title bar (minimize, maximize, close buttons). The interface is for a login system. At the top left is the UAEH logo with the text 'Universidad Autónoma del Estado de Hidalgo'. At the top right is another UAEH logo with the text 'Dirección de Servicio Médico Universitario'. In the center, there is a red rectangular box containing the text 'Sistema de Gestion de Expedientes Clinicos'. Below this box, the text 'Area de Psicologia' is displayed. The login form consists of two input fields: 'Numero de Cuenta:' followed by a text input field, and 'Contraseña:' followed by a password input field. Below the password field is a checkbox labeled 'Mostrar contraseña'. At the bottom of the form are two buttons: 'Ingresar' and 'Nuevo Usuario'. A solid red horizontal bar is at the very bottom of the window.

Figura 3.1: Interfaz del inicio de sesión *Fuente: Elaboración propia*

La Figura 3.1 muestra la interfaz en la cual se inicia sesión, los botones que dan la validación de las credenciales ingresadas y la opción de crear un nuevo usuario.



The image shows a web application window titled "Nuevo Perfil". The window has a standard OS-style title bar with minimize, maximize, and close buttons. The main content area has a light blue background. At the top left, there is a circular icon with a left-pointing arrow. To its right, the title "Nuevo Perfil" is displayed in a large, bold, black font. Below the title, the instruction "Ingrese los datos solicitados" is written in a smaller, gray font. In the top right corner, there is a blue button labeled "Registrar". The form consists of several input fields: "Numero de Cuenta:" and "Correo:" are at the top, followed by "Nombre:" and "Telefono:". Below these are "Contraseña:" and "Horario:". The "Contraseña:" field has a small checkbox labeled "Mostrar contraseña" next to it. A large, empty rectangular box is positioned to the right of the "Nombre:" and "Telefono:" fields.

Figura 3.2: Interfaz del creación de nuevo psicólogo *Fuente: Elaboración propia*

En la Figura 3.2 se muestra la interfaz en la cual los psicólogos que no tienen una cuenta se puedan registrar para después poder ingresar al sistema y empezar a crear y organizar sus expedientes.

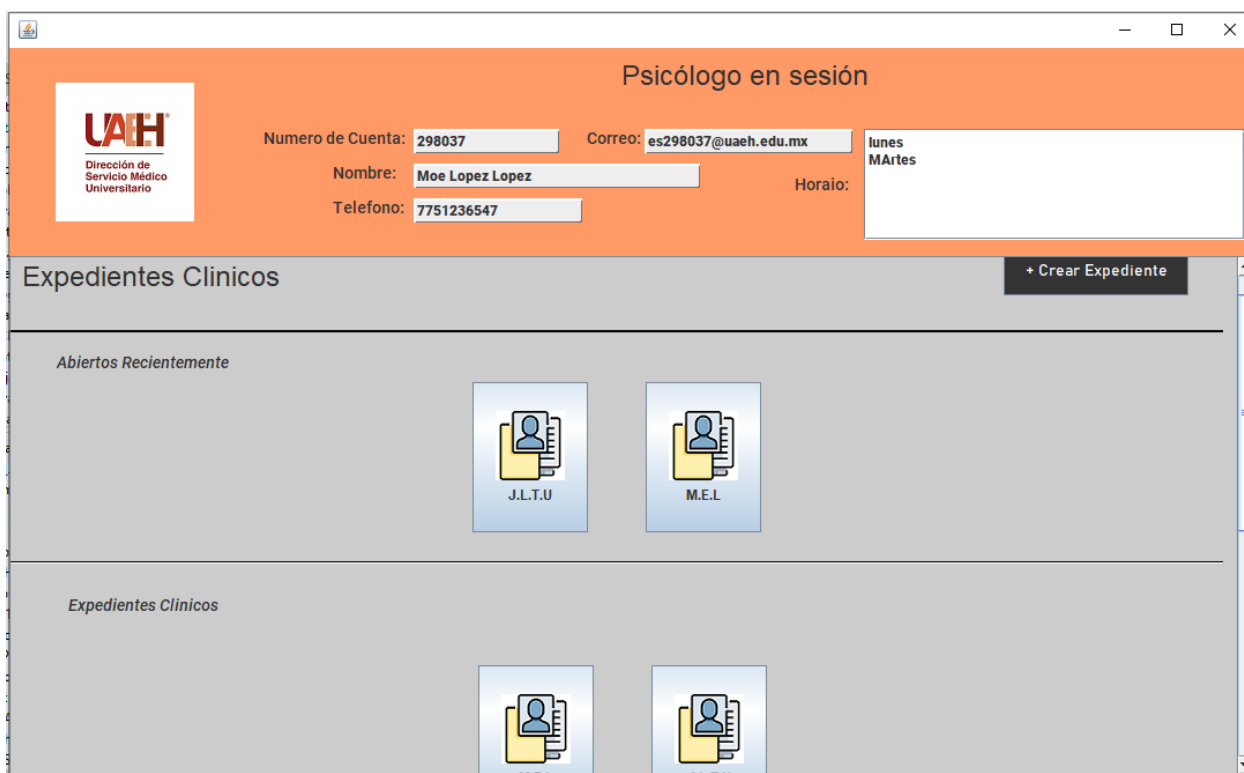


Figura 3.3: Interfaz principal de los expedientes *Fuente: Elaboración propia*

En la Figura 3.3 se muestra la interfaz principal en la cual se mostrarán los expedientes que tiene el psicólogo así como visualizar los datos del psicólogo que entró al sistema y la posibilidad de crear un nuevo expediente si es que lo necesita.

Creación de expediente Registrar

Ingrese los datos solicitados

Nombre del paciente: Monserrat Escarcega Lopez No. De Cuenta: 298039 Fecha de ingreso: 12 nov. 2024

Sexo: Femenino Edad: 22 Sem: 4 Gpo: 55 Escuela/Instituto: ICSA

Peso: 56 Talla: 160 Telefono: 7752348650 Fecha de Nac: 6 may. 1998 Lugar de Nac: Tulancingo

Lugar de residencia actual: Pachuca de Soto Estado Civil: Casado/a Religión: Cristiana Ocupación: Estudiante

Motivo de Consulta:

No puedo dormir

Figura 3.4: Interfaz del creación de nuevo psicólogo *Fuente: Elaboración propia*

En la Figura 3.4 se observa cuáles son los datos que se deben de ingresar para crear un nuevo expediente y así poder realizar después manipulaciones más a fondo de este expediente.

Pruebas

Para validar el correcto funcionamiento del desarrollo de la primera iteración se aplicaron las pruebas de aceptación correspondientes. Estas se explican detalladamente en las Tablas de la 3.29 a la 3.32.

Caso de prueba	
Código: 1	Nº Historia de usuario: 1
Historia de usuario: Acceso al Sistema.	
Condiciones de ejecución: Cada psicólogo tendrá un número identificador y una contraseña para acceder al sistema.	
Entradas/Pasos de ejecución: 1. Iniciar el sistema 2. Ingresar identificador y contraseña 3. Clic en el botón <i>Ingresar</i> 4. Visualizar la interfaz principal de expedientes.	
Resultado esperado: Acceso a la sesión del psicólogo para visualizar los expedientes correspondientes.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 3.29: Caso de prueba 1 *Fuente: Elaboración propia*

Caso de prueba	
Código: 2	Nº Historia de usuario: 2
Historia de usuario: Registro de nuevos psicólogos en el sistema.	
Condiciones de ejecución: Si es que el psicólogo no esta registrado en el sistema esté tendrá la oportunidad de registrarse ingresando datos como su nombre, su identificador, su contraseña, teléfono, correo electrónico y horario dentro de la institución.	
Entradas/Pasos de ejecución: 1. Iniciar el sistema 2. Clic en el botón <i>Nuevo Usuario</i> 3. Ingresar sus datos correspondientes 4. Dar clic en el botón <i>Registrar</i> 5. Regresar a la interfaz de inicio de sesión.	
Resultado esperado: Registrar un nuevo psicólogo en el sistema para que pueda ingresar al mismo.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 3.30: Caso de prueba 2 *Fuente: Elaboración propia*

Caso de prueba	
Código: 3	Nº Historia de usuario: 3
Historia de usuario: Registro de nuevos expedientes.	
Condiciones de ejecución: Desde la interfaz principal de expedientes dar clic en el botón de <i>Crear Expediente</i> dar los datos de la ficha de identificación del paciente y que en la interfaz principal se tenga el expediente.	
Entradas/Pasos de ejecución: 1. Clic en el botón <i>Crear Expediente</i>. 2. Ingresar los datos correspondientes del paciente. 3. Clic en el botón <i>Registrar</i> 4. Al regresar a la interfaz principal de expedientes tener el nuevo expediente creado.	
Resultado esperado: Registrar un nuevo expediente para que al regresar a la interfaz principal de los mismos esté listo para ser llenado de más información.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 3.31: Caso de prueba 3 *Fuente: Elaboración propia*

Caso de prueba	
Código: 4	Nº Historia de usuario: 4
Historia de usuario: Visualización de los expedientes clínicos.	
Condiciones de ejecución: Al ingresar un psicólogo al sistema en la interfaz principal de expedientes aparezcan todos los expedientes pertenecientes a los pacientes del psicólogo que haya creado.	
Entradas/Pasos de ejecución: 1. Ingresar al sistema con su identificador y contraseña. 2. Al cargar la interfaz de expedientes que se visualicen todos los expedientes que tiene a cargo el psicólogo que entro al sistema.	
Resultado esperado: Cuando un psicólogo ingrese al sistema visualice todos los expedientes a su cargo.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 3.32: Caso de prueba 4 *Fuente: Elaboración propia*

Con las pruebas de aceptación completadas y resultados positivos se pasara a la siguiente iteración para continuar con el desarrollo del sistema.

3.3.2 Segunda iteración

En esta segunda iteración se empezaron a crear las funcionalidades básicas de los apartados que contiene un expediente clínico, haciendo que la ficha de identificación del paciente que se ingresó cuando se registró sea visible adentro del expediente, se pueda tener una tabla de contactos de emergencia y trabajar en las primeras 2 funciones de las pestañas de lo que compone el expediente clínico, el historial clínico y la tabla de sintomatología psicológica, en la Tabla 3.33 se especifican las tareas de ingeniería que se llevarán a cabo en esta iteración.

Número de tarea	Historia	Nombre de la tarea
11	5	Crear interfaz principal del expediente del paciente
12	5	Interacciones de la base de datos para mostrar y actualizar los datos ya ingresados del expediente seleccionado
13	6	Crear la tabla de contactos dentro de la interfaz principal del expediente.
14	6	Crear la tabla de la base de datos para ingresar contactos del paciente
15	6	Interacción de la base de datos para crear, mostrar y actualizar los contactos del paciente
16	7	Crear dentro de la interfaz principal del expediente una interfaz que contenga los componentes que contiene el expediente con diferentes pestañas en la cual este el cuestionario del historial clínico
17	7	Crear la tabla de la base de datos para almacenar los datos del historial clínico.
18	7	Interacción con la base de datos para almacenar, mostrar y actualizar los datos ingresados en el historial clínico.
19	8	Crear dentro de la interfaz de los componentes del expediente la pestaña en la cual se muestren todas las tablas de síntomas del paciente
20	8	Crear una interfaz en la cual se muestren la tabla de síntomas
21	8	Crear las tablas de la base de datos que contengan, los 46 síntomas que siempre se evalúan, y las respuesta del paciente conforme a estos síntomas
22	8	Interacciones de la base de datos para almacenar, mostrar y actualizar los datos ingresados en la tabla de síntomas

Tabla 3.33: Tareas a realizar para la segunda iteración *Fuente: Elaboración propia*

Para poder observar a detalle cómo es que se realizarán las tareas descritas en la Tabla 3.33 se describirán cada una de ellas en las Tablas 3.34 a 3.45.

Número de tarea: 11	Número de historia: 5
Nombre de tarea: Crear interfaz principal del expediente del paciente	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha inicio: 10/09/24	Fecha fin: 14/09/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Realizar una interfaz en la cual se muestre la ficha de identificación del paciente	

Tabla 3.34: Tarea de Ingeniería 11 *Fuente: Elaboración propia*

Número de tarea: 12	Número de historia: 5
Nombre de tarea: Interacciones de la base de datos para mostrar y actualizar los datos ya ingresados del expediente seleccionado	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha inicio: 14/09/24	Fecha fin: 17/09/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear la funciones necesarias para poder mostrar lo datos ya ingresados al crear el expediente y actualizarlos	

Tabla 3.35: Tarea de Ingeniería 12 *Fuente: Elaboración propia*

Número de tarea: 13	Número de historia: 6
Nombre de tarea: Crear la tabla de contactos dentro de la interfaz principal del expediente	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.6
Fecha inicio: 17/09/24	Fecha fin: 19/09/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear la tabla en la cual se mostrarán y se guardarán los contactos de emergencia del paciente dentro del panel principal de la interfaz del expediente	

Tabla 3.36: Tarea de Ingeniería 13 *Fuente: Elaboración propia*

Número de tarea: 14	Número de historia: 6
Nombre de tarea: Crear la tabla de la base de datos para ingresar contactos del paciente	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.6
Fecha inicio: 19/09/24	Fecha fin: 21/09/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear la tabla en la base de datos en la cual se puedan ingresar los contactos de emergencia del paciente	

Tabla 3.37: Tarea de Ingeniería 14 *Fuente: Elaboración propia*

Número de tarea: 15	Número de historia: 6
Nombre de tarea: Interacción de la base de datos para crear, mostrar y actualizar los contactos del paciente	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.6
Fecha inicio: 21/09/24	Fecha fin: 24/09/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear los métodos necesarios para que en la tabla de la interfaz se muestren los datos ingresados y se agreguen más si es que son necesarios	

Tabla 3.38: Tarea de Ingeniería 15 *Fuente: Elaboración propia*

Número de tarea: 16	Número de historia: 7
Nombre de tarea: Crear dentro de la interfaz principal del expediente una interfaz que contenga los componentes que contiene el expediente con diferentes pestañas en la cual esté el cuestionario del historial clínico	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha inicio: 24/09/24	Fecha fin: 27/09/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear dentro de la interfaz principal otra ventana la cual contenga varias pestañas en la que la primera será la del historial clínico, dentro de esta ventana se creará el cuestionario de 61 preguntas del historial clínico	

Tabla 3.39: Tarea de Ingeniería 16 *Fuente: Elaboración propia*

Número de tarea: 17	Número de historia: 7
Nombre de tarea: Crear la tabla de la base de datos para almacenar los datos del historial clínico	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha inicio: 27/09/24	Fecha fin: 29/09/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear la tabla de la base de datos donde se va a almacenar los datos del historial clínico	

Tabla 3.40: Tarea de Ingeniería 17 *Fuente: Elaboración propia*

Número de tarea: 18	Número de historia: 7
Nombre de tarea: Interacción con la base de datos para almacenar, mostrar y actualizar los datos ingresados en el historial clínico	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha inicio: 29/09/24	Fecha fin: 01/10/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Codificar las funciones necesarias para poder almacenar, mostrar y actualizar los datos que sean ingresados en el historial clínico	

Tabla 3.41: Tarea de Ingeniería 18 *Fuente: Elaboración propia*

Número de tarea: 19	Número de historia: 8
Nombre de tarea: Crear dentro de la interfaz de los componentes del expediente la pestaña en la cual se muestren todas las tablas de síntomas del paciente	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.75
Fecha inicio: 01/10/24	Fecha fin: 03/10/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear dentro de la ventana de los componentes del expediente una pestaña en la cual se muestren las tablas de síntomas ya creadas o crear una nueva	

Tabla 3.42: Tarea de Ingeniería 19 *Fuente: Elaboración propia*

Número de tarea: 20	Número de historia: 8
Nombre de tarea: Crear una interfaz en la cual se muestre la tabla de síntomas para poder ser editada	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.75
Fecha inicio: 03/10/24	Fecha fin: 07/10/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear una interfaz amigable en la cual el psicólogo pueda manipular la tabla de síntomas cada vez que lo necesite	

Tabla 3.43: Tarea de Ingeniería 20 *Fuente: Elaboración propia*

Número de tarea: 21	Número de historia: 8
Nombre de tarea: Crear las tablas de la base de datos que contengan, los 46 síntomas que siempre se evalúan, y las respuestas del paciente conforme a estos síntomas	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.75
Fecha inicio: 07/10/24	Fecha fin: 11/10/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear en la base de datos las tablas necesarias para guardar esta tabla de síntomas tomando en cuenta la optimización de esta base de datos	

Tabla 3.44: Tarea de Ingeniería 21 *Fuente: Elaboración propia*

Número de tarea: 22	Número de historia: 8
Nombre de tarea: Interacciones de la base de datos para almacenar, mostrar y actualizar los datos ingresados en la tabla de síntomas	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.75
Fecha inicio: 11/10/24	Fecha fin: 15/10/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear los métodos necesarios para poder mostrar, ingresar y actualizar los datos necesarios de la tabla de síntomas	

Tabla 3.45: Tarea de Ingeniería 22 *Fuente: Elaboración propia*

Diseño y codificación

En esta sección se mostrará el diseño y codificación de las interfaces que responden a las necesidades de las historias de usuario y las tareas de ingeniería que ya fueron explicadas con anterioridad, así como mostrar el diseño de las tarjetas CRC para el manejo de las clases que se utilizaron en esta iteración del sistema.

En las Tablas 3.46 a la 3.48 se muestran los detalles de las tarjetas CRC que se utilizaron para crear las clases de esta iteración.

Contactos	
Responsabilidades	Colaboradores
Crear contactos	Expediente
Editar contactos	
Dar de baja contactos	
Obtener contactos dependiendo del expediente	
Validar campos	
Validar campos obligatorios	

Tabla 3.46: Tarjeta CRC Contactos *Fuente: Elaboración propia*

Historial clínico	
Responsabilidades	Colaboradores
Crear historial clínico	Expediente
Editar el historial clínico	
Dar de baja el historial clínicos	
Obtener la información del expediente clínico dependiendo del expediente	
Validar campos	

Tabla 3.47: Tarjeta CRC historial Clínico *Fuente: Elaboración propia*

Síntomas	
Responsabilidades	Colaboradores
Crear tablas de síntomas	Expediente
Editar la tabla de síntomas	
Dar de baja una tabla de síntomas	
Obtener las tablas de síntomas creadas dependiendo del expediente	
Obtener la información del la tabla de síntomas dependiendo del expediente	
Validar campos	

Tabla 3.48: Tarjeta CRC historial Clínico *Fuente: Elaboración propia*

Después de que estos diseños fueran aceptados se pasó a la creación y codificación de las interfaces correspondiente para el funcionamiento del sistema.



UAH
Dirección de
Servicio Médico
Universitario

Numero de Cuenta: 298037

Correo: es298037@uaeh.edu.mx

Nombre: Moe Lopez Lopez

Telefono: 7751236547

lunes
Martes

Horaio:

 **Ficha de Identificación** Guardar

Nombre del paciente: Juan Luis Torres Uribe

No. De Cuenta: 378732

Fecha de ingreso: 7 nov. 2024

Sexo: Masculino

Edad: 22

Sem: No apli...

Gpo: 3

Escuela/Instituto: Icbi

Peso: 68

Talla: 180

Telefono: 7751691518

Fecha de Nac: 22 feb. 2002

Lugar de Nac: Tulancingo

Lugar de residencia actual: Pachuca

Estado Civil: Union Libre

Religion: Ninguna

Ocupación: Estudiante

Nombre	Ocupación	Teléfono
Escarcega Lopez MOnserrát (esposa) jsjs	Estudia	7751691518

Motivo de Consulta

El paciente se preseta a sesion mencionando que se ha sentido angustiado ya q ue "no ha hecho su proyecto final" esto le causa sentimientos de culpa y verguanz a.

Historial Clínico Sintomas Familiograma Notas Clínicas Plan de Tratamiento

Historial Clínico

Antecedentes médicos

Figura 3.5: Interfaz del principal del expediente *Fuente: Elaboración propia*

En la Figura 3.5 se puede observar la interfaz principal de la información del expediente escogido, mostrando la ficha de identificación del paciente, la tabla de contactos y la ventana secundaria que contendrá los demás componentes del expediente clínico. También cuenta con el botón guardar el cual es el encargado de almacenar la información integrada al expediente.

The screenshot displays a web application window titled "Psicólogo en sesión". The interface is divided into several sections:

- Header:** Features the UAH logo (Dirección de Servicio Médico Universitario) and a session status indicator "Psicólogo en sesión".
- Patient Information:** Includes fields for "Numero de Cuenta: 298037", "Correo: es298037@uaeh.edu.mx", "Nombre: Moe Lopez Lopez", "Telefono: 7751236547", and "Horaio: lunes Martes".
- Motivo de Consulta:** A text area containing the text: "El paciente se preseta a sesion mencionando que se ha sentido angustiado ya que 'no ha hecho su proyecto final' esto le causa sentimientos de culpa y verguanza." Below this is an "Observaciones" section with the text "Apoco si".
- Clinical History Module:** A tabbed interface with tabs for "Historial Clínico", "Sintomas", "Familiograma", "Notas Clínicas", and "Plan de Tratamiento". The "Historial Clínico" tab is active, showing a form titled "Historial Clínico" with the following questions:
 - ¿Padece alergias? No ☐ ¿Cuales? ninguna
 - ¿Padece alguna enfermedad crónica degenerativa o visual? ¿Cuál?
 - ¿Toma algún medicamento regular bajo prescripción médica? ¿Para qué?
 - ¿Sufrió alguna enfermedad durante la infancia?
 - ¿Ha sido intervenido quirúrgicamente alguna vez? Especifique
 - ¿Es alérgico a algún medicamento?
 - ¿Algún medicamento que que consuma sin receta médica?

Figura 3.6: Interfaz secundaria en el modulo del historial clínico *Fuente: Elaboración propia*

Se despliega en la Figura 3.6 cómo se creó la interfaz secundaria que contendrá los módulos de la información necesaria para llenar el expediente. En dicha interfaz, se aprecia el módulo del historial clínico el cual contiene un cuestionario de 61 preguntas.



Figura 3.7: Interfaz secundaria en el modulo de las tablas de síntomas *Fuente: Elaboración propia*

Pasando al módulo de la Tabla de sintomatología psicológica de la interfaz secundaria, en la Figura 3.7 se muestra cómo es que aparecen los botones en los cuales se puede ingresar a una tabla de síntomas ya creada siendo identificada por la fecha en la que se integró o crear una nueva tabla de síntomas.

Tabla de sintomatología psicológica

Fecha de creación: 7 nov. 2024

Indique si en su vida ha presentado algo de lo que a continuación se le presenta:

Síntoma	Presencia	Tiempo	Nota
Aislamiento	si		
Apatía	no		
Asco	si	Cuando como	
Autolesiones	si	En la secundaria solamente	
Cambios de humor	no		
Cambios en el apetito	no		
Cansancio excesivo	no		
Comerse las uñas	no		
Convulsiones	si	Desde que naci	
Deseos de muerte	si	Desde que termine con mi ex	
Dificultad en la concentración	no		
Dolor estomacal	no		
Dolores de cabeza	no		
Escalofríos	no		
Escuchar voces	no		

Otro:

Guardar

Figura 3.8: Interfaz de la tabla de síntomas *Fuente: Elaboración propia*

Por último, se realizó la interfaz en la cual se editan o llenan los datos correspondientes de la Tabla de síntomas como se muestra en la Figura 3.8 en la cual se muestra la Tabla y un botón de guardar para insertar o actualizar todos los cambios.

Pruebas

Así es cómo se pasó a la validación de los módulos codificados aplicando las pruebas de aceptación correspondientes que se detallan en las Tablas 3.49 a 3.52 .

Caso de prueba	
Código: 5	Nº Historia de usuario: 5
Historia de usuario: Visualización de la ficha de identificación del paciente dentro del expediente.	
Condiciones de ejecución: Cuando el psicólogo escoja un expediente se mostrará la interfaz principal en la cual se muestran la ficha de identificación del paciente, además de un botón para guardar los cambios.	
Entradas/Pasos de ejecución: 1. Darle clic al expediente con el que se quiera trabajar 2. Se mostrará la interfaz con la ficha de identificación del paciente 3. Clic en el botón guardar si así lo requiere	
Resultado esperado: Mostrar los datos de la ficha de identificación del paciente en la interfaz principal del expediente seleccionado	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 3.49: Caso de prueba 5 *Fuente: Elaboración propia*

Caso de prueba	
Código: 6	Nº Historia de usuario: 6
Historia de usuario: Tabla de contactos.	
Condiciones de ejecución: Dentro de la interfaz principal del expediente se visualizará abajo de la ficha de identificación una tabla para anotar los contactos de emergencia del paciente	
Entradas/Pasos de ejecución: 1. Darle clic al expediente con el que se quiera trabajar 2. Se mostrará la interfaz con la tabla de contactos 3. Editar los datos correspondientes a los contactos del paciente	
Resultado esperado: Al observar la tabla de contactos se puede agregar y actualizar los datos de la tabla	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 3.50: Caso de prueba 6 *Fuente: Elaboración propia*

Caso de prueba	
Código: 7	Nº Historia de usuario: 7
Historia de usuario: Historial Clínico.	
Condiciones de ejecución: Cuando se esté trabajando en la interfaz secundaria en la pestaña del historial clínico este se pueda llenar y actualizar	
Entradas/Pasos de ejecución: 1. Darle clic al expediente con el que se quiera trabajar 2. Estar en la pestaña del historial clínico de la interfaz secundaria del expediente 3. Editar los datos correspondientes del historial clínico 4. Dar clic en el botón de guardar	
Resultado esperado: Que se guardé correctamente los datos ingresados o actualizados en el historial clínico	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 3.51: Caso de prueba 7 *Fuente: Elaboración propia*

Caso de prueba	
Código: 8	Nº Historia de usuario: 8
Historia de usuario: Tabla de Síntomas.	
Condiciones de ejecución: Cuando se esté trabajando en la interfaz secundaria en la pestaña de la tabla de síntomas se mostrarán los botones de las tablas ya creadas o hacer una nueva	
Entradas/Pasos de ejecución: 1. Darle clic al expediente con el que se quiera trabajar 2. Estar en la pestaña de la tabla de síntomas de la interfaz secundaria del expediente 3. Dar clic en una tabla de síntomas ya integrada o crear una nueva 4. Ser llevado a la interfaz de las tablas con los datos cargados correspondientes 5. Editar o ingresar más datos a la tabla 6. Clic botón <i>Guardar</i> para actualizar los cambios correspondientes	
Resultado esperado: Ser llevado a la interfaz de la tabla de síntomas vacía si es que se crea una nueva o con los datos correspondientes a la ya existente y poder editarlos y guardarlos	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 3.52: Caso de prueba 8 *Fuente: Elaboración propia*

Al concluir la fase de pruebas satisfactoriamente se comenzará con el desarrollo de la tercera iteración para casi culminar con el desarrollo del mismo.

3.3.3 Tercera iteración

En la tercera iteración se termina de construir los 3 módulos correspondientes a la interfaz secundaria del expediente, los cuales serían el familiograma, las notas cónicas y el plan del tratamiento del paciente. Las tareas de ingeniería pertinentes para el desarrollo de estos 3 módulos se exponen en la tabla 3.53.

Número de tarea	Historia	Nombre de la tarea
23	9	Crear interfaz del módulo del familiograma
24	9	Crear la tabla de la base de datos que contenga los datos del familiograma
25	9	Interacción con la base de datos para poder crear, mostrar y actualizar los datos del familiograma
26	10	Crear la interfaz del módulo de las notas clínicas
27	10	Crear la interfaz en la cual se muestren los datos de la nota clínica
28	10	Crear la tabla de la base de datos que contenga los datos de las notas clínicas
29	10	Obtener las interacciones con la base de datos para ingresar, mostrar y actualizar los datos de las notas clínicas
30	11	Crear la interfaz del módulo del plan de tratamiento
31	11	Crear la interfaz en la cual se muestren los datos del plan de tratamiento
32	11	Crear la tabla de la base de datos que contenga los datos del plan de tratamiento
33	11	Obtener las interacciones con la base de datos para ingresar, mostrar y actualizar los datos del plan de tratamiento

Tabla 3.53: Tareas a realizar para la tercera iteración *Fuente: Elaboración propia*

Los detalles de las tareas de ingeniería propuestas en la Tabla 3.53 se muestran en las Tablas 3.54 a 3.64 describiendo cada una de ellas.

Número de tarea: 23	Número de historia: 9
Nombre de tarea: Crear interfaz del módulo del familiograma	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.6
Fecha inicio: 15/10/24	Fecha fin: 17/10/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Realizar una interfaz en el módulo del familiograma la cual contenga las preguntas relacionadas a las relaciones familiares y una tabla para ingresar la familia del paciente	

Tabla 3.54: Tarea de Ingeniería 23 *Fuente: Elaboración propia*

Número de tarea: 24	Número de historia: 9
Nombre de tarea: Crear la tabla de la base de datos que contenga los datos del familiograma	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.6
Fecha inicio: 17/10/24	Fecha fin: 19/10/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear tablas en la base de datos que contengan las preguntas de las relaciones familiares y la tabla de la familia	

Tabla 3.55: Tarea de Ingeniería 24 *Fuente: Elaboración propia*

Número de tarea: 25	Número de historia: 9
Nombre de tarea: Interacción con la base de datos para poder crear, mostrar y actualizar los datos del familiograma	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.6
Fecha inicio: 19/10/24	Fecha fin: 22/10/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Codificar los métodos necesarios para poder extraer, ingresar y actualizar los datos del familiograma	

Tabla 3.56: Tarea de Ingeniería 25 *Fuente: Elaboración propia*

Número de tarea: 26	Número de historia: 10
Nombre de tarea: Crear la interfaz del módulo de las notas clínicas	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 22/10/24	Fecha fin: 24/10/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear una interfaz la cual tenga el espacio para crear botones dependiendo de las notas clínicas ya creadas y el botón de una nueva nota clínica	

Tabla 3.57: Tarea de Ingeniería 26 *Fuente: Elaboración propia*

Número de tarea: 27	Número de historia: 10
Nombre de tarea: Crear la interfaz en la cual se muestren los datos de la nota clínica	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 24/10/24	Fecha fin: 25/10/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear una interfaz amigable para poder ingresar todos los datos correspondientes de una nota clínica	

Tabla 3.58: Tarea de Ingeniería 27 *Fuente: Elaboración propia*

Número de tarea: 28	Número de historia: 10
Nombre de tarea: Crear la tabla de la base de datos que contenga los datos de las notas clínicas	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 25/10/24	Fecha fin: 27/10/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Estructurar la tabla de la base de datos que guarde los datos del historial clínico	

Tabla 3.59: Tarea de Ingeniería 28 *Fuente: Elaboración propia*

Número de tarea: 29	Número de historia: 10
Nombre de tarea: Obtener las interacciones con la base de datos para ingresar, mostrar y actualizar los datos de las notas clínicas	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 27/10/24	Fecha fin: 29/10/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Integrar métodos codificados para la obtención, ingreso y actualización de los datos de la nota clínica en la base de datos	

Tabla 3.60: Tarea de Ingeniería 29 *Fuente: Elaboración propia*

Número de tarea: 30	Número de historia: 11
Nombre de tarea: Crear la interfaz del módulo del plan de tratamiento	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 29/10/24	Fecha fin: 31/10/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Diseñar una interfaz la cual tenga la disponibilidad para crear botones dependiendo de los planes de tratamientos ya creados y el botón de un nuevo plan de tratamiento	

Tabla 3.61: Tarea de Ingeniería 30 *Fuente: Elaboración propia*

Número de tarea: 31	Número de historia: 11
Nombre de tarea: Crear la interfaz en la cual se muestren los datos del plan de tratamiento	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 31/10/24	Fecha fin: 01/11/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear una interfaz en la cual se ingresen los datos correspondientes a un plan de tratamiento	

Tabla 3.62: Tarea de Ingeniería 31 *Fuente: Elaboración propia*

Número de tarea: 32	Número de historia: 11
Nombre de tarea: Crear la tabla de la base de datos que contenga los datos del plan de tratamiento	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 01/10/24	Fecha fin: 03/11/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Insertar una tabla en la base de datos que guarde la información del plan de tratamiento	

Tabla 3.63: Tarea de Ingeniería 32 *Fuente: Elaboración propia*

Número de tarea: 33	Número de historia: 11
Nombre de tarea: Obtener las interacciones con la base de datos para ingresar, mostrar y actualizar los datos del plan de tratamiento	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 01/10/24	Fecha fin: 03/11/24
Programador responsable: Juan Luis Torres Uribe	
Descripción: Describir los métodos necesarios para poder obtener, insertar y actualizar los datos del plan de tratamiento	

Tabla 3.64: Tarea de Ingeniería 33 *Fuente: Elaboración propia*

Ya teniendo las tareas a realizar en la iteración se pasará al diseño de las clases y la codificación de las interfaces para empezar con sus pruebas.

Diseño y codificación

En este apartado se describirán las tarjetas CRC así como las interfaces codificadas para esta iteración para así poder pasar a su evaluación en las pruebas de aceptación.

Las tarjetas CRC que se crearon para las clases de esta iteración se describen en las Tablas 3.65 a 3.67.

Familiograma	
Responsabilidades	Colaboradores
Crear un familiograma que contenga las preguntas y la tabla	Expediente
Editar las preguntas y la tabla del familiograma	
Dar de baja la información del familiograma	
Obtener la información del familiograma dependiendo del expediente	
Validar campos	
Validar campos obligatorios	

Tabla 3.65: Tarjeta CRC Familiograma *Fuente: Elaboración propia*

Notas clínicas	
Responsabilidades	Colaboradores
Crear una nota clínica	Expediente
Editar la nota clínica	
Dar de baja la nota clínica	
Obtener las notas clínicas vinculadas al expediente	
Obtener la información de la nota clínica seleccionada	
Validar campos	
Validar campos obligatorios	

Tabla 3.66: Tarjeta CRC Nota clínica *Fuente: Elaboración propia*

Plan de tratamiento	
Responsabilidades	Colaboradores
Crear un plan de tratamiento	Expediente
Editar un plan de tratamiento	
Dar de baja un plan de tratamiento	
Obtener los planes de tratamiento vinculados al expediente	
Obtener la información del plan de tratamiento seleccionado	
Validar campos	
Validar campos obligatorios	

Tabla 3.67: Tarjeta CRC Plan de Tratamiento *Fuente: Elaboración propia*

Al ser aceptados los diseños de las clases se empezaron a diseñar y codificar las interfaces que van a ser respuesta a los resultados de las tareas de ingeniería por lo cual fueron las siguientes.

The screenshot shows a web application window titled 'Psicólogo en sesión'. The header bar is orange and contains the UAH logo, user account details (Numero de Cuenta: 298037, Correo: es298037@uah.edu.mx, Nombre: Moe Lopez Lopez, Telefono: 7751236547), and the day of the week (lunes, Martes). Below the header, there are tabs for 'Historial Clínico', 'Sintomas', 'Familiograma' (selected), 'Notas Clínicas', and 'Plan de Tratamiento'. The 'Familiograma' tab displays a form titled 'Tabla de familiograma y relaciones interpersonales' with several questions and text input fields. The 'Motivo de Consulta' sidebar contains a text area with a patient's statement. The 'Observaciones' sidebar contains a text area with the word 'Apoco si'.

Figura 3.9: Interfaz del modulo del Familiograma *Fuente: Elaboración propia*

Para comenzar en la Figura 3.9 se observa cómo se construyó el módulo del familiograma en la interfaz secundaria de los expedientes el cual tiene las preguntas relacionadas con las relaciones familiares y una tabla donde se anotan sus familiares más cercanos y cómo es la relación con ellos.

The screenshot displays a web application window titled "Psicólogo en sesión". The header bar is orange and contains the UAEM logo on the left. In the center of the header, there are input fields for "Numero de Cuenta: 298037", "Correo: es298037@uaeh.edu.mx", "Nombre: Moe Lopez Lopez", and "Telefono: 7751236547". On the right, a box shows the date and time: "lunes Martes" and "Horaio:". Below the header, a navigation bar includes tabs for "Historial Clinico", "Sintomas", "Familiograma", "Notas Clínicas" (which is selected), and "Plan de Tratamiento". The main content area is titled "Notas clinicas mensuales" and contains the instruction "Cree o escoja una nota clinica mensual para editar:". Below this instruction are two blue buttons: "Primera Nota Clinica" and "Crear un nueva nota clinica". On the left side of the interface, there are two text areas: "Motivo de Consulta" containing the text "El paciente se preseta a sesion mencionando que se ha sentido angustiado ya q ue 'no ha hecho su proyecto final' esto le causa sentimientos de culpa y verguanza." and "Observaciones" containing the text "Apoco si".

Figura 3.10: Interfaz del modulo de Notas Clínicas *Fuente: Elaboración propia*

En la Figura 3.10 se observa la construcción de la interfaz de las notas clínicas en las cuales se crea un botón por cada nota clínica realizada y teniendo la oportunidad de crear una nota nueva si es que lo requiera.

 **Nota Clínica** De  A 

Impresión diagnóstica:

Motivo de consulta:

Exploración física:

Pronóstico: Reservado a devolución

Temas abordados durante las sesiones:

Primera sesión:

Figura 3.11: Interfaz de la Nota Clínica *Fuente: Elaboración propia*

Dentro de la Figura 3.11 se observa la interfaz que se creó para llenar los datos que debe contener una nota clínica, la cual puede ser llenada o editada cuando sea necesario. Adicionalmente, se incorporó un botón de guardar para asegurar todos los datos integrados del paciente.

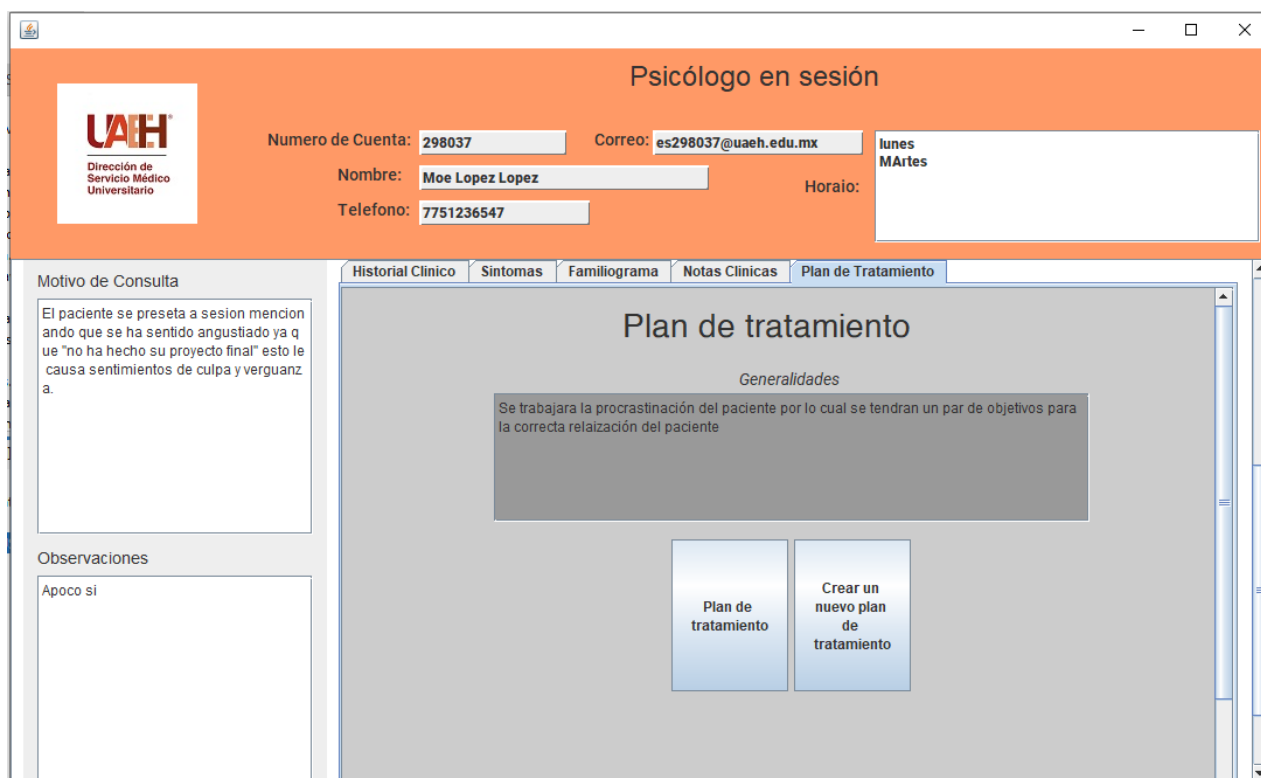


Figura 3.12: Interfaz del modulo del Plan de Tratamiento *Fuente: Elaboración propia*

En el siguiente módulo de la Figura 3.12 se observa el diseño que se dió para crear los botones dependiendo de los planes de tratamiento que han sido creados dentro del expediente y con la posibilidad de crear un plan de tratamiento nuevo.

Plan de tratamiento

Objetivo: Quiere mejorar sus habitos

Actividad: economia de fichas

Psicoeducación: Se busca explicar la satisfaccion de hacer coosas

Figura 3.13: Interfaz del Plan de Tratamiento *Fuente: Elaboración propia*

Por último en la Figura 3.13 se muestra cómo queda la interfaz en la que se ingresa la información necesaria para el plan de tratamiento, así como su botón de guardar para poder integrar los datos nuevo o actualizados que han sido introducidos en el plan de tratamiento escogido.

Para comprobar que estas interfaces codificadas responden a las necesidades descritas en las historias de usuario se pasarán por pruebas de aceptación en las cuales se comprueban estos requisitos.

Pruebas

En esta sección se comprobarán las codificaciones que fueron realizadas para los requisitos expuestos de las historias de usuario que pertenecen a esta iteración. En las Tablas 3.68 a 3.70 se detallarán estas pruebas de aceptación y sus resultados.

Caso de prueba	
Código: 9	Nº Historia de usuario: 9
Historia de usuario: Familiograma.	
Condiciones de ejecución: Al entrar a un expediente y estar en el módulo del familiograma este debe poder editar las preguntas respecto a las relaciones familiares del paciente y aparte agregar en la tabla los integrantes familiares más cercanos del paciente y cómo es la relación con ellos.	
Entradas/Pasos de ejecución: 1. Darle clic al expediente con el que se quiera trabajar 2. Dar clic en la interfaz secundaria en el modulo del familiograma 3. Editar las preguntas con las respuestas del paciente. 4. Dar clic en el botón <i>Nuevo integrante</i> para agregar un integrante familiar e insertar sus datos 5. Dar clic en el botón de <i>guardar</i> de la interfaz principal del paciente para actualizar o insertar los datos agregados.	
Resultado esperado: Los datos ingresados se guarden correctamente puedan ser modificados y actualizados, y que al cargar nuevamente este expediente se muestren.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 3.68: Caso de prueba 9 *Fuente: Elaboración propia*

Caso de prueba	
Código: 10	Nº Historia de usuario: 10
Historia de usuario: Notas clínicas.	
Condiciones de ejecución: Al entrar a un expediente y estar en el módulo de las notas clínicas el usuario debe de elegir si crear una nueva nota o escoger alguna que ya creó para editarla, lo debe de cambiar de interfaz a la de la nota y ahí poder guardar los cambios realizados.	
Entradas/Pasos de ejecución: 1. Darle clic al expediente con el que se quiera trabajar 2. Dar clic en la interfaz secundaria en el módulo de las notas clínicas 3. Escoger si se crea una nueva nota clínica o escoger alguna ya integrada 4. Editar o ingresar los datos de la nota clínica escogida 5. Dar clic en el botón de <i>guardar</i> de la interfaz de las notas clínicas para guardar cualquier cambio	
Resultado esperado: Los datos ingresados se guarden correctamente puedan ser modificados y actualizados, y que al cargar nuevamente este expediente se muestren las notas clínicas creadas y que al escoger alguna de estas notas se muestren los datos guardados en esta nota.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 3.69: Caso de prueba 10 *Fuente: Elaboración propia*

Caso de prueba	
Código: 11	Nº Historia de usuario: 11
Historia de usuario: Plan de tratamiento	
Condiciones de ejecución: Al entrar a un expediente y estar en el módulo de los planes de tratamiento el usuario debe de elegir si crear un nuevo plan o escoger alguno que ya creó para editarlo, lo debe de cambiar de interfaz a la de el plan y ahí poder guardar los cambios realizados.	
Entradas/Pasos de ejecución: 1. Darle clic al expediente con el que se quiera trabajar 2. Dar clic en la interfaz secundaria en el módulo de los planes de tratamiento 3. Escoger si se crea un nuevo plan de tratamiento o escoger alguno ya integrado 4. Editar o ingresar los datos del plan de tratamiento escogido 5. Dar clic en el botón de <i>guardar</i> de la interfaz del plan de tratamiento para guardar cualquier cambio	
Resultado esperado: Los datos ingresados se guarden correctamente puedan ser modificados y actualizados, y que al cargar nuevamente este expediente se muestren los planes de tratamiento creados y que al escoger alguno de estos planes se muestren los datos guardados en este plan.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 3.70: Caso de prueba 11 *Fuente: Elaboración propia*

Con la aceptación de las pruebas de la codificación que se hicieron se termina esta iteración para poder pasar a la siguiente y última iteración de esta fase del desarrollo.

3.3.4 Cuarta Iteración

Se concluirá el desarrollo del proyecto en esta iteración por lo cual se construirá los últimos módulos para terminar con los requerimientos propuestos por los encargados del área. Estos apartados son la creación de un módulo para ingresar los resultados de algunos test psicométricos con los cuales se evalúan a los pacientes, así como un apartado administrativo para poder llevar el control de los psicólogos ingresados al sistema y los expedientes para poder modificarlos, eliminarlos, exportarlos y traspasar expedientes entre psicólogos. Por lo cual en la Tabla 3.71

se mostrarán cuáles son las tareas de ingeniería que se necesitan para construir esto en el sistema.

Número de tarea	Historia	Nombre de la tarea
34	12	Crear interfaz del módulo de los test psicométricos
35	12	Crear la tabla de la base de datos que contenga los datos de los test psicométricos
36	12	Interacción con la base de datos para poder crear, mostrar y actualizar los datos de los test psicométricos
37	13	Crear la interfaz que contendrá las funciones de los administradores divida en pestañas
38	13	Interacción de la IU con el usuario para la primera pestaña del panel administrador para poder buscar, mostrar, actualizar o eliminar a un psicólogo
39	13	Interacción de la IU con el usuario para la segunda pestaña del panel de administrador para poder buscar, mostrar y traspasar a otro psicólogo uno o varios expedientes
40	13	Diseñar la clase modelo para poder poner los datos de la BD en un Excel y agregar sentencias SQL para poder obtener los datos de la BD
41	13	Interacción de la IU con el usuario para poder buscar, mostrar, seleccionar la cantidad de expedientes, poder eliminarlos y exportarlos a un Excel

Tabla 3.71: Tareas a realizar para la cuarta iteración *Fuente: Elaboración propia*

Los detalles de las tareas de ingeniería propuestas en la Tabla 3.71 se especifican de la Tabla 3.72 a la 3.79.

Número de tarea: 34	Número de historia: 12
Nombre de tarea: Crear interfaz del módulo de los test psicométricos	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.6
Fecha inicio: 13/01/25	Fecha fin: 17/01/25
Programador responsable: Juan Luis Torres Uribe	
Descripción: Realizar una interfaz en el módulo de los test psicométricos que contenga todos los resultados del test MMPI 2 RF, pensamientos automáticos, y los test de ansiedad y depresión así como un campo de texto para poder ingresar observaciones o interpretaciones de los resultados obtenidos en estos test.	

Tabla 3.72: Tarea de Ingeniería 34 *Fuente: Elaboración propia*

Número de tarea: 35	Número de historia: 12
Nombre de tarea: Crear la tabla de la base de datos que contenga los datos de los test psicométricos	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.6
Fecha inicio: 18/01/25	Fecha fin: 21/01/25
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear una tabla en la BD que contenga todos los requerimientos para que se ingresen todos los datos pensados para almacenarlos aquí.	

Tabla 3.73: Tarea de Ingeniería 35 *Fuente: Elaboración propia*

Número de tarea: 36	Número de historia: 12
Nombre de tarea: Interacción con la base de datos para poder crear, mostrar y actualizar los datos de los test psicométricos	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.8
Fecha inicio: 22/01/25	Fecha fin: 03/02/25
Programador responsable: Juan Luis Torres Uribe	
Descripción: Diseñar todas las operaciones lógicas para poder tener todas las relaciones necesarias con estos datos y la interfaz que son insertar, mostrar, actualizar y eliminar de esta tabla.	

Tabla 3.74: Tarea de Ingeniería 36 *Fuente: Elaboración propia*

Número de tarea: 37	Número de historia: 13
Nombre de tarea: Crear la interfaz que contendrá las funciones de los administradores divida en pestañas	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.6
Fecha inicio: 03/02/25	Fecha fin: 07/02/25
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear la interfaz del panel de administración subdividida en pestañas internas para seccionar cada tarea que se tendrá, como la administración de los psicólogos, el traspaso de los expedientes y la exportación de los mismos.	

Tabla 3.75: Tarea de Ingeniería 37 *Fuente: Elaboración propia*

Número de tarea: 38	Número de historia: 13
Nombre de tarea: Interacción de la IU con el usuario para la primera pestaña del panel administrador para poder buscar, mostrar, actualizar o eliminar a un psicólogo	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 08/02/25	Fecha fin: 14/02/25
Programador responsable: Juan Luis Torres Uribe	
Descripción: Diseñar las instrucciones lógicas para hacer que en la tabla principal de su pestaña se muestre todos los psicólogos dados de alta en el sistema así como buscarlos para poder actualizarlos en la misma tabla o seleccionarlos y eliminarlos.	

Tabla 3.76: Tarea de Ingeniería 38 *Fuente: Elaboración propia*

Número de tarea: 39	Número de historia: 13
Nombre de tarea: Interacción de la IU con el usuario para al segunda pestaña del panel de administrador para poder buscar, mostrar y traspasar a otro psicólogo uno o varios expedientes	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 15/02/25	Fecha fin: 21/02/25
Programador responsable: Juan Luis Torres Uribe	
Descripción: Diseñar las instrucciones lógicas para que en la tabla principal se muestren todos los expedientes ya dados de alta así como poder buscarlos para poder seleccionar uno o varios y de ahí escoger en un una lista de los psicólogos dados de alta y elegir a cuál de ellos traspasarle los expedientes seleccionados.	

Tabla 3.77: Tarea de Ingeniería 39 *Fuente: Elaboración propia*

Número de tarea: 40	Número de historia: 13
Nombre de tarea: Diseñar la clase modelo para poder poner los datos de la BD en un Excel y agregar sentencias SQL para poder obtener los datos de la BD	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.8
Fecha inicio: 22/02/25	Fecha fin: 07/03/25
Programador responsable: Juan Luis Torres Uribe	
Descripción: Crear la clase que inicializará el Excel y contendrá los datos exportados de los expedientes, así como revisar y modificar la lógica para obtener estos datos de la BD y poder acomodarlos precisamente en el Excel.	

Tabla 3.78: Tarea de Ingeniería 40 *Fuente: Elaboración propia*

Número de tarea: 41	Número de historia: 13
Nombre de tarea: Interacción de la IU con el usuario para poder buscar, mostrar, seleccionar la cantidad de expedientes, poder eliminarlo y exportarlos a un Excel	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.6
Fecha inicio: 08/03/25	Fecha fin: 14/03/25
Programador responsable: Juan Luis Torres Uribe	
Descripción: Manejar las funciones de la clase de la creación del Excel para tenerla lógicamente en la interfaz y así sea capaz no solo de mostrar y buscar expedientes dentro de la tabla de la IU sino también seleccionarlos para poder exportarlos o eliminarlos.	

Tabla 3.79: Tarea de Ingeniería 41

Una vez teniendo descritas las tareas de ingeniería que se harán en esta iteración, ahora se diseñarán las clases e interfaces que se necesitan para esta iteración.

Diseño y codificación

En este apartado se describirán las tarjetas CRC que se necesitan para poder mostrar las clases a usarse en esta iteración así como la codificación de las interfaces necesarias en esta fase.

Las tarjetas CRC desarrolladas para esta iteración se describen de la Tabla 3.80 a la 3.82.

Test psicométricos	
Responsabilidades	Colaboradores
Crear un instancia del test psicométricos que contenga todos los campos necesarios	Expediente
Editar todos los resultados de los test psicométricos que se necesiten	
Borrar información de este apartado	
Obtener la información del test psicométrico dependiendo del expediente	
Validar campos	

Tabla 3.80: Tarjeta CRC Test Psicométricos *Fuente: Elaboración propia*

Exportar Excel	
Responsabilidades	Colaboradores
Crear el archivo que contendrá los datos exportados	Expediente
Crear una hoja por cada tabla de la BD	
Acomodar cada hoja con los datos según su naturaleza	
Imponer reglas de estilización a las celdas y títulos de cada hoja	
Calcular el progreso en porcentaje para mostrarlo en la IU	

Tabla 3.81: Tarjeta CRC Exportación a Excel *Fuente: Elaboración propia*

Obtener datos Excel	
Responsabilidades	Colaboradores
Traer los datos formateados para que los pueda usarse en Excel	Expediente

Tabla 3.82: Tarjeta CRC Obtención de datos para el Excel *Fuente: Elaboración propia*

Al ser aceptados los modelos de las clases se procedió a el diseño de las interfaces que pondrán estos requerimientos.

Psicólogo en sesión

Logo: UAH Dirección de Servicio Médico Universitario

Numero de Cuenta: 378732 Correo: to378732@uaeh.edu.mx Siempre estoy aqui

Nombre: Juan Luis Torres Uribe Horaio:

Telefono: 7751691518

Motivo de Consulta: No se sinete motivado con su vida

Observaciones:

Historial Clínico Sintomas Familiograma Notas Clínicas Plan de Tratamiento **Test Psicométricos**

Test Psicométricos

Ingresa los resultados de la aplicación de los siguientes test:

MMPI 2 RF

INVA-r: 2	AP: 46	CR9: 21	ANS: 62	ESO: 2
INVE-r: 65	AC/PE: 21	MAL: 62	TEN: 62	TIM: 62
F-r: 45	CRd: 25	QGI: 62	LCM: 48	DES: 26
Fpsi-r: 12	CR1: 12	QDC: 68	MEM: 21	IEL: 65
Fs: 21	CR2: 65	QNEU: 21	PCIJ: 8	IFM: 24
FVS-r: 56	CR3: 21	QCO: 56	ABS: 2	AGGR-r: 15
St: 21	CR4: 8	ISU: 65	AG: 12	PSYC-r: 35
L-r: 18	CR6: 53	Im/D: 18	EUF: 26	DISC-r: 12
K-r: 65	CR7: 18	DSM: 23	PFA: 25	NEGE-r: 18

Figura 3.14: Interfaz del módulo de los Test Psicométricos: MMPI Fuente: *Elaboración propia*

Psicólogo en sesión

UAH
Dirección de Servicio Médico Universitario

Numero de Cuenta: 378732 Correo: to378732@uaeh.edu.mx Siempre estoy aqui

Nombre: Juan Luis Torres Uribe Horaio:

Telefono: 7751691518

Motivo de Consulta
No se sinete motivado con su vida

Observaciones

Historial Clínico Sintomas Familiograma Notas Clinicas Plan de Tratamiento **Test Psicométricos**

Pensamientos Automáticos

Tipo de Distorsión	Puntuación Total
Filtraje	12
Pensamiento Polarizado	21
Sobregeneralización	32
Interpretación del Pensamiento	1
Visión Catastrófica	52
Personalización	6
Falacia de Control	13
Falacia de Justicia	21
Razonamiento Emocional	53
Falacia de Cambio	32
Etiquetas Globales	12
Culabilidad	21

Figura 3.15: Interfaz del módulo de los Test Psicométricos: Pensamientos automáticos *Fuente: Elaboración propia*

The screenshot displays a web application window titled "Psicólogo en sesión". The interface is divided into several sections:

- Header/Top Bar:** Contains the UAH logo (Dirección de Servicio Médico Universitario) on the left. In the center, there are input fields for "Numero de Cuenta:" (378732), "Correo:" (to378732@uaeh.edu.mx), "Nombre:" (Juan Luis Torres Uribe), and "Telefono:" (7751691518). On the right, there is a "Siempre estoy aqui" session timer and a gear icon for settings.
- Main Content Area:**
 - Top Section:** A tab labeled "Palacia de Recompensa Divina" with a dropdown menu showing "1".
 - Test Results:** Two side-by-side sections:
 - Test Depresión:** A dropdown menu showing "Depresión Leve".
 - Test Ansiedad:** A dropdown menu showing "Ansiedad Moderada".
 - Observaciones/Interpretación:** A large text area containing the text "La verdad esta mal pero definitivamente yo estoy peor jajajaajajaaj".
 - Observaciones:** A separate text area on the left side of the main content.

Figura 3.16: Interfaz del módulo de los Test Psicométricos: Test Ansiedad y Depresión *Fuente: Elaboración propia*

Como se observa en las Figuras 3.14 a la 3.16 se muestra la construcción real de las interfaz para el módulo de los test psicométricos, en el se observan todos los resultados para cada uno de estos test empezando con el de MMPI 2 RF, siguiendo con el de pensamientos automáticos y acabando con las categorías de los resultados de los test de depresión y ansiedad además de un cuadro de texto para que puedan poner la interpretación de los resultados obtenidos u observaciones de las mismas y tener registro de eso.

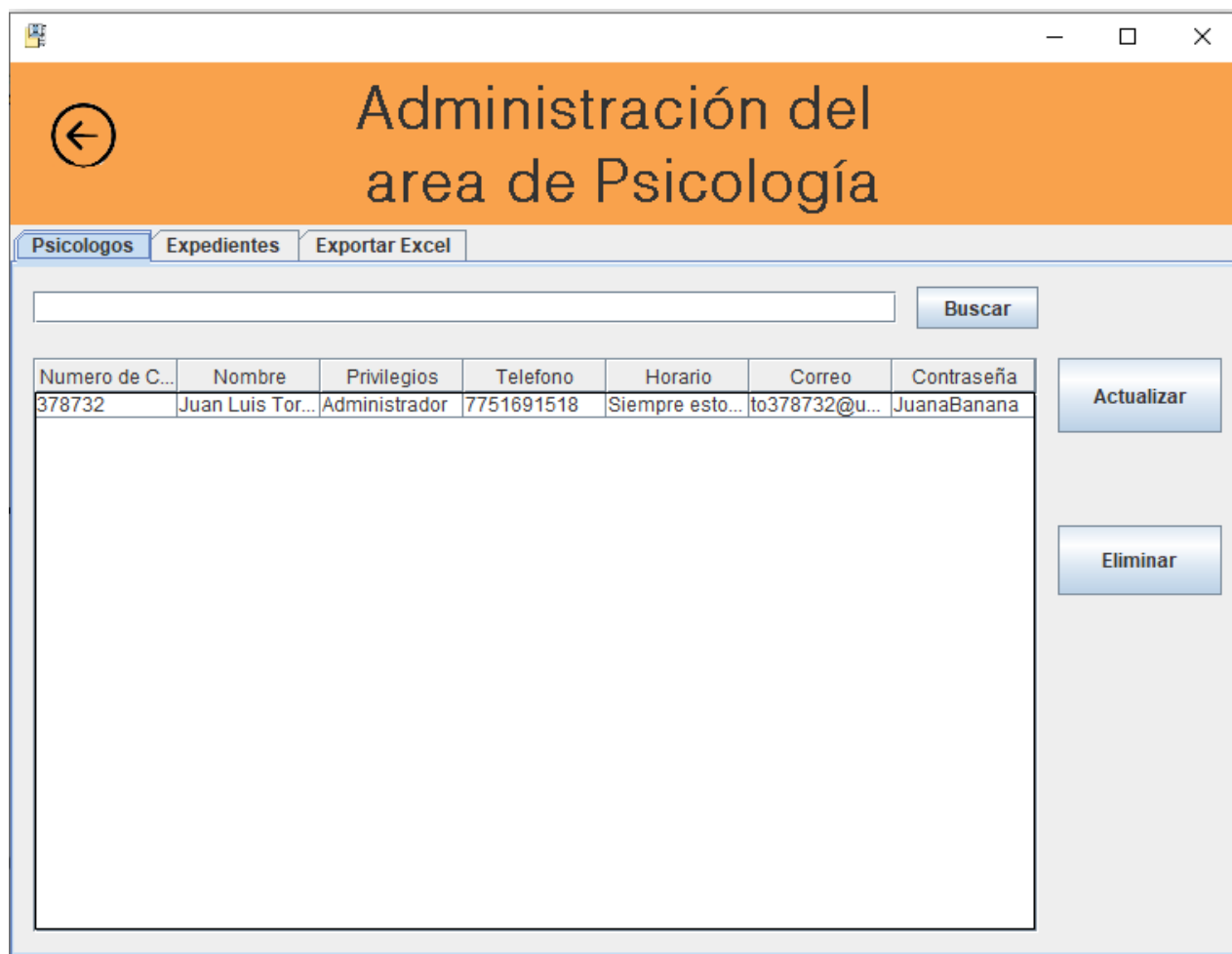


Figura 3.17: Interfaz del administrador: Módulo de psicólogos *Fuente: Elaboración propia*

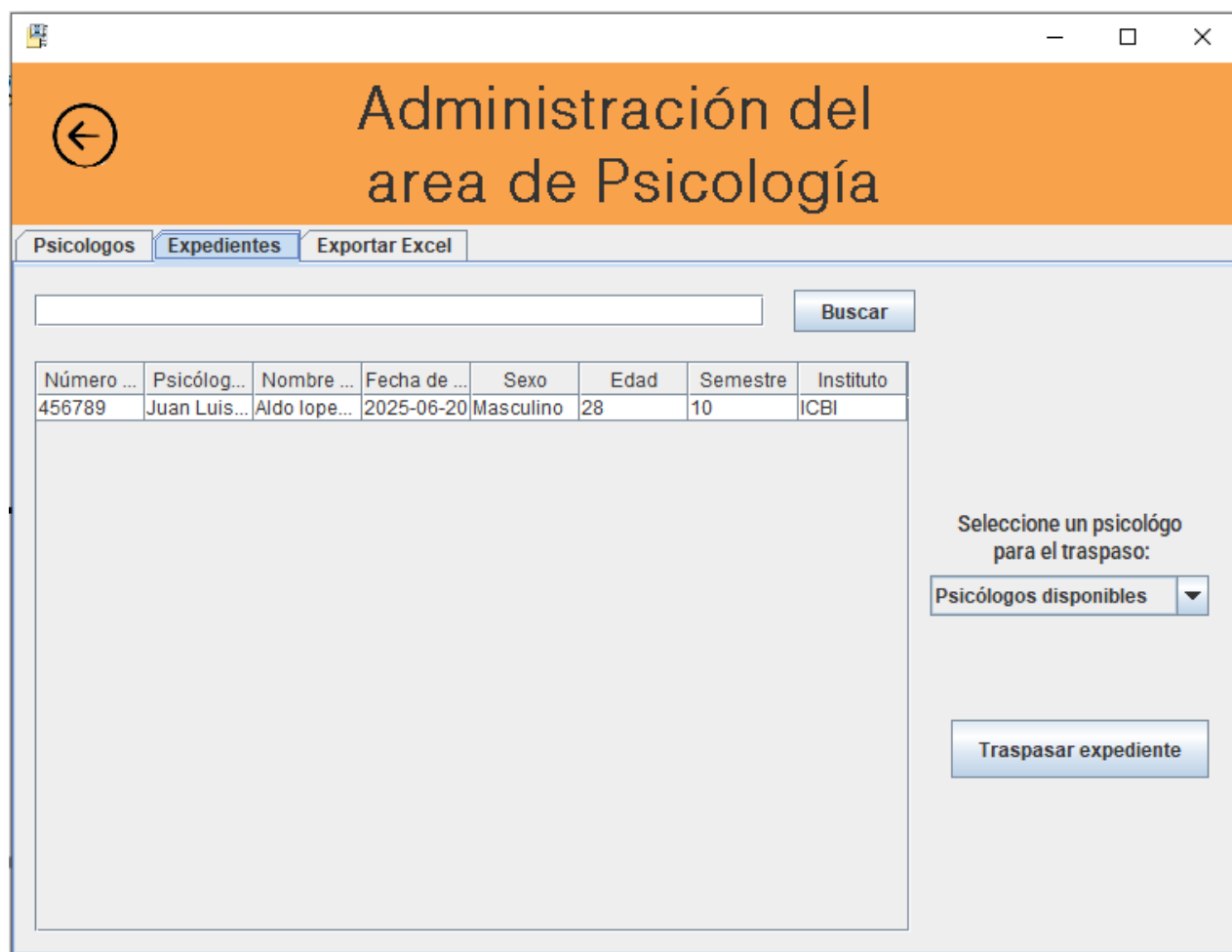


Figura 3.18: Interfaz del administrador: Módulo de expedientes *Fuente: Elaboración propia*

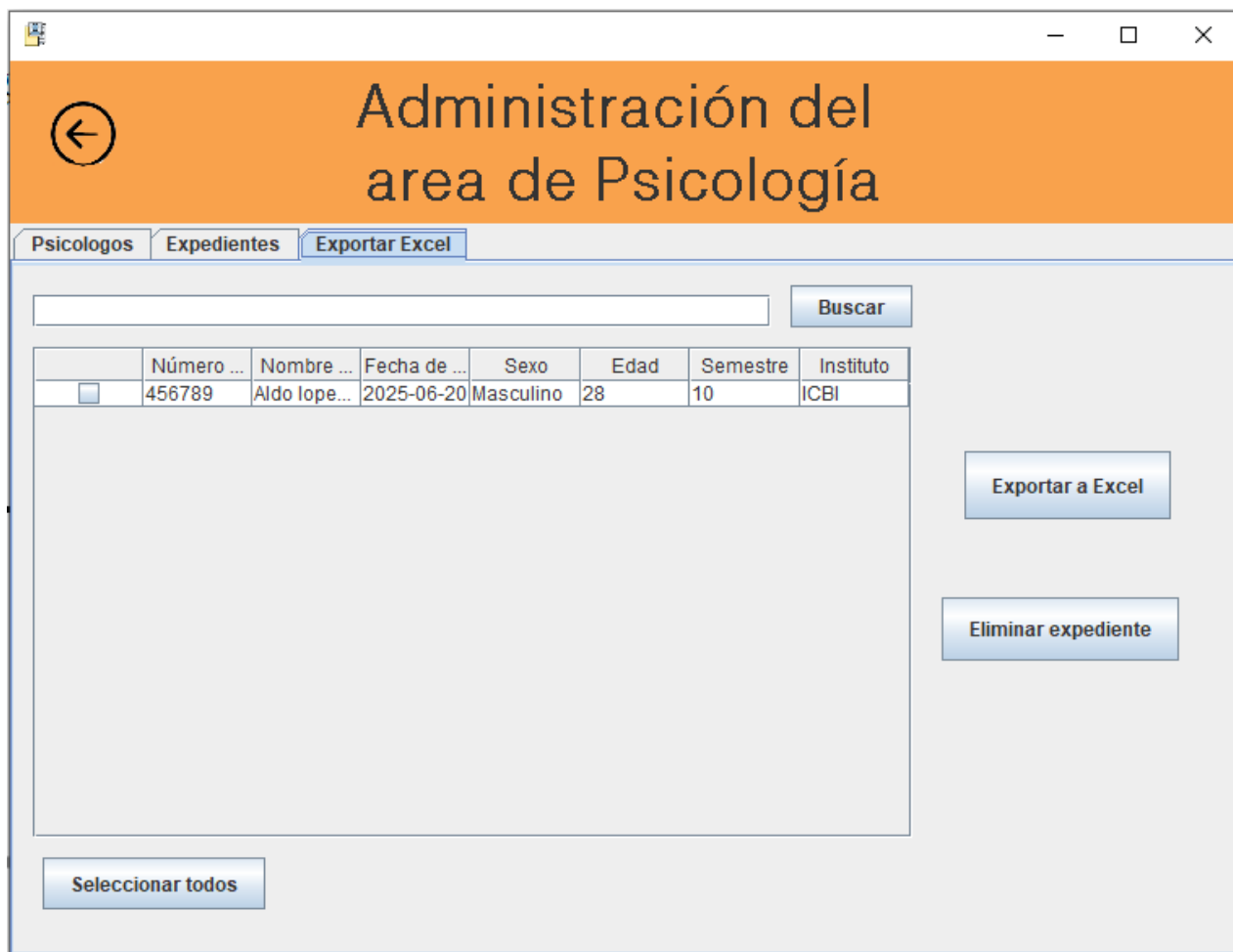


Figura 3.19: Interfaz del administrador: Módulo de exportación de los expedientes *Fuente: Elaboración propia*

Ahora en las Figuras 3.17 a la 3.19 se observa la construcción de la interfaz para el uso del administrador del sistema contado con sus 3 módulos para cada una de las tareas que puede realizar el administrador, como lo es la pestaña psicólogos donde se muestra a todos los psicólogos registrados y se pueden hacer modificaciones a ellos, la pestaña de expedientes donde se puede traspasar expedientes entre psicólogos y la pestaña de exportación donde se pueden escoger los expedientes que se muestran para poder exportarlos a un Excel y así aprovechar la información que se tiene de ellos.

Terminado con el desarrollo de este diseño y para comprobar que estas codificaciones realizan lo especificado en los requerimientos que se pidieron, se pasará a la parte de las pruebas para comprobar el funcionamiento de estas.

Pruebas

En esta última sección de la iteración se verificará que se tiene lo necesario para cumplir con los requerimientos propuestos para las historias de usuario que se propusieron para la iteración. Por tal motivo, en las Tablas 3.83 y 3.84 se tienen detalladamente las pruebas hechas y los resultados obtenidos de estas mismas.

Caso de prueba	
Código: 9	Nº Historia de usuario: 12
Historia de usuario: Resultados de test psicométricos aplicados.	
Condiciones de ejecución: Al entrar a un expediente y estar en el módulo del los test psicométricos se puede modificar las entradas a los resultados de los test requeridos sin la condición de llenar todos a la vez o tener el resultado de todos los test integrados para así poder hacer que estas pruebas se hagan ya sea dentro del tiempo requerido con el paciente y al mismo tiempo llenar sus resultados en el momento necesitado.	
Entradas/Pasos de ejecución: 1. Darle clic al expediente con el que se quiera trabajar 2. Dar clic en la interfaz secundaria en el modulo del test psicométrico 3. Editar los resultados de los test que se vayan a ingresar. 4. Dar clic en el botón de <i>guardar</i> de la interfaz principal del paciente para actualizar o insertar los datos agregados.	
Resultado esperado: Los datos ingresados se guarden correctamente puedan ser modificados y actualizados, y que al cargar nuevamente este expediente se muestren.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 3.83: Caso de prueba 12 *Fuente: Elaboración propia*

Caso de prueba	
Código: 9	Nº Historia de usuario: 13
Historia de usuario: Psicólogo administrador.	
Condiciones de ejecución: Al entrar en sesión un psicólogo como administrador debe de ver un botón especial para ingresar a las funciones de administrador, dentro de esta debe de ver las 3 pestañas con las actividades administrativas que puede realizar, debe de poder realizar cualquiera de las 3 sin ningún tipo de contratiempo.	
Entradas/Pasos de ejecución: 1. Que un psicólogo administrador entre en sesión al programa. 2. Dar clic en el botón que solo ven los administradores para que los lleve a la interfaz de administración. 3. Verificar que si se pudo ingresar a la interfaz de administrador. 4. Estando en la primera pestaña que es la de los <i>psicólogos</i>, mostrar que si se muestran todos los que están registrados, buscar uno de ellos, actualizar la información de uno y eliminar a uno. 5. Dar clic a la siguiente pestaña la de los <i>expedientes</i> y ver que en la interfaz se muestren todos los expedientes ingresados, buscar uno y traspasar uno y después varios a algún psicólogo diferente al que ya los tiene. 6. Dar clic a la siguiente pestaña la que es para la <i>exportación a Excel</i> y verificar que se muestren todos los expedientes ingresados, verificar que se puedan eliminar alguno y exportar la información de uno y después varios al mismo tiempo y verificar como se exportan estos datos.	
Resultado esperado: Que todos los cambios hechos dentro de los diferentes apartados se vean actualizados en las sesiones de los demás psicólogos así como confirmar que el Excel obtenido a base de la exportación contenga los datos que se mandaron para su exportación.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 3.84: Caso de prueba 13 *Fuente: Elaboración propia*

Con la finalización de las pruebas satisfactoriamente y con la aprobación de los directores de psicología del hospital el desarrollo del programa ha terminado y se podrá pasar a las últimas fases del desarrollo para poner el programa y observar que no haya ningún problema al estar en funcionamiento.

3.4 Fase de producción

En esta sección de la metodología se explicará cómo es que se instala el sistema en esta institución, describiendo las estructura informática de la institución y la construcción de la base de datos en la cual va a ser almacenada.

Construcción informática

El diseño del sistema se planea que se use en cada computadora de los consultorios que se tiene. Actualmente solo se cuenta con la computadora de la directora del área de psicología así como otras dos en los demás consultorios para integrar el sistema pero se quiere llegar a tener una en cada consultorio e integrarse en una red local para comunicar la información y guardarla en la base de datos. El servidor que contendrá la base de datos será la computadora de la directora debido a que ella es la única que debe de tener accesos a los datos confidenciales de raíz.

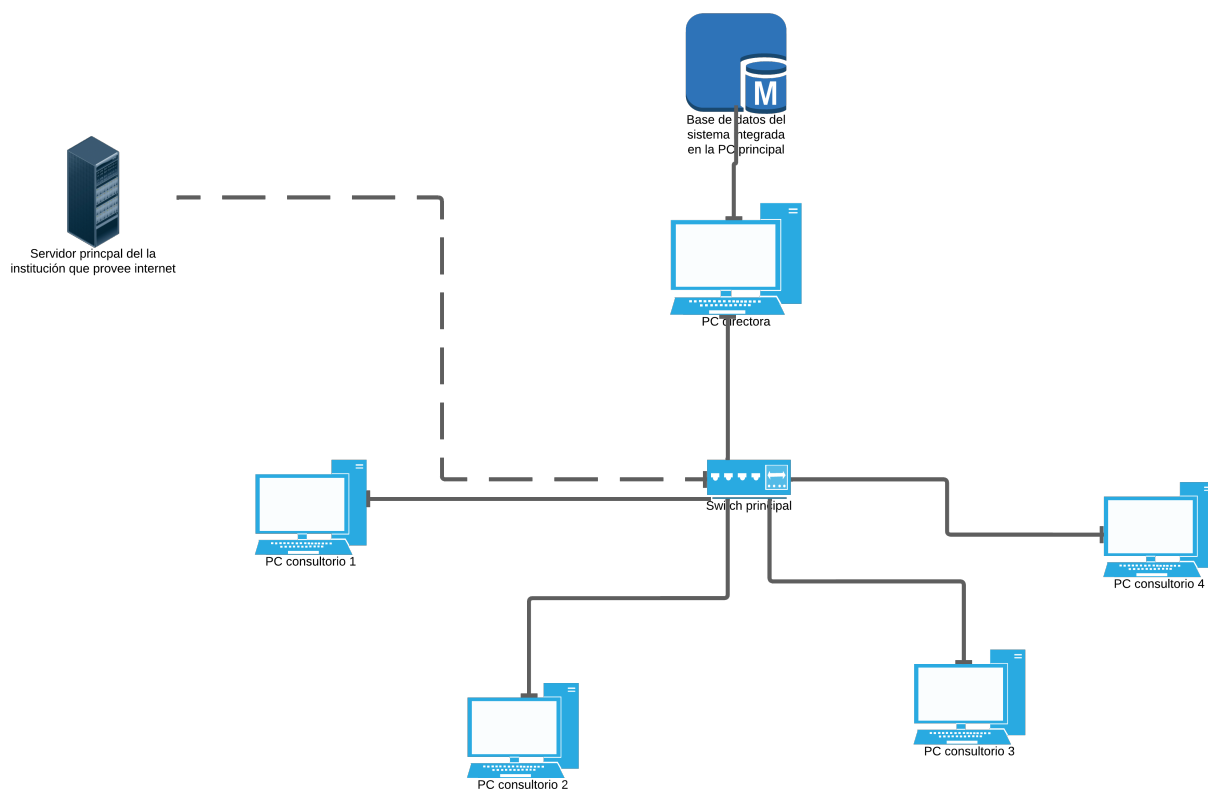


Figura 3.20: Diagrama de la estructura de red *Fuente: Elaboración propia*

En la Figura 3.20 se observa el diagrama con la estructura que se quiere llevar acabo para

poder realizar una buena comunicación entre el servidor de la base de datos con los demás consultorios dentro de la red local creada para esta área.

Desarrollo de la base de datos

Para la creación de la base de datos se crearon 2 modelos los cuales ayudaron a saber cómo es que se iba a relacionar y obtener la base de datos que cumpla con los objetivos necesarios en la información que va a guardar el sistema.

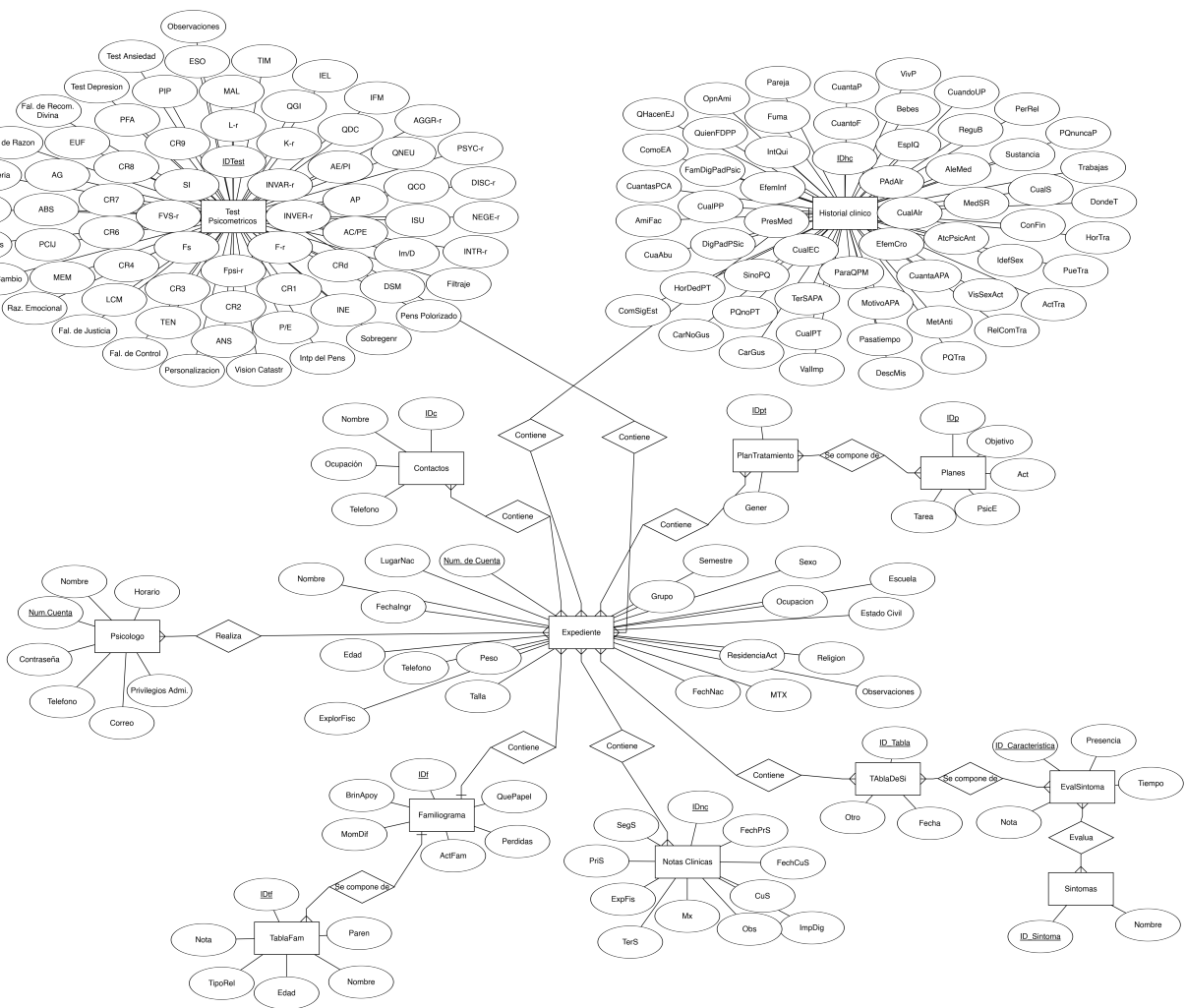


Figura 3.21: Modelo entidad-relación de la base de datos *Fuente: Elaboración propia*

En la Figura 3.21 se observa el modelo de entidad-relación que tiene la base de datos. Éste contiene todas las entidades que van a formar nuestras relaciones y cómo funcionan entre sí, las

más destacables son la de psicólogos que crea los expedientes y como esta identidad de expediente se relaciona con todas las demás identidades para poder obtener y guardar correctamente cada dato en la base de datos.

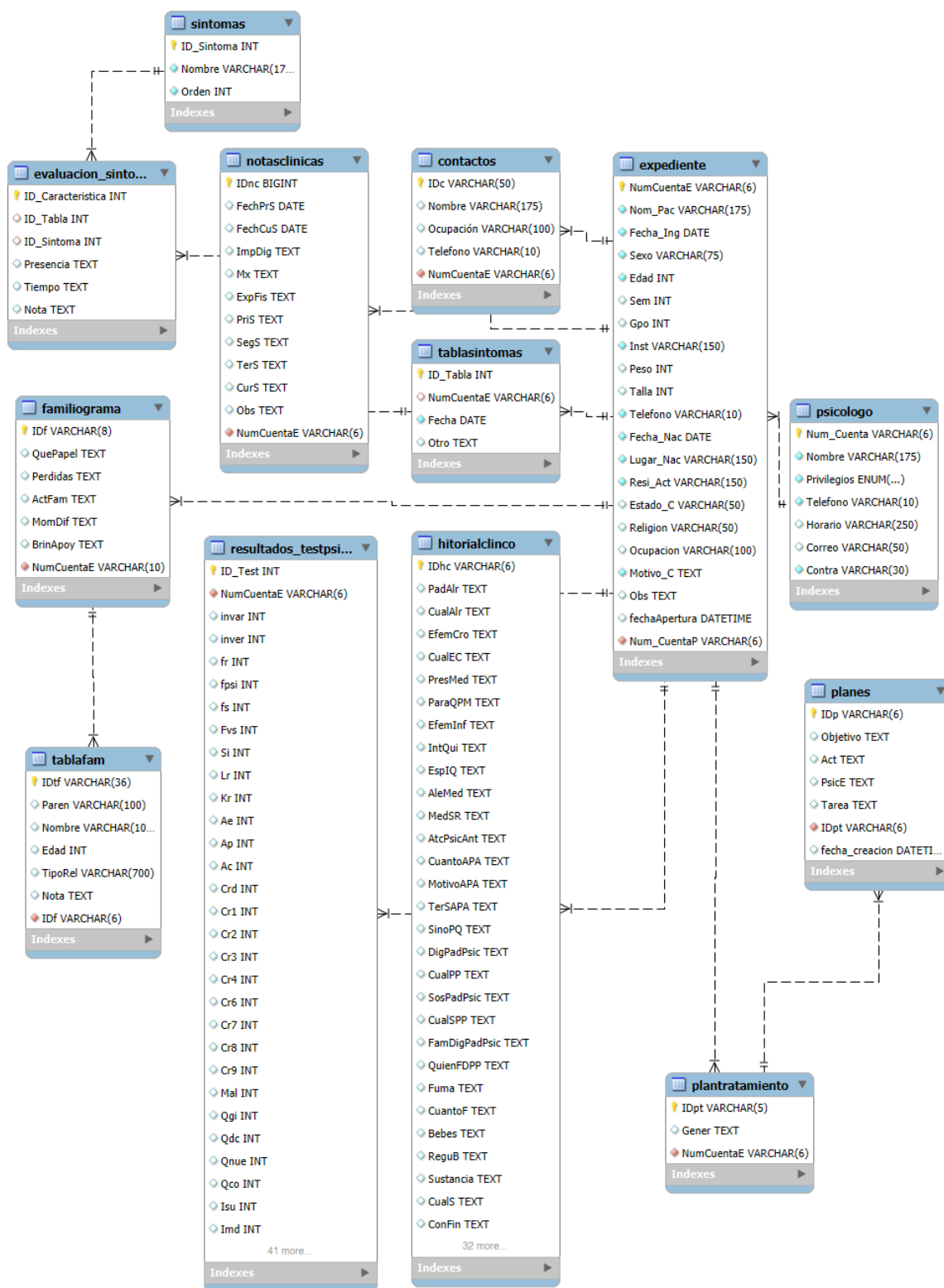


Figura 3 22: Modelo relacional de la base de datos *Fuente: Elaboración propia*

Así se llegó a la realización del modelo relacional de la base de datos en la Figura 3.22. Dicha figura expone como es que está constituido, teniendo una forma más visual de la construcción de las tablas de la base de datos y sus relaciones entre las relaciones de las llaves principales y foráneas.

De esta manera, se da por concluida la fase de producción, marcando el hito final del ciclo de desarrollo iterativo e incremental de XP. La liberación exitosa del sistema en el entorno real del Área de Psicología de la DSMU no sólo representa la materialización de las historias de usuario planificadas, sino la entrega de valor tangible al cliente, cumpliendo con el objetivo primordial del proyecto: proveer una herramienta funcional, robusta y en operación activa que optimice la gestión de expedientes clínicos.

Una vez que el sistema se encontró en operación estable, se procedió a la siguiente etapa fundamental: la evaluación de su impacto y la percepción de los usuarios mediante la aplicación de un instrumento de evaluación, cuyos resultados se presentan en la siguiente fase.

3.5 Fase de mantenimiento

Una vez que el Sistema de Gestión de Expedientes Clínicos fue desplegado y se encontró en pleno funcionamiento dentro del Área de Psicología de la DSMU, el proyecto entró formalmente en la fase de mantenimiento establecida por Extreme Programming. Para dar inicio a este ciclo de mejora continua y obtener una evaluación cuantificable del sistema en su entorno real, se implementó el mecanismo de retroalimentación por excelencia: la aplicación de un instrumento de evaluación dirigido a los usuarios finales. Esta acción concreta representa la materialización del principio de feedback rápido y constituye la base para las iteraciones futuras del software.

3.5.1 Instrumento de Evaluación: Construcción y Aplicación

Para materializar el proceso de evaluación en esta fase de mantenimiento, se diseñó un instrumento *ad hoc* que permitiera recabar datos tanto cuantitativos como cualitativos. La decisión de construir un cuestionario de estas características se basó en la necesidad de obtener una valoración integral del sistema, que combinara una métrica estandarizada de usabilidad con una evaluación específica de su impacto en los procesos clínicos de la DSMU.

Diseño y Estructura del Cuestionario

El instrumento se estructuró en dos secciones fundamentales:

1. **Evaluación de la Usabilidad (Preguntas 1-10):** Se incorporó la **Escala de Usabilidad del Sistema (SUS, por sus siglas en inglés)**, creada por John Brooke en 1986. Se seleccionó este instrumento por ser una herramienta robusta, validada internacionalmente y de rápida aplicación, que proporciona una medida confiable de la usabilidad percibida. Está compuesta por 10 ítems que se responden en una escala Likert de 5 puntos, y su diseño, que alterna afirmaciones positivas y negativas, mitiga el sesgo de respuesta.
2. **Evaluación de la Eficacia y Eficiencia (Preguntas 11-18):** Esta sección, de elaboración propia, fue diseñada para medir el impacto concreto del sistema en el trabajo diario del Área de Psicología. Su construcción se fundamentó en dos pilares teóricos:
 - La **norma ISO/IEC 25010**, que provee un modelo de calidad para software. De ella, se tomaron los atributos de “Eficiencia de rendimiento” y “Funcionalidad” para evaluar cómo el sistema optimiza el tiempo y los recursos, y si satisface las necesidades explícitas de los usuarios.
 - El **manual de la Organización Mundial de la Salud (OMS) “Monitoring and Evaluating Digital Health Interventions”**, el cual proporciona un marco para evaluar el éxito de las herramientas digitales en salud. Este documento fundamenta la evaluación del **impacto en la atención clínica**, enfocándose en la mejora de los procesos, la reducción de errores y la calidad de la información.

De esta sección, 5 ítems (preguntas 11 a 15) utilizan una escala Likert para cuantificar percepciones clave, mientras que las 4 preguntas finales (16 a 18) son abiertas, con el objetivo de capturar retroalimentación cualitativa rica en detalles, sugerencias de mejora y narrativas sobre el cambio en la práctica profesional.

Aplicación del Instrumento

El cuestionario se distribuyó de forma digital mediante un formulario de Google Forms a un total de 10 integrantes de la comunidad de la DSMU. El grupo de participantes estuvo conformado por los 2 Jefes del Área de Psicología y 8 voluntarios, practicantes y pasantes del mismo departamento que han utilizado el sistema de manera activa en su trabajo clínico. La aplicación se realizó una vez que los usuarios tuvieron un periodo de uso suficiente para familiarizarse con todas las funcionalidades, garantizando así que sus respuestas estuvieran basadas en una experiencia real y sustancial.

3.5.2 Metodología de Análisis de Datos

Para garantizar un análisis riguroso y sistemático de los datos recabados mediante el instrumento de evaluación, se definieron procedimientos específicos para cada tipo de dato: cuantitativo (escalas Likert) y cualitativo (preguntas abiertas).

Procesamiento de la Escala de Usabilidad del Sistema (SUS)

El análisis de las primeras 10 preguntas, correspondientes a SUS, se llevará a cabo siguiendo el algoritmo estandarizado propuesto por Brooke (1986):

1. **Recodificación de las Respuestas:** A cada una de las 10 respuestas se le asignará un nuevo valor numérico (valor de contribución) en una escala de 0 a 4, de la siguiente manera:
 - Para los ítems con connotación **positiva** (1, 3, 5, 7, 9): El valor de contribución se calculará restando 1 a la puntuación original del usuario ($Contribución = Respuesta_{Usuario} - 1$).
 - Para los ítems con connotación **negativa** (2, 4, 6, 8, 10): El valor de contribución se calculará restando la puntuación original a 5 ($Contribución = 5 - Respuesta_{Usuario}$).
2. **Cálculo del Puntaje SUS Global:** Una vez recodificadas las 10 respuestas, se sumarán todos los valores de contribución. Esta suma total se multiplicará por 2.5 para obtener el puntaje final del SUS, el cual se sitúa en una escala de 0 a 100. La fórmula global es:

$$\text{Puntaje SUS} = \left(\sum_{i=1}^{10} C_i \right) \times 2.5 \quad (3.1)$$

donde C_i representa el valor de contribución del i-ésimo ítem.

3. **Interpretación del Puntaje SUS Global:** El puntaje obtenido se interpretará utilizando las bandas de percentiles y adjetivos ampliamente aceptadas en la literatura Bangor (2008):
 - **Puntaje ≥ 80.3 :** Usabilidad **Excelente**
 - **Puntaje entre 68.0 y 80.3:** Usabilidad **Buena**
 - **Puntaje entre 51.0 y 68.0:** Usabilidad **Mala**
 - **Puntaje ≤ 51.0 :** Usabilidad **Pobre**
4. **Análisis de Ítems Individuales:** Adicionalmente, con el fin de identificar fortalezas y debilidades específicas en la usabilidad del sistema, se calculará la **media** y la **desviación estándar** para cada uno de los 10 ítems, utilizando para este fin las respuestas originales en la escala Likert (1 a 5).

Procesamiento de las Preguntas de Impacto y Eficiencia (Escala Likert)

Para el análisis de las preguntas 11 a 15, que utilizan una escala Likert de 5 puntos, se emplearán estadísticas descriptivas:

1. **Cálculo de Métricas de Tendencia Central y Dispersión:** Para cada una de las 5 preguntas, se calcularán las siguientes métricas:
 - **Media (Promedio):** Para determinar la tendencia central de las respuestas.
 - **Mediana:** Para identificar el valor central y robusto ante posibles valores atípicos.
 - **Desviación Estándar:** Para medir el grado de dispersión o consenso en las respuestas de los usuarios.
2. **Interpretación Cualitativa de las Medias:** Con el fin de facilitar la interpretación, los promedios obtenidos se categorizarán de la siguiente forma:
 - **1.00 – 2.49:** Percepción **Negativa**.
 - **2.50 – 3.49:** Percepción **Neutral o Indecisa**.
 - **3.50 – 5.00:** Percepción **Positiva**.

Procesamiento de las Preguntas Abiertas

Las respuestas a las preguntas abiertas (16 a 18) serán analizadas mediante la técnica de **análisis de contenido temático**, siguiendo el proceso propuesto por Braun y Clarke (2006):

1. **Familiarización:** Se realizará una lectura repetida y activa de todas las respuestas para lograr una inmersión completa en los datos.
2. **Generación de Códigos Iniciales:** Se identificarán y etiquetarán fragmentos de texto significativos relacionados con los objetivos de evaluación (utilidad, dificultades, mejoras, impacto).
3. **Búsqueda de Temas:** Los códigos similares se agruparán para formar temas o categorías coherentes que capturen patrones relevantes en los datos (ej: “Dificultades Técnicas”, “Sugerencias de Funcionalidad”, “Beneficios en la Organización”).
4. **Revisión y Definición de Temas:** Los temas preliminares se refinarán, asegurando que representen fielmente el conjunto de datos. Se definirán con claridad y se les asignará un nombre.
5. **Elaboración del Reporte:** Los hallazgos se presentarán de forma narrativa. Para cada tema, se describirá su naturaleza, se citará su frecuencia de aparición entre los usuarios y se incluirán fragmentos de texto (citados anónimos) que ejemplifiquen de manera contundente el tema identificado.

3.5.3 Presentación y Análisis de los Resultados

Una vez aplicado el instrumento de evaluación y procesados los datos conforme a la metodología descrita en la Sección 3.5.2, se presentan a continuación los hallazgos obtenidos, iniciando con el análisis de usabilidad.

Resultados de la Escala de Usabilidad del Sistema (SUS)

Tras aplicar el algoritmo de cálculo estandarizado a las respuestas de los 10 participantes, se obtuvo un **puntaje SUS global de 87.5** para el Sistema de Gestión de Expedientes Clínicos. De acuerdo con la escala de interpretación de Bangor (2008), detallada en la sección 3.5.2, este resultado sitúa al sistema en el rango de «**Usabilidad Excelente**», lo que indica una percepción sumamente positiva por parte de los usuarios finales.

El detalle de los puntajes individuales, presentado en la Tabla 3.85 y en la Figura 3.23, donde D1 y D2 representan a la director y subdirector de esta área dentro de la institución, y P3 al P10 representan a los pasantes revela una alta consistencia en las respuestas. El 80 % de los participantes (8 de 10) calificó la usabilidad del sistema como «Excelente», mientras que el 20 % restante (2 de 10) la ubicó en el rango «Bueno». Es significativo destacar que no se obtuvieron puntajes por debajo de 70.0, lo que refuerza la solidez de la alta usabilidad percibida y la efectiva adopción del sistema en el entorno clínico.

Tabla 3.85: Puntajes individuales de la escala SUS *Fuente: Elaboración propia*

Participante	Puntaje SUS	Clasificación
D1	72.5	Buena
D2	70.0	Buena
P3	85.0	Excelente
P4	87.5	Excelente
P5	100.0	Excelente
P6	90.0	Excelente
P7	100.0	Excelente
P8	87.5	Excelente
P9	100.0	Excelente
P10	82.5	Excelente

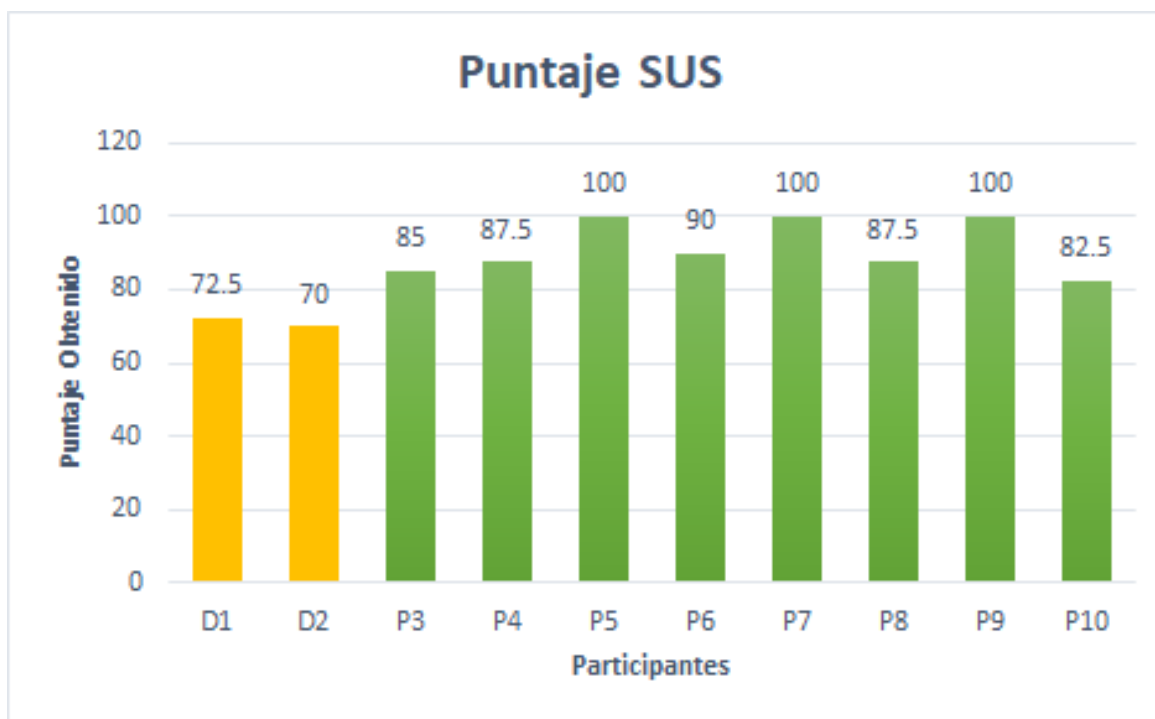


Figura 3.23: Distribución de puntajes SUS individuales. *Fuente: Elaboración propia*

Para identificar las fortalezas y áreas de oportunidad específicas del sistema, se procedió a calcular la media y la desviación estándar para cada uno de los 10 ítems del SUS, utilizando las respuestas originales en la escala Likert (1 a 5), tal como se estableció en la metodología. Los resultados de este análisis se presentan en la Tabla 3.86. Es crucial recordar que los ítems pares (2, 4, 6, 8, 10) están enunciados negativamente; por lo tanto, un promedio *alto* en ellos representa una percepción *desfavorable*, mientras que un promedio *bajo* indica una *fortaleza*. La Figura 3.24 presenta un análisis visual de los promedios obtenidos, donde se han ajustado los ítems pares (negativos) para que un valor alto siempre represente una característica favorable.

Tabla 3.86: Estadísticas descriptivas de los ítems del SUS (Escala 1-5) *Fuente: Elaboración propia*

Ítem	Descripción Breve	Promedio	Desv. Estándar
1	Uso frecuente	4.5	0.67
2	Complejidad	1.4	0.80
3	Facilidad de uso	4.7	0.46
4	Necesidad de apoyo técnico	1.2	0.40
5	Integración de funciones	4.5	0.67
6	Inconsistencia	1.2	0.40
7	Facilidad de aprendizaje	4.7	0.46
8	Engorroso	1.2	0.40
9	Confianza en el uso	4.6	0.49
10	Necesidad de aprender mucho	1.1	0.30

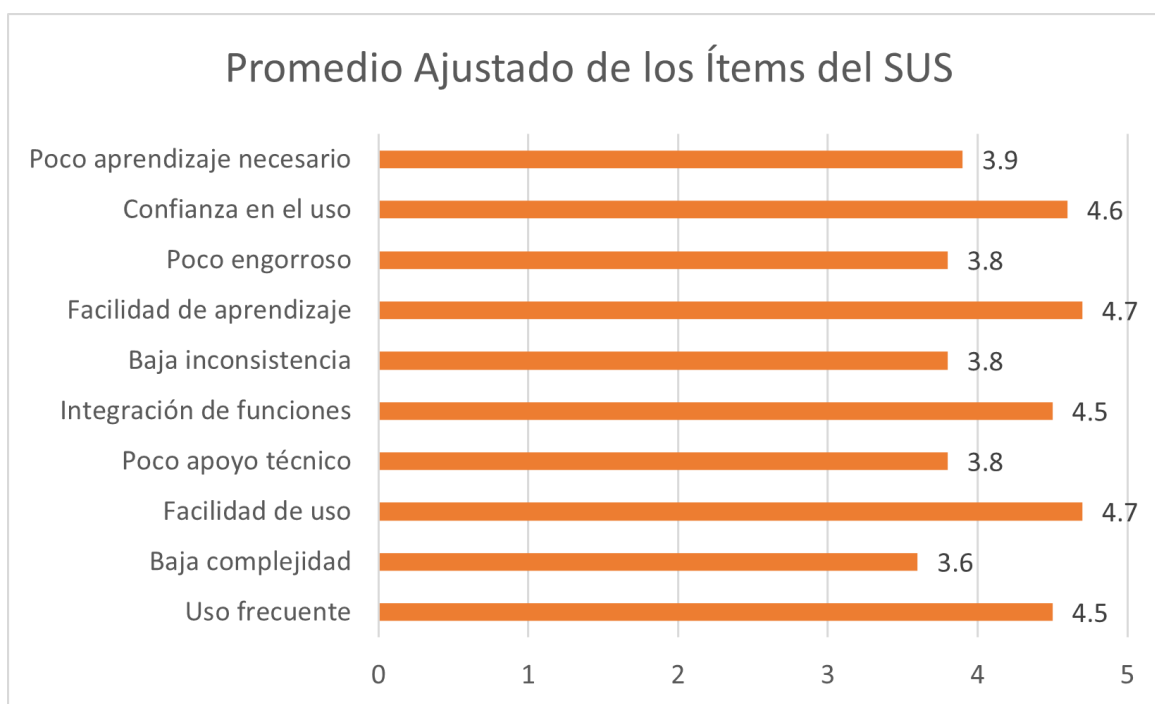


Figura 3.24: Análisis visual de los ítems del SUS. Los ítems pares han sido invertidos para una interpretación coherente (valor alto = favorable). *Fuente: Elaboración propia*

Hallazgos Clave del Análisis por Ítems:

- **Fortalezas Principales:** Los ítems 3 («Considero que el sistema fue fácil de usar») y 7 («Imagino que la mayoría de las personas aprendería a usar este sistema rápidamente»)

recibieron las calificaciones promedio más altas (4.7), seguidas muy de cerca por el ítem 9 («Me sentí confiado al usar el sistema») con 4.6. Esto demuestra que el sistema es percibido como extremadamente intuitivo, fácil de aprender e inspirador de confianza. Asimismo, los ítems pares relacionados con la complejidad (2), la necesidad de apoyo (4, 10), la inconsistencia (6) y lo engorroso (8) obtuvieron promedios muy bajos (inferiores a 1.5), lo que confirma que estas percepciones negativas están prácticamente ausentes entre los usuarios.

- **Área de Oportunidad:** El ítem 1 («Considero que me gustaría usar este sistema frecuentemente»), si bien tiene un promedio alto (4.5), es ligeramente inferior al resto de los ítems positivos. Esto sugiere que, mientras la usabilidad es excelente, la integración perfecta del sistema en la rutina diaria de *todos* los usuarios puede ser el aspecto con mayor margen de mejora, posiblemente mediante la optimización de flujos de trabajo muy específicos.
- **Consenso entre Usuarios:** Las desviaciones estándar calculadas para cada ítem fueron generalmente bajas (en su mayoría por debajo de 0.5), lo que indica un alto grado de consenso entre los participantes sobre la excelente usabilidad del sistema. No se observaron diferencias significativas entre las respuestas de los distintos perfiles de usuarios.

En conclusión, los resultados de SUS confirman de manera contundente que el desarrollo del software fue un éxito desde la perspectiva del usuario final. La usabilidad excelente, respaldada por un consenso generalizado, constituye un pilar fundamental para la adopción efectiva y a largo plazo del sistema en el Área de Psicología de la DSMU.

Resultados de la Evaluación de Impacto y Eficiencia

Para evaluar el impacto concreto del sistema en los procesos de trabajo del Área de Psicología, se analizaron las respuestas a las preguntas 11 a 15, las cuales miden dimensiones de eficacia y eficiencia basadas en los marcos ISO 25010 y OMS. La Tabla 3.87 presenta las estadísticas descriptivas calculadas para cada ítem, conforme a la metodología establecida en la sección 3.5.2.

Tabla 3.87: Estadísticas descriptivas de las preguntas de impacto y eficiencia (Escala 1-5) *Fuente: Elaboración propia*

Ítem	Descripción	Media	Med.	Moda	DE	Interp.
P11	Reducción tiempo registro expediente	4.2	4.5	5	0.98	Positiva
P12	Menos errores en captura de información	3.9	4.0	5	1.52	Positiva
P13	Acceso fácil al historial clínico	4.7	5.0	5	0.46	Muy Positiva
P14	Facilita seguimiento terapéutico	4.4	5.0	5	0.80	Positiva
P15	Mejora desempeño profesional	4.4	5.0	5	1.02	Positiva

*Leyenda: Med. = Mediana; DE = Desviación Estándar; Interp. = Interpretación según escala: 1.00-2.49 (Negativa), 2.50-3.49 (Neutral), 3.50-5.00 (Positiva)

Como se observa en la Tabla 3.87, todas las dimensiones evaluadas obtuvieron medias superiores a 3.5, lo que las sitúa en el rango de **percepción positiva** según los criterios establecidos. El ítem P13, relacionado con la facilidad de acceso al historial clínico, destaca con la media más alta (4.7) y la menor dispersión (DE = 0.46), indicando un consenso prácticamente unánime sobre este beneficio.

La Figura 3.25 presenta una visualización comparativa de los promedios obtenidos en cada dimensión, permitiendo apreciar de manera integral el perfil de impacto del sistema.

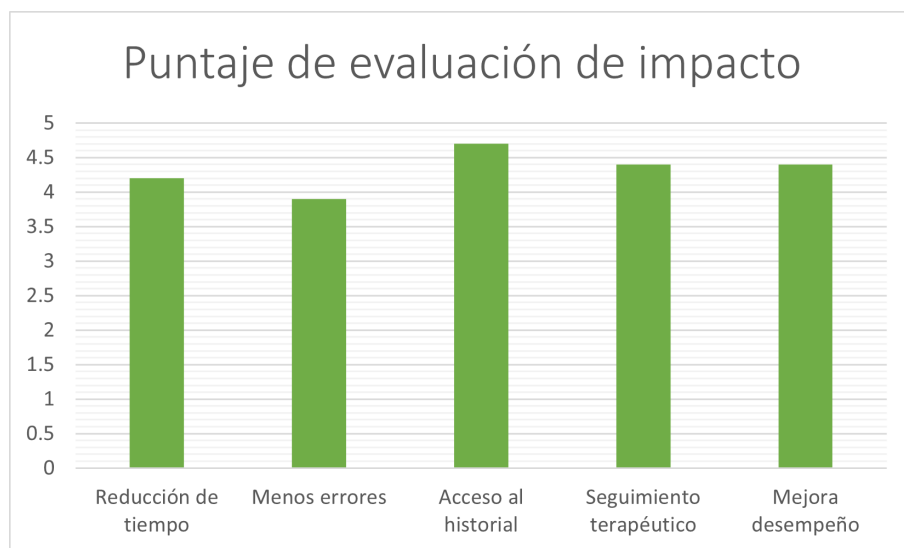


Figura 3.25: Perfil de impacto del sistema en las dimensiones evaluadas (valores promediados)
Fuente: *Elaboración propia*

Análisis Detallado:

- **Beneficio Principal:** La mejora en el **acceso a la información clínica** (P13) es el impacto más reconocido y consistente. Este hallazgo es particularmente relevante en un contexto clínico donde la disponibilidad inmediata de historiales es crucial para la toma de decisiones.
- **Impacto General en la Productividad:** Las preguntas relacionadas con la **reducción de tiempo** (P11) y la **mejora del desempeño profesional** (P15) también obtuvieron calificaciones altas (4.2 y 4.4 respectivamente), confirmando que el sistema cumple su objetivo de optimizar los procesos administrativos-clínicos.
- **Variabilidad en la Reducción de Errores:** La dimensión con mayor dispersión en las respuestas fue P12 (reducción de errores), con una desviación estándar de 1.52. Esto sugiere que, mientras algunos usuarios perciben una disminución significativa en errores, otros mantienen percepciones más moderadas, posiblemente relacionadas con la naturaleza de sus tareas específicas o su nivel de experiencia previa con sistemas digitales.
- **Correlación con Usabilidad:** Los altos puntajes en impacto coinciden consistentemente con la usabilidad excelente medida por el SUS (87.5). Esta correlación respalda la premisa de que un sistema *fácil de usar* es un requisito fundamental para generar *impacto positivo* en los procesos de trabajo.

En síntesis, los resultados confirman que el Sistema de Gestión de Expedientes Clínicos ha generado un **impacto positivo y medible** en la eficiencia operativa del Área de Psicología de

la DSMU, particularmente en el acceso a información, la optimización de tiempo y la mejora general del desempeño profesional.

Resultados del Análisis Cualitativo (Preguntas Abiertas)

Mediante la aplicación de la técnica de análisis de contenido temático descrita en la sección 5.3.2, se procesaron las respuestas a las preguntas abiertas 16, 17 y 18. A continuación, se presentan los temas identificados y su interpretación.

P16: Funcionalidad Más Útil

Como se detalla en la Tabla 3.88, se identificaron **4 temas principales** en las respuestas sobre la funcionalidad más útil, ordenados por frecuencia de mención:

Tabla 3.88: Temas identificados en la funcionalidad más útil *Fuente: Elaboración propia*

Tema	Frecuencia	Cita Representativa
Gestión y Organización de Pacientes en el Panel Principal	40 % (4/10)	“Como organiza a mis pacientes en el panel principal” (Part. 6)
Funcionalidades de Búsqueda y Acceso Rápido	30 % (3/10)	“La función de búsqueda en mi panel de psicólogo” (Part. 5)
Herramientas Específicas de Documentación (Tabla de Síntomas)	20 % (2/10)	“La tabla de síntomas es bastante útil y practica” (Part. 4)
Funciones de Control y Exportación	10 % (1/10)	“El poder exportar los expedientes a un Excel” (Part. 8)

La distribución porcentual de estos temas se visualiza en la Figura 3.26.

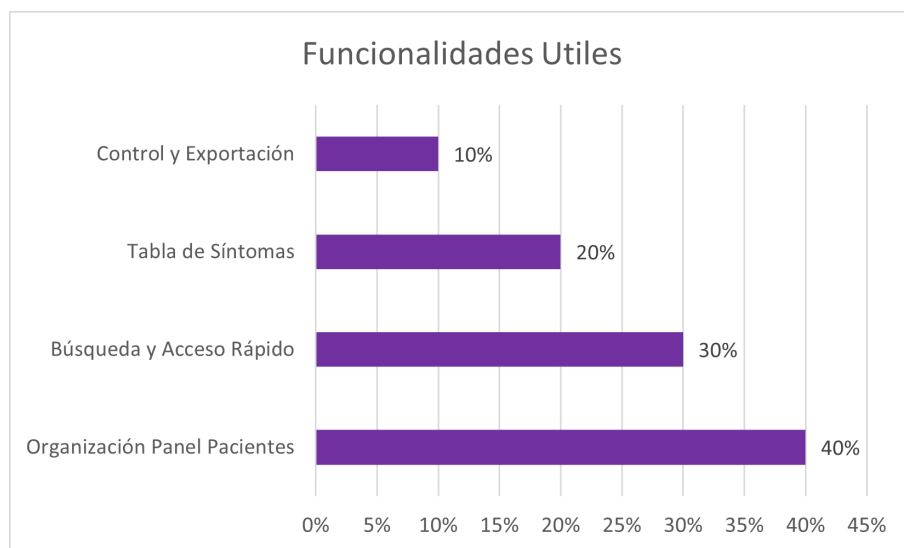


Figura 3.26: Frecuencia de mención de las funcionalidades más útiles del sistema *Fuente: Elaboración propia*

Interpretación: Los usuarios valoran principalmente las funciones que optimizan su flujo de trabajo diario. La organización centralizada de pacientes y el acceso rápido a la información son percibidos como los aportes más valiosos, seguidos por herramientas que estandarizan la documentación clínica, como la tabla de síntomas.

P17: Dificultades Encontradas

El análisis reveló un hallazgo destacado que se resume en la Figura 3.27: el **90 % de los usuarios (9 de 10) no reportó dificultades significativas**, utilizando respuestas como “Ninguna”, “Ninguno” o “Ningún”. Este consenso abrumador corrobora cuantitativamente la usabilidad excelente medida por el SUS.

Únicamente un usuario (10 %) mencionó una observación específica: “*Al ingresar la información no es de lo más cómoda*” (Part. 2). Esta retroalimentación señala un área de oportunidad concreta, aunque menor, relacionada con la ergonomía durante la captura de datos.

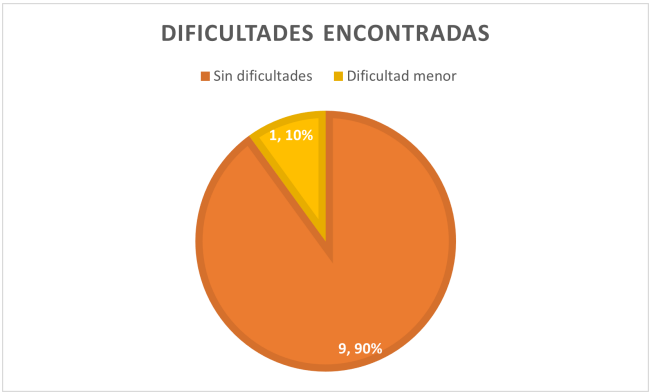


Figura 3.27: Distribución de dificultades reportadas por los usuarios *Fuente: Elaboración propia*

P18: Cambio en el Trabajo

Los impactos positivos reportados por los usuarios se categorizan en la Tabla 3.89, donde se observa que algunos participantes mencionaron más de un tipo de beneficio.

Tabla 3.89: Impactos percibidos en el trabajo clínico *Fuente: Elaboración propia*

Tipo de Impacto	Frecuencia*	Cita Representativa
Mejora en la Organización y Estructura	50 % (5/10)	“Tengo una mejor organización de mis pacientes” (Part. 5)
Incremento en la Eficiencia y Rapidez	40 % (4/10)	“Rapidez y efectividad” (Part. 3)
Acceso y Consistencia en la Documentación	30 % (1/10)	“Puedo llevar mis notas mas consistentemente” (Part. 4)

La comparación visual de la frecuencia de estos impactos se presenta en la Figura 3.28.

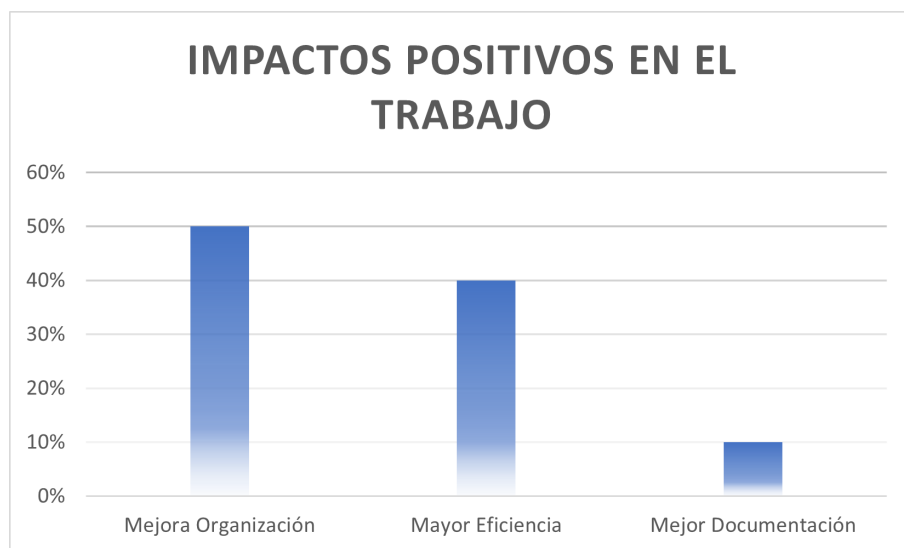


Figura 3.28: Tipos de impacto positivo reportados en el trabajo clínico *Fuente: Elaboración propia*

Interpretación: La implementación del sistema ha generado un cambio positivo y medible en la práctica profesional. Los impactos más citados se relacionan directamente con los objetivos del proyecto: **organizar** la información clínica, hacer los procesos más **eficientes** y garantizar la **consistencia** en los registros.

Síntesis del Análisis Cualitativo

Los hallazgos cualitativos forman una narrativa coherente y potente que complementa y enriquece los datos cuantitativos:

- **Confirmación de Usabilidad:** La casi nula reporte de dificultades (90 %) valida empíricamente el puntaje SUS excelente.
- **Valor Claro y Específico:** Los usuarios no solo están satisfechos, sino que pueden identificar y articular las funciones que resuelven sus problemas operativos concretos (organización, búsqueda, documentación estructurada).
- **Retroalimentación Accionable:** La mención a la comodidad en el ingreso de datos proporciona un insumo valioso y específico para la siguiente iteración de desarrollo dentro de la fase de mantenimiento de XP.
- **Impacto Transformacional:** El cambio en el trabajo se describe en términos de eficiencia, estructura y calidad, demostrando que el sistema ha superado el umbral de ser

una mera herramienta para convertirse en un facilitador clave de la práctica clínica en la DSMU.

Los resultados presentados en este capítulo demuestran de manera contundente el éxito del desarrollo e implementación del Sistema de Gestión de Expedientes Clínicos bajo la metodología Extreme Programming (XP). El sistema ha cumplido y superado las expectativas en sus tres dimensiones fundamentales. En primer lugar, alcanzó una **usabilidad excelente** con un puntaje SUS de 87.5, confirmando que el enfoque centrado en el usuario de XP produjo una interfaz intuitiva y confiable. En segundo lugar, generó un **impacto operativo significativo**, con percepciones positivas (medias superiores a 3.5) en todas las dimensiones de eficacia y eficiencia evaluadas, destacando especialmente la facilidad de acceso al historial clínico (media de 4.7). Finalmente, logró una **alta aceptación y valoración por los usuarios**, quienes no solo reportaron ausencia de dificultades significativas (90 % de los casos), sino que identificaron claramente funcionalidades valiosas y mejoras concretas en su trabajo diario.

Estos hallazgos proporcionan insumos precisos para el siguiente ciclo de mejora continua dentro de la fase de mantenimiento de XP. La única dificultad menor mencionada (ergonomía en la captura de datos) señala una oportunidad de optimización prioritaria, mientras que las funciones más apreciadas (organización del panel, búsqueda, tabla de síntomas) deben potenciarse en futuras iteraciones. De esta manera, se materializan los valores esenciales de XP: la **comunicación** constante con el cliente (DSMU), la **simplicidad** evidenciada en la usabilidad, la **retroalimentación** formal obtenida a través de esta evaluación, el **valor** entregado mediante mejoras tangibles en procesos clínicos, y el **coraje** de implementar una nueva solución en un entorno de salud crítico.

Conclusiones

El presente proyecto de tesis logró exitosamente el diseño, desarrollo e implementación de un sistema de software especializado para la gestión integral de expedientes clínicos en el área de psicología de la Dirección del Servicio Médico Universitario de la Universidad Autónoma del Estado de Hidalgo. Este sistema representa una solución tecnológica robusta que aborda las problemáticas identificadas en la gestión manual de documentos clínicos, garantizando no solo la seguridad y confidencialidad de los datos sensibles, sino también la integridad y disponibilidad de la información médica.

Una de las contribuciones más significativas de este trabajo reside en el establecimiento de un formato estandarizado para la recopilación y organización de los documentos que componen el expediente clínico psicológico. Esta estandarización facilita considerablemente el proceso de transferencia de pacientes entre profesionales de la salud mental dentro de la institución, permitiendo que cada nuevo psicólogo asignado pueda acceder de manera inmediata al historial terapéutico completo, conocer las intervenciones previas realizadas y comprender el plan de tratamiento establecido, lo que se traduce en una continuidad de cuidado más efectiva y una mejor experiencia para el paciente.

En el ámbito metodológico, este proyecto demostró empíricamente los beneficios de implementar la metodología ágil Extreme Programming (XP) en el desarrollo de software para entornos clínicos. La naturaleza iterativa e incremental de XP, junto con su énfasis en la comunicación continua con el cliente, resultó fundamental para comprender las necesidades específicas del personal médico y traducirlas eficientemente en funcionalidades tecnológicas. La práctica de llevar al cliente mas cerca en la etapa del desarrollo, los ciclos de retroalimentación cortos y las pruebas continuas no solo mejoraron la calidad del código sino que aseguraron que el producto final se alineara perfectamente con las expectativas y flujos de trabajo reales de los usuarios finales. Esta aproximación metodológica demostró generar satisfacción tanto en el equipo de desarrollo como en el cliente, validando su efectividad para proyectos de software en el sector salud.

Si bien el sistema implementado cumple cabalmente con los objetivos establecidos inicialmente, se identificaron diversas oportunidades para futuras actualizaciones y expansiones funcionales. Entre estas se incluyen: el desarrollo de módulos avanzados de analítica de datos para identificar tendencias en salud mental estudiantil, la implementación de funcionalidades de telepsicología para consultas remotas, la integración con otros sistemas institucionales existentes, y el fortalecimiento de las capacidades de reporte automático para investigación clínica. Es-

tas mejoras potenciales no demeritan el valor del sistema actual, sino que representan caminos naturales para su evolución futura.

En términos de impacto institucional, la implementación de este sistema de gestión de expedientes clínicos representa un avance significativo en la modernización de los procesos del Servicio Médico Universitario. La solución tecnológica entregada proporciona una estructura de datos sólida y eficiente que optimiza la obtención, almacenamiento y recuperación de información clínica relevante, sentando las bases para una transformación digital más amplia dentro de la institución. Este proyecto no solo resuelve una necesidad inmediata sino que establece un precedente valioso para la incorporación de tecnologías de información en beneficio de los servicios de salud universitaria.

Finalmente, este trabajo contribuye al campo de la informática médica al demostrar cómo principios de ingeniería de software aplicados metódicamente pueden generar soluciones tecnológicas que verdaderamente impacten positivamente en la calidad de la atención en salud mental. El éxito de este proyecto sugiere que aproximaciones similares podrían ser replicadas en otras especialidades médicas dentro del contexto universitario, potencialmente escalando los beneficios a toda la comunidad universitaria.

Bibliografía

- [Abuchar Porras, 2023] Abuchar Porras, A. (2023). *Metodologías ágiles para el desarrollo de software*. Editorial Universidad Distrital Francisco José de Caldas.
- [Aguilar, 1996] Aguilar, L. J. (1996). *Programacion orientada a objetos: conceptos, modelado, diseño, y codificacion en C+*. McGraw-Hill.
- [Aguirre, 2013] Aguirre, J. M. O. (2013). *El expediente clínico*. Editorial El Manual Moderno.
- [Association, 2018] Association, A. P. (2018). Clinical psychology. *APA Dictionary of Psychology*.
- [Bajaña García, 2019] Bajaña García, E. L. (2019). Diseño de un software de registro de historia clínica de pacientes para la clínica odontológica albodental de la ciudad de guayaquil 2018. B.S. thesis, Instituto Superior Tecnológico Bolivariano de Tecnología.
- [Bayona et al., 2016] Bayona, J. C., Pineda, O. L., and Pardo, O. D. (2016). El papel de la ingeniería de software en el desarrollo de aplicaciones. *Tecnología Investigación y Academia*, 4(1):3–14.
- [Cabello, 2010] Cabello, M. V. N. (2010). *Introducción a las bases de datos relacionales*. Vision libros.
- [Camps Paré et al., 2010] Camps Paré, R., Casillas Santillán, L. A., Costal, D., Gibert Ginestà, M., Martín Escofet, C., and Pérez Mora, Ó. (2010). Bases de datos, febrero 2007.
- [Carvajal Nagua and Solano Cedeño, 2020] Carvajal Nagua, K. A. and Solano Cedeño, C. S. (2020). Desarrollo de una aplicación web para el control de citas y manejo de historial médico en la unidad médica family care de la ciudad de guayaquil. B.S. thesis.
- [Celi-Párraga et al., 2023] Celi-Párraga, R. J., Boné-Andrade, M. F., Mora-Olivero, A. P., et al. (2023). Ingeniería del software i: Requerimientos y modelado del software.
- [Cubero, 2020] Cubero, H. (2020). Procesos de ingeniería de software.
- [de Jiménez, 2015] de Jiménez, R. L. M. (2015). Metodologías ágiles de desarrollo de software aplicadas a la gestión de proyectos empresariales. *Revista tecnológica*, 8.

- [Delgado et al., 2023] Delgado, C. E., Siqueira, F. M., Freitas, G. K. S. d., Carvalho, D. B. F., Fortes, F. L. d. S., and Cavalcante, R. B. (2023). Desarrollo de un prototipo de software basado en la cartilla de salud de la persona mayor. *Cogitare Enfermagem*, 28:e88597.
- [Elmasri et al., 2007] Elmasri, R., Navathe, S. B., Castillo, V. C., Pérez, G. Z., and Espiga, B. G. (2007). *Fundamentos de sistemas de bases de datos*. Pearson educación.
- [Fernandez Leon, 2022] Fernandez Leon, E. M. (2022). Desarrollo de un sistema web de historias clínicas electrónicas para mejorar la gestión en el proceso de consultas ambulatorias del hospital belén de lambayeque.
- [García Rodríguez et al., 2015] García Rodríguez, M. J. et al. (2015). Estudio comparativo entre las metodologías ágiles y las metodologías tradicionales para la gestión de proyectos software.
- [Gomez Fuentes et al., 2019] Gomez Fuentes, M. D. C., Cervantes Ojeda, J., and Gonzales Perez, P. P. (2019). Fundamentos de ingeniería de software.
- [Gonzalez-Longatt, 2007] Gonzalez-Longatt, F. M. (2007). Introducción a los sistemas de información: fundamentos. *Universidad Experimental Politécnica de la Fuerza Armada*, pages 1–7.
- [Heredia et al., 2012] Heredia, C., Santaella, G., and Somarriba, L. (2012). Informe psicológico. *México: UNAM*.
- [James, 1890] James, W. (1890). The principles of psychology. *Henry Holt*.
- [Ladrón de Guevara, 2015] Ladrón de Guevara, J. M. (2015). Fundamentos de programación en java. Disponible en http://www.unipamplona.edu.co/unipamplona/portalIG/home_15/recursos/01_general/09062014/n_icontec.pdf.
- [López, 2018] López, D. A. (2018). *Programación orientada a objetos I*. Servicio de Publicaciones de la Universidad Católica de Ávila.
- [Maida and Pacienza, 2015] Maida, E. G. and Pacienza, J. (2015). Metodologías de desarrollo de software.
- [Mazón Olivo et al., 2015] Mazón Olivo, B. E., Cartuche Calva, J. J., Chimarro Chipantiza, V. L., and Rivas Asanza, W. (2015). Fundamentos de programación orientada a objetos en java.
- [Molina Ríos et al., 2015] Molina Ríos, J. R., Honores Tapia, J. A., and Zea Ordóñez, M. P. (2015). *Nociones de Ingeniería de Software*. Machala: Ecuador.

- [Ortiz, 2013] Ortiz, C. J. L. (2013). Estudio de la metodología msf “microsoft solution framework” aplicada al desarrollo de un sistema de facturación “software libre” para la unión de papelerías de la ciudad de ibarra.
- [Palomo and Gil, 2020] Palomo, S. R. G. and Gil, E. M. (2020). *Aproximación a la ingeniería del software*. Editorial Centro de Estudios Ramon Areces SA.
- [Pressman and Contreras, 2010] Pressman, R. S. and Contreras, T. B. (2010). Desarrollo ágil. *Ingeniería del Software: Un Enfoque Práctico* (7a ed., p. 69). University of Connecticut.
- [Riano Nossa, 2021] Riano Nossa, N. D. (2021). Estudio comparativo de metodologías tradicionales y ágiles aplicadas en la gestión de proyectos.
- [Rivera López et al., 2021] Rivera López, M., Santander Acosta, R., and Sixto Fuentes, S. (2021). Arquitectura de información para la gestión de la historia clínica digital en oftalmopediatría. *Revista de Ciencias Médicas de Pinar del Río*, 25(2).
- [Silberschatz et al., 2006] Silberschatz, A., Korth, H. F., Sudarshan, S., Pérez, F. S., Santiago, A. I., and Sánchez, A. V. (2006). *Fundamentos de bases de datos*, volume 11. McGraw-Hill Ciudad de México, México.
- [Sommerville, 2011] Sommerville, I. (2011). Introducción a la ingeniería de software. *I. Sommerville, Ingeniería de Software*. Pearson Educación.
- [Spigariol, 2005] Spigariol, L. (2005). Fundamentos teóricos de los paradigmas de programación.
- [Velásquez et al., 2019] Velásquez, S. M., Montoya, J. D. V., Adasme, M. E. G., Zapata, E. J. R., Pino, A. A., and Marín, S. L. (2019). Una revisión comparativa de la literatura acerca de metodologías tradicionales y modernas de desarrollo de software. *Revista Cintex*, 24(2):13–23.