UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN

Desarrollo de un sistema informático para la gestión y control de la información referente a servicios de optometría.

TESIS

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN INFORMÁTICA

PRESENTA:

CASTAÑEDA CORONEL HIRAM

RAMÍREZ OSORIO YESENIA ESTEPHANIE

ASESOR:

MGTI. LÓPEZ SALAZAR LEONEL GUALBERTO

CUAUTITLÁN IZCALLI, ESTADO DE MÉXICO, 2018

UNIVERSIDAD NACIONAL AVTÓNOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
UNIDAD DE ADMINISTRACIÓN ESCOLAR
DEPARTAMENTO DE EXÁMENES PROFESIONALES

U. N. A. M.
FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN

ASUNTO: VOTO APROBATORIO

M. en C. JORGE ALFREDO CUÉLLAR ORDAZ
DIRECTOR DE LA FES CUAUTITLAN
PRESENTE

ATN: I.A. LAURA MARGARITA CORTAZAR FIGUEROA
Jefa del Departamento de Exámenes Profesionales
de la FES Cuautitlán.

Con base en el Reglamento General de Exámenes, y la Dirección de la Facultad, nos permitimos comunicar a usted que revisamos el: **Trabajo de Tesis**

Desarrollo de un sistema informático para la gestión y control de la información referente a servicios de optometría

Que presenta el pasante: **HIRAM CASTAÑEDA CORONEL**
Con número de cuenta: **30902728-3** para obtener el Título de la carrera: **Licenciatura en Informática**

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el **EXAMEN PROFESIONAL** correspondiente, otorgamos nuestro **VOTO APROBATORIO**.

ATENTAMENTE
"POR MI RAZA HABLARÁ EL ESPÍRITU"
Cuautitlán Izcalli, Méx. a 13 de octubre de 2017.

PROFESORES QUE INTEGRAN EL JURADO

	NOMBRE	FIRMA
PRESIDENTE	MCC. María Araceli Nivón Zaghi	
VOCAL	MGTI. Leonel Gualberto López Salazar	
SECRETARIO	L.I. Mauricio Jaques Soto	
1er. SUPLENTE	L.I. Elizabeth Barrera Romero	
2do. SUPLENTE	I.E. María Guadalupe Vázquez Salazar	

NOTA: los sinodales suplentes están obligados a presentarse el día y hora del Examen Profesional (art. 127).

LMCF/ntm*

VNIVERSIDAD NACIONAL AVFNOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN
UNIDAD DE ADMINISTRACIÓN ESCOLAR
DEPARTAMENTO DE EXÁMENES PROFESIONALES

U. N. A. M.
FACULTAD DE ESTUDIOS
SUPERIORES CUAUTITLAN

ASUNTO: VOTO APROBATORIO

M. en C. JORGE ALFREDO CUÉLLAR ORDAZ
DIRECTOR DE LA FES CUAUTITLAN
PRESENTE

ATN: I.A. LAURA MARGARITA CORTAZAR FIGUEROA
Jefa del Departamento de Exámenes Profesionales
de la FES Cuautitlán.

DEPARTAMENTO DE
EXAMENES PROFESIONALES

Con base en el Reglamento General de Exámenes, y la Dirección de la Facultad, nos permitimos comunicar a usted que revisamos el: **Trabajo de Tesis**

Desarrollo de un sistema informático para la gestión y control de la información referente a servicios de optometría

Que presenta la pasante: **YESENIA ESTEPHANIE RAMÍREZ OSORIO**
Con número de cuenta: **30931597-1** para obtener el Título de la carrera: **Licenciatura en Informática**

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el **EXAMEN PROFESIONAL** correspondiente, otorgamos nuestro **VOTO APROBATORIO**.

ATENTAMENTE
"POR MI RAZA HABLARÁ EL ESPÍRITU"
Cuautitlán Izcalli, Méx. a 14 de octubre de 2017.

PROFESORES QUE INTEGRAN EL JURADO

	NOMBRE	FIRMA
PRESIDENTE	MCC. María Araceli Nivón Zaghi	
VOCAL	MGTI. Leonel Gualberto López Salazar	
SECRETARIO	L.I. Mauricio Jaques Soto	
1er. SUPLENTE	L.I. Elizabeth Barrera Romero	
2do. SUPLENTE	I.E. María Guadalupe Vázquez Salazar	

NOTA: los sinodales suplentes están obligados a presentarse el día y hora del Examen Profesional (art. 127).

LMCF/ntm*

Dedicatorias Hiram:

A mi familia: Quiero agradecer a mi papá Héctor y mamá Paula por ser los pilares de mi formación, por todos los sacrificios que hicieron para darme lo mejor, por su comprensión y su cariño, los amo. A mi hermano por estar siempre a mi lado

A mis amigos: Por hacer de mis días los más divertidos, por todo su apoyo, porque siempre han estado y estarán ahí

A Yesenia: Por ser mi complemento en muchos aspectos de mi vida, por tu entrega, tu disciplina y todo tu cariño, eres la mejor compañera que alguien puede pedir, significas mucho para mí.

A Milton y Eduardo: Por darme la oportunidad de trabajar con ustedes y enseñarme tanto, por su paciencia para que aprendiera y el apoyo que me dieron les estoy muy agradecido.

A mi asesor: Por ser un ejemplo a seguir como profesor y como persona, la dedicación que pone a cada uno de sus alumnos y en la carrera como tal, siempre preocupado por el aprendizaje y desarrollo de nosotros sus alumnos.

A mis profesores: Por la paciencia para soportarme como alumno y por cada minuto que dedicaron para que nos pudiéramos convertir en personas profesionales

Dedicatorias Yesenia:

A mis padres:

Por cada consejo que me brindaron, por cada hora de desvelo que me acompañaron, por apoyarme en los momentos de duda y por no rendirse en los momentos difíciles y por todo su amor. Los amo.

A mis hermanos:

Por apoyarme todo el tiempo y darme café.

A Hiram:

Por trabajar a mi lado y soportarme en los momentos más duros, por animarme a seguir con el proyecto cuando el cansancio llegaba, por ser el mejor compañero, por el apoyo incondicional, por todos los momentos divertidos y por todo tu cariño.

A nuestro asesor Leonel:

Por brindarnos su experiencia y sus consejos, por interesarse en mejorar la enseñanza en la carrera, por interesarse en el aprendizaje de los alumnos y por ser un excelente profesor.

A Milton y a Eduardo:

Por darnos la experiencia, los conocimientos en desarrollo de software y sus consejos profesionales para realizar el proyecto.

Contenido

Introducción

Con respecto al uso de las nuevas tecnologías, Iglesias y Ortiz (2009) mencionan que “Ante un cambio, avance y desarrollo sumamente acelerado de la informática la sociedad se encuentra en un momento importante de transformación, en el cual existen grandes transiciones, estás deberán acoplarse en cada sociedad. Cualquier organización, institución, comercio o empresa cuyo deseo sea el permanecer y sobresalir competitivamente, deberán adaptar la innovación tecnológica de la informática para administrar su información pero siempre cuidando los principios éticos que determinen políticas y procedimientos a seguir con el uso de la información.”, tomando en cuenta lo anterior, y observando la situación en la que se encuentran los pequeños negocios orientados a la salud visual (optometrías), la cual consiste en una forma de administración inexistente o un sistema de administración obsoleto basado en archivos impresos, los cuales es preciso etiquetar y almacenar , perdiendo así el control total de la información, lo que conllevaría a pérdidas financieras y de inventario. Para evitar esto se desarrolló un sistema de información en conjunto con un pequeño negocio llamado “**Satori Vision**” con el objetivo de agilizar los procesos administrativos de inventario y fomentar con esto una mejor toma de decisiones para ser más competitivos dentro del mercado.

Este proyecto se desarrolla con el fin de beneficiar tanto a los clientes como al negocio con giro de optometría aprovechando los conocimientos adquiridos en la carrera y con la experiencia profesional sobre tecnologías de la información.

Para poder conocer la metodología actual de los diferentes procesos y de las necesidades se contó con la ayuda del administrador del negocio, con él se realizó el levantamiento de requerimientos mediante un cuestionario que se presentara posteriormente.

Se realizó una investigación bibliográfica sobre temas relacionados con los sistemas de información como son: bases de datos, paradigmas de programación, lenguajes de programación, tecnologías web, ingeniería de software, etc. Esto con la finalidad de crear un sistema eficiente, escalable y adecuado a las necesidades y posibilidades financieras del cliente.

Para lograr lo anterior se nos brindó información real del negocio para el mejor entendimiento del mismo y así demostrar que el sistema cumple con lo requerido y que funciona en su totalidad. Por último, se realizaron pruebas del sistema con “Satori Vision” llevadas a cabo por el administrador del negocio quien nos indicó una gran cantidad de retroalimentaciones, así como sugerencias para el mejor rendimiento del sistema.

En el capítulo 1 se definirán los conceptos que necesarios para comprender algunos aspectos del proyecto presentado. Se definen los sistemas de información así como sus tipos y características para que un sistema pueda considerarse de utilidad; posteriormente se hablara sobre la ingeniería de software contemplando los paradigmas de desarrollo de software, bases de datos (indicando cuales son las fases que se deben cumplir para tener una base de datos eficiente además del software que las gestiona), la programación orientada a objetos y los principales lenguajes de este tipo de programación, también se definirá cuáles son las tecnologías web que se emplearan en el sistema, la arquitectura que se aplicara y que Diagramas UML serán de utilidad.

En el capítulo 2 se evaluará la situación actual del negocio; cómo se llevan a cabo los procesos tanto administrativos como de interacción con sus clientes para obtener las necesidades del usuario y los objetivos que tendrá que alcanzar el sistema, además de analizar sus limitaciones; se analizan los riesgos que podría generar el desarrollo del sistema, se clasifican y se indica una solución, todo esto basándonos en la investigación del paradigma de desarrollo de software en espiral y específicamente en las primeras dos etapas de este (definición de objetivos y evaluación de riesgos.)

En el capítulo 3 se presenta la parte del desarrollo de software, de acuerdo con los procesos a seguir del modelo en espiral y basándonos en los riesgos principales observados, se decidió trabajar con un modelo de desarrollo de software por prototipos, es decir se fueron diseñando y programando los módulos individuales para mostrarlos al cliente y que este pudiera retroalimentarnos con sus observaciones. Para el desarrollo se analiza cómo será el funcionamiento de cada

módulo involucrado con ayuda de diagramas de casos de uso y de actividades y se crea una base de datos que guarde la información que se consideró necesaria. Para finalizar se indica cómo se implementa el sistema y se agrega un manual de uso.

Alcance de sistema

Brindar el acceso a las pequeñas y medianas empresas a un sistema para la administración de su información.

Proporcionar un sistema que permita un seguimiento a las personas con recursos limitados para que tengan una solución adecuada respecto a su padecimiento visual.

Dar a las Pequeñas y medianas empresas una herramienta que les permita brindar un mejor servicio y con ello tener más oportunidades de crecimiento.

Justificación

En la actualidad la automatización de los procesos en las pequeñas y medianas empresas se ha convertido en un punto muy relevante para el control de la información.

Con el sistema se reduce la perdida de información y se obtiene un mejor análisis de esta, ya que se guarda en una base de datos que permite una manipulación sencilla y eficaz. Se logra un acercamiento entre el usuario y el cliente al generar importancia en la salud visual del cliente, con ayuda de la información almacenada.

El usuario se beneficia teniendo un mejor control de su inventario, con menos pérdidas y un mejor análisis del rendimiento de los productos y servicios que brinda el negocio.

Objetivos

General

Desarrollar un sistema informático para automatizar y controlar la información de manera eficiente derivada de los procesos relacionados con servicios de optometría.

Particulares

- Identificar los procesos más relevantes que intervienen en la prestación de servicio de optometría.
- Mejorar el análisis del presupuesto al momento de realizar la venta.
- Tener un seguimiento de los clientes que puedan facilitar la labor del optometrista y la rapidez de atención.
- Evitar o reducir las pérdidas con base en un registro de bienes y ventas.
- Lograr que el negocio sea más competente dentro del mercado

Hipótesis

Alternativa

Desarrollar e implementar un sistema informático automatiza y controla la información derivada de los procesos relacionada con servicios de optometría.

Nula

Desarrollar e implementar un sistema informático no simplifica, automatiza y mejora los procesos en la gestión y control de la información orientado a los servicios de optometría.

Metodología

Este trabajo se basó en un modelo en espiral[1], el cual, se llevó a cabo de la siguiente manera:

1. Para empezar a realizar el sistema de información se levantaron los requerimientos con el cliente, quién es el encargado de la administración y atención de una optometría y nos indicó las características de ése negocio en particular.
2. Una vez teniendo conocimiento tanto de las áreas de oportunidad como de sus puntos fuertes, se realizó una investigación sobre paradigmas de desarrollo de software encontrando así que el más adecuado para este sistema de información es el modelo en espiral de acuerdo a sus características.
3. Nos volvimos reunir con el cliente donde establecimos la forma de trabajo basándonos en los pasos del modelo en espiral.
4. Determinamos en conjunto con el cliente los objetivos del sistema acorde con sus necesidades y requerimientos.
5. En base a estos requerimientos y necesidades se detectaron algunos riesgos que podrían ocurrir durante el desarrollo del sistema.
6. Teniendo en cuenta las características que debe cumplir el sistema se investigó sobre las herramientas para el desarrollo de aplicaciones web llegando a la conclusión de utilizar Java como lenguaje de programación principal apoyado por los frameworks Java Server Faces y Primefaces; así como el uso de MaríaDB como gestor de base de datos relacional.
7. Con lo anterior se realiza un prototipo del sistema que se muestra al usuario para que nos de su retroalimentación y continuar con las fases del modelo en espiral de nueva cuenta.
8. Ya aceptado el último prototipo se libera como versión final, se implementa y se pone a prueba durante un mes.

[1] En desarrollo de software se representa como una espiral y cada ciclo en la espiral representa una fase en el proceso de software. Así, el ciclo más interno podría referirse a la viabilidad de sistema, el siguiente ciclo a la definición de requerimientos, el siguiente ciclo al diseño del sistema, y así sucesivamente.(Somerville,2005)

Capítulo 1: Marco Teórico

1.1 Sistemas de información

A continuación, se describen las distintas plataformas, software y métodos analizados para el desarrollo del sistema:

1.1.1 Definición de sistema de información

“Un sistema de información realiza las funciones de entrada, procesamiento y salida, e incluye funciones de retroalimentación y control, la salida de un sistema de información es un producto de información de alguna clase. La entrada de un sistema de información son los datos, o hechos, acerca de otros subsistemas de la empresa u otros sistemas del entorno, como las descripciones de las necesidades del cliente, los materiales comprados y las transacciones de ventas. La función del procesamiento organiza y ordena los datos de forma que las personas pueden entender y utilizar. Un sistema de información también tiene una función de almacenamiento para guardar datos y productos de información para un uso futuro. La función de control asegura que las salidas del producto de información son de alta calidad y que son útiles para los usuarios de información para resolver problemas y tomar decisiones.

Un sistema de información implica a personas que utilizan información y a tecnologías de información para ejecutar procesos comerciales, o tareas, que son importantes para la misión y los objetivos de la empresa dentro de un entorno empresarial.” (Beekman. 2005)

“Red de informaciones formalizadas y estructuradas en función de las necesidades y posibilidades de la entidad, apoyándose en un en un sistema evolucionado de tratamiento de datos, para suministrar a los distintos niveles de dirección, a tiempo y fácil de usar, las informaciones necesarias para la gestión y dirección de la organización.” (Rivas G, 1989)

¿Para qué sirven los sistemas de información?

“Casi el 80% de un día típico está dedicado a la información, recibiéndola, comunicándola y utilizándola en una amplia variedad de tareas. En virtud de que la información es la base de virtualmente de todas las actividades realizadas en una compañía, deben desarrollarse sistemas para producirla y administrarla. El objetivo

de tales sistemas es asegurar que información exacta y confiable esté disponible cuando se le necesite y que se le presente en forma fácilmente aprovechable". (Senn, 1990)

1.1.2 Requisitos de los sistemas de información

De acuerdo con Senn (1990) los principales requisitos que los sistemas de información deben tener son:

- Reducción de los datos.

No todos los datos disponibles para una persona serán necesarios para una tarea específica.

Los sistemas de información pueden proporcionar reducción de los datos, disponiendo muchos de los detalles en un formato con el cual puede trabajar el usuario. Sin la reducción de los datos los usuarios tendrán muchas dificultades para aplicar todos los datos al problema que están manejando. Quizá estarían sobrecargados de detalles, pero a la vez carentes de información relevante.

- Manejo del retraso de la información
- El retraso en la información necesario es uno de los mayores problemas en la toma de decisiones. Los administradores dependen de la disponibilidad de los datos y de la capacidad de un sistema de información para recuperar y procesar datos con oportunidad. Por lo tanto, las funciones de los sistemas de información se pueden diseñar para suministrar tal información de acuerdo con las necesidades esperadas.
- Previsión de las necesidades de la gerencia.

Los sistemas de información se deben diseñar con gran flexibilidad. Tendrán que permitir a los usuarios desarrollar y recuperar información, a menudo de variadas formas. Aunque los requisitos del proceso demanden un programaje computacional más poderoso, los sistemas deben ser fáciles de usar por las personas. Deben constituirse en instrumentos para la toma de decisiones y no en obstáculos para la resolución de problemas.

- Manipulación de distintos tipos de información.

La transmisión de muchos tipos de información es un hecho rutinario para la organización. Las fuentes de información tanto internas como externas son importantes, así como la información que proporciona. La tecnología de la computación incluye la capacidad de administrar estos diversos tipos de información.

1.1.3 Tipos de sistemas de información

Según K. Laundon y J. Laundon (1996) los sistemas de información se pueden clasificar de la siguiente forma:

a) Sistema de Procesamiento de Operaciones (SPO): sistemas informáticos encargados de la administración de aquellas operaciones diarias de rutina necesarias en la gestión empresarial (aplicaciones de nóminas, seguimiento de pedidos, auditoría, registro y datos de empleados). Estos sistemas generan información que será utilizada por el resto de sistemas de información de la compañía siendo empleados por el personal de los niveles inferiores de la organización (Nivel Operativo).

b) Sistemas de Trabajo del Conocimiento (STC): aquellos sistemas de información encargados de apoyar a los agentes que manejan información en la creación e integración de nuevos conocimientos para la empresa (estaciones de trabajo para la administración); forman parte del nivel de conocimiento.

c) Sistemas de automatización en la oficina (SAO): sistemas informáticos empleados para incrementar la productividad de los empleados que manejan la información en los niveles inferiores de la organización (procesador de textos, agendas electrónicas, hojas de cálculo, correo electrónico, etc.); se encuentran encuadrados en el nivel de conocimiento al igual que los Sistemas de Trabajo del Conocimiento.

d) Sistemas de información para la administración (SIA): sistemas de información a nivel administrativo empleados en el proceso de planificación, control y toma de decisiones proporcionando informes sobre las actividades ordinarias (control de inventarios, presupuesto anual, análisis de las decisiones de inversión y financiación). Son empleados por la gerencia y directivos de los niveles intermedios de la organización.

e) Sistemas para el soporte de decisiones (SSD): sistemas informáticos interactivos que ayudan a los distintos usuarios en el proceso de toma de decisiones, a la hora de utilizar diferentes datos y modelos para la resolución de problemas no estructurados (análisis de costes, análisis de precios y beneficios, análisis de ventas por zona geográfica). Son empleados por la gerencia intermedia de la organización.
f) Sistemas de Soporte Gerencial (SSG): sistemas de información a nivel estratégico de la organización diseñados para tomar decisiones estratégicas mediante el empleo de gráficos y comunicaciones avanzadas. Son utilizados por la alta dirección de la organización con el fin de elaborar la estrategia general de la empresa.

1.2 Desarrollo de software

Somerville (2005) describe los procesos del software como un conjunto de actividades que conducen a la creación de un producto software. Aunque existen muchos procesos diferentes de software, algunos fundamentales son comunes para todos ellos:

1. Especificación del software. Se debe definir la funcionalidad del software y las restricciones en su operación.
2. Diseño e implementación del software. Se debe producir software que cumpla con su especificación.
3. Validación del software. Se debe validar el software para asegurar que hace lo que el cliente desea.
4. Evolución del software. El software debe evolucionar para cubrir las necesidades cambiantes del cliente.

Así mismo Cortés R. (1998) menciona que todo proyecto debe pasar por al menos tres fases:

- La primera es el qué de la aplicación por construir. En esta fase, el ingeniero recabará los requisitos y planteará, a partir de una investigación analítica, las posibles soluciones al problema planteado. Genéricamente esta fase es conocida como análisis.

- La segunda fase es el cómo de la aplicación por construir. En ella se diseñan los componentes técnicos de la solución por construir: módulos que reflejen los procedimientos de entrada y salida, bases de datos, controles etc.
- La tercera fase es la implementación de la solución, donde incluimos:
    - La codificación (programación) de los elementos diseñados.
    - Las pruebas de la aplicación (control y calidad) y
    - La puesta en producción de la aplicación.

1.2.1 Paradigmas de desarrollo de software

Somerville (2005) clasifica a los principales paradigmas de desarrollo de software como se indica a continuación:

1. Modelo en cascada:

   Considera las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución, y los representa como fases separadas del proceso, tales como la especificación de requerimientos, el diseño del software, la implementación, las pruebas etc.

   Características.

   Las principales etapas de éste modelo se transforman en actividades fundamentales de desarrollo.

   - Análisis y definición de requerimientos.

   Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces, se definen en detalle y sirve como una especificación del sistema.

   - Diseño del sistema y del software

   El proceso de diseño del sistema divide los requerimientos en sistema hardware y software. Establece una arquitectura completa del sistema.

   - Implementación y prueba de unidades.

   Durante esta etapa, el diseño del software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada uno cumpla con su especificación.

   - Integración y prueba del sistema.

Los programas o las unidades individuales de programas se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software después de las pruebas, el sistema software se entrega al cliente.

- Funcionamiento y mantenimiento.

El sistema se instala y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida.

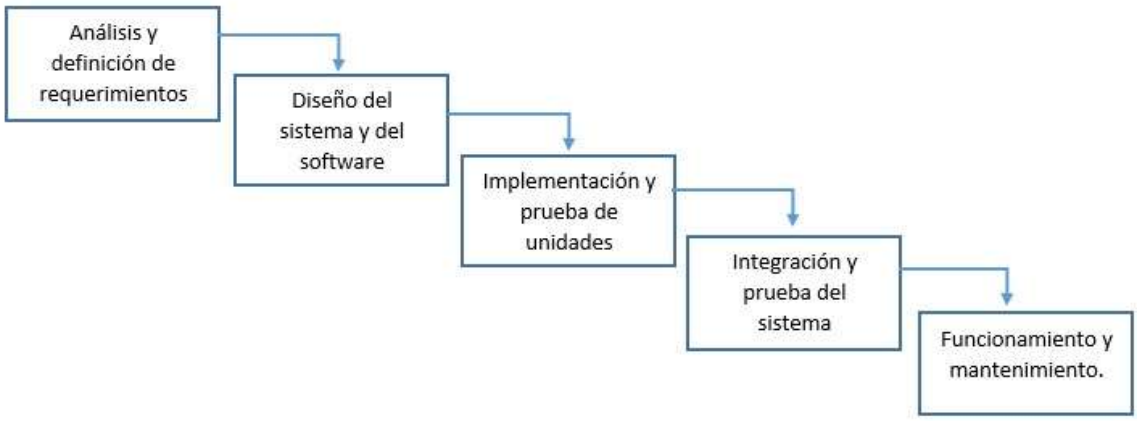


Figura 1.- Modelo en cascada

2. Modelo evolutivo:

   El desarrollo evolutivo se basa en la idea de desarrollar una implementación inicial exponiéndola a los comentarios de usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado. Existen 2 tipos de desarrollo evolutivo:

   - Desarrollo Exploratorio, donde el objetivo del proceso es trabajar con el cliente para explorar sus requerimientos y entregar un sistema final. El desarrollo empieza con las partes del sistema que se comprenden mejor. El sistema evoluciona agregando nuevos atributos propuestos por el cliente.
   - Prototipos desechables, donde el objetivo del proceso de desarrollo evolutivo es comprender los requerimientos del cliente y entonces desarrollar un definición mejorada de los requerimientos para el sistema.

El prototipo se centra en experimentar con los requerimientos del cliente que no se comprenden del todo.

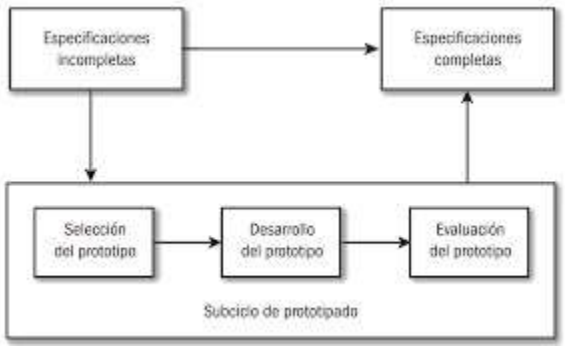


Figura 2.- Modelo evolutivos por prototipos.

3. Modelo en espiral:

   Cada ciclo en la espiral representa una fase en el proceso de software. Así, el ciclo más interno podría referirse a la viabilidad de sistema, el siguiente ciclo a la definición de requerimientos, el siguiente ciclo al diseño del sistema, y así sucesivamente.

   Cada ciclo de la espiral se divide en cuatro sectores:

   - Definición de objetivos. Para esta fase del proyecto se definen los objetivos específicos. Se identifican las restricciones del proceso y el producto, y se traza un plan detallado de la gestión. Se identifican los riesgos del proyecto. Dependiendo de estos riesgos, se plantean estrategias alternativas.
   - Evaluación y reducción de riesgos. Se lleva cabo un análisis detallado para cada uno de proyectos identificados. Se definen los pasos para reducir dichos riesgos, por ejemplo, si existe el riesgo de tener requerimientos inapropiados, se puede desarrollar un prototipo del sistema.
   - Desarrollo y validación. Después de la evaluación de los riesgos, se elige un modelo para el desarrollo del sistema. Por ejemplo, si los riesgos en la interfaz del usuario son dominantes, un modelo de desarrollo apropiado podría ser la construcción de prototipos

evolutivos. Si los riesgos de seguridad son la principal consideración, un desarrollo basado en transformaciones formales podría ser el más apropiado, y así sucesivamente.

- Planificación. El proyecto se revisa y se toma la decisión de si se debe continuar con un ciclo posterior de la espiral. Si se decide continuar, se desarrollan los planes para la siguiente fase del proyecto.

La diferencia principal entre en modelo en espiral y los otros modelos del proceso de software es la consideración explicita del riesgo en el modelo en espiral. Informalmente, el riesgo significa sencillamente algo que puede ir mal. Por ejemplo, si la intención es utilizar un nuevo lenguaje de programación, el riesgo es que los compiladores disponibles sean poco fiables o que no produzcan código objeto suficientemente eficiente. Los riesgos originan problemas en el proyecto, como los de confección de agendas y excesos en los costos; por lo tanto, la disminución de riesgos es una actividad muy importante en la gestión de un proyecto.

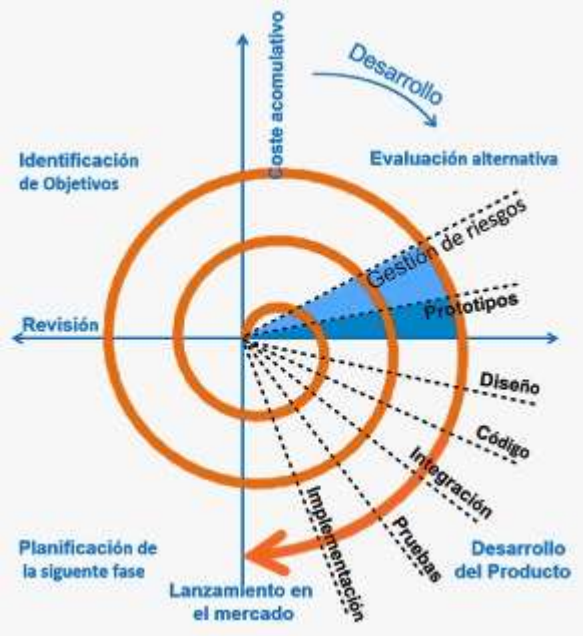


Figura 3.- Modelo en espiral.

4. Metodología ágil:

   Método que permite incorporar cambios con rapidez en el desarrollo del software. En muchas ocasiones, los modelos de gestión tradicionales no sirven para afrontar un reto que hoy en día resulta fundamental: incorporar cambios con rapidez y en cualquier fase del proyecto.

   Un proceso es ágil cuando el desarrollo del software es:

   - Incremental. Entregas pequeñas de software, con ciclos rápidos.
   - Cooperativo. Cliente y desarrolladores trabajan juntos constantemente con una cercana comunicación.
   - Sencillo. El método en sí mismo es simple, fácil de aprender y modificar. Está bien documentado y es adaptable. Permite realizar cambios de último momento.

   Sus elementos claves son:

   - Poca documentación
   - Simplicidad
   - Análisis como una actividad constante
   - Diseño evolutivo
   - Integraciones
   - Testeos diarios

   Ventajas de las metodologías ágiles

   Las metodologías ágiles presentan diversas ventajas como:

   - Rápida respuesta a cambios de requisitos a lo largo del desarrollo.
   - Entrega continua y en plazos cortos de software funcional.
   - Trabajo conjunto entre el cliente y el equipo de desarrollo.
   - Minimiza los costos frente a cambios.
   - Importancia de la simplicidad, al eliminar el trabajo innecesario.
   - Atención continua a la excelencia técnica y al buen diseño.
   - Mejora continua de los procesos y el equipo de desarrollo.
   - Evita malentendidos de requerimientos entre el cliente y el equipo.

- El equipo de desarrollo no malgasta el tiempo y dinero del cliente desarrollando soluciones innecesariamente generales y complejas que en realidad no son un requisito del cliente.
- Cada componente del producto final ha sido probado y satisface los requerimientos.

Desventajas de los métodos ágiles

Como en cualquiera otra metodología, también hay desventajas y problemas que surgen a la hora de implementarlas:

- Falta de documentación del diseño. El código no puede tomarse como una documentación. En sistemas de tamaño grande se necesitar leer los cientos o miles de páginas del listado de código fuente.
- Problemas derivados de la comunicación oral. Este tipo de comunicación resulta difícil de preservar cuando pasa el tiempo y está sujeta a muchas ambigüedades.
- Falta de calidad. Probar el código de forma constante no genera productos de calidad, sólo revela falta de análisis y diseño.
- Fuerte dependencia de las personas. Como se evita en lo posible la documentación y los diseños convencionales, los proyectos ágiles dependen críticamente de las personas.
- Falta de procesos de revisión del código.
- Falta de reusabilidad. La falta de documentación hace difícil que pueda reutilizarse el código ágil.
- Sobre costos y retrasos derivados de la refactorización continua. Para un sistema de ciertas proporciones, los costos y retrasos derivados de la refactorización no pueden despreciarse.
- Restricciones en cuanto a tamaño de los proyectos abordables.
- Rigidez. Algunos métodos ágiles son muy rígidos: deben cumplirse muchas reglas de una forma estricta para garantizar el éxito del proyecto.
- Problemas derivados del fracaso de los proyectos ágiles. Si un proyecto ágil fracasa no hay documentación o hay muy poca; lo mismo

ocurre con el diseño. La comprensión del sistema se queda en las mentes de los desarrolladores.

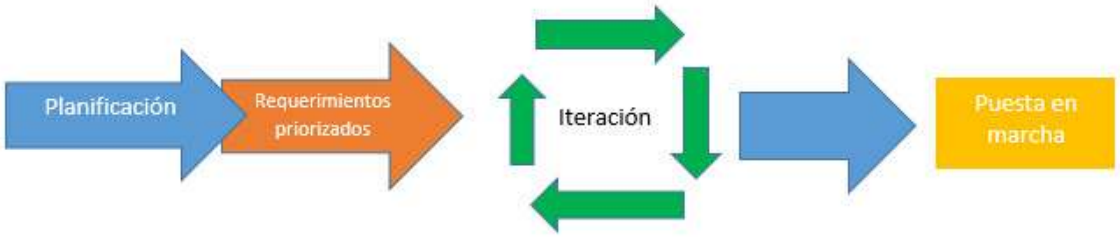


Figura 4.- Metodología ágil

Con respecto a los paradigmas conocidos, principalmente se encuentran tres, de acuerdo con Cortés R. (1998), los cuales son:

1. Modelo de la cascada: este modelo indica que el producto de cada una de las fases de entrada de la siguiente fase. Tiene una evolución lineal y sumativa.

   Este modelo es el más tradicional. Sin embargo, hoy en día tiende a ser obsoleto debido a las siguientes razones:

   - Casi ningún proyecto en la vida real sigue el patrón lineal que tiene este paradigma.
   - Si un error no es detectado en el inicio será muy costoso, pues solo hasta el final del proyecto (en la implementación) será evidente dicho error.
   - Este modelo se basa en las tecnologías de antes las cuales forzaban a una especialización en el campo del análisis, del diseño, etc. Hoy en día, las herramientas tienden a combinar las capacidades del profesional en informática en muchos campos, deshaciendo consecuentemente esa división.

2. El modelo por prototipos: Los prototipos nacieron como un método para acelerar la definición de los requisitos del software por construir. Un prototipo en un programa. La idea principal es hacer un modelo de la aplicación y presentársela al cliente, sobre todo a nivel de interfaces y otras salidas, (consultas, reportes). El cliente hará sus observaciones

acerca de lo que ve en ese modelo, y el programador modificará ése modelo de acuerdo con dichas observaciones.

3. El modelo de la espiral: Este modelo busca pasar controladamente, y de manera cíclica, por 4 actividades en el desarrollo de la aplicación. Estas actividades son:
    a. Planificación: Aquí se determinan objetivos, alternativas de selección, y restricciones que se den para el proyecto. Encontramos el qué de la aplicación por construir, así como aspectos de la administración de proyectos y de administración de los recursos necesarios por utilizar.
    b. Análisis de riesgos: Identificación y resolución de los riesgos que pueda tener el proyecto.
    c. Ingeniería: Es la construcción del software. Aquí se combina el cómo (aspectos del diseño) y parte de la implementación (programación, pruebas, puesta en producción).
    d. Evaluación del cliente: Aquí se cumplen otra parte de la implementación (pruebas del cliente). Es importante esta actividad debido al control de calidad y a la participación activa de los clientes en la construcción del producto.

1.3 Bases de datos

“Una base de datos es un conjunto de elementos de datos que se describe a si mismo con relaciones entre estos elementos, que presenta una interfaz uniforme de servicio” (Johnson, 2000)

Una base de datos es una herramienta para recopilar y organizar información. Las bases de datos pueden almacenar información sobre personas, productos, pedidos u otras cosas. Muchas bases de datos comienzan como una lista en una hoja de cálculo o en un programa de procesamiento de texto. A medida que la lista aumenta su tamaño, empiezan a aparecer redundancias e inconsistencias en los datos. Cada vez es más difícil comprender los datos en forma de lista y los métodos de búsqueda o extracción de subconjuntos de datos para revisión son limitados.

1.3.1 Modelos de bases de datos

El autor Carlos Coronel (2011) nos da una breve descripción de los modelos de bases de datos.

1.3.1.1 Modelo jerárquico:

El modelo jerárquico se desarrolló en la década de 1960 para manear grandes cantidades de datos para complejos proyectos de manufactura. Su estructura lógica básica está representada por un árbol invertido. La estructura jerárquica contiene o niveles, o segmentos. Un segmento es equivalente de un tipo de registro de un sistema de archivos. Dentro de una jerarquía se percibe una capa más alta como el padre del segmento directamente bajo ella, que se denomina hijo. El modelo jerárquico describe un conjunto de relaciones uno a muchos (1:M) entre un padre y sus segmentos hijos (cada padre puede tener muchos hijos, pero cada hijo tiene solo un padre).

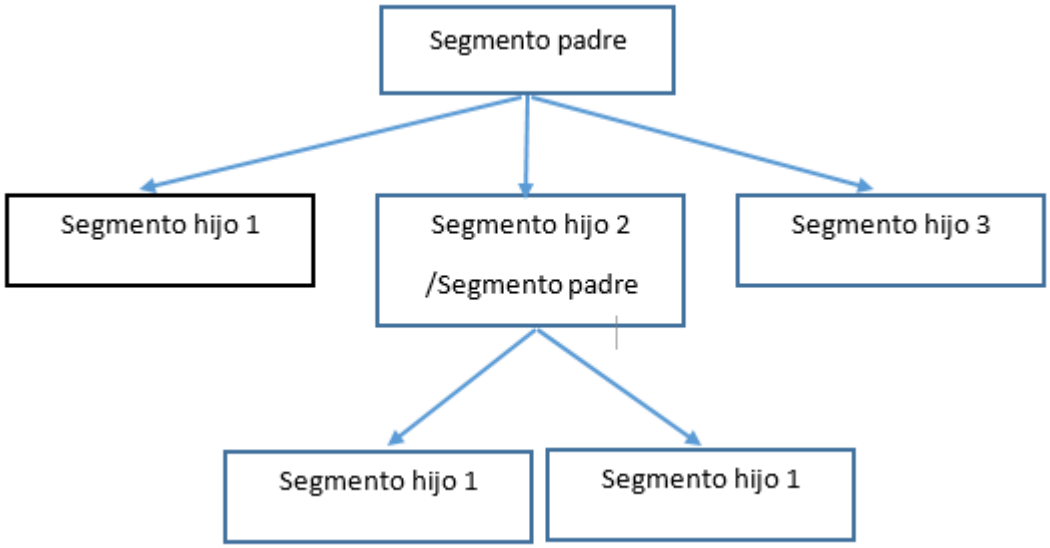


Figura 5.- Modelo jerárquico

1.3.1.2 Modelo de red:

El modelo de red fue creado para representar complejas relaciones de datos en forma más efectiva que el modelo jerárquico, para mejorar la operación de una base de datos y para imponer un estándar de base de datos. En el modelo de red, el usuario percibe la base de datos de red como un conjunto de registros de relaciones

1:M, pero, a diferencia del modelo jerárquico, el modelo de red permite que un registro tenga más de un padre. En tanto que el modelo de base de datos de red generalmente no se usa en la actualidad, las definiciones de conceptos estándar de bases de datos que emergieron con el modelo de red todavía se usan en modelos modernos de datos. Algunos conceptos importantes que se definieron son:

- El esquema, que es la organización conceptual de toda la base de datos según la ve el administrador.
- El subesquema, que define la parte de la base de datos “vista” por los programas de aplicación que en realidad producen la información deseada a partir de los contenidos dentro de la base de datos.
- Un lenguaje de administración de datos (DML), que define el ambiente en el que los datos se pueden administrar y trabajar con los datos de las bases de datos.
- Un esquema de lenguaje de definición de datos (DDL), que hace posible que el administrador de una base de datos defina los componentes del esquema.

A medida que crecieron las necesidades de información y se requirió de refinadas bases de datos el modelo de red se volvió engorroso. La falta de capacidad de consultas ad hoc puso una fuerte presión sobre los programadores para generar código requerido para producir incluso los informes más sencillos. Y aun cuando las bases de datos existentes brindaban una limitada independencia de datos, cualquier cambio estructural en la base de datos podía hacer estragos en todos los programas de aplicación que sacaban datos de la base de datos. Debido a las desventajas, los modelos jerárquicos y de red, fueron sustituidos en gran parte por el modelo de datos relacional en la década de 1980.

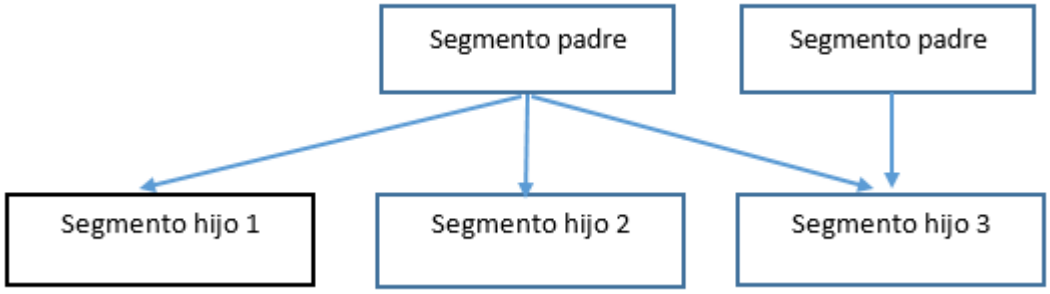


Figura 6. Modelo de red

1.3.1.3 Modelo Orientado a Objetos

Los cada vez más complejos problemas reales demostraron la necesidad de un modelo de base de datos que se apegara más al mundo real. En el modelo de datos orientado a objetos (OODM), tanto los datos como sus relaciones están contenidos en una sola estructura conocida como objeto. A su vez el OODM es el fundamento para el sistema de administración de una base de datos orientada a objetos (OODBMS).

Un OODM refleja una forma muy diferente de definir y usar entidades. Al igual que la entidad de un modelo relacional, un objeto esta descrito por su contenido fáctico. Pero a diferencia, de una entidad, un objeto incluye información acerca de las relaciones entre los datos dentro del objeto, así como información acerca de sus relaciones con otros objetos. Por tanto, a los datos dentro de los datos se les da un mayor significado. Se dice que el OODM es un modelo de datos semántico porque semántico indica significado.

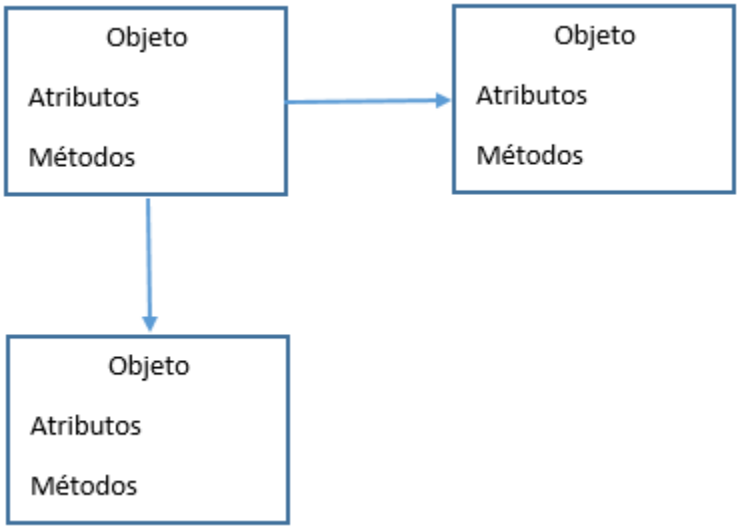


Figura 7.- Modelo orientado a objetos.

1.3.1.4 Bases de datos relacionales

"Es una colección compartida de datos lógicamente relacionados, junto con una descripción de estos datos, que están diseñados para satisfacer las necesidades de información de una organización." (Conolly, 2005)

Pérez (2007) indica que, quizás, el problema fundamental que suele plantearse al realizar una base de datos real, formada por varias tablas, es la repetición de datos, es decir, campos repetidos en diferentes tablas (redundancia), lo cual va a dificultar su gestión, la actualización, inserción, eliminación, consulta, etc.

Para resolver estos problemas es necesario que exista integración entre las distintas tablas y que esté controlada la repetición de datos. Un Sistema de Gestión de Bases de Datos Relacionales, en el caso de los microordenadores están concebidos como un conjunto de programas de propósito general que permiten controlar el acceso y la utilización de las bases de forma que satisfagan las necesidades del usuario y que actúen con independencia de los datos y con ello, las bases de datos relacionales puedan resolver, mejor que otras organizaciones, las dificultades de redundancia y la generación de los datos.

La teoría relacional se basa en el concepto matemático de relación de E.F Codd, quien desarrollo una sólida fundamentación teórica.

Además, concluye que, al utilizar las bases de datos relacionales, se tienen las siguientes ventajas:

- Actúan sobre las tablas en su conjunto, en lugar de hacerlo sobre los registros como ocurre en otros sistemas.
- Se pueden realizar consultas complejas que utilizan varias tablas de forma simple.
- Son fáciles utilizar (la organización física de los datos es independiente de su tratamiento lógico)
- La organización relacional se caracteriza por que las tablas de la base de datos tienen estructura de matriz o tabla bidimensional, donde las filas son los registros y las columnas los campos.

También Conolly (2005) indica que existen factores críticos en el diseño de una base de datos.

Las siguientes directrices suelen resultar críticas a la hora de diseñar de la forma adecuada una base de datos.

- Es necesario trabajar interactivamente con los usuarios lo más posible.
- Hay que seguir una metodología estructurada durante todo el proceso del modelado de los datos.
- Hay que emplear una técnica centrada en los datos.
- Es preciso incorporar las consideraciones estructurales y de integridad dentro de los modelos de datos.
- Hay que combinar técnicas de conceptualización, normalización y validación de transacciones en la metodología de modelado de los datos.
- Es necesario emplear diagramas para representar una parte lo mayor posible de los modelos de los datos.
- Es necesario usar un lenguaje de diseño de bases de datos (Data Base Design Language, DBDL) para representar información semántica acerca de los datos que no puedan representarse fácilmente en un diagrama.

- Es preciso construir un diccionario de datos para complementar los diagramas del modelo de datos y el DBDL.
- Hay que estar dispuestos a repetir diversos pasos en determinadas ocasiones.

En la investigación de Pérez (2007), define que para que la estructura de las tablas cumpla las leyes de la teoría relacional deben satisfacerse las siguientes condiciones:

Todos los registros dela tabla deben tener el mismo número de campos, aunque alguno de ellos este vacío, deben ser registros de longitud fija.

Cada campo tiene un nombre o etiqueta que hay que definir previamente a su utilización.

- La base de datos estará formada por muchas tablas, una por cada tipo de registro.
- Dentro de una tabla cada nombre de campo debe ser distinto.
- Los registros de una misma tabla tienen que diferenciarse, al menos, en el contenido de alguno de sus campos, no debe haber dos campos "idénticos".
- Los registros de una tabla pueden estar dispuestos en cualquier orden.
- El contenido de cada campo está delimitado por un rango de valores posibles.
- Permite la creación de nuevas tablas a partir de las ya existentes, relacionando campos de distintas tablas anteriores. Esta condición es la esencia de las bases de datos relacionales, formando lo que se llama un fichero "virtual" (temporalmente en memoria).

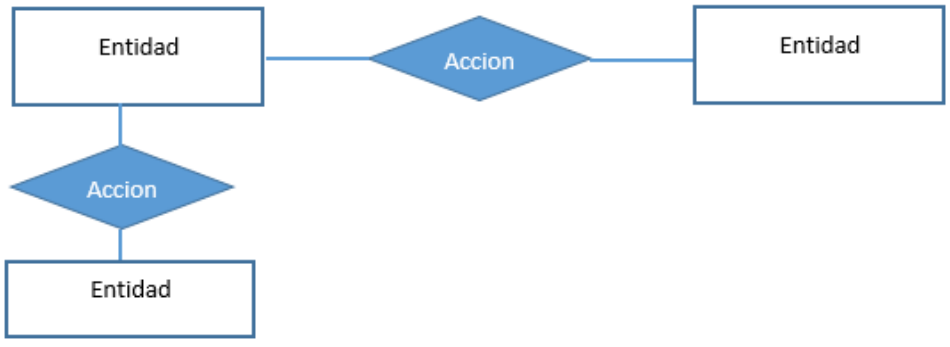


Figura 8. Modelo relacional.

1.3.2 SQL

“SQL es el lenguaje utilizado para la administración y el análisis de bases de datos y para el desarrollo de aplicaciones. Se trata de un súper conjunto de del lenguaje estándar definido por la American National Standards Institute (ANSI) y la International Standars Organization (ISO). “Pérez (2007)

Houlete (2003) describe que SQL está formado por tres lenguajes, cada uno con un propósito. Por lo general esta estructura refleja una realidad. Por lo general, hablamos de las bases de datos de tres maneras distintas. En las siguientes tres secciones siguientes se describen las tres formas de hablar acerca de una base de datos.

- DDL (Data Definition Language o Lenguaje de Definición de Datos) Es necesario describir los datos que entran en las bases de datos. ¿Cómo definimos los datos? En primer lugar, tendremos que crear las tablas que almacenarán la información. El lenguaje de definición de datos incluye las declaraciones que usamos para crear, modificar, y destruir objetos de las bases de datos. Decimos objetos porque la definición de los datos va más allá de las tablas. Incluye las vistas y los procedimientos almacenados, además de los usuarios que interactúan con ellos. Todos ellos caen sobre la rúbrica DDL.
- DML (Data Modification Languaje o Lenguaje de modificación de Datos)

  Está integrado por las declaraciones con las que se selecciona, inserta, actualiza o elimina datos. Además de las declaraciones y los predicados para formato y presentación de los resultados de las consultas pertenecen al reino del lenguaje de manipulación de datos.

  Sus ventajas derivan de su capacidad de permitirle declarar la estructura de los datos como lo desea. Cuando se emite una declaración DML, no se conoce la forma en la que los datos están ordenados y almacenados en el momento, ni le interesa conocerla.

- DCL (Data Control Language o Lenguaje de control de datos)

  Describirá quién tiene acceso a los datos, usualmente los programadores no tienen acceso a este lenguaje, pero el administrador sí ya que sino no habría seguridad en la base de datos.

  Dos declaraciones SQL integran el corazón de DCL: GRANT y REVOKE. Con GRANT se otorgan acceso a los datos. Con REVOKE se limita el acceso.

1.3.3 Sistemas de gestión de bases de datos relacionales

Un sistema de gestión de bases de datos relacionales (RDBMS) es un programa que te permite crear, actualizar y administrar una base de datos relacional.

"Los sistemas de gestión de base de datos (abreviado mediante SGBD o DBMS) organizan y estructuran los datos de tal modo que puedan ser recuperados y manipulados por usuarios y programas de aplicación. Las estructuras de los datos y las técnicas de acceso proporcionados por un DBMS particular se denominan modelo de dato. El modelo de datos determina la personalidad de un DBMS y las aplicaciones para las cuales están particularmente bien conformado" (Pérez, 2007)

La mayoría de los RDBMS comerciales utilizan el lenguaje de consultas estructuradas (SQL) para acceder a la base de datos, aunque SQL fue inventado después del desarrollo del modelo relacional y no es necesario para su uso.

1.3.3.1 Gestores de bases de datos

1.3.3.1.1 MySQL - MariaDB

MySQL es un sistema gestor de bases de datos relacionales, que además ofrece compatibilidad con PHP, Perl, C y HTML, y funciones avanzadas de administración y optimización de bases de datos para facilitar las tareas habituales. Implementa funciones web, permitiendo un acceso seguro y sencillo a los datos a través de internet. Este sistema gestor de bases de datos incluye capacidades de análisis integradas, servicios de transformación y duplicación de datos y funciones de programación mejoradas.

MySQL es un sistema gestor de bases de datos relacional cliente- servidor de coste mínimo que incluye un servidor SQL, programas cliente para acceder al servidor, herramientas administrativas y una interfaz de programación para escribir

programas. MySQL es portable y se ejecuta en sistemas operativos comerciales como Windows y Linux.

MariaDB Server es uno de los servidores de bases más populares en el mundo. Hecho por los desarrolladores originales de MySQL y garantizada para mantenerse de código abierto. (Pérez, 2008)

MariaDB convierte los datos en información estructurada en una amplia gama de aplicaciones, que van desde la banca hasta sitios web. Se trata de una mejora, en el reemplazo para MySQL.

Ventajas:

- Cuenta con diversos plugins.
- No requiere una licencia para su uso.
- Más motores de almacenamiento, con la finalidad de optimizar el almacenamiento de datos.
- La velocidad en diversos aspectos, como las inserciones, la velocidad en consultas muy grandes y en el número de conexiones en paralelo a la base de datos.
- Incluye nuevas extensiones y características que MySQL no tenía.
- Fuente verdaderamente abierta, no cuenta con módulos de código cerrado como lo tenía MySQL.
- Compatible con Windows XP, Vista, 7.8 y 10, Solaris, Linux, Debian, Red Hat, Fedora.

Desventajas:

- Si se cuenta con una amplia base de datos realizada en MySQL resultara complicada la migración.

1.3.3.1.2 Database Oracle

Database Oracle es una colección de datos tratados como una unidad. El propósito de una base de datos es almacenar y recuperar información relacionada. Un servidor de base de datos es la clave para resolver los problemas de gestión de la información. En general, un servidor gestiona de forma fiable una gran cantidad de datos en un entorno multiusuario por lo que muchos usuarios pueden acceder simultáneamente a los mismos datos. Todo esto se realiza al tiempo que ofrece un

alto rendimiento. Un servidor de base de datos también impide el acceso no autorizado y ofrece soluciones eficientes para la recuperación de errores.

Base de Datos Oracle es la primera base de datos diseñada para grid computing empresarial, la manera más flexible y económica para gestionar la información y las aplicaciones Enterprise Grid Computing crea grandes grupos de estándares de la industria, almacenamiento modular y servidores. Con esta arquitectura, cada nuevo sistema se puede aprovisionar rápidamente de la piscina de los componentes. No hay ninguna necesidad de que los picos de trabajo, ya que la capacidad puede ser fácilmente añadidos o reasignarse los grupos de recursos según sea necesario.

La base de datos tiene estructuras lógicas y estructuras físicas. Debido a que las estructuras físicas y lógicas están separados, el almacenamiento físico de los datos se puede manejar sin afectar el acceso a las estructuras de almacenamiento lógico.

Roldán, (2011) menciona que la arquitectura del servidor de base de datos Oracle se divide en tres componentes:

- Los procesos relacionados con el usuario, que dependerán, al igual que los recursos que utilizan, de si el servidor es dedicado o compartido.
- La instancia de Oracle, formada por los procesos de la instancia y las estructuras de memoria en que se apoyan.
- La base de datos propiamente dicha, y que se compone de un conjunto de ficheros físicos en los que se guardan los datos e información adicional necesaria para el correcto funcionamiento de la base de datos.

Ventajas:

- Incluye mecanismos para que el control de la concurrencia de datos en un sistema multiusuario sea maximizado.
- Para gestionar la consistencia de datos multiversión, Oracle debe crear un conjunto coherente de lectura de datos cuando se consulta una tabla (lectura) y al mismo tiempo actualizado (escritura).
- Manejo de transacciones

- Cuenta con mecanismos de bloqueo para garantizar la seguridad de la información.

1.3.3.1.3 SQL Server

Microsoft SQL Server es un sistema relacional de gestión de base de datos o RDBMS, que soporta una amplia variedad de procesamiento de transacciones, la inteligencia empresarial y aplicaciones de análisis en entornos de TI corporativos. Es una de las tres tecnologías de bases de datos líderes en el mercado, junto con la base de datos Oracle y DB2 de IBM.

Características:

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente – servidor, dónde la información y los datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información.

Desventajas:

- Usa Address Windowing Extension (AWE) para hacer el direccionamiento de 64 – bit. Esto impide usar la administración dinámica de memoria y solo le permite alojar máximo de 64 GB de memoria compartida.
- No maneja compresión de datos (en SQL Server 2005 y 2000, solamente la versión del 2008 tiene esta característica) por lo que ocupa mucho espacio en disco.
- Está atado a la plataforma del sistema operativo sobre la cual se instala.

1.3.3.1.4 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

La última serie de producción es la 9.3. Sus características técnicas la hacen una de las bases de datos más potentes y robustos del mercado. Su desarrollo comenzó hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.

Ventajas:

- Programación / Desarrollo
- Funciones/procedimientos almacenados (stored procedures) en numerosos lenguajes de programación, entre otros PL/pgSQL (similar al PL/SQL de Oracle), PL/Perl, PL/Python y PL/Tcl
- Bloques anónimos de código de procedimientos (sentencias DO)
- Numerosos tipos de datos y posibilidad de definir nuevos tipos. Además de los tipos estándares en cualquier base de datos, tenemos disponibles, entre otros, tipos geométricos, de direcciones de red, de cadenas binarias, UUID, XML, matrices, etc.
- Soporta el almacenamiento de objetos binarios grandes (gráficos, videos, sonido, ...)
- APIs para programar en C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, PHP, Lisp, Scheme, Qt y muchos otros.

1.3.3.1.5 SyBase

SyBase fue fundada en 1984 y se especializa en infraestructura empresarial e integración de plataformas, bases de datos y aplicaciones. Se expandió en 1995, cuando adquirió Powersoft.

La familia de productos de SyBase incluye bases de datos, herramientas de desarrollo, equipo intermedio de integración, portales empresariales y servicios inalámbricos móviles. Oferta soluciones de principio a fin en bases de datos.
La vida de SyBase (servidor empresarial adaptable), comenzó como Sybase SQL Server, el primer sistema en manejo de bases de datos relacionales (relational database management system RDBMS), creado y vendido por Sybase.
Ventajas:

1. Incrementar la productividad.
2. Dar disponibilidad e integridad en los datos cuando los sistemas de red fallan.
3. Distribuir datos heterogéneos a través de múltiples locaciones.
4. Simplificar la migración de datos de sistemas históricos a nuevas plataformas incluyendo el Web.
5. Proteger los datos permitiendo que los sistemas históricos y nuevos corran conjuntamente hasta que las organizaciones estén listas para cambiar totalmente hacia la nueva plataforma.
6. Sincronizar los datos a través de Sybase y/o bases de datos heterogéneas.
7. Poblar los almacenes de bases de datos.
8. Mejorar el desarrollo de sus sistemas OLTP.
9. Proveer más salidas a usuarios finales y más locaciones de acceso en tiempo real con el propósito de analizar y soportar la toma de decisiones.
10. Liberar los recursos de la red.
11. Expedir de desplegados para nuevas aplicaciones.

Desventajas de la base de datos de SyBase:

1. No se encuentran desventajas propiamente establecidas. La única desventaja podría ser el costo del programa a implementar que se compensa con el retorno de inversión.

1.3.4 Fases para el diseño de bases de datos

Para Miguel Castaño, Mario Piattini y Esperanza Marcos (2000) las fases que debe cumplir el diseño de una base de datos contemplan las siguientes etapas:

1. Estudio previo y plan de trabajo

a) Decisión política y fijación de objetivos (estudio de viabilidad)

Esta fase, a veces llamada de análisis previo, estudio de oportunidad o viabilidad, debe preceder a cualquier operación de concepción y diseño de una base de datos; en ella se ha de definir unos objetivos claros y concretos que sirvan de pauta para todo el desarrollo.

b) Evaluación previa de medios y costes:

Una vez definidos los objetivos generales del proyecto, es preciso realizar una evaluación aproximada de los medios y de los costes que requerirá la puesta en marcha del sistema.

Se trata solo de dar un orden de magnitud del coste global del proyecto, ya que será prácticamente imposible, sin un estudio más profundo del sistema que se va a desarrollar, hacer un análisis detallado de los costes.

c) Aprobación de una estructura orgánica.

Antes de comenzar el desarrollo del sistema será preciso definir la organización de la unidad administrativa que tendrá la responsabilidad de la gestión y control de la base de datos, así como determinar la estructura y los componentes del equipo encargado del desarrollo.

Las funciones del administrador de la base de datos, su responsabilidad respecto al contenido de la base, la actualización de la misma, la estandarización de la información etc., son aspectos fundamentales que han de ser considerados desde u principio y que pueden ser decisivos para conseguir que el proyecto llegue a buen fin.

d) Plan de trabajo detallado.

Sera preciso crear un plan de trabajo detallado en el que se especifiquen las distintas fases, con los plazos y medios que se requerirán para ello.

2. Concepción de la base de datos y selección del equipo

En esta fase se realiza un análisis del a información que se ha de integrar en la base de datos a fin de alcanzar los objetivos propuestos y se representa en un modelo conceptual de datos independiente del SGBD que se vaya a utilizar.

a) Concepción de la base de datos:

En esta fase se determinarán las necesidades de los usuarios, concretándose en las funciones que hay que integrar en la base de datos. También habrá que describir las actividades de la organización analizándolas en términos de sistema, de subsistema y de entorno. Todo ello permitirá determinar, por un lado, las características del sistema (requisitos en cuanto a protección de los datos, flexibilidad, etc.) y de su arquitectura y, por otro lado, el contenido de la base de datos, con especificación en su volumen, volatilidad, normas de validación y una lista de reglas de gestión.

b) Especificaciones de las necesidades de equipo físico y lógico:

Sera preciso evaluar las exigencias en cuanto a equipo (memorias principales y secundarias, capacidad de proceso, etc.). En cuanto a la selección del SGBD, se deberá proceder a realizar un estudio de los SGBD existentes en el mercado, de sus características y de las posibilidades que ofrece, para poder elegir aquel que mejor se adapte a los requisitos específicos del sistema de información.

3. Diseño y carga

Esta fase comprende tanto el diseño lógico de la base de datos y su codificación, como la carga de datos y la prueba de programas.

a) Diseño Lógico y Físico

Modelado conceptual: su objetivo es tener una buena representación de los recursos de la información de la empresa, con independencia de usuarios o aplicaciones en particular, y fuera de consideraciones sobre eficiencia del computador.

Diseño Lógico: su objetivo es transformar el esquema conceptual obtenido en la etapa anterior, adaptándolo al modelo de datos en el que se apoya el SGBD que se va a utilizar.

Diseño Físico: Su objetivo en conseguir una implementación, lo más eficiente posible, del esquema lógico.

Diseño Lógico o diagrama entidad relación

En esta etapa se transforma el esquema de la base de datos (diseño conceptual), en una serie de estructuras lógicas (tablas, campos, claves primarias y ajenas, etc.),también conocido como diagrama entidad relación, que permitirán almacenar los datos de una forma óptima, sin redundancia de y garantizando la integridad referencial: que no se pueda relacionar un dato A con otro dato B, si este último no existe todavía en la base de datos.

El objetivo es definir correctamente los campos y claves de las tablas, y las relaciones entre ellas, para que el sistema gestor de base de datos pueda avisar con un mensaje de error si el usuario está intentando realizar una operación incorrecta sobre la base de datos, y que no corresponde con el diseño del esquema inicial.

Un diagrama entidad relación es una herramienta de diseño que permite representar las entidades importantes de un sistema de información, así como la relaciones entre ellas y las propiedades que las describen. El modelo de datos entidad-relación es una representación del mundo real que consta de una colección de objetos básicos, llamados entidades, y de relaciones entre esos objetos.

Diseño Físico

Transformamos el esquema de la base de datos (diseño conceptual), en una serie de estructuras lógicas (tablas, campos, claves primarias y ajenas, etc.), que permitirán almacenar los datos de una forma óptima, sin redundancia de datos (que no haya duplicidad de información; que no se repita el mismo dato) y garantizando la integridad referencial: que no se pueda relacionar un dato A con otro dato B, si este último no existe todavía en la base de datos.

El objetivo es definir correctamente los campos y claves de las tablas, y las relaciones entre ellas, para que el sistema gestor de base de datos pueda avisar con un mensaje de error si el usuario está intentando realizar una operación incorrecta sobre la base de datos, y que no corresponde con el diseño del esquema inicial.

El diseño físico de la base de datos optimiza el rendimiento a la vez que asegura la integridad de los datos al evitar repeticiones innecesarias de datos. Durante el diseño físico, se transforman las entidades en tablas, las instancias en filas y los atributos en columnas.

Una vez completado el diseño lógico de la base de datos, se pasa al diseño físico. El personal que realiza el diseño debe tomar decisiones que afectan al diseño físico, algunas de las cuales se listan a continuación.

- Cómo convertir entidades en tablas físicas
- Qué atributos utilizar para las columnas de las tablas físicas
- Qué columnas de las tablas deben definirse como claves
- Qué índices deben definirse en las tablas
- Qué vistas deben definirse en las tablas
- Cómo resolver relaciones de varios con varios
- Qué diseños pueden beneficiarse del acceso hash

b) Carga y optimización de la base

Definida la estructura física de la base de datos, es preciso cargar los datos en la misma.

1.4 Paradigma de programación orientada a objetos (POO)

El paradigma de orientación a objetos es una metodología de desarrollo de aplicaciones en la cual éstas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representa una instancia de alguna clase, y cuyas clases son todos miembros de una jerarquía de clases unidas mediante relaciones de herencia. Como su mismo nombre indica, la programación orientada a objetos se basa en la idea de un objeto, que es una combinación de variables locales y

procedimientos llamados métodos que juntos conforman una entidad de programación. (Pérez, 2014)

El elemento fundamental de la programación orientada a objetos es, como su nombre indica, el objeto. Podemos definir un objeto como un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización. Esta definición especifica varias propiedades importantes de los objetos. En primer lugar, un objeto no es un dato simple, sino que contiene en su interior cierto número de componentes bien estructurados. En segundo lugar, cada objeto no en un ente aislado, sino que forma parte de una organización jerárquica de otro tipo.

Un objeto puede considerarse como una especie de capsula dividida en tres partes: relaciones, propiedades y métodos. Cada uno de estos componentes desempeña un papel totalmente independiente. Las relaciones permiten que el objeto se inserte en la organización y están formadas esencialmente por punteros a otros objetos. Las propiedades de un objeto pueden ser heredadas a sus descendientes en la organización. Los métodos son las operaciones que pueden realizarse sobre el objeto, que normalmente estarán incorporados en forma de programas (código) que el objeto es capaz de ejecutar y que también pone a disposición de sus descendientes a través de la herencia. (Pérez, 2008)

La programación orientada a objetos es una metodología de diseño de software y un paradigma de programación que define lo programas en términos de “clases de objetos”, objetos que son entidades que combinan estado (es decir, datos) y comportamiento (esto es, procedimientos o métodos).

La idea principal es construir programas que utilizan objetos de software. Un objeto puede considerarse como una entidad independiente de cómputo con sus propios datos e información. Por tanto, en POO, los objetos de software tienen una correspondencia estrecha con los objetos reales relacionados con el área de la aplicación. Esta correspondencia facilita la comprensión y el manejo del programa de la computadora. En contraste la programación tradicional trabajaba con bytes, variables, matrices, índices y otros artefactos de programación que resultaba difícil relacionar con el problema actual. Además, la programación tradicional se concentra en procedimientos paso a paso, llamados algoritmos, para realizar tareas

deseadas por esta razón a la programación tradicional también se le llama programación orientada a procedimientos

Ventajas de POO

POO ofrece las siguientes ventajas principales:

- Simplicidad: como los objetos de software son modelos de objetos reales en el dominio de la aplicación, la complejidad del programa se reduce y su estructura se vuelve clara y simple.
- Modularidad: cada objeto forma parte de una entidad separada cuyo funcionamiento interno está desacoplado de otras partes del sistema.
- Facilidad para hacer las modificaciones: es sencillo hacer cambios menores en la representación de los datos o los procesos utilizados en un programa OO (orientado a objetos) las modificaciones hechas en el interior de un objeto no afectan ninguna otra parte del programa, siempre y cuando se preserve su comportamiento externo.
- Posibilidad de extenderlo: la adición de nuevas funciones o la respuesta a ambientes operativos cambiantes pueden lograrse con sólo introducir algunos objetos nuevos y variar algunos existentes.
- Flexibilidad: un programa OO puede ser manejable al adaptarse a diferentes situaciones porque es posible cambiar los patrones de interacción entre los objetos sin alterarlos.
- Facilidad para darles mantenimiento: los objetos pueden mantenerse por separado, lo que facilita su localización y arreglo de problemas, así como la adición de otros elementos.
- Reusabilidad: los objetos pueden emplearse en diferentes programas.

Además, POO, permite el polimorfismo, que es la capacidad de un programa para trabajar con diferentes objetos como cajas negras intercambiables. Se permite la creación de objetos compatibles que son transferibles. La modificación y transferibles. La modificación y mejoramiento de un programa polimórfico puede ser tan sólo cuestión de enlazar objetos actualizados. (Paul S. 1999)

Tucker y Noonan (2003), definen como principales características de la programación orientada a objetos las siguientes:

- Herencia: El paradigma orientado a objetos, soporta la reutilización de código a través de la herencia. En un lenguaje orientado a objetos, las clases existen en una jerarquía de clases. Podemos declarar una clase como una subclase de otra clase, que llamamos clase padre o superclase. La subclase hereda las variables y métodos de sus padres. La herencia e utiliza comúnmente como un medio para que una clase padre comparta el código con sus subclases.
- Clases Abstractas: Una clase abstracta no proporciona código para el método, sino que son sus subclases las que deben suministrarlo. Las clases abstractas se utilizan para definir una interfaz pública que deben compartir un conjunto de clases.
- Interfaces: Una interfaz, declara un conjunto de características identificadas por métodos abstractos. Una interfaz, siempre es abstracta, se declare o no abstracta. Se asegura de que cualquier clase que implemente la clase interfaz proporcione definiciones concretas de todos sus métodos, independientemente de la naturaleza de la clase que la implementa.
- Polimorfismo y unión tardía: En los lenguajes orientados a objetos, el polimorfismo se refiere a la unión tardía de una llamada con un método específico de un objeto.

Del mismo modo que Tucker y Noonan, Joyanes y Fernández (2002) mencionan que dentro de las características más importantes de la programación orientada a objetos se encuentran el encapsulamiento, la herencia y el polimorfismo, como lo describen a continuación:

“El encapsulamiento, que se produce al combinar en un objeto los datos con las operaciones que se pueden ejecutar sobre los mismos y establecer sus públicas, accesibles al usuario, y privadas, no accesibles lo que impide usos indebidos. Las partes no accesibles al usuario se dice que están encapsuladas en la clase. En

general los datos miembros de un objeto que son privados y que para acceder a ellos utilizan las rutinas del objeto.

La herencia es la capacidad para crear nuevas clases (descendientes) que se construyen sobre otras existentes, permitiendo que estas les transmitan sus propiedades. En programación orientada a objetos la reutilización de código se efectúa creando una subclase que constituye una restricción o extensión de la clase base o superclase, de la cual hereda sus propiedades.

El polimorfismo consigue que un mismo mensaje pueda actuar sobre diferentes tipos de objetos y comportarse de modo distinto. El polimorfismo adquiere su máxima expresión de la derivación o extensión de clases; es decir cuando se obtienen nuevas clases a partir de una ya existente mediante la propiedad de derivación de clases o herencia."

1.4.1 Lenguajes de programación orientada a objetos

1.4.1.1 Java

Java fue concebido por James Gosling, Patrick Naughton, Chris Warth, Ed Frank y Mike Sheridan en Sun Mycrosystems Inc en 1991.

El desarrollo de la primera versión duró dieciocho meses y se le llamo "Oak". El motivo principal era la necesidad de un lenguaje independiente de la plataforma que se pudiese utilizar para crear software para diversos dispositivos electrónicos como hornos microondas y controles remotos. Sin embargo, con la aparición de la World Wide Web, java ha sido impulsado al frente del diseño de los lenguajes de programación, ya que la red también exigía programas portables. Java está relacionado con C++, que es un descendiente directo de C. La mayor parte del carácter de Java está heredado de estos 2 lenguajes. De C, Java deriva su sintaxis. La mayoría de las características orientadas a objetos están basadas en C++. (ITM, 2007)

Java es desde su diseño, un lenguaje capaz de superar las diferencias que hay entre ordenadores de una red heterogénea, ya que al ser independiente del sistema operativo puede ejecutarse indistintamente. Es un lenguaje de los que se podría

decir fácil de aprender, ya que superada la primera etapa de aprendizaje de los conceptos básicos de programación orientada a objetos el programador tiene la seguridad de que lo que pueda hacer con el Java, lo podrá hacer él mismo y además va a funcionar. (Moldes, 2008)

Características de Java

- Orientación a objetos; Java está totalmente orientado a objetos. No hay funciones sueltas en un programa de Java. Todos los métodos se encuentran dentro de clases. Los tipos de datos primitivos, como los enteros o los dobles, tienen empaquetadores de clases, siendo estos objetos por sí mismos, lo que permite que el programa los manipule.
- Simplicidad: la sintaxis de Java es similar a ANSI C y C++ y, por tanto, fácil de aprender; aunque es mucho más simple y pequeño que C++. Elimina encabezados de archivos, preprocesador, aritmética de apuntadores, herencia múltiple, sobrecarga de operadores, struct, union, y plantillas. Además, realiza automáticamente la recolección de basura. Lo que hace innecesario el manejo explícito de memoria.
- Compactibilidad: Java está diseñado para ser pequeño. La versión más compacta puede utilizarse para controlar pequeñas aplicaciones. El intérprete de Java y el soporte básico de las clases que se mantienen pequeños al empaquetar por separado otras bibliotecas.
- Portabilidad: sus programas se compilan en el código de bytes de arquitectura neutra y se ejecutarán en cualquier plataforma con un intérprete de Java. Su compilador y otras herramientas están programadas en Java. Su intérprete está escrito en ANSI C. De cualquier modo, la especificación del lenguaje Java no tiene características dependientes de la implantación.
- Amigable para el trabajo en red: java tiene elementos integrados para comunicación en red, applets Web, aplicaciones cliente – servidor, además de acceso remoto a bases de datos, métodos y programas.
- Soporte a GUI: la caja de herramientas para la creación de ventanas abstractas (Abstract Windowing Toolkit) de Java simplifica y facilita la

escritura de programas GUI orientados a eventos con muchos componentes de ventana.

- Carga y vinculación incremental dinámica: las clases de Java se vinculan dinámicamente al momento de la carga. Por lo tanto, la adición de nuevos métodos y campos de datos a clases, no requieren de recompilación de clases del cliente.
- Internacionalización: los programas de java están escritos en Unicode, un código de carácter de 16 bits que incluye alfabetos de los lenguajes más utilizados del mundo. La manipulación de los caracteres de Unicode y el soporte para la fecha/hora local hacen que Java sea bienvenido a todo el mundo.
- Hilos: Java proporciona múltiples flujos de control que se ejecutan de manera concurrente dentro de uno de sus programas. Los hilos permiten que su programa emprenda varias tareas de cómputo al mismo tiempo, una característica que da soporte a programas orientados a eventos, para trabajo en red y de animación.
- Seguridad: entre las medidas de seguridad de Java se incluyen restricciones en sus applets, implantación redefinible de sockets y objetos de administrador de seguridad definidos por el usuario. Hacen que las applets sean confiables y permiten que las aplicaciones implanten y se apeguen a las reglas de seguridad personalizadas. (Paul S. 1999)

1.4.1.2 PHP

PHP es un lenguaje de script de propósito general de código abierto. Fue creado por Rasmus Lerdof en 1994 y empezó siendo acrónimo de Personal Home Page ya que se diseñó originalmente para la creación de páginas web dinámicas debido a que se puede embeber con HTML. A medida que sus capacidades fueron evolucionando y ampliándose paso de ser el acrónimo de Hypertext Preprocessor.

Está inspirado principalmente en los lenguajes C y Perl. Aunque orientado a la programación web, también se puede utilizar desde la línea de comandos (como Perl o Python) en su versión PHP-CLI (Command Line Interface) o como lenguaje de propósito general con una interfaz gráfica utilizando la extensión PHP-Qt o PHP-

GTK. Las versiones recientes tienen soporte para orientación a objetos, aunque no es propiamente un lenguaje orientado a objetos.

PHP se puede ejecutar en la mayoría de los servidores web (Apache e Internet Information Services (IIS)) de Microsoft entre otros), sistemas operativos (Linux, variantes de UNIX -HP-UX, Solaris, OpenBSD-, Microsoft Windows, Mac OS X, RISC OS), así como varios sistemas de gestión de bases de datos relacionales, bien mediante una extensión (MySQL), con una capa de abstracción PDO, utilizando el estándar Open Database Connection a través de la extensión ODBC o sockets. PHP puede trabajar como un módulo del servidor o como un procesador CGI.

Además de generar salida HTML, PHP puede generar ficheros en otros formatos como XHTML, XML o PDF. PHP incluye capacidades para generar procesamiento de cadenas de texto que son compatibles con las de Perl. También incluye extensiones y herramientas para analizar y procesar documentos XML. (Gortázar et al., 2012)

PHP 5 cubriendo las carencias de las versiones anteriores, nos provee un soporte muy completo para la Programación Orientada a Objetos. El nuevo motor Zend (Zend Engine 2) ha sido rediseñado por completo y proporciona todos los mecanismos necesarios asociados a este paradigma de programación. Así desde esta versión podemos usar PHP indistintamente, según el gusto personal, en sus dos facetas Procedural y Orientada a Objetos (OO).

A diferencia de otros lenguajes OO nativos tipo Java, C++, Eiffel, etc., PHP es un lenguaje de script en el que no es indispensable dominar y conocer toda la parafernalia de clases que incorpora el lenguaje.

PHP proporciona las siguientes posibilidades:

- Soporte para múltiples sistemas operativos: UNIX (entre otras, Linux, HP-UX, Solaris y OpenBSD), Microsoft Windows, MAC OS X, RISC OS. Actualmente está en preparación para las plataformas IBM OS/390 y AS/400.

- Soporte para múltiples servidores web: Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro server, Caundium, Xitami, OmniHTTPd y muchos otros.
- Soporte para más de 25 gestores de bases de datos.
- Soporte ODBC y extensiones DBX.
- Soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros.
- Puede utilizar objetos Java de forma transparente, como objetos PHP.
- La extensión de CORBA puede ser utilizada para acceder a objetos remotos.
- PHP soporta WDDX para intercambio de datos entre lenguajes de programación en Web.
- Generación de resultados en múltiples formatos como XHTML, XML, ficheros de imágenes, ficheros PDF y películas Flash.
- Parser de documentos XML, soporte de estándares SAX y DOM. Manejo de XSLT para transformar documentos XML.
- Manejo de expresiones regulares POSIX Extended o Perl.
- Funciones de comercio electrónico, como Cybercash, CyberMUT, VeriSign PayFlow Proy CCVS para las pasarelas de pago.
- Otras extensiones muy interesantes son las funciones del motor de búsquedas mnoGoSearch, funciones para pasarelas de IRC, utilidades de compresión (gzip, bz2), conversión de calendarios, traducción. (Gutiérrez y Bravo 2005)

1.4.1.3 C#

Sobre el lenguaje de programación orientado a objetos C#, Katcheroff(2008) escribe en su manual de programación los antecedentes descripción y características de la siguiente manera:

Microsoft C# es un nuevo lenguaje de programación diseñado para crear un amplio número de aplicaciones empresariales que se ejecutan en .NET Framework. Supone una evolución de Microsoft C y Microsoft C++; es sencillo, moderno,

proporciona seguridad de tipos y está orientado a objetos. El código creado mediante C# se compila como código administrado, lo cual significa que se beneficia de los servicios de Common Language Runtime. Estos servicios incluyen interoperabilidad entre lenguajes, recolección de elementos no utilizados, mejora de la seguridad y mayor compatibilidad entre versiones.

C# se presenta como Visual C# en el conjunto de programas Visual Studio .NET. Visual C# utiliza plantillas de proyecto, diseñadores, páginas de propiedades, asistentes de código, un modelo de objetos y otras características del entorno de desarrollo. La biblioteca para programar en Visual C# es .NET Framework.

C# fue creado por Microsoft con el propósito de ser el mejor lenguaje de programación que exista para describir aplicaciones destinadas a la plataforma .NET. Combina la facilidad de desarrollo propia de Visual Basic con el poderío del lenguaje C++, un lenguaje con el cual se ha escrito la mayor parte de la historia del software y de los sistemas operativos de todos los tiempos. En líneas generales, podemos decir que es un lenguaje orientado a objetos simple y poderoso. Una muestra de esto es que las aplicaciones desarrolladas en él podrán funcionar tanto en Windows como en dispositivos móviles, ya sea en un teléfono celular o en una PDA.

Características fundamentales

Lo primero que debemos saber es que podemos crear una diversidad de programas utilizando este lenguaje: desde aplicaciones de consola, aplicaciones para Windows o aplicaciones Web, hasta software para dispositivos móviles, drivers y librerías para Windows. C# ha evolucionado notablemente desde sus primeras versiones, al incorporar funcionalidades que mejoran y facilitan la escritura de código, la seguridad de tipos y el manejo automático de memoria.

Con respecto a las principales características del lenguaje, Joyanes y Fernández (2002) las enlistan como:

- C# permite la llamada a librerías nativas del API de Windows, la llamada a componentes COM, y el acceso a bajo nivel si es necesario, Además, los programas y componentes diseñado en C# se integran perfectamente con los realizados en cualquier otro lenguaje del entorno .NET.

- Con C# pueden crearse aplicaciones muy complejas con un gran número de componentes, lo que justifica la aparición del ensamblado (assembly) o unidad de ensamblaje. El ensamblado agrupa los archivos que contienen el código con los recursos que precisa la aplicación y describe sus dependencias, constituyendo una pieza fundamental en los procesos de distribución.
- C# es un lenguaje fuertemente tipificado y orientado a objetos que trabaja en un entorno administrado, es decir controlado por el CLR, con seguridad de tipos. Ser fuertemente tipificado implica que todos los datos utilizados deben tener sus tipos declarados explícitamente, lo que determina cómo ejecutar las operaciones y facilita al compilador la detección de errores en las mismas. La seguridad de tipos proporciona programas robustos, al evitar la ejecución de código incorrecto, como referencias no válidas conversiones que desbordan la variable u objeto destino.
- Todos los tipos comparten la misma funcionalidad base y pueden ser convertidos en una cadena, serializados o guardados en una colección.
- Permite declarar clases y métodos no seguros.
- Utiliza los denominados espacios calificados o espacios de nombres (Namesapaces) para clasificar tipos, que pueden ser otros espacios de nombres, estructuras, interfaces, enumeraciones, delegados o clases, o agrupar los que se encuentren lógicamente relacionados.
- Parte fundamental del código de C# son las clases, que sirven como plantilla para construir objetos que pueden extender o heredar a otra.
- Todas las clases de C# tienen como base la clase Object e incluso todos los tipos se pueden tratar como un Object.
- C# permite la implementación de interfaces y soluciona el problema de la herencia múltiple permitiendo que una clase derive de múltiples interfaces. En C# una subclase no puede derivar de varias súper clases.
- C# compila un código intermedio, denominado Intermideate Language (IL) similar al bytecode de Java. Este código intermedio denominado Entorno

Común de Ejecución (Common Language Runtime), que interpreta también cualquier otro lenguaje que compile el mencionado código.

- Libera a los programadores de la necesidad de gestionar el manejo de la memoria y evita problemas con punteros, fugas de memoria y referencias circulares, apoyándose en el CLR. C# no elimina los punteros, pero los hace innecesarios para la mayor parte de las tareas de programación.
- Utiliza el sistema de tipos común (Common Types Sysrem) que comprende los tipos de datos soportados por el entorno común de ejecución (CLR).
- Ofrece soporte para la programación multihilos, que es la capacidad de un programa de ejecutar varias tareas simultáneamente. Los hilos sincronizados son muy útiles en la creación de aplicaciones distribuidas en la red.
- Utiliza excepciones para el manejo de errores. las excepciones son los objetos que describen y permiten controlar errores y problemas inesperados en el funcionamiento de un programa sin oscurecer el diseño del mismo y se gestionan mediante un código especial que no sigue el flujo normal de ejecución de dicho programa.

1.5 Tecnologías para desarrollo web

1.5.1 HTML y XHTML

El lenguaje de marcas de hipertexto, HTML o (HyperText Markup Language) se basa en el metalenguaje SGML (Standard Generalized Markup Language) y es el formato de los documentos de la World Wide Web. El World Wide Web Consortium (W3C) es la organización que desarrolla los estándares para normalizar el desarrollo y la expansión de la Web y la que publica las especificaciones relativas al lenguaje HTML.

XHTML (Lenguaje de Marcado de Hipertexto Extensible) es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de remplazar a HTML ante su limitación de uso con las cada vez más abundantes herramientas basadas en XML. XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñado para describir los datos.

¿Para qué sirve?

Ante la llegada al mercado de un gran número de dispositivos, XHTML surge como el lenguaje cuyo etiquetado, más estricto que HTML, va a permitir una correcta interpretación de la información independientemente del dispositivo desde el que se accede a ella. XHTML puede incluir otros lenguajes como MathML, SMIL o SVG, al contrario que HTML.

¿Cómo funciona?

XHTML es el lenguaje de marcas creado para sustituir HTML. Se podría decir que XHTML es la versión XML de HTML, ya que tienen básicamente las mismas funcionalidades, cumpliendo con las especificaciones más estrictas de XML.

Su objetivo es avanzar en el proyecto de W3C (World Wide Web Consortium), de lograr páginas web donde la información y la forma de presentarla estén claramente separadas. XHTML es, por lo tanto, un lenguaje semántico, que quiere decir que no definimos el aspecto de las cosas sino lo que significan.

Esta característica se aprecia claramente en su estructuración semántica, ya que XHTML sirve únicamente para transmitir la información que contiene un documento dejando el aspecto y el diseño para las hojas de estilo (CSS), y la interactividad y funcionalidad para JavaScript, por ejemplo.

Como ya hemos comentado, la diferencia entre ambos lenguajes es principalmente un cambio en el concepto y forma de la estructuración del documento. Se busca una sintaxis coherente dentro del documento, donde los distintos elementos deben estar correctamente anidados, todas las etiquetas en minúsculas, los elementos cerrados correctamente, los valores de los atributos entrecomillados, etc. De esta forma, se evita la anarquía existente en muchos códigos web.

El lenguaje XHTML se basa en el uso de etiquetas, también llamadas marcas, directivas o comandos (tags). Estas etiquetas son fragmentos de texto delimitados por el símbolo menor que (<) y mayor que (>). Básicamente, estas etiquetas indican al navegador la forma de representar los elementos (texto, gráfico, etc.) que contiene el documento.

Reglas a tener en cuenta

1.- Un documento XHTML no puede tener etiquetas abiertas. Un debe cerrarse con un un elemento <li> debe cerrarse con un elemento y así con todos los elementos.

2.- Los elementos en un documento XHTML deben estar anidados de forma lógica, por lo tanto, deben cerrarse en el orden inverso del que fueron abiertos. Por ejemplo: debe cerrarse en el orden .

3.- Todas las etiquetas y sus atributos deben escribirse siempre en minúsculas. No ocurre lo mismo con el valor del atributo que sí puede ir con mayúsculas.

4.- Todos los valores de los atributos de las etiquetas deben estar siempre entrecomillados.

5.- El primer elemento del documento siempre será <html> y deberemos colocar el <head> y el <body> en su orden correcto sin olvidar cerrarlos al final del

documento. El uso de <title> es de carácter obligado y se tendrá que poner justo después de <head>.

6.- El atributo id reemplaza el atributo name. Sólo en casos de compatibilidad con navegadores antiguos está permitido el empleo del atributo name usando XHTML transicional.

7.- XTHML es un lenguaje semántico, por lo tanto, cuando desarrollemos un documento hay que pensar en la organización y en la estructuración del mismo más que en su maquetación. Se deberán utilizar las etiquetas de acuerdo al fin con el que fueron pensadas, por lo tanto, es recomendable usar para un título la etiqueta <h1> en vez de un con una clase asociada.

(Orós, J. 2012)

1.5.2 JavaScript

Es el lenguaje interpretado orientado a objetos desarrollado por Netscape que se utiliza en millones de páginas web y aplicaciones de servidor en todo el mundo.

Las capacidades dinámicas de JavaScript incluyen construcción de objetos en tiempo de ejecución, listas variables de parámetros, variables que pueden contener funciones, creación de scripts dinámicos, introspección de objetos y recuperación de código fuente (los programas de JavaScript pueden descompilar el cuerpo de funciones a su código fuente original).

En el libro “Ajax en J2EE primera edición”, Antonio Martin (2007) describe que, un script es un bloque de código que se incluye directamente en el documento HTML y que puede ser implementado en el navegador. Mediante estos scripts se pueden incluir instrucciones de código que respondan a acciones de usuario y que sean capaces de modificar dinámicamente el aspecto de las páginas.
De todos los lenguajes de script existentes, es sin duda alguna JavaScript el que se ha ido imponiendo al resto, debido fundamentalmente a su sport en prácticamente todos los modelos de navegadores que actualmente se utilizan para acceder a internet.

Un script de JavaScript se incluye dentro de la página HTML delimitándolo por la etiqueta <script>.

Igualmente menciona algunas consideraciones importantes de este lenguaje:

- JavaScript hace distinción entre mayúsculas y minúsculas
- Las sentencias terminan con ;
- Para comentarios de una línea se utiliza //
- Los comentaos de más de una línea se encierran entre /* y */
- Los bloques de instrucciones se delimitan con { y }

1.5.3 Framework

Un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente suelen incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

1.5.3.1 Java Server Faces

Java Server Faces es un framework basado en el MVC (Modelo Vista Controlador) para aplicaciones Java basadas en la web.

La tecnología JavaServer Faces surge como una solución a la separación entre la presentación y el comportamiento en una aplicación Web, de forma tal que las actividades de los autores de las páginas puedan separarse de las actividades de los desarrolladores de la lógica.

Características

- Conectar eventos generados en el cliente con código de la aplicación en el servidor
- Mapear componentes UI a la página de datos del servidor
- Construir un UI con componentes reutilizables y extensibles
- Grabar y restaurar el estado del UI más allá de la vida de las peticiones del servidor.

Esta tecnología se ejecuta del lado del servidor y es orientado a componentes.

Sergio Rios (2017) menciona que Java Server Faces (JSF) es una tecnología y framework para aplicaciones basadas en web que simplifica el desarrollo de

interfaces de usuario en aplicaciones JavaEE. JSF usa Java Server Pages, XUL o XHTML como la tecnología que permite hacer el despliegue de las páginas.

JSF incluye:

- ✓ Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, mejorar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- ✓ Un conjunto por defecto de componentes de la interfaz de usuario.
- ✓ Un modelo de eventos en el lado del servidor.
- ✓ Administración de estados.
- ✓ Beans administrados.

1.5.3.2 Primefaces

PrimeFaces es una biblioteca de componentes para JavaServer Faces (JSF) de código abierto que cuenta con un conjunto de componentes enriquecidos que facilitan la creación de las aplicaciones web. Está bajo la licencia de Apache License V2.

Características:

- Conjunto de componentes para desarrollo web (Editor de HTML, autocompletar, graficas paneles, etc).
- Soporte de Ajax con despliegue parcial, lo que permite controlar qué componentes de la página actual se actualizarán y cuáles no.
- 25 temas prediseñados
- Grid de desarrollo responsivo.

1.5.4 CSS

Las llamadas hojas de estilo en cascada, CSS o Cascading Style Sheets abren un nuevo abanico de posibilidades para los creadores de páginas web. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación o aspecto, como por ejemplo la combinación XHTML y CSS.

Usando hojas de estilo, los webmasters pueden, por ejemplo, generar un estilo para todo el resto de los documentos de una web, con el consiguiente ahorro de tiempo de diseño y mantenimiento.

Las hojas de estilo constituyen el complemento ideal para HTML Y XHTML. Su misión es definir la apariencia y el estilo de sus elementos.

Niveles de las CSS

Las hojas de estilo tienen diferentes niveles y perfiles (estándares). Prácticamente todos los navegadores actuales implementan el nivel 1 y 2, aunque las últimas versiones de algunos de ellos también implementan caso al 100% el nivel 3. Veamos a continuación sus principales características.

- El nivel 1 de CSS (CSS1) define las propiedades de tipos de letra, márgenes, colores y otras herramientas de estilo que son comunes a casi todos los perfiles CSS.
- El nivel 2 de CSS (CSS2) incluye todo lo del nivel 1 de CSS añadiendo elementos de posicionamiento absoluto, numeración automática, salto de página, texto de derecha a izquierda y otras características de maquetación avanzada.
- El nivel de CSS3 (CSS3), incluye todo lo del nivel 2 e incorpora nuevos selectores, hipertexto enriquecido, bordes y fondos, texto vertical, interacción del usuario, reproducción en dispositivos multimedia y muchos más.

Pros y contras de las hojas de estilo.

- Podemos modificar la presentación de todos los elementos estándar del documento sin tener que modificar el código XHTML estructural.
- Disponemos de comandos y atributos más potentes y precisos con los que poder maquetar exactamente un documento.
- Es un lenguaje muy sencillo, ya que se basas en el uso de propiedades muy intuitivas, similares a las de un procesador de texto inglés.

- Podemos generar un estilo externo que contenga todas las definiciones de estilo de un documento y modificar éste únicamente para efectuar cambios en una o varias páginas web.
- Es uno de los pilares del DHTML y puede combinarse con JavaScript, VBScript, etc.
- Su uso estructurado y razonado permite ahorrar muchas líneas de código.
- Respeto a los estándares, consiguiendo un sitio web será más funcional para la mayoría de los navegadores.

Sin embargo, no todo son ventajas su único lado negativo es la incompatibilidad entre navegadores. Aunque todos aseguran la compatibilidad, lo cierto es que entre versiones de un mismo navegador ya hay irregularidades. La solución a éste inconveniente pasa por conocer las propiedades implementadas es cada versión de navegador probando nuestros estilos en varios de ellos.

1.6 Aplicaciones web

En las aplicaciones web suelen distinguirse tres niveles (como en las arquitecturas cliente / servidor de tres niveles): el nivel superior que interacciona el usuario (el cliente web, normalmente un navegador), el nivel inferior que proporciona los datos (la base de datos) y el nivel intermedio que procesa los datos (el servidor web).

Una aplicación web (web-based application) es un tipo especial de aplicación cliente/servidor, donde tanto el cliente (el navegador, explorador o visualizador) como el servidor (servidor web) y el protocolo mediante el cual se comunican (HTTP) están estandarizados y no han de ser creados por el programador de aplicaciones.

- El cliente

El cliente web es un programa con el interacciona el usuario para solicitar a un servidor web el envío de los recursos que desea obtener mediante HTTP.

La parte cliente de las aplicaciones web suele estar formados por código HTML que forma la página web más algo más de código ejecutable realizado en lenguaje script del navegador (JavaScript o VBScript) o mediante pequeños programas (applets) realizados en Java. También se suelen emplear plug-ins que permiten visualizar

otros contenidos multimedia (como Macromedia Flash), aunque no se encuentran tan extendidos como las tecnologías anteriores y plantean problemas de incompatibilidad entre distintas plataformas. Por tanto, la misión del cliente web es interpretar las páginas HTML y los diferentes recursos que contienen (imágenes, sonidos, etc.).

- El servidor

El servidor es un programa que está esperando permanentemente las solicitudes de conexión mediante el protocolo HTTP por parte de los clientes web. En los sistemas Unix suele ser un “demonio” y en Windows un servicio

- Ventajas y desventajas

El desarrollo explosivo de internet en especial de la WWW se debe a la aceptación por todo el mundo de los estándares y tecnologías que emplea.

Una ventaja clave del uso de aplicaciones web es que gestionar el código en el cliente se reduce drásticamente. Suponiendo que existe un navegador o explorador estándar en cada cliente, todos los cambios, tanto de interfaz como de funcionalidad, que se deseen realizar en la aplicación se realizan cambiando el código que resida en el servidor web.

Una segunda ventaja, relacionada con la anterior, es que se evita la gestión de versiones. Se evitan problemas de inconsistencia en las actualizaciones, ya que no existen clientes con distintas versiones de la aplicación.

Una tercera ventaja es que, si la empresa ya está utilizando Internet, no se necesita comprar ni instalar herramientas adicionales para los clientes.

Otra ventaja es que, de cara al usuario, los servidores externos (Internet) e internos (Intranet) aparecen integrados, lo que facilita el aprendizaje y uso.

Para que una aplicación web se pueda ejecutar en distintas plataformas (hardware y sistema operativo), sólo se necesita disponer de un navegador para cada una de las plataformas, y no es necesario adaptar el código de la aplicación a cada una de ellas. Además, las aplicaciones web ofrecen una interfaz gráfica independiente de la plataforma.

Una desventaja, que sin embargo está desapareciendo rápidamente, es que la programación web no es tan versátil o potente como la tradicional. El lenguaje HTML

presenta varias limitaciones, como es el escaso repertorio de controles disponibles para crear formularios. Por otro lado, al principio las aplicaciones web eran básicamente de “sólo lectura “: permitían una interacción con el usuario prácticamente nula. Sin embargo, con la aparición de nuevas tecnologías de desarrollo como Java, JavaScript y ASP, esta limitación tiende a desaparecer.
(Luján 2012)

1.6.1 Arquitectura de tres capas

Las capas que componen esta arquitectura, son las siguientes:

- Capa cliente.
- Capa intermedia.
- Capa de datos.

Capa cliente: Se trata de la capa con la que interactúa el usuario de la aplicación normalmente a través de un navegador web. Realiza principalmente dos funciones: por un lado, se encarga de capturar los datos del usuario con los que opera la capa intermedia y enviárselos a esta, y por oro presentar al usuario resultados generados por la aplicación.

Las páginas web, construidas mediante XHTML, css, son las encargadas de implementar esta funcionalidad, apoyándose como ya hemos visto de código JavaScript/ AJAX para mejorar la experiencia del usuario con la aplicación.

Capa intermedia

En una arquitectura de tres capas la capa intermedia está constituida por la aplicación en sí. Esta se encuentra instalada en una maquina independiente conocida como servidor.

La aplicación de la capa intermedia es ejecutada por un motor, los servidores necesitan otro software conocido como servidor Web, que sirva de interfaz entre la aplicación y el cliente.

Las funciones de la capa intermedia consisten en:

- Recoger los datos enviados desde l capa cliente.
- Procesar la información y, en general implementar la lógica de aplicación, incluyendo el acceso a datos.
- Generar las respuestas para el cliente.

Capa de datos: La capa de datos tiene como misión en almacenamiento permanente de la información manejada por la aplicación y la gestión de la seguridad de los mismos.

Para esta tarea se utiliza, en la mayoría de los casos, las llamadas bases de datos relacionales. Una base de datos relacional distribuye la información entre diferentes tablas, relacionándose entre sí a través de un campo común que permita identificar los registros de una tabla que correspondan con los de otra.

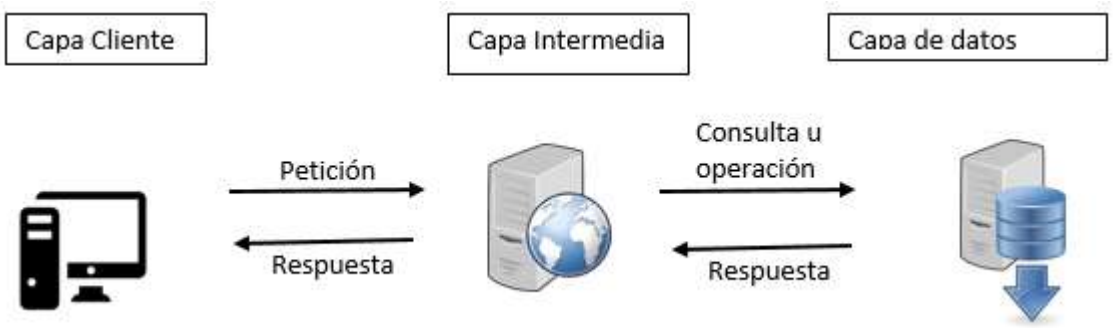


Figura 9.- Modelo de 3 capas.

1.6.1.1 Modelo Vista Controlador

De cara a afrontar con éxito el desarrollo de la capa intermedia de una aplicación web, se hace necesario establecer un modelo o esquema que permita estructurar esta capa en una serie de bloques o componentes, de modo que cada uno de estos bloques tenga unas funciones definidas dentro de la aplicación y pueda desarrollarse de manera independiente.

Uno de estos esquemas es la arquitectura Modelo Vista Controlador (MVC), la cual proporciona una clara separación entre las distintas responsabilidades de la aplicación.

Según esta arquitectura. La capa intermedia de una aplicación web puede dividirse en tres grandes bloques funcionales:

- Controlador
- Vista
- Modelo

➢ Controlador.

Se puede decir que el controlador es el “cerebro” de la aplicación. Todas las peticiones a la capa intermedia que se realicen desde el cliente son dirigidas al

controlador, cuya misión es determinar las acciones a realizar para cada una de estas peticiones e invocar el resto de los componentes de la aplicación para que realicen las acciones requeridas en cada caso.

- Vista:

Tal y como se puede deducir de su nombre, la vista es la encargada de generar las respuestas, que deben ser enviadas al cliente. Cuando esta respuesta incluye datos proporcionados por el controlador, el código XHTML de la página no será fijo sino que deberá ser generado de forma dinámica.

- El Modelo:

En la arquitectura MVC la lógica de negocio de la aplicación, incluyendo el acceso a los datos y su manipulación, esta encapsulada dentro del modelo. En una aplicación web el modelo puede ser implementado mediante clases estándar. (Martin, 2007)

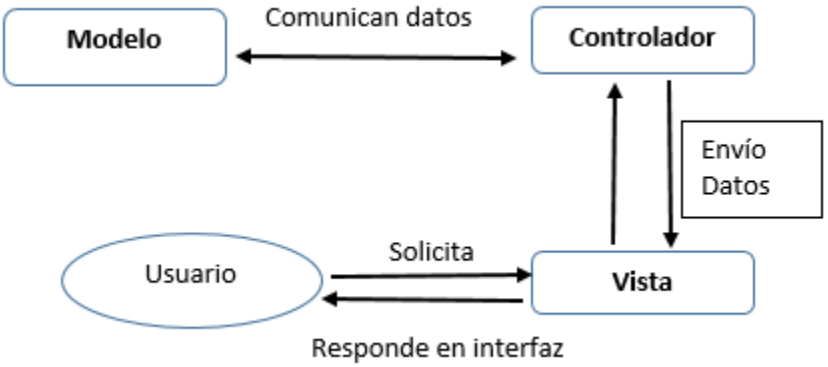


Figura 10.- Modelo Vista Controlador

1.7 Diagramas UML

El lenguaje unificado de modelado o UML (Unified Modelillg Language) es un lenguaje de modelado, y no un método. La mayor parte de los métodos consisten, al menos en principio, en un lenguaje y en un proceso para modelar. El lenguaje de modelado es la notación (principalmente gráfica) de que se valen los métodos para expresar los diseños. El proceso es la orientación que nos dan sobre los pasos a seguir para hacer el diseño.

“UML provee un conjunto estandarizado de herramientas para documentar el análisis y el diseño de un sistema software. El conjunto de herramientas de UML

incluye diagramas que permiten a las personas visualizar la construcción de un sistema orientado a objetos. El UML consiste en cosas, relaciones y diagramas.
El UML es una potente herramienta que puede mejorar en forma considerable la calidad del análisis y el diseño de sistemas, y se espera que las prácticas mejoradas se traduzcan en sistemas de mayor calidad.
Al usar UML en forma iterativa podemos lograr una mejor comprensión en relación con los requerimientos del sistema los procesos que deben ocurrir en el sistema para cumplir con esos requerimientos." (Kendall y Kendall, 2011)

1.7.1 Tipos de diagramas UML

Kendall y Kendall (2011), concuerdan que los diagramas más utilizados son:

- Un diagrama de casos de uso que describe la forma en que se utiliza el sistema.
- Un escenario de caso de uso. Este escenario es una articulación verbal de excepciones para el comportamiento principal descrito por el caso de uso principal.
- Un diagrama de actividad, que ilustra el flujo de actividades en general. cada caso puede crear un diagrama de actividad.
- Los diagramas de secuencia, que muestran la secuencia de las actividades y las relaciones entre las clases. El diagrama de comunicación es la alternativa a un diagrama de secuencia, el cual contiene la misma información, pero enfatiza la comunicación en vez de la sincronización.
- Los diagramas de clases, que muestran las clases y sus relaciones.
- Los diagramas de estados, que muestran las transiciones de estado.

Fowlder & Scott (1999) definen algunos tipos de diagramas UML como se muestra a continuación.

- Diagrama de actividades:

El diagrama de actividades combina ideas de varias técnicas: el diagrama de eventos de Jim Odelt las técnicas de modelado de estados de SOL y las redes de retri. Estos diagramas son particularmente útiles en conexión con el nujo de trabajo

y para la descripción del comportamiento que tiene una gran cantidad de proceso paralelo.

- Diagramas de estados:

Los diagramas de estados son una técnica conocida para describir el comportamiento de un sistema. Describen todos los estados posibles en los que puede entrar un objeto particular y la manera en que cambia el estado del objeto, como resultado de los eventos que llegan a él. En la mayor parte de las técnicas los diagramas de estados se dibujan para una sola clase, mostrando el comportamiento de un solo objeto durante todo su ciclo de vida

- Diagramas de emplazamiento:

El diagrama de emplazamiento es aquel que muestra las relaciones físicas entre los componentes de software y de hardware en el sistema entregado. Así, el diagrama de emplazamiento es un buen sitio para mostrar cómo se mueven los componentes y los objetos, dentro de un sistema distribuido.

- Diagramas de interacción:

Los diagramas de interacción son modelos que describen la manera en que colaboran grupos de objetos para cierto comportamiento. Habitualmente, un diagrama de interacción capta el comportamiento de un solo caso de uso. El diagrama muestra cierto número de ejemplos de objetos y los mensajes que se pasan entre estos objetos dentro del caso de uso.

- Diagramas de casos de uso

Con la ayuda de un diagrama de casos de uso, puede analizar y comunicar:

- Los escenarios en los que el sistema o aplicación interactúa con personas, organizaciones o sistemas externos.
- Los objetivos que el sistema o aplicación contribuye a lograr.
- El ámbito del sistema.

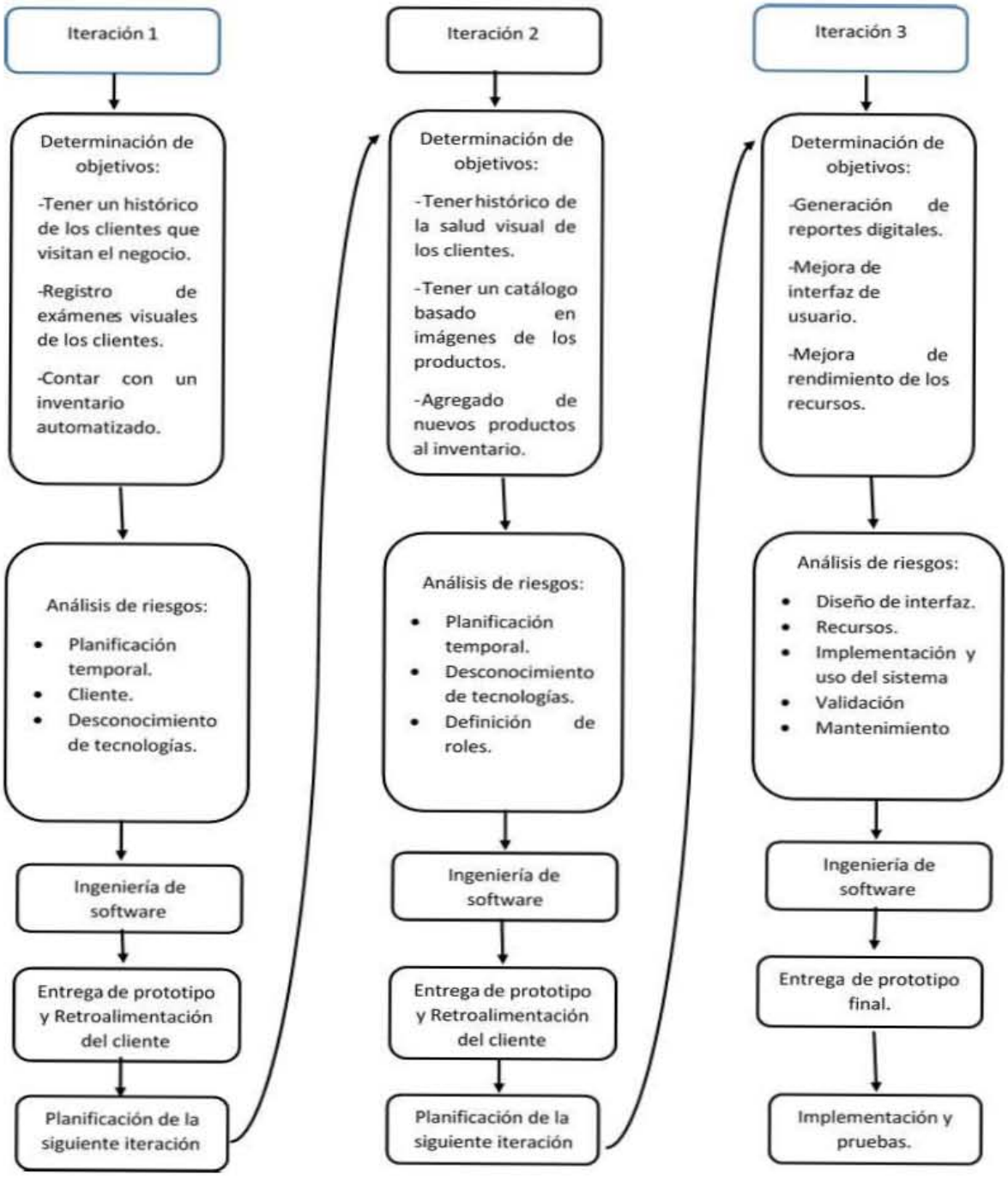
En un diagrama de casos de uso no se muestran los casos de uso en detalle; solamente se resumen algunas de las relaciones entre los casos de uso, los actores y los sistemas. En concreto, en el diagrama no se muestra el orden en que se llevan a cabo los pasos para lograr los objetivos de cada caso de uso. Esos detalles

pueden describirse en otros diagramas y documentos, que pueden vincularse a cada caso de uso.

En las descripciones que se proporcionen de los casos de uso se usarán diversos términos relacionados con el dominio en el que trabaja el sistema, como Ventas, Menú, Cliente, etc. Es importante definir de manera clara estos términos y sus relaciones y, para ello, puede resultar útil un diagrama de clases de UML.

Capítulo 2: Planificación del sistema.

El siguiente diagrama muestra las iteraciones que fueron necesarias para concluir con el desarrollo del software de acuerdo con el paradigma en espiral.



2.1 Análisis y determinación de objetivos del sistema

Para comenzar con el desarrollo de software, es necesario conocer cómo funciona actualmente la empresa, así como la ejecución de cada uno de los procesos involucrados en la funcionalidad de la misma.

2.1.1 Análisis funcional del negocio

De acuerdo con lo analizado, se describen los procesos que se llevan a cabo por el administrador del negocio y por los clientes del mismo, y se identifican a continuación:

Procedimiento del cliente

- El cliente llega al negocio a solicitar una revisión visual.
- El optometrista toma algunos datos del cliente para tener una breve descripción de los antecedentes que tiene con respecto a su visión.
- El optometrista comienza a realizar el examen de vista para verificar cual es la condición del cliente y hace sus respectivas anotaciones en un formulario impreso.
- De acuerdo con los resultados y con previo conocimiento de los gustos del cliente, el optometrista muestra los armazones que tiene en existencia.
- Si el cliente decide hacer la compra, el optometrista hace el cálculo del precio final, entrega una nota de compra y guarda una copia de dicha nota para inventario.
- El cliente acude el día que se indicó para entrega del producto.

Procedimiento de Administración de inventario.

- Con las notas de compra el optometrista lleva el control de las ventas y de las compras del negocio en un archivo impreso.

Para visualizar de manera más el procedimiento de interacción del usuario con el cliente, así como el manejo de inventario que se realiza actualmente se realizó la figura 11.

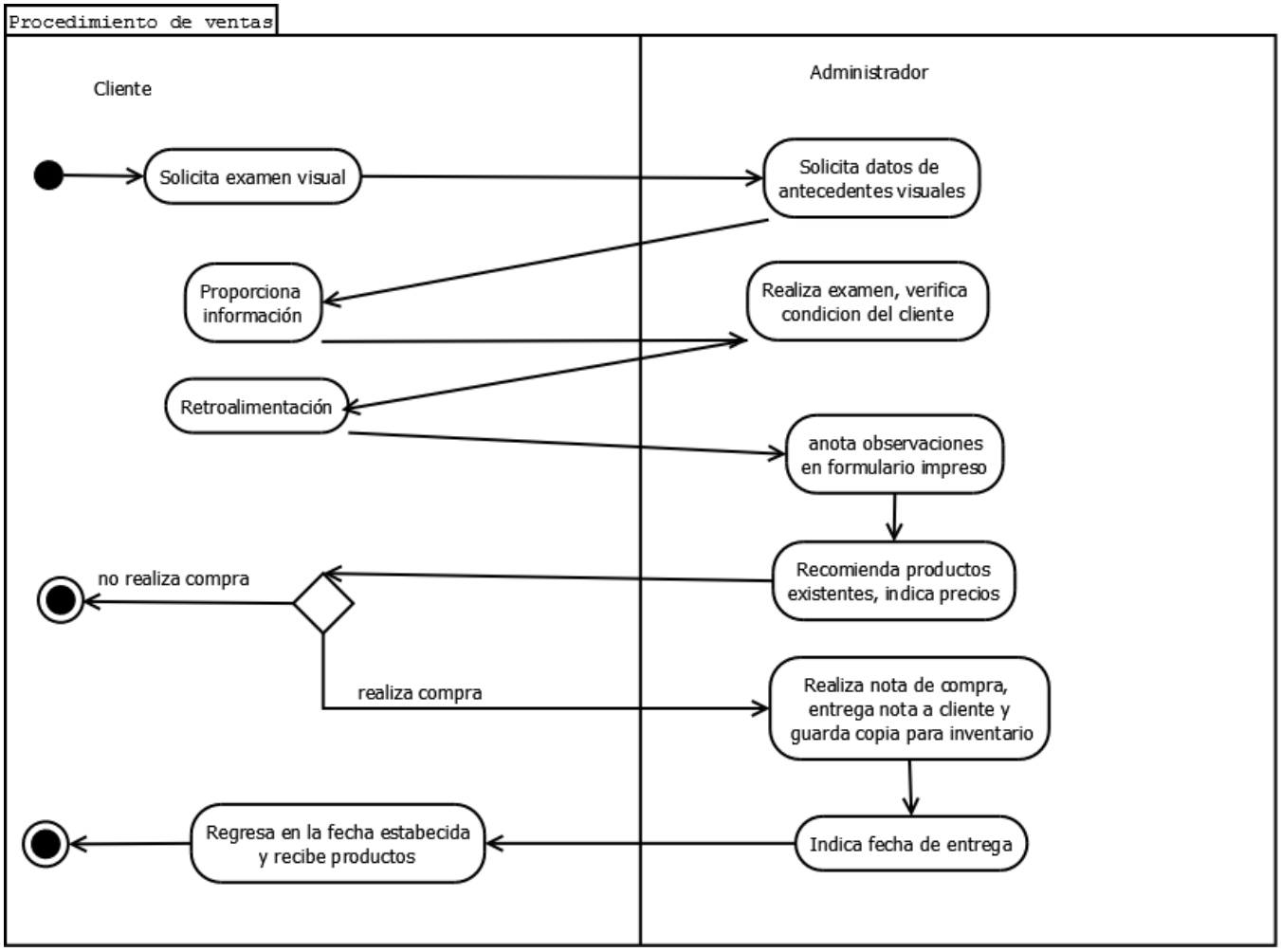


Figura 11. Diagrama de flujo de procesos actuales del negocio.

2.1.2 Determinación de requerimientos

Una vez analizados los procedimientos actuales, en conjunto con el administrador del negocio, se determinaron los siguientes requerimientos del sistema:

- ✓ Tener un histórico de los clientes que visitan el negocio.
- ✓ Tener un control del histórico de la salud visual de cada cliente.
- ✓ Poder registrar cada examen visual que se haya realizado al cliente con el fin de tener un historial de la salud visual del cliente.
- ✓ Contar con un catálogo de productos.
- ✓ Poder agregar los productos que va adquiriendo para el negocio.
- ✓ Poder registrar y ver el detalle de las ventas realizadas.
- ✓ Contar con un inventario automatizado y controlado.

- ✓ Generar diversos reportes en formatos digitales, que pueda guardar e imprimir.

2.1.3 Determinación de objetivos del sistema

Al realizar un análisis funcional de la organización, concluir con la encuesta de determinación de requerimientos (Anexo 1), y de acuerdo con la primera etapa del ciclo de vida del desarrollo de software, se proponen los siguientes objetivos del sistema:

- Permitir al usuario del sistema tener un historial del cliente, de acuerdo a los exámenes visuales realizados durante sus visitas, las preferencias en productos del cliente, etc., para una mejor atención futura; ingresando la información del examen visual en el sistema para guardarla en cada visita.
- Dar al usuario un control sobre su inventario (conocer los productos existentes, así como el histórico de ventas realizadas), para una mejor toma de decisiones, esto será posible mediante la generación automática y resguardo de un Kárdex.
- Tener un catálogo digital de productos, al poder agregar una imagen a la descripción del producto.
- Sugerir un precio estimado de los productos a vender, para hacer más ágil el proceso, mediante la generación automática de dicho cálculo por el sistema.
- Poder generar diversos reportes digitales, para tener un conocimiento profundo del negocio, así como de sus clientes, éstos generados automáticamente por el sistema.
- Permitir al usuario generar importancia al cliente, mejorando la atención hacia ellos, mediante recordatorios enviados por el sistema.

2.2 Análisis del riesgo

Una vez analizados los requerimientos del sistema, como segunda fase del ciclo de desarrollo de software en espiral, se analizan los riesgos que este proyecto puede generar asimismo se indican posibles soluciones a estos problemas, se dividen en dos tipos:

Riesgo del proyecto

Riesgo	**Solución**
Planificación Temporal	Crear un Diagrama de Gantt con fechas estipuladas para termino de cada etapa de desarrollo, agregar tiempo para solucionar problemas que lleguen a surgir
Cliente (Falta de entendimiento con el cliente)	Mantener buena relación con el cliente a fin de tener una comunicación fluida y altamente útil para el desarrollo del sistema.
Desconocimiento de tecnologías	Realizar una investigación sobre las mejores tecnologías en la actualidad para el desarrollo de un sistema de este tipo.
Definición de roles	Realizar análisis de conocimiento a los integrantes del equipo y de acuerdo con los roles de desarrollo asignar tareas conociendo sus puntos fuertes.
Recursos	Optimizar los recursos con los que se cuenta para tener un desarrollo sin interrupciones.

Riesgo Técnico

Riesgo	**Solución**
Diseño de Interfaz	Analizar los problemas que se puedan tener y sugerir ideas que los minimicen así como tener una retroalimentación con el cliente.
Implementación y uso del sistema	Indicar al cliente cual es la infraestructura deseada para el óptimo funcionamiento del sistema y brindar capacitación al usuario sobre el uso del sistema así como dar un manual de usuario.

Validación	Se establece un mes de prueba con el sistema para validar que cumpla con las necesidades del usuario.
Mantenimiento	Fijar tiempos para el mantenimiento del sistema.

2.3 Limitaciones del sistema

En cada fase del proyecto se fueron obteniendo diversas limitaciones que el proyecto tiene al momento de concluir con su desarrollo los cuales son:

- Acceso solo como administrador general del sistema.
- Personalización de reportes.
- En algunos procesos el tiempo de término de la actividad es superada por la habilidad y/o experiencia del usuario.
- Solución temporal a la realización de cancelaciones de ventas.

2.4 Requerimientos de software para el desarrollo del sistema.

De acuerdo con la investigación realizada anteriormente, y de acuerdo con nuestras necesidades y del sistema, así como por las ventajas que ofrecen, se optó por realizar el proyecto utilizando como lenguaje principal para el desarrollo Java en su entorno Web y MariaBD para base de datos.

Para desarrollar éste proyecto se necesitaron 3 herramientas:

- Java, JDK (Java Development Kit)
- NetBeans 8.1 y
- XAMPP
- Instalación Java 8

Descargamos Java desde el siguiente enlace: https://www.java.com/es/dowload/

- Una vez hecho esto la leyenda del botón cambiará a “Aceptar e iniciar descarga gratuita” y debemos volver a dar clic.

- Para instalar el software es necesario dar clic en ejecutar el archivo descargado y a continuación nos mostrara una pantalla para comenzar la instalación. Damos clic en instalar.
- Después comenzará la instalación y sólo debemos esperar.
- Si la instalación es correcta aparecerá la siguiente pantalla y únicamente debemos dar clic en “Cerrar”.

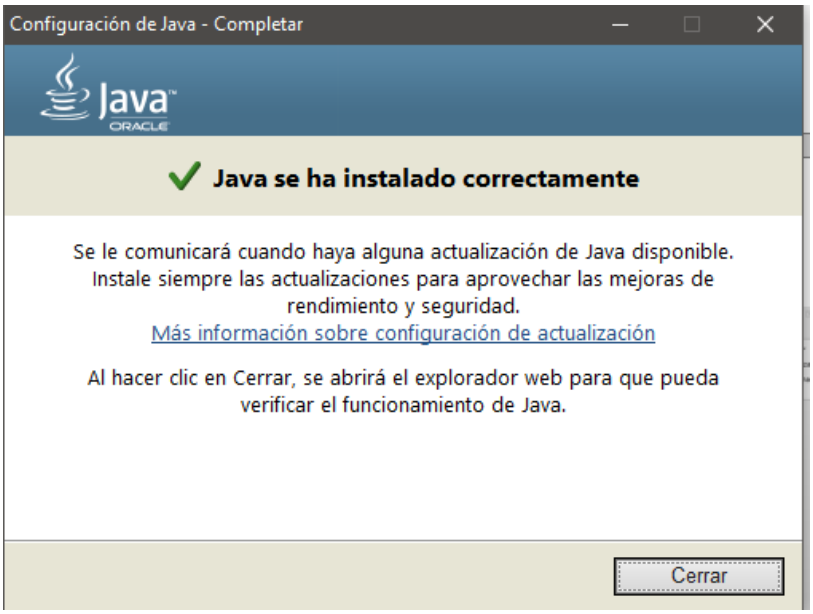


Figura 12.- Instalación java

Instalar esto nos ayudará a poder visualizar y utilizar todas las funcionalidades de java, ya sean desde aplicaciones de escritorio hasta páginas web.

- JDK

El JDK o Java Development Kit es una herramienta esencial (obligatoria) para poder programar en java. Para poder instalar el JDK debemos ingresar a la siguiente página: http://www.oracle.com/technetwork/es/java/javase/downloads/index.html

y seguir con una serie de pasos para su instalación.

- XAMPP

Es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB, el cual es el Sistema gestor de Base de Datos que usaremos.

Para poder descargar esta herramienta debemos ingresar en la siguiente dirección

https://www.apachefriends.org/es/index.html

Una vez terminada la descarga, damos clic derecho y seleccionamos ejecutar, lo cual comenzará la instalación.

Figura 13.- Instalación XAMPP.

Damos clic en siguiente y seleccionamos las herramientas que deseamos instalar.

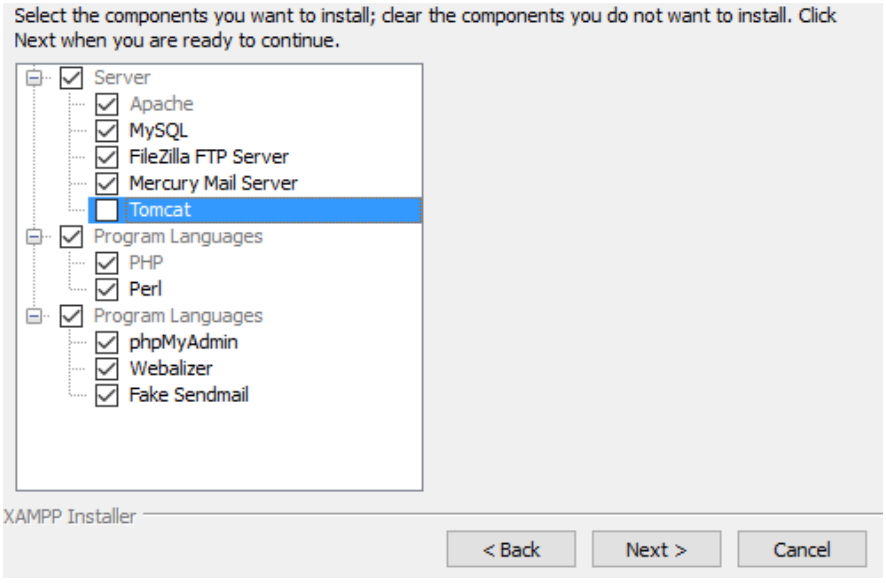


Figura 14.- Selección de herramientas

Ya instalado y después de seleccionar el idioma esta será la pantalla inicial.

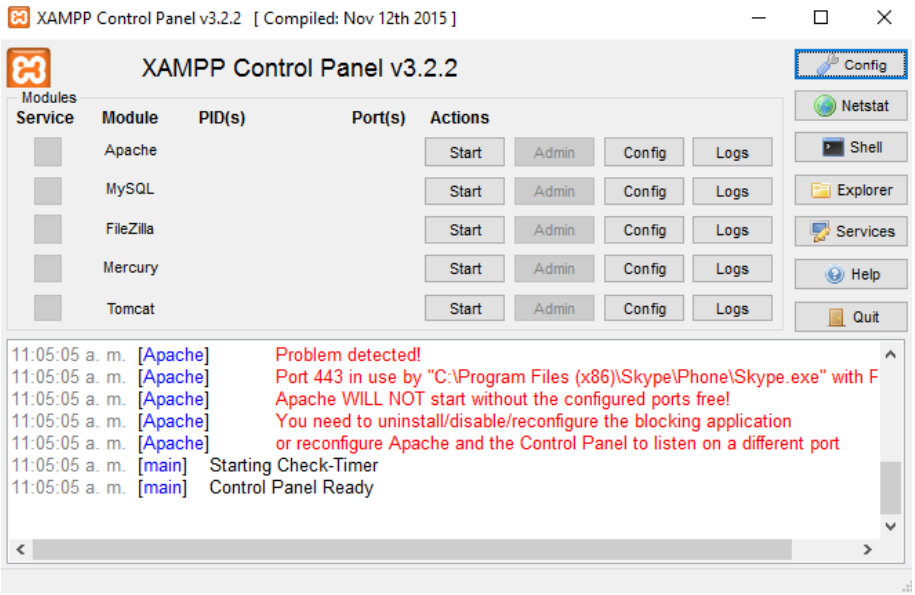


Figura 15.- Panel de control XAMPP

Damos clic en “Start MySQL”, y cambiara a color verde una vez que haya iniciado, esto se realiza para poder acceder a esta herramienta.

2.4.1 Especificaciones de software y hardware para implementación del sistema.

Para implementar el sistema en el negocio se contó con un equipo de cómputo con las siguientes características:

- Procesador Core I3
- 2GB de memoria RAM.
- Navegador web (Chrome y Mozilla Firefox).
- Acceso a internet (solo para algunas funcionalidades).
- Excel 2013
- Acrobat Reader DC

Capítulo 3: Desarrollo e implementación del sistema.

3.1 Diagramas UML

Para generar una mejor visión de cómo serán estructurados los métodos y las clases, nos apoyamos de los diagramas UML, de acuerdo con los objetivos y requerimientos del sistema se crearon los siguientes diagramas de ayuda.

3.1.1 Diagramas de caso de uso

La figura 16 es un diagrama que muestra de manera general como funcionará el sistema, indica quienes serán los actores, es decir, quienes usaran el sistema; también con que podrá interactuar el actor y que acciones hará el sistema automáticamente. Así como que procesos dependen de otros.

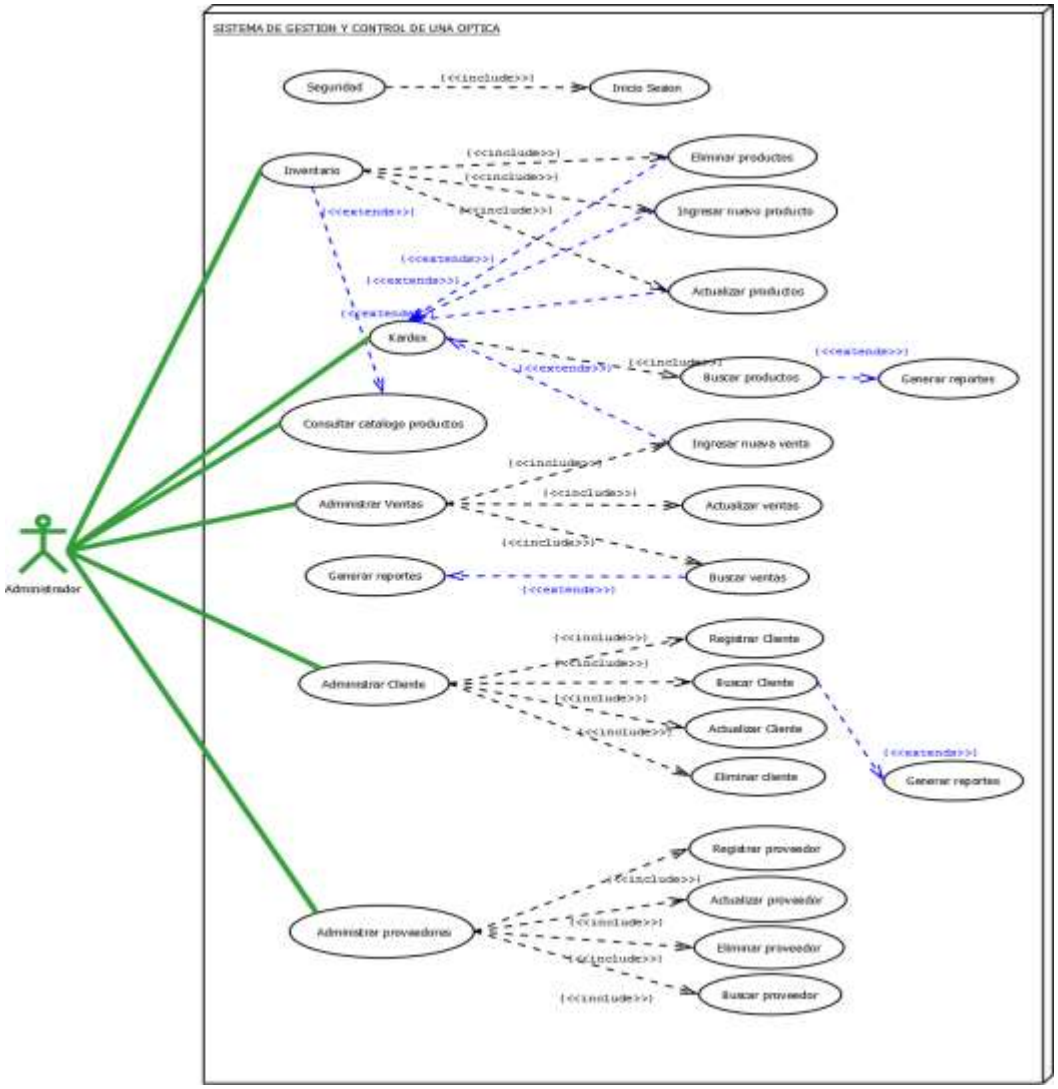


Figura 16.- Diagrama de casos de uso del usuario

3.1.2 Diagramas de actividades

Cada diagrama de actividades nos ayuda a observar cómo será el flujo de acciones realizadas tanto del usuario como del sistema automáticamente, y con esto definir que módulos serán creados.

- Diagrama de ventas:

La figura 17 que se presenta a continuación es un diagrama que muestra los pasos detallados para realizar una venta con el sistema.

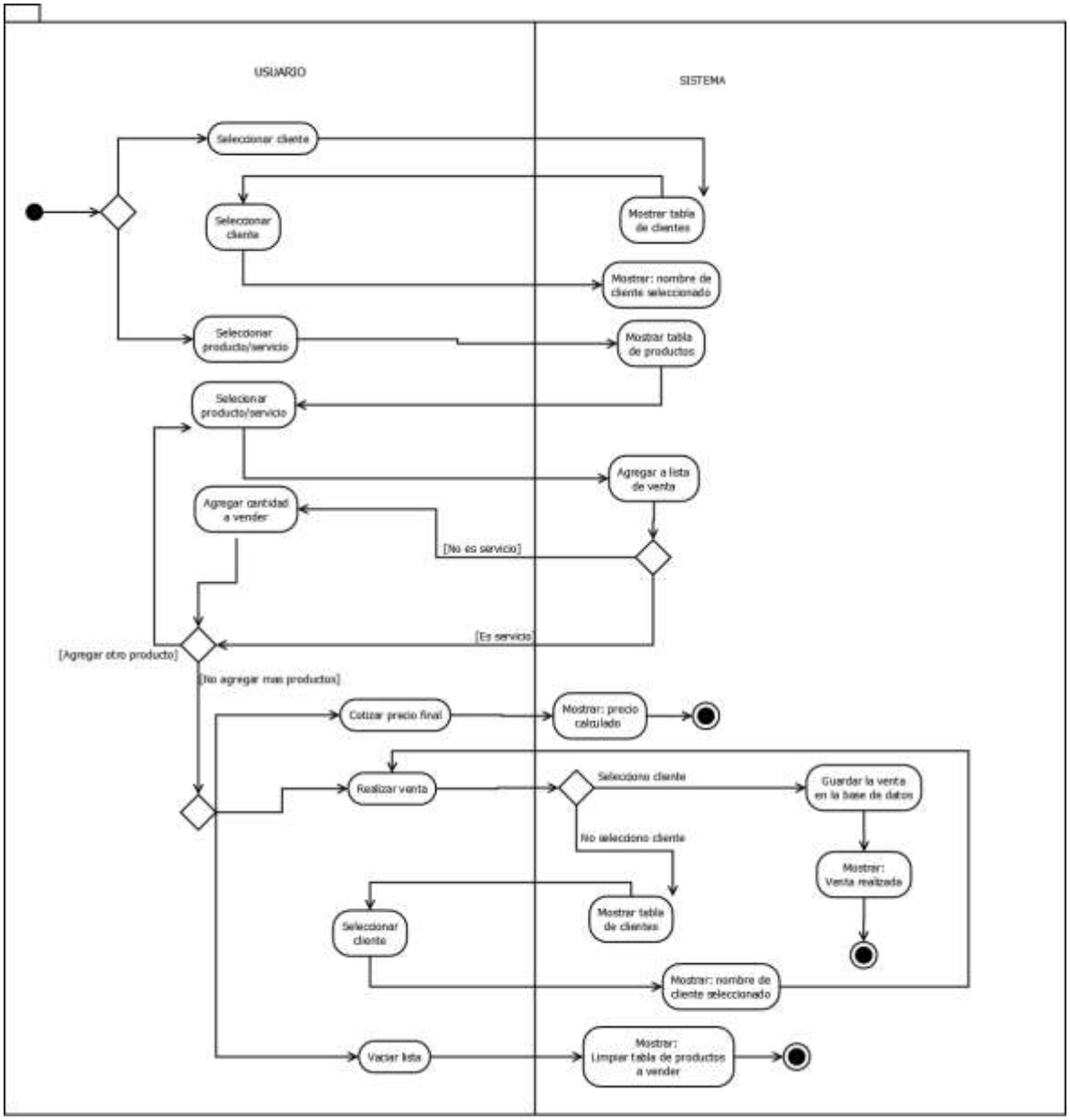


Figura 17. Diagrama de actividades: Proceso de ventas.

- Diagrama de clientes:

La figura 18 que se presenta a continuación es un diagrama donde se indica los procesos que se pueden realizar con los clientes, va desde el examen de la vista, actualizar información, etc.

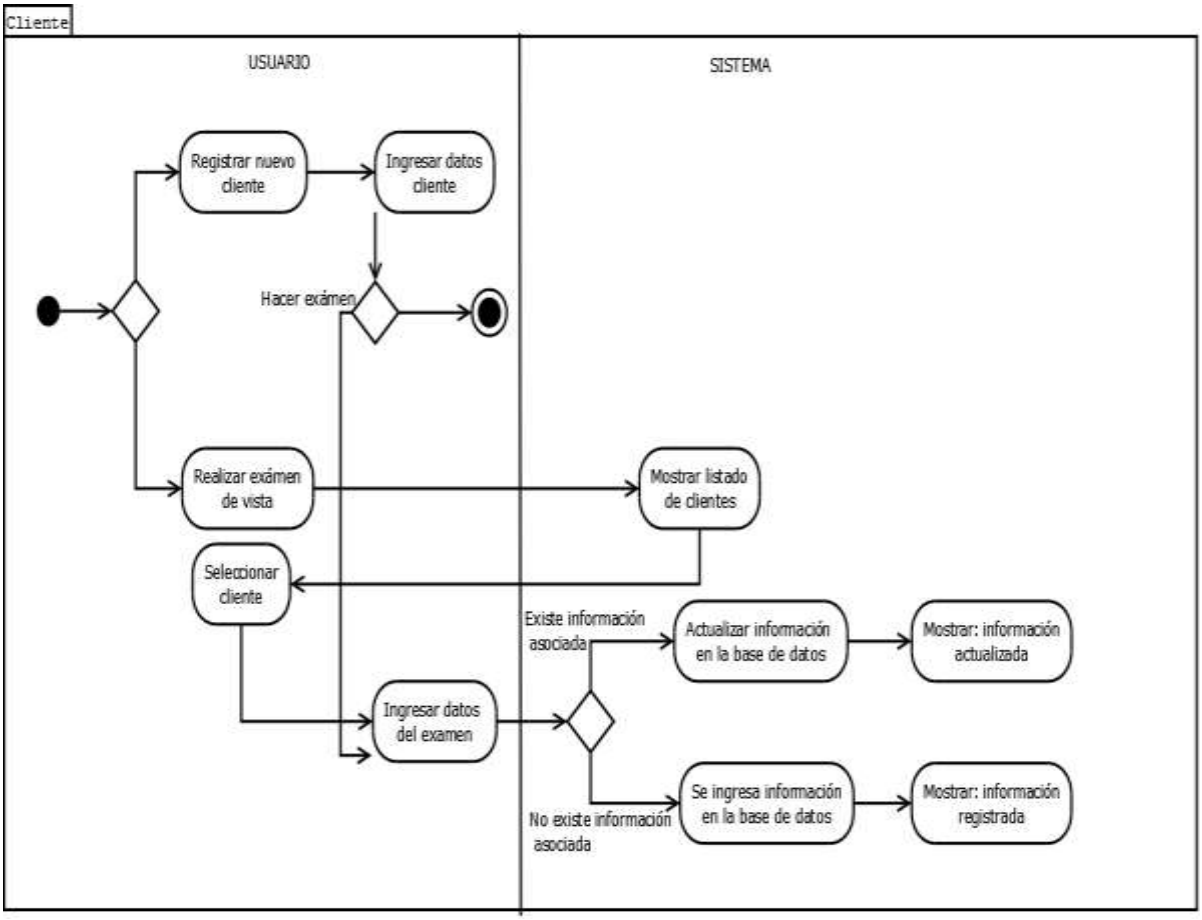


Figura 18.- Diagrama de actividades: Proceso de clientes

- Diagrama de Acceso al sistema:

La figura 19 que se presenta a continuación es un diagrama de actividades donde al ingresar los datos de acceso el usuario tiene una serie de opciones a seleccionar en un menú.

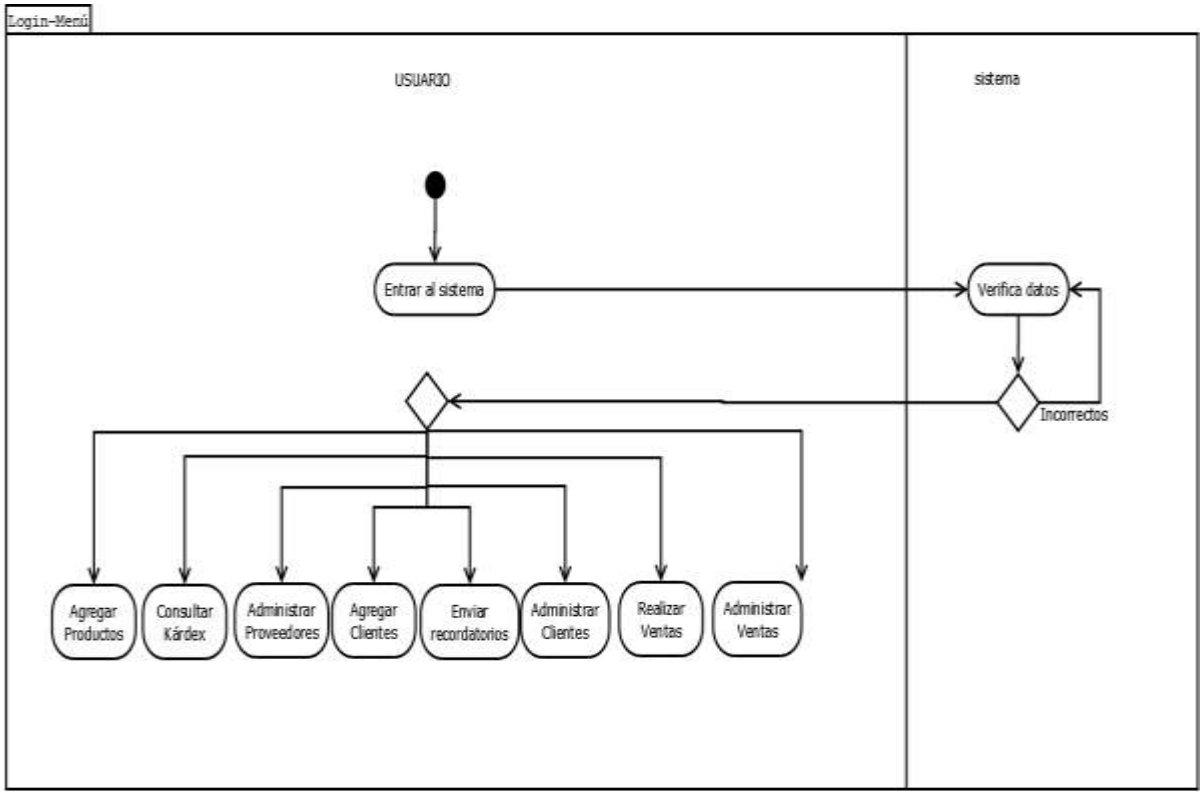


Figura 19.- Diagrama de actividades: Proceso de acceso al sistema

- Diagrama de productos:

En la figura 20 se puede apreciar la forma en la que el usuario interactúa con el sistema para manejar toda la información relacionada con los productos.

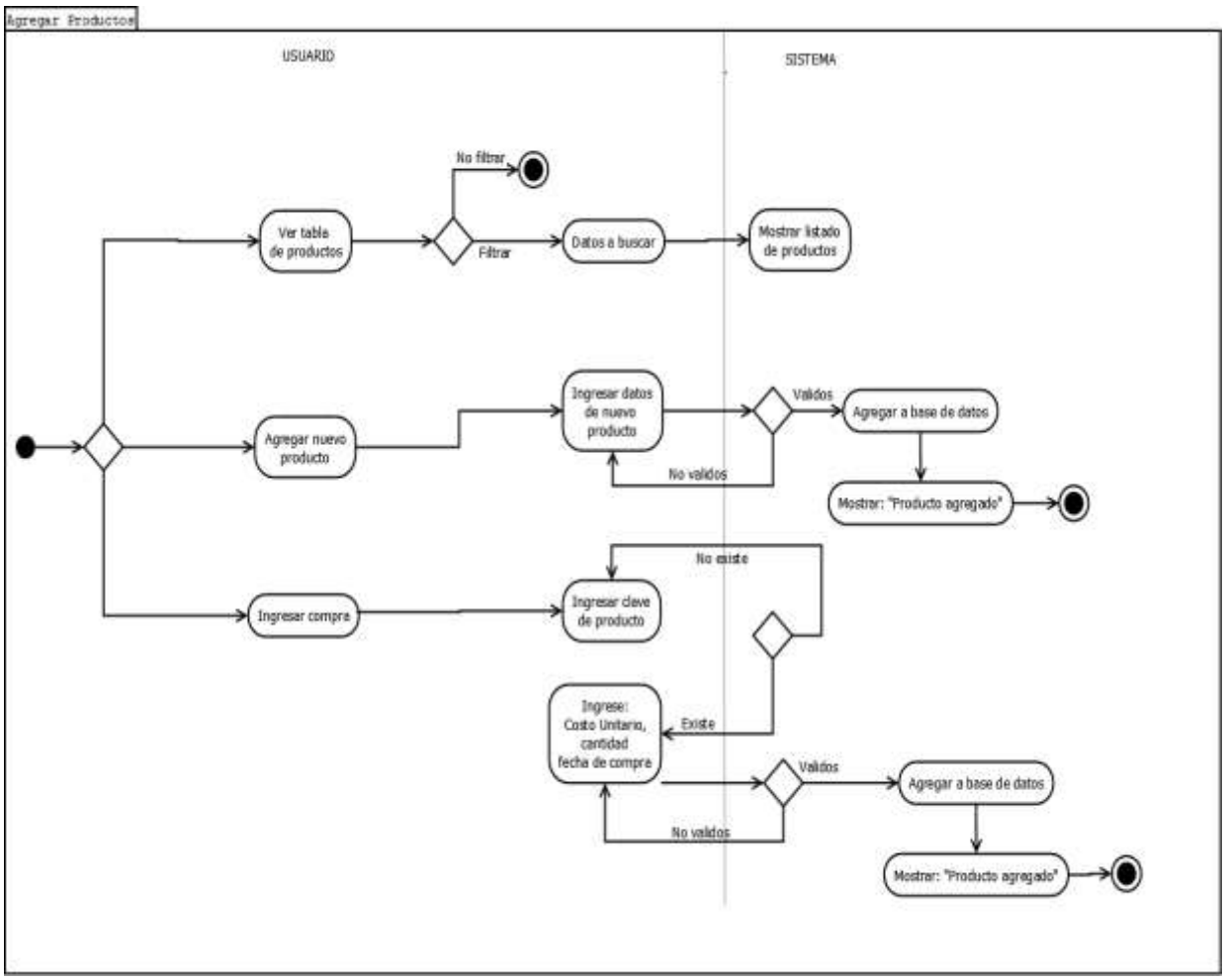


Figura 20.- Diagrama de actividade: Proceso de productos

- Diagrama de Administración de ventas:

En la figura 21 indica la forma en la que el usuario interactúa con el sistema para realizar ventas.

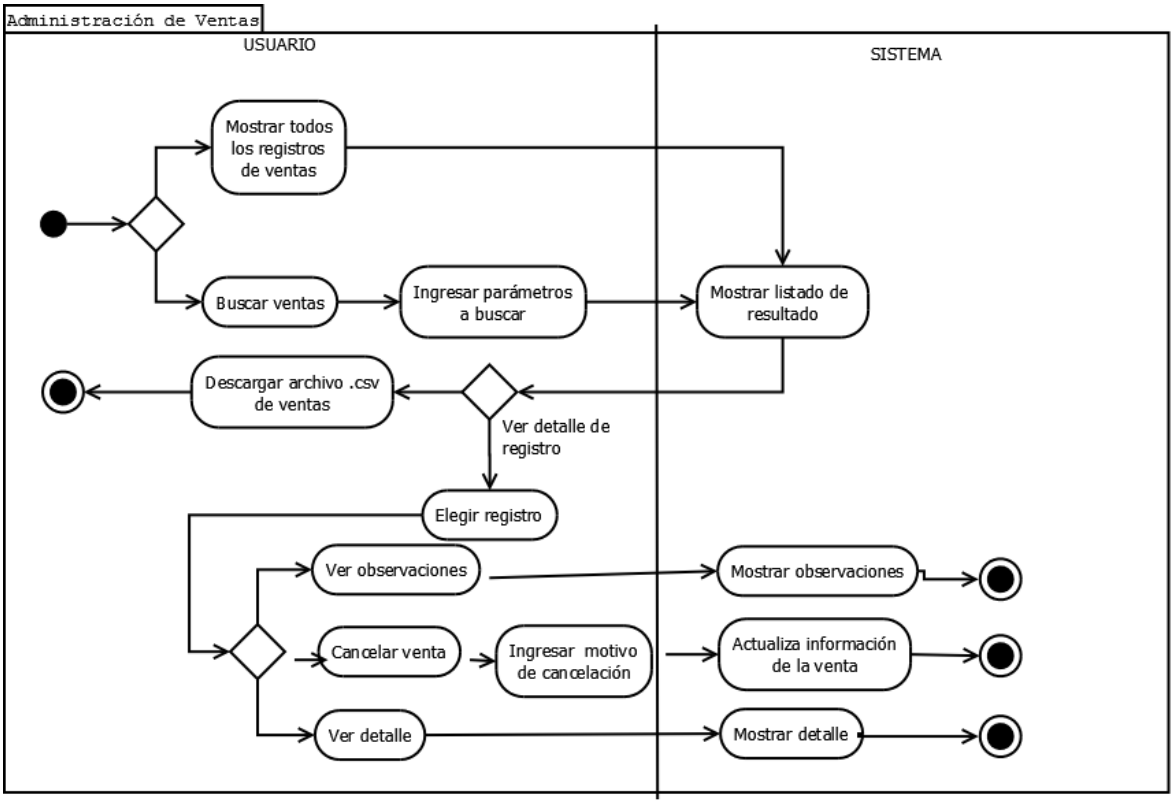


Figura 21.- Diagrama de actividades .- Proceso de ventas

- Diagrama de administración de proveedores:

En la figura 22 señala los procesos que el usuario realiza con el sistema para realizar administrar a los proveedores del negocio.

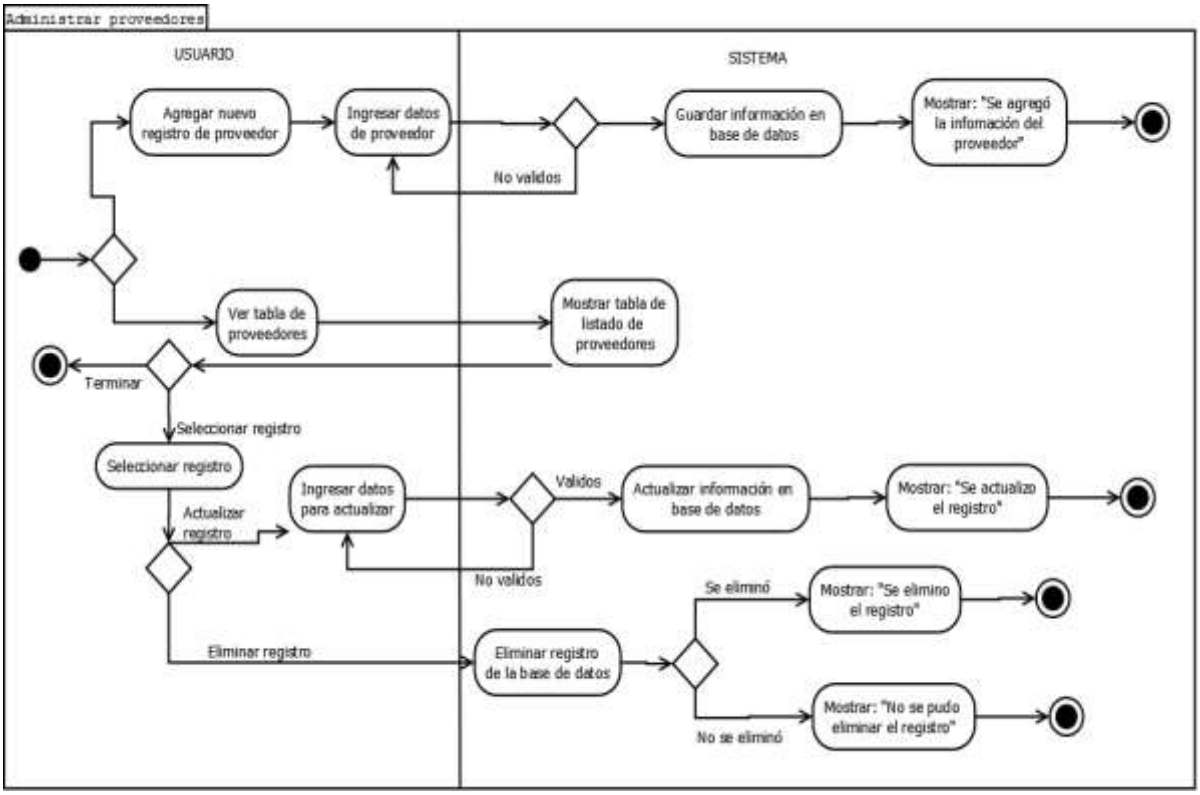


Figura 22.- Diagrama de actividades .- Proceso de administración de proveedores

- Diagrama de administración de clientes:

En la figura 23 se muestra los procesos que puede realizar el usuario con la información del cliente.

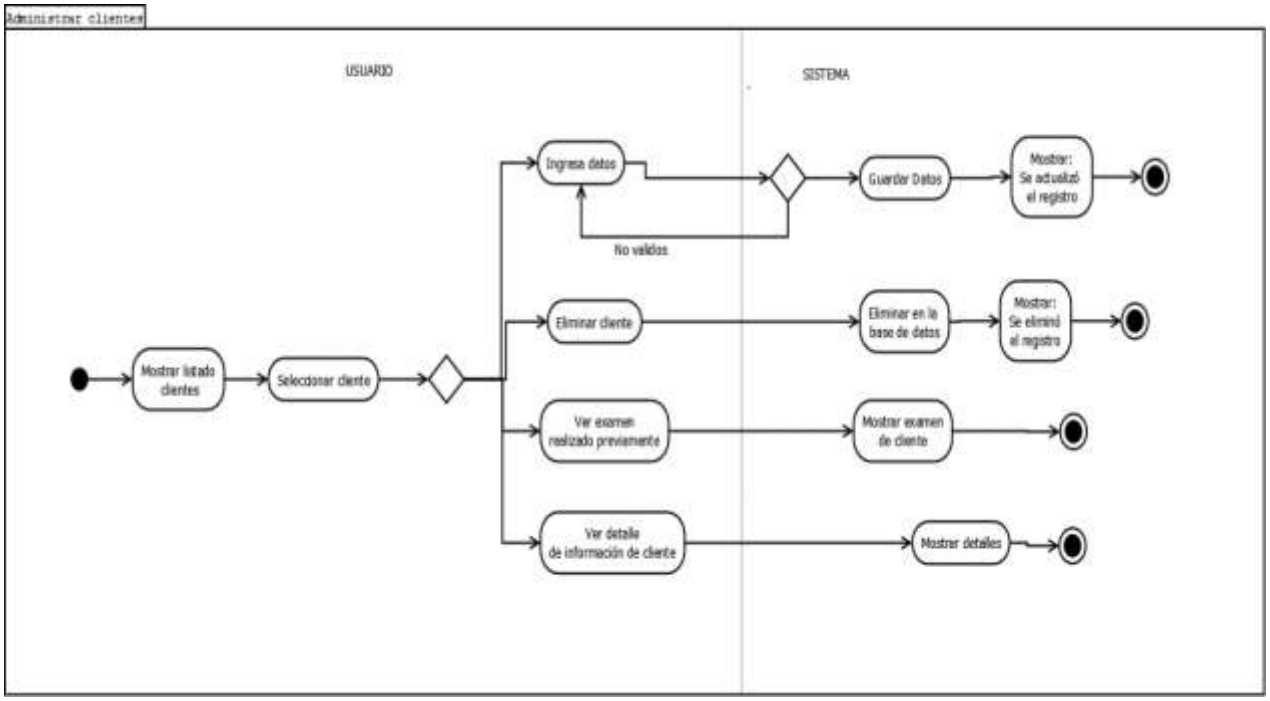


Figura 23.- Diagrama de actividades: Proceso de administración de clientes

3.2 Diseño de la base de datos relacional.

Una vez obtenido un panorama más específico de los módulos y acciones del sistema, se procede con el diseño de la base de datos, en este punto ya analizaron los datos que es necesario guardar dentro de la base de datos, por lo que, con la creación del diseño lógico, podremos entender con mayor facilidad como será e funcionamiento y las relaciones que tendrá la base de datos.

3.2.1 Diseño Lógico

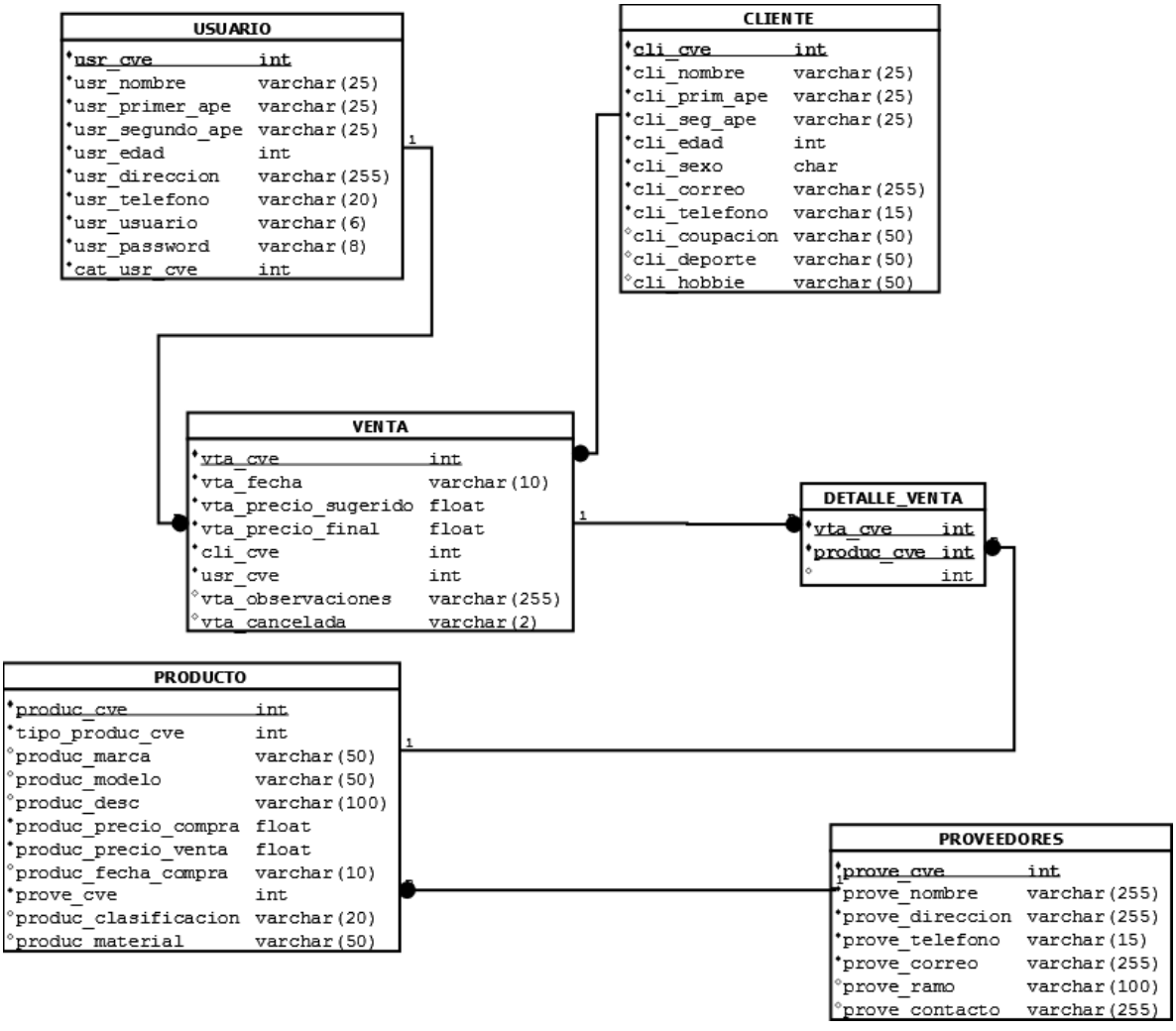


Figura 24.- Diagrama entidad relación

3.2.2 Diseño físico

```
CREATE DATABASE OPTICA;

USE OPTICA;

CREATE table USUARIO(
usr_cve int not null,
usr_nombre varchar(25) not null,
usr_primer_ape varchar(25) not null,
usr_segundo_ape varchar(25) not null,
usr_edad int not null,
usr_direccion varchar(255) not null,
usr_telefono varchar(20) not null,
usr_usuario varchar(6) not null,
usr_password varchar (8) not null,
usr_tipo int not null,
PRIMARY KEY (usr_cve),
);
CREATE table CLIENTE(
cli_cve int not null,
cli_nombre varchar(25) not null,
cli_primer_ape varchar(25) not null,
cli_segundo_ape varchar(25) not null,
cli_edad int not null,
cli_sexo char not null,
cli_correo varchar(255) not null,
cli_telefono  varchar(20) not null,
cli_ocupacion varchar(50),
cli_deporte varchar(50),
cli_hobbie varchar(50),
```

```
PRIMARY KEY (cli_cve)
);

CREATE Table PROVEEDOR(
        prove_cve int not null PRIMARY KEY,
        prove_nombre varchar(255) not null,
        prove_direccion varchar(255) not null,
        prove_telefono varchar(15) not null,
        prove_correo varchar(255),
        prove_ramo varchar(100),
        prove_contacto varchar(255)
);
CREATE Table PRODUCTO(
        produc_cve varchar(10) not null,
        produc_tipo int not null,
        produc_marca varchar(50),
        produc_modelo varchar(50),
        produc_desc varchar(100),
        produc_precio_compra float not null,
        produc_precio_venta float not null,
        produc_fecha_compra varchar(10) not null,
        produc_proveedor int not null,
        produc_material varchar(50),
        produc_clasificacion varchar(20),
        PRIMARY KEY (produc_cve),
        FOREIGN KEY (produc_proveedor) REFERENCES
PROVEEDOR(prove_cve)
);
```

```
CREATE table VENTA(
        vta_cve int not null,
        vta_fecha varchar(10),
        vta_precio_sugerido float,
        vta_precio_final float,
        vta_cliente_cve int,
        vta_usuario int,
        PRIMARY KEY (vta_cve),
        FOREIGN KEY (vta_cliente_cve) REFERENCES CLIENTE(cli_cve),
        FOREIGN KEY (vta_usuario) REFERENCES USUARIO(usr_cve)
);

CREATE Table DETALLE_VENTA(
        det_cve_vta int not null,
        det_cve_prod varchar(10) not null,
        det_cantidad int not null,
        PRIMARY KEY(det_cve_vta,det_cve_prod),
        FOREIGN KEY (det_cve_vta) REFERENCES VENTA (vta_cve),
        FOREIGN KEY (det_cve_prod) REFERENCES PRODUCTO(produc_cve)
);
```

3.2.3 Diccionario de datos

Con la creación de un diccionario de datos, se tendrá pleno conocimiento de a que se refiere cada campo dentro de la base de datos, así como el tipo de dato que espera etc., esto para ayudar en cambios futuros que se puedan realizar. Para el sistema se diseñó el diccionario de datos como se muestra en las figuras 11 – 16.

USUARIO

CAMPO	TIPO	LONGITUD	DESCRIPCION	LLAVE	NULO
usr_cve	entero	11	Codigo del usuario	PK	no
usr_nombre	cadena	25	Nombre del usuario del sistema		no
usr_primer_apellido	cadena	25	Pprimer apellido del usuario		no
usr_segund apellido	cadena	25	Segundo apellido del usuario		si
usr_edad	entero	3	Edad del usuario del sistema		si
usr_telefono	cadena	20	Telefono del usuario		no
usr_direccion	cadena	255	Direccion de usuario		no
usr_usuario	cadena	6	Usuario con el que ingresa al sistema		no
usr_password	cadena	8	Contraseña con la que ingresara al sistema		no
usr_tipo	entero	11	Clave de tipo de usuario	FK	no

Figura 25.- Diccionario de datos de Usuario

CLIENTE

CAMPO	TIPO	LONGITUD	DESCRIPCION	LLAVE	NULO
cli_cve	entero	10	clave del cliente	PK	no
cli_nombre	cadena	25	Nombre del cliente		no
cli_primer_apellido	cadena	25	Primer apellido del cliente		no
cli_segundo_apellido	cadena	25	Segundo apellido del cliente		si
cli_edad	entero	3	Edad del cliente		no
cli_sexo	carácter	1	Genro del cliente		no
cli_correo	cadena	255	Correo electronico del cliente		si
cli_telefono	cadena	20	Telefono del cliente		si
cli_ocupacion	cadena	50	Ocupacion del cliente		si
cli_deporte	cadena	50	Deporte que parctica el cliente		si
cli_hobbie	cadena	50	Hobbie del cliente		si

Figura 26.- Diccionario de datos de Cliente

PROVEEDOR					
CAMPO	TIPO	LONGITUD	DESCRIPCION	LLAVE	NULO
prove_cve	entero	10	Clave del proveedor	PK	no
prove_nombre	cadena	255	Nombre del proveedor		no
prove_direccion	cadena	255	Direccion del proveedor		no
prove_telefono	cadena	15	Telefono del proveedor		no
prove_correo	cadena	255	Correo electronico del proveedor		si
prove_ramo	cadena	100	Especialidad del proveedor		si
prove_contacto	cadena	255	Nombre del contacto de proveedor		no

Figura27.- Diccionario de datos de Proveedor

VENTA					
CAMPO	TIPO	LONGITUD	DESCRIPCION	LLAVE	NULO
vta_cve	entero	10	Clave de venta	PK	no
vta_fecha	cadena	10	Fecha de realizacion de venta		no
vta_precio_sugerido	entero con decimales	10	Precio del producto con ganancia		no
vta_precio_final	entero con decimales	10	Precio final de venta de producto		si
vta_cliente_cve	entero	10	Cliente al que se vendio el producto	FK	si
vta_usuario	entero	10	Usuario que vendio el producto	FK	no

Figura 28.- Diccionario de datos de Venta

DETALLE_VENTA					
CAMPO	TIPO	LONGITUD	DESCRIPCION	LLAVE	NULO
det_cve_vta	entero	10	Clave de detalle de venta	PK	no
det_cve_prod	cadena	10	Clave de producto a vender	Pk,FK	no

Figura 29.- Diccionario de datos de Detalle de Venta

PRODUCTO					
CAMPO	TIPO	LONGITUD	DESCRIPCION	LLAVE	NULO
produc_cve	cadena	10	Clave del producto	PK	no
tipo_produc_cve	entero	10	Clave del tipo de producto	FK	no
produc_marca	cadena	50	Marca del producto		si
produc_modelo	cadena	50	Modelo del producto		si
produc_desc	cadena	100	Descripcion breve del producto		si
produc_precio_compra	entero con decimales	10	Precio de compra de producto		no
produc_precio_venta	entero con decimales	10	Precio de venta del producto		no
produc_fecha_compra	cadena	10	Fecha en que se compro el producto		no
produc_proveedor	entero	10	Clave de proveedor al que referencia	FK	no
produc_material	cadena	50	Material del producto		si
produc_clasificacion	cadena	20	Genero para venta del producto		si

Figura 30.- Diccionario de datos de Usuario

3.3 Desarrollo de software

3.3.1 Descripción de los módulos

- Clientes:

Este módulo es dedicado al manejo de la información histórica de los clientes, se podrá registrar un nuevo cliente, actualizar la información del cliente y (en caso de no tener realizado un examen de la vista o una compra) eliminar el cliente. También se podrá consultar la información de uno o todos los clientes y ver todos los exámenes de vista realizados.

- Proveedores:

Este módulo contiene información sobre los proveedores del negocio, se podrá agregar un nuevo proveedor, editar la información del proveedor y eliminar la información; de igual manera se tendrá acceso a la consulta del listado de proveedores existente.

- Productos:

Para el módulo de productos, se podrá dar de alta un nuevo producto, agregar en cantidad productos, se podrán actualizar algunos campos de su información, tendrá acceso a consultas del histórico de comportamiento del producto.

- Ventas:

En este módulo se controlarán las ventas que se realicen, será posible cancelar alguna venta si así lo consideran, se podrán consultar las ventas por filtros específicos y generar un reporte en archivo PDF y guardarlo o imprimirlo directamente.

- Inventario:

Este módulo será solo de consultas realizadas de acuerdo a diversos filtros, y de la información obtenida se podrá generar un reporte en formato PDF o CSV.

3.4 Implementación del sistema

3.4.1 Narrativa de implementación

Para implementar el sistema se requiere:

- Equipo de cómputo con procesador Pentium en adelante, con al menos 2GB de memoria RAM.
- Navegador web.
- Contar con acceso a internet (solo para algunas funcionalidades)
- Software para visualizar archivos .csv (para visualizar los reportes)
- Software para visualizar archivos .pdf (para visualizar los reportes)

Pasos a seguir para poner en marcha el sistema:

Proporcionaremos el software del servidor de aplicaciones (glassfish versión 4.1.1), y lo copiaremos en C:\

Una vez copiado el archivo, levantaremos el servidor de la siguiente manera:

Abrir consola de windows y ubicarnos en la ruta donde está instalado:

```
C:\>glassfish4\bin>
```

Para levantar el servidor escribir:

```
C:\>glassfish4\bin>asadmin start-domain
```

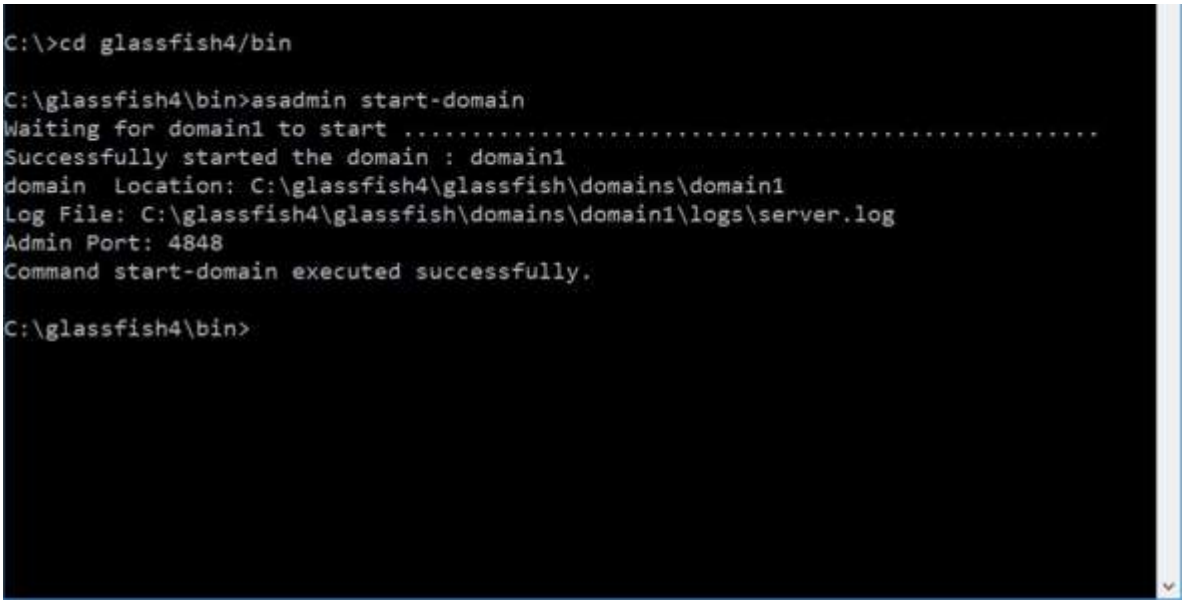


Figura 31.- Levantamiento de servidor

Ahora es necesario desplegar la aplicación web en el servidor, para ello ingresaremos a la liga: http://localhost:4848/common/index.jsf

Figura 32.- Desplegado de aplicación

Y en la parte de “Aplications” dar clic en “Deploy”, y seleccionar el archivo .war que contiene la aplicación web (el cual proporcionaremos).

Después de esto será requerido instalar

Con esto, queda listo el sistema para acceder a él.

3.4.2 Uso del sistema

1) Inicio Sesión

Al ingresar al sistema, se mostrará la siguiente pantalla.

Para ingresar al sistema se da clic al botón “Iniciar Sesión”, que se encuentra en la parte superior derecha.

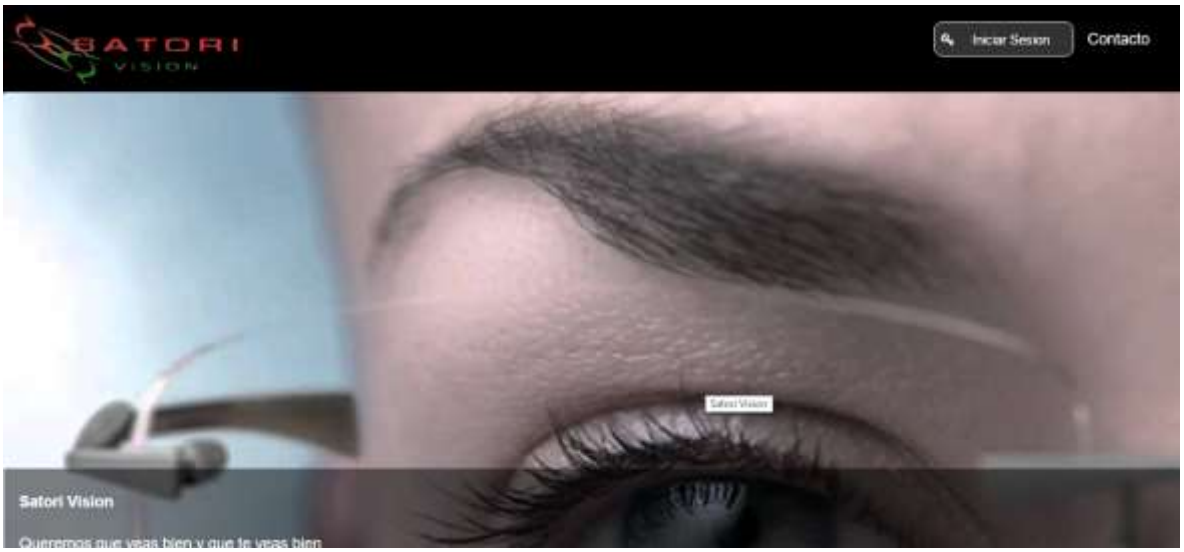


Figura 33.- Inicio de sistema

El sistema solicitará ingresar usuario y contraseña de acceso.

Figura 34.- Ingreso de credenciales

En caso de ingresar datos erróneos el sistema mostrara el siguiente mensaje:

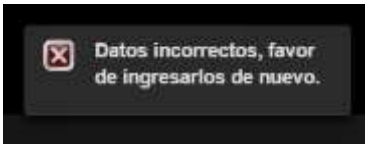


Figura 35.- Credenciales incorrectas

Si los datos de usuario existen en la base de datos se permitirá el acceso al sistema y se mostrará el siguiente menú general.

2) Menú General

Se podrá seleccionar cualquier opción dando clic sobre el botón.

Una vez ingresado a cualquier opción se podrá cambiar de opción mediante el menú que se encuentra en la parte superior de la pantalla que se muestra a continuación:

Figura 36.- Menú y área de trabajo

El área de trabajo cambiara con respecto a la opción que haya seleccionado

3) Clientes

Al seleccionar la opción “Clientes” aparecerá una pantalla como la siguiente:

Figura 37.- Alta de cliente 1

3.1 Realizar Examen

3.1.1 Registrar Cliente

Para ingresar a ella desde el menú superior: clic en “Clientes” y seleccionar “Realizar Examen”; esta pantalla también será mostrada al seleccionar “Agregar Cliente”.

Figura 38.- Submenú de clientes

3.1.2 Realizar Examen

Para realizar el examen se podrá comenzar por seleccionar un cliente al cual se hará el examen, si el cliente no está registrado deberá agregarlo en el apartado “Registrar Paciente” ingresando sus datos.

Figura 38.- Alta de cliente 2

Es necesario seleccionar un cliente, para esto dar clic en “Selecciona cliente” y el sistema mostrará una tabla de todos los clientes registrados.

Figura 39.- Selección de cliente para examen

NOTA: Seleccionar cliente se podrá hacer en el momento que sea, pero es necesario para guardar el examen

Una vez seleccionado el cliente sus datos se mostrarán como en la siguiente imagen.

1) Para guardar un examen de vista es necesario llenar la información solicitada del ojo derecho y del ojo izquierdo.

   En esta parte hay algunos campos en donde no se permite la edición que son los que están sombreados de color gris, estos datos el sistema autocompletará de acuerdo a la información que el usuario vaya ingresando
2) Para agregar la adición solo de debe dar clic en alguno de los tres botones

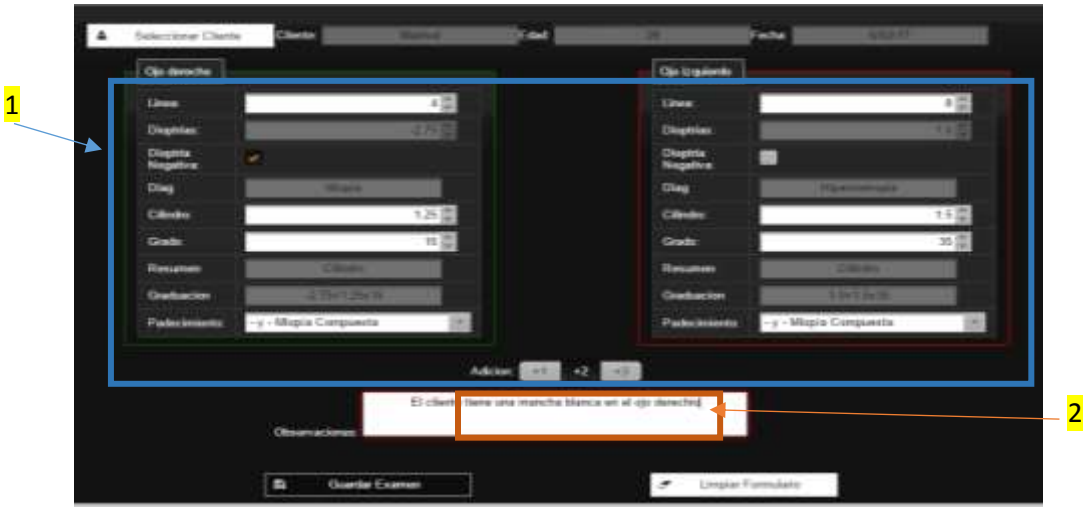


Figura 40.- Examen de la vista

Dar clic en “Guardar Examen” para guardar el registro: si los datos son válidos, el sistema mostrará el siguiente mensaje: “Examen almacenado con éxito”.

4) Administrar Clientes

Para acceder a esta opción, dar clic en “Clientes” y luego “Administrar clientes”

4.1.1Mostrar Clientes

Se mostrará una tabla con toda la información de los clientes registrados cuando se haya dado clic en “Mostrar clientes”

Figura 41.- Listado de clientes

En esta tabla se podrá filtrar la información desde la parte superior de la tabla

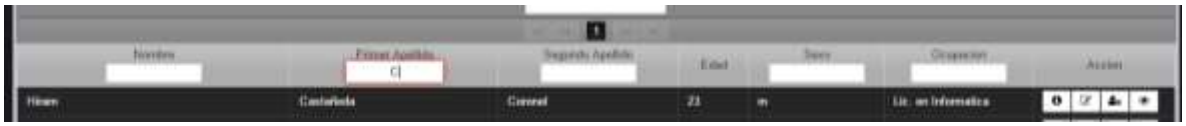


Figura 42.- Filtros de búsqueda

En la columna “Acción” se muestra un menú en el que se puede ver la información completa del cliente, eliminar el cliente, actualizar datos del cliente y ver el historial de exámenes que se le han realizado. Al dar clic en  “Ver información” se mostrara una tabla con la información:

Figura 43.- Detalles del cliente

4.1.2 Actualizar Información de Cliente

Al dar clic en  “Actualizar información” se mostrara la siguiente tabla y se editara la información que se requiera al dar clic en “Actualizar Información”.

Figura 44.- Actualización de cliente

4.1.3 Eliminar Cliente

Al dar clic en  “Eliminar cliente” se mostrara un mensaje de confirmación:

Figura 45.- Confirmación de borrado de cliente

En caso de confirmar la eliminación y si el cliente ya tiene al menos un examen de vista realizado no se podrá completar la acción y se mostrara un mensaje de error con la leyenda “El cliente cuenta con examen de la vista, no puede ser borrado”; en este caso será necesario eliminar todos los exámenes de vista que se tengan relacionados con el cliente y volver a intentar.

4.1.4 Historial de exámenes de vista del cliente

Al dar clic en | | se mostrara una tabla con la información de los exámenes que se le hayan realizado al cliente.

Figura 46.- Historial de exámenes de la vista

En caso de que el cliente no tenga por lo menos un examen de vista relacionado aparecerá un mensaje.

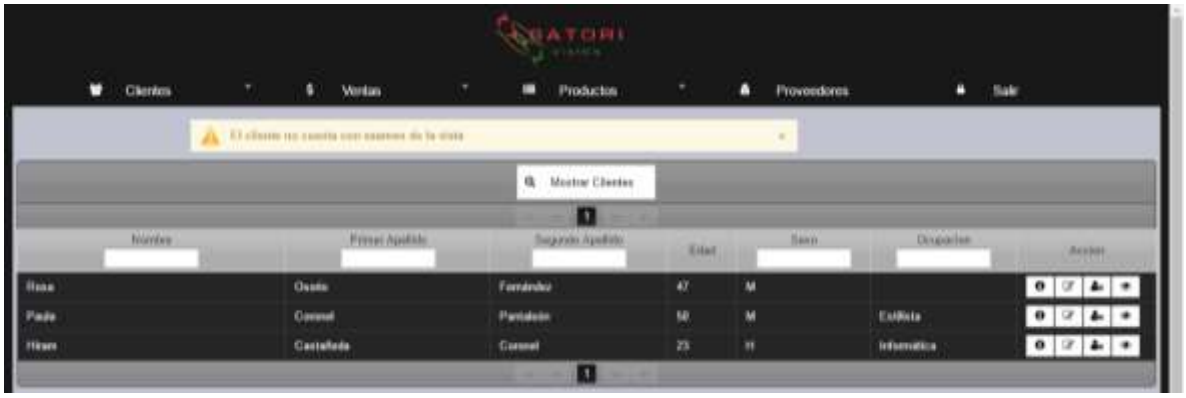


Figura 47.- Advertencia de examen de la vista

4.2 Agregar Cliente

Para registrar un nuevo cliente, en el apartado “Registrar Paciente” ingresar sus datos.

Figura 48.- Formulario de registro de cliente.

Si todos los datos son válidos se mostrará un mensaje como el siguiente:

Figura 49.- Mensaje de cliente agregado.

5) Ventas

5.1 Nueva Venta

Se mostrará la pantalla principal

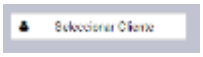
Figura 50.- Formulario de ventas.

El “No. de venta” (número de venta) se genera automáticamente de manera consecutiva por el sistema, en la fecha se mostrará la fecha actual y con esa se registrará la venta.

Figura 51.- Número de venta

5.1.1 Seleccionar cliente

NOTA: En esta parte es necesario que se elija un cliente, en caso de no estar registrado se tendrá que añadir en la opción “Agregar cliente” dentro del menú “Clientes”. Este paso lo podrá hacer antes o durante la captura de productos a vender.

Al dar clic en 

“Seleccionar cliente” el sistema mostrara una tabla con el listado de todos los clientes registrados, en la última columna se encuentra el botón para seleccionar el cliente.

Figura 52.- Selección de cliente en ventas.

Una vez seleccionado se verá el nombre del cliente en la pantalla

Figura 53.- Vista de cliente seleccionado

5.1.2 Realizar venta

5.1.2.1 Agregar productos

Para poder agregar productos a la venta se dará clic en el tipo de producto a vender deseado de las siguientes opciones: Armazones, Micas, Servicios y Otros.

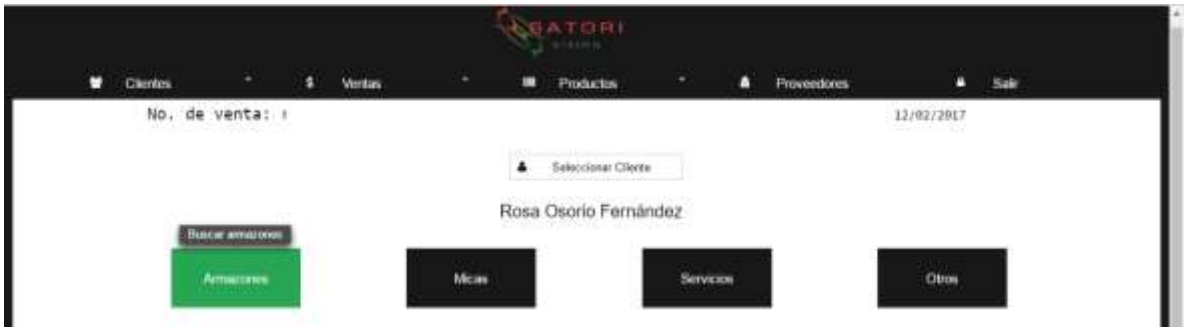


Figura 54.- Búsqueda de armazones

Para cada uno de las opciones de producto anteriores se mostrara una tabla con el listado de todos los productos del tipo seleccionado, en la cual se podrá filtrar la información desde la parte superior de la tabla (1).

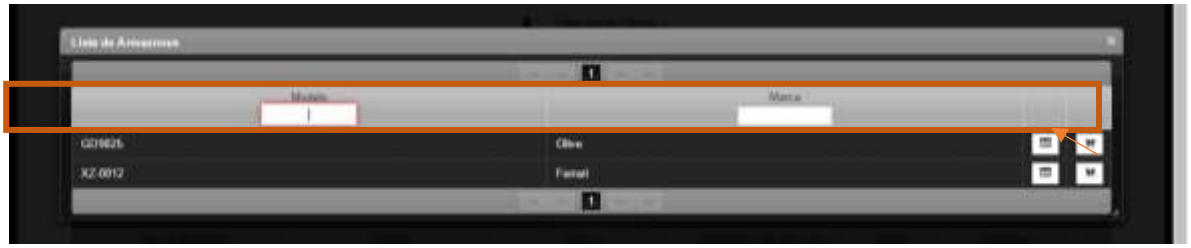


Figura 55.- Filtros de búsqueda

Para ver el detalle del producto o servicio dar clic en “Mostrar detalles”, la información se vera de la siguiente manera:

Figura 56.- Detalle de armazones.

Y para agregar el producto o servicio a la compra, dar clic en. Ya que se agregó se añade el a la tabla de los productos o servicios seleccionados titulada “Productos a vender”, tendrá que ingresar la cantidad a vender y si existe alguna observación de la venta se anotara en el campo correspondiente.

Figura 57.- Productos seleccionados.

Para quitar un producto o servicio de la venta dar clic en “Quitar de la lista”

5.1.2.2 Realizar Venta

Para completar la venta dar clic en  “Realizar venta”, se mostrara el precio sugerido y un campo para ingresar el precio en que se realizó la venta (como predeterminado estará el precio sugerido, el cual se podrá modificar)

Figura 58.- Precio de venta y confirmación.

En caso de no existir suficiente producto en inventario, el sistema mostrará un mensaje de error.

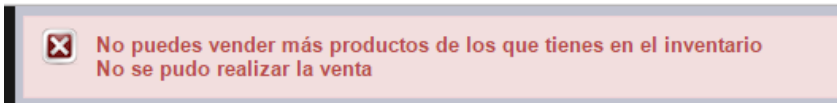


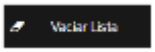
Figura 59.- Mensaje de error de validación en ventas

En caso de ingresar 0 en cantidad se mostrará un mensaje de error:

Figura 60.- Mensaje de error de validación en ventas 2

Si la venta se realizó, el sistema mostrará un mensaje de éxito.

5.1.2.3 Vaciar lista

Para cancelar una venta, dar clic en  “Vaciar listado de ventas”, esto todos los campos como estaban al inicio.

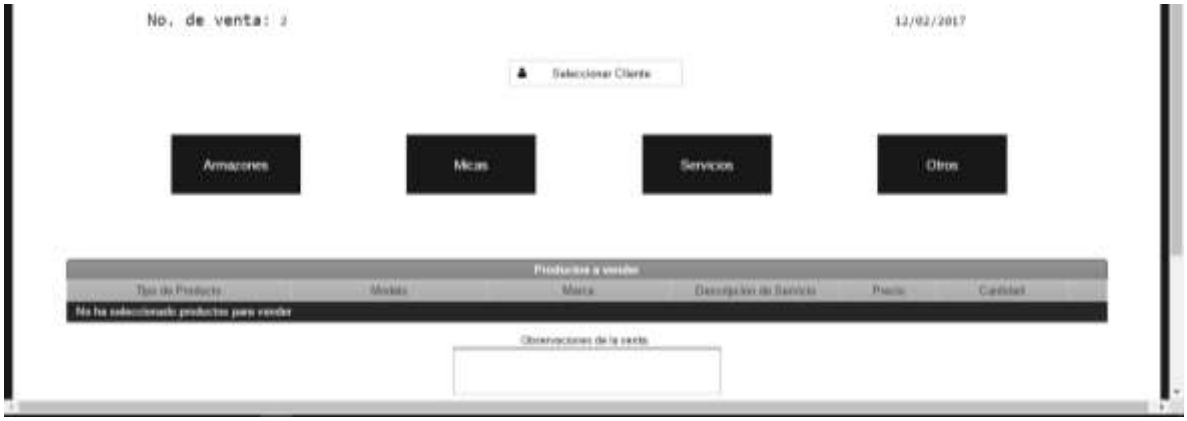


Figura 61.- Limpiar lista.

5.2 Administrar Ventas

5.2.1 Buscar ventas

Para ir a Administrar clientes dar clic en: Administrar ventas

La pantalla principal de esta opción es:

Figura 62.- Formulario búsqueda de ventas

En el apartado “Criterios de búsqueda” se podrá filtrar la información,

En la pestaña “Fecha” ingresar el periodo de fechas que requiera.

Figura 63.- Selección de fecha

En la pestaña “Cliente” dar clic en el botón “Seleccionar Cliente”, el sistema mostrara un listado de todos los clientes registrados. Para seleccionar dar clic en el botón de la última columna.

Figura 64.- Listado de clientes en reporte de ventas.

En la pestaña “Precio”, se ingresara el rango de precios que desea.

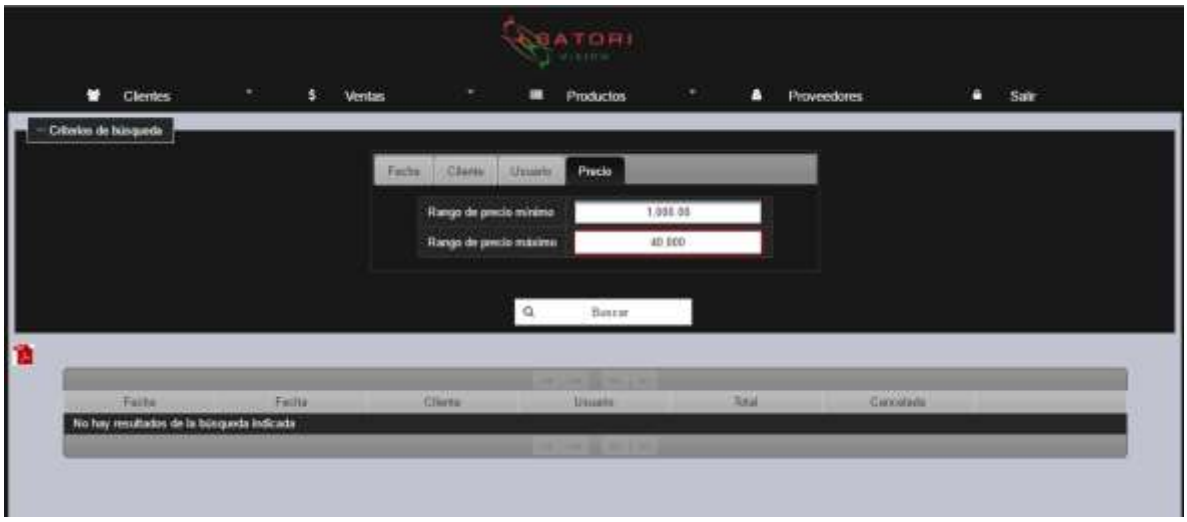


Figura 65.- Criterio de búsqueda por precio

NOTA: Para buscar se podrá ingresar como filtro uno o todos los campos mostrados anteriormente, para ver todas las ventas realizadas (sin filtros) será suficiente dar clic en “Buscar”

En los resultados de la búsqueda se podrá ver el detalle de la venta, cancelar la venta y ver las observaciones de la venta. Al posicionarse sobre los botones de la última columna se muestra una breve descripción de la acción que realiza cada uno.

Figura 66.- Resultados de búsqueda de venta.

Al dar clic en  “Ver detalle” se mostrara una tabla con los detalles de la venta.

Figura 67.- Detalle de venta

Para ver las observaciones de la venta dar clic en  y se mostrara la siguiente pantalla:

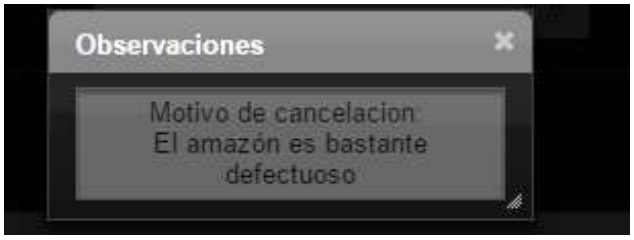


Figura 68.- Observaciones de venta.

5.2.2 Generar reportes

Para generar el reporte de los resultados obtenidos en la búsqueda, dar clic en  , el sistema mostrara una pantalla en la que se podrá guardar el archivo en formato PDF o Imprimir el documento directamente.

Figura 69.- Reporte de venta

6) Productos

En esta sección se hará la administración de los productos, registrarlos, incrementar las existencias, añadirles imagen a los productos y buscarlos.

6.1 Nuevo Producto

En esta pantalla el usuario podrá dar de alta un nuevo producto, se le tendrá que asignar un clave de producto, un tipo de producto, un proveedor de forma obligatoria, los demás campos pueden ir vacío si así lo considera el usuario. El sistema en los campos de Tipo de producto, Proveedor, Marca y modelo autocompleto con registros previos, en el caso de proveedor y tipo de producto autocompleta con registros de la base de datos. Antes de agregar un producto deberán agregar proveedores para poder relacionarlos.

Figura 70.- Formulario nuevo producto.

Figura 71.- Añadir imagen a producto.

Figura 72.- Listado de productos

6.1.1.1 Agregar Imagen

6.1.2 Registrar Nueva Compra

6.1.2.1 Registrar Nuevo tipo de producto

6.1.2.2 Agregar Imagen

Hay 2 formas de poder añadir una imagen a un producto.

Al momento de agregar un nuevo producto aparecerá la pantalla para poder agregarle una imagen

Figura 73.- Seleccionar imagen

Y en la tabla de productos al dar clic al botón  del extremo derecho de la tabla aparecerá la misma pantalla que en la imagen anterior.

Figura 74.- Botón agregar imagen.

Ya viendo esta pantalla se presiona el botón “Seleccionar” y mostrará el explorador para poder seleccionar la ubicación de la imagen que quiere darle al producto.

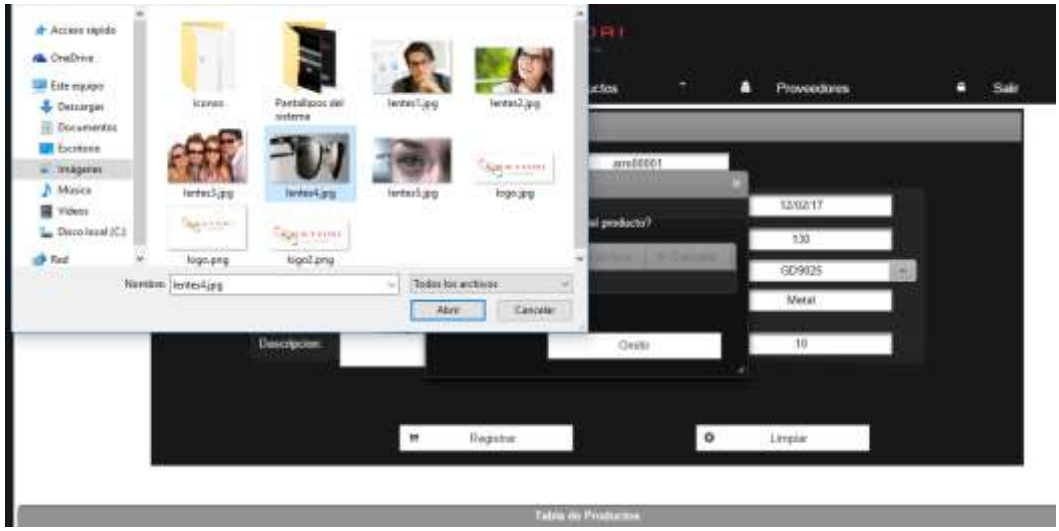


Figura 75.- Selección de archivo

Ya seleccionado el archivo aparecerá la siguiente pantalla y se deberá presionar el botón subir archivo.

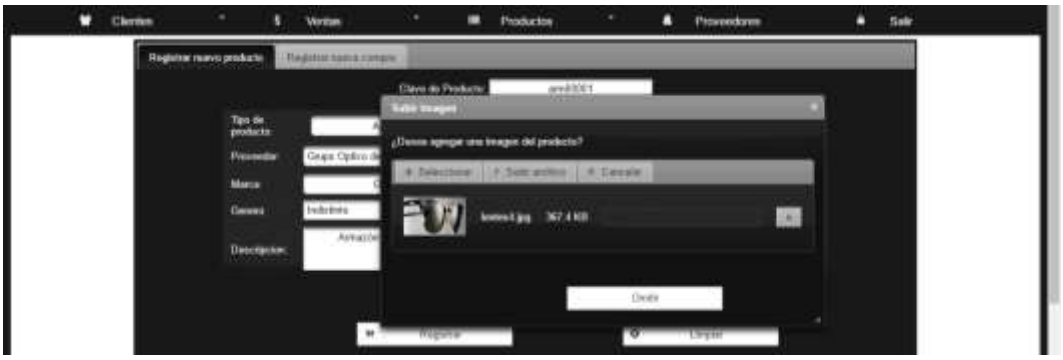


Figura 76.- Vista previa imagen.

Después el sistema avisará que la imagen se añadió correctamente.

Figura 77.- Mensaje de éxito al agregar imagen.

6.1.3 Buscar Producto

Figura 78.- Listado de productos y búsqueda.

Figura 79.- Detalle de producto.

6.2 Kárdex (Inventario)

La función de esta parte del sistema es la de informar al usuario sobre los movimientos que se van realizando en los productos, altas (adquisición de productos) o bajas (venta de productos).

6.2.1 Buscar registros

Se deberá ingresar la clave del producto del que se desea saber sus movimientos, en caso de no saberlo hay un botón con el ícono de una lupa a un costado del campo, este mostrará una tabla con la lista de productos y se podrá seleccionar un producto.

Además de la clave de producto se puede seleccionar o hacer más específica la búsqueda escogiendo fechas y mostrará la información del producto seleccionado entre las fechas indicadas.

Figura 80.- Rango de fechas Kárdex.

Y también se podrá especificar un rango de precios.

Figura 81.- Rango de precios Kárdex.

Ya seleccionado el producto e introducido otros criterios de búsqueda (el único obligatorio es la clave del producto) se desplegará una tabla con los movimientos de ese producto. Ésta tabla tiene las siguientes características:

- Información del producto

- Información de las entradas del producto
- Información de las salidas del producto
- Número de unidades adquiridas en un movimiento, su valor unitario y total
- Número de productos vendidos en un movimiento su valor unitario y total
- Y las existencias que tiene dicho producto.

Figura 82.- Información de Kárdex.

6.2.2 Generar Reportes

Se podrá descargar o imprimir el reporte del producto dándole clic en los íconos de PDF y de Excel ( ) en la parte superior de derecha de la pantalla, se descargará la información visualizada en la tabla.

Si presiona el ícono de PDF aparecerá a siguiente pantalla dando la opción de guardar el documento o de imprimirlo (estas opciones se cambian en el apartado que dice destino dando clic al botón que dice “Cambiar”).

Figura 83.- Reporte Kárdex.

Si se presiona el ícono de Excel descargará un archivo.csv que se abre con Excel o un editor de archivos como Bloc de Notas. Este archivo se encontrará en la parte de “Descargas” de Windows.

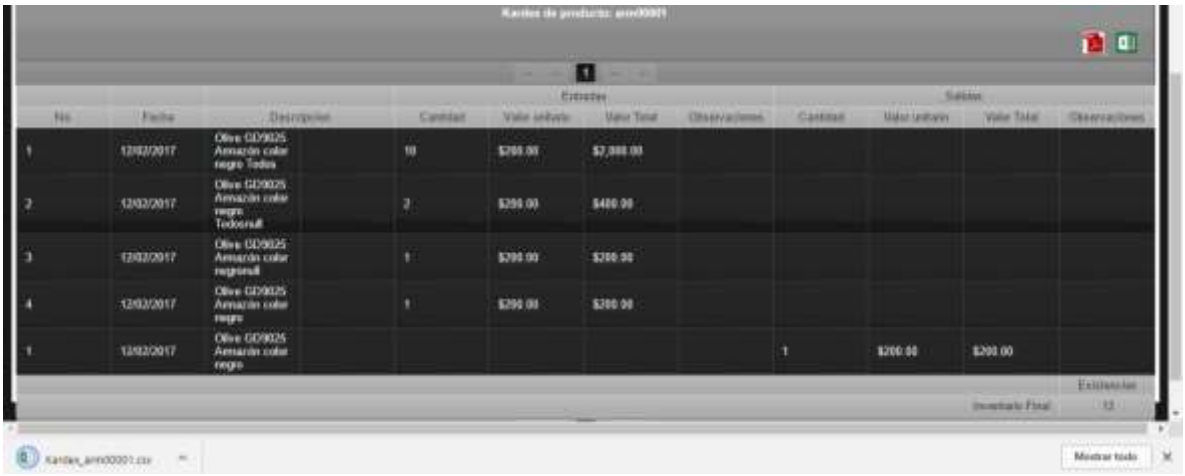


Figura 84.- Descarga de reporte de Kárdex.

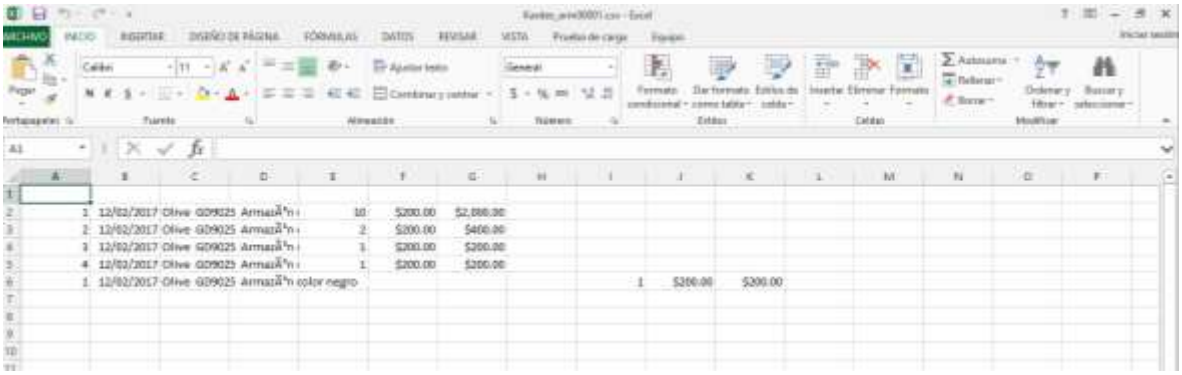


Figura 85.- Archivo de reporte de Kárdex.

7 Proveedores

En este apartado el usuario podrá agregar todos los proveedores con los que tenga acuerdos

7.1 Agregar proveedor

Figura 86.- Formulario nuevo proveedor.

En esta pantalla capturará los datos del proveedor los cuales son: El nombre del proveedor, dirección, teléfono, correo, el contacto con quién se comunicarán y el ramo o producto que éste proporcionará al negocio.

Una vez capturados estos datos se deberá dar clic en el botón “Registrar”. Posteriormente el sistema mandará un mensaje de que el proveedor fue agregado correctamente.

7.2 Mostrar Proveedores

Una vez agregado el proveedor en la parte inferior de la pantalla hay una tabla dónde se mostrará la lista de proveedores con su información, en el extremo derecho de la pantalla hay 2 botones.

Figura 87.- Listado proveedores.

Para actualizar el proveedor se debe dar clic en el boton . Este desplegará una venta con los campos de texto para poder actualizar y después apretar el botón “Actualizar Proveedor”.

Figura 88.- Actualizar proveedor.

Para eliminar el proveedor en caso de ser posible (si está ligado a algún producto el proveedor no podrá ser eliminado) se debe presionar el botón y desplegará una pantalla de confirmación.

Figura 89.- Eliminar proveedor.

8 Notificaciones

Las notificaciones es una funcionalidad del sistema el cuál se encuentra sólo en la primera pantalla después de haber ingresado al sistema.

Apretar este botón del menú mostrará una tabla con los clientes que tengan 1 año de haberse realizado su último examen de la vista y les mandará un correo informándoles que por el bienestar de su vista sería conveniente que se volviera a realizar el examen de la vista.

8.1 Enviar notificación por correo electrónico

8.2 Generar reporte

9 Salir del sistema

Para salir del sistema hay 3 formas.

- Dando clic dónde dice “Salir” en el menú que se encuentra en todas las pantallas del sistema
- Dando clic en dónde dice salir en el menú de la primera pantalla una vez ingresado al sistema.
- Cerrando el navegador.

Conclusiones

- Basándonos en la información recopilada de distintos autores y poniéndola en práctica, podemos asegurar que el sistema desarrollado ayuda a automatizar y acelerar los procesos realizados en un negocio de optometría ya que cumple con los requisitos para crear un sistema eficiente que define el paradigma de desarrollo en espiral en conjunto con el paradigma de desarrollo por prototipos.
- El sistema ayuda de gran manera a conservar, dar un mejor uso y tener un rápido acceso a la información y con esto llevar un mejor control para una toma de decisiones más efectiva que lleva a tener menor perdida financiera.
- En cuanto a los clientes, se genera un sentido de importancia ya que ahora se guarda su información, la cual se analiza junto con sus necesidades para brindarles una mejor atención y como consecuencia se tiene un negocio más competitivo.
- Se observó que para personas que cuentan con una amplia experiencia realizando exámenes de visión, resulta menos ágil usar el sistema para este propósito en comparación con sus anotaciones en papel; sin embargo, concuerdan que es de mucha utilidad tener el registro de dichos exámenes.
- Con lo anterior se puede confirmar la hipótesis planteada al inicio de este proyecto y que se logran los objetivos definidos con nuestro colaborador “Satori Vision”, quien quedó convencido de la gran utilidad que supone el sistema para su negocio.

Bibliografía

Beekman, G. (2005) *Introducción a la Informática 6ta Edición*. Madrid, España: Pearson Educación

Somerville, I. (2005) *Ingeniería del software 7ma Edición.* Madrid, España: Pearson Educación

Conolly, T.; Begg, C. (2005) *Sistemas de bases de datos.* Madrid, España: Pearson Educación

Leobardo, R. (2006) *Metodología de la programación orientada a objetos.* México, México: AlfaOmega Grupo Editor SA. De CV.

Norris, M.; Rigby, P. (1994). *Ingeniería de software explicada*.BT, Gran Bretaña: Megabyte Noriega Editores, Editorial Limusa S.A. de C.V.

Long, L.; Long N (1999). *Introducción a las computadoras y a los sistemas de información* 5ta Edición. México, México: Prentice Hall

Perdomo, C. (2009). Fundamentos lentes oftálmicos. Bogotá, Colombia: Universidad de la Salle.

Iglesias, G.; Ortiz, E. (2009). Introducción a la Informática Edición 11.5. México, México: Editorial no publicada.

Canchala, Luis (s.f). UML, *Ejemplo sencillo sobre Modelado de un Proyecto.* Recuperado de: https://msdn.microsoft.com/es-es/library/bb972214.aspx.

Kendall, K.; Kendall, J. (2005). *Análisis y Desarrollo de Sistemas 6ta edición.* México, México: Pearson Educación

Paul S. (1999) *Java. Programación orientada a objetos y aplicaciones en la World Wide Web* México, México: International Thompson Editores, S.A. de C.V.

Orós J. (2012) *Guía práctica de XHTML, JavaScript y CSS* 1ª Edición. México, México Alfaomega Grupo Editor S.A. de C.V.

Pérez, M.; Bastos B, A. I. (2010*). Introducción a la gestión de stocks:*

El proceso de control, valoración y gestión de stocks. Ideas propias Editorial S.L.

Rivas, G. (1989). Auditoría Informática. Ediciones Díaz de Santos 1a edición.

Cortés R. (1998*). Introducción al análisis de sistemas de la ingeniería de software.* EUNED, Costa Rica.

Katcheroff P. (2008) *Desarrollador .NET Curso teórico y práctico de programación,*

Creative Andina Corp.;

Luján M. *(2012) Programas de aplicaciones web: historia, principios básicos y clientes web.* Editorial Club Universitario, Alicante España.

Metodologías Agiles, s.f., https://www.ecured.cu/Metodolog%C3%ADas_Agiles

Gortázar F.; Martínez R.; Fresno V., (2012) *Lenguajes de programación y procesadores*, Editorial Universitaria Ramon Areces, España.

Arias Á, (2016) *Fundamentos de programación y bases de datos segunda edición*, IT Campus Academy.

Coronel C.,Morris S.; Rob P. (2011); *Bases de datos Diseño, Implementación y Administración,* Cenage Learning Editores S.A de C.V.,México.

Rios S.(2015),*JSF 2 + Hibernate 4 + Spring 4: PrimeFaces 5 with JAX-WS y EJB'S*,Sergio Rios

Referencias web:

Benimeli E., (2010) Bases de Datos (II). Esquema con el Modelo Entidad-Relación, Esfera TIC. Recuperado de: http://www.esferatic.com/2012/04/bases-de-datos-ii-esquema-con-el-modelo-entidad-relacion/

Anónimo, (2017), PrimeFaces, recuperado de: https://es.wikipedia.org/wiki/PrimeFaces

Hender, P., Ciclo de Vida (Conceptos), [Figura 2], Recuperado de:http://sings-ufps.blogspot.mx/search/label/Ingenier%C3%ADa%20de%20Software

Herramientas automatizadas, 2016, [figura 3], recuperado de: http://h2cm41a.blogspot.mx/2016/03/espiral.html

Anexo 1: Cuestionario para levantamiento de requerimientos.

Sistema

¿Quiénes serán los usuarios del sistema?

¿Qué problemas se espera que resuelva el sistema?

¿Qué problemas podría crear el sistema?

Usuario

¿Cómo se resuelve el problema actualmente?

¿Qué nivel de habilidades tecnológicas tienen los usuarios?

¿Genera reportes?

En caso de responder SI

¿Cuales?

¿Qué medios utiliza para generar reportes?

¿Qué reportes le gustaría tener que ahora no tiene?

¿Cuáles son los procesos y/o actividades que se realizan para lograr las metas?

Procesos y/o actividades (se responde por cada proceso y/o actividad involucrado)

¿Nombre y finalidad del proceso y/o actividad?

¿Con que actividad inicia el proceso?

¿Cómo concluye el proceso y/o actividad?

¿Qué información es necesaria para poder realizar este proceso?

¿Recibe información de otros procesos?

¿Quiénes tienen acceso a la información en este proceso?

Anexo 2: Carta de liberación de proyecto.

Fecha: 15/06/2017

Nombre del proyecto: Sistema de Gestión y Control de Optometría

Justificación o propósito del proyecto:

Con el sistema se reduce la pérdida de información y se obtiene un mejor análisis de esta, guardándola en una base de datos que permite una manipulación sencilla y eficaz. Se logra un acercamiento entre el usuario y el cliente al generar importancia en la salud visual del cliente, con ayuda de la información almacenada.

Tener un mejor control del inventario, con menos pérdidas y un mejor análisis del rendimiento de los productos y servicios que brinda el negocio

Objetivos del proyecto:

- Permitir al usuario del sistema tener un historial del cliente, de acuerdo a los exámenes visuales realizados durante sus visitas, las preferencias en productos del cliente, etc., para una mejor atención futura; ingresando la información del examen visual en el sistema para guardarla en cada visita.
- Dar al usuario un control sobre su inventario (conocer los productos existentes, así como el histórico de ventas realizadas), para una mejor toma de decisiones, esto será posible mediante la generación automática y resguardo de un Kárdex.
- Tener un catálogo digital de productos, al poder agregar una imagen a la descripción del producto.
- Sugerir un precio estimado de los productos a vender, para hacer más ágil el proceso, mediante la generación automática de dicho cálculo por el sistema.
- Poder generar diversos reportes digitales, para tener un conocimiento profundo del negocio, así como de sus clientes, éstos generados automáticamente por el sistema.

- Permitir al usuario generar importancia al cliente, mejorando la atención hacia ellos, mediante recordatorios enviados por el sistema.

Criterios de aceptación:

Cumplir con los requerimientos iniciales generando mejora en ellos.

Inversión inicial no elevada.

Pruebas de sistema con funcionalidad completa.

Descripción del proyecto:

Este proyecto automatiza los procesos que se llevan a cabo en negocios de optometría, cuenta con 5 módulos generales que son:

Productos: Para el módulo de productos, se podrá dar de alta un nuevo producto, agregar en cantidad productos, se podrán actualizar algunos campos de su información, tendrá acceso a consultas del histórico de comportamiento del producto.

Ventas: En este módulo se controlarán las ventas que se realicen, será posible cancelar alguna venta si así lo consideran, se podrán consultar las ventas por filtros específicos y generar un reporte en archivo PDF y guardarlo o imprimirlo directamente.

Productos: Para el módulo de productos, se podrá dar de alta un nuevo producto, agregar en cantidad productos, se podrán actualizar algunos campos de su información, tendrá acceso a consultas del histórico de comportamiento del producto.

Inventario: Este módulo será solo de consultas realizadas de acuerdo a diversos filtros, y de la información obtenida se podrá generar un reporte en formato PDF o CSV.

Clientes: Este módulo es dedicado al manejo de la información histórica de los clientes, se podrá registrar un nuevo cliente, actualizar la información del cliente y (en caso de no tener realizado un examen de la vista o una compra) eliminar el cliente. También se podrá consultar la información de uno o todos los clientes y ver todos los exámenes de vista realizados.

Limitaciones del proyecto:

- Acceso sólo como administrador general del sistema.
- Personalización de reportes.
- En algunos procesos el tiempo de término de la actividad es superada por la habilidad y/o experiencia del usuario.
- Solución temporal a la realización de cancelaciones de ventas.

El Lic. Héctor Castañeda, en representación de “Satori Vision” indica que quedan concluidas las pruebas del sistema y se aceptan las especificaciones del mismo para liberar el proyecto.

_____