UNIVERSIDAD CENTRAL

VICERRECTORÍA ACADÉMICA

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DESARROLLO DE UN SISTEMA INFORMÁTICO DE FACTURACIÓN Y CONTROL DE INVENTARIO PARA FARMACIAS ASOCIADAS A LA COMPAÑÍA FARMACÉUTICA S.A.

TRABAJO FINAL DE GRADUACIÓN MODALIDAD DE TESIS PARA OPTAR POR EL GRADO ACADÉMICO DE BACHILLERATO EN INGENIERÍA INFORMÁTICA

ELABORADO POR: CÉSAR AUGUSTO RUIZ FAJARDO

TUTOR: ING. MAURICIO TORRES

SEDE CENTRAL

JULIO, 2024

Índice general

Índice de figuras

Índice de tablas

Dedicatoria y agradecimiento

Le dedico este trabajo a las personas más importantes de mi vida: a mi esposa Yenory, quien es la roca que me sostiene cada vez que lo necesito. Sin tu confianza, ayuda y soporte, jamás habría podido finalizar este trabajo de investigación.

A mi madre Rosa, quien me ha brindado todo su apoyo a lo largo de la vida; tus consejos y ejemplos me han servido como guía para emprender el camino que me ha traído hasta aquí.

A mi hija Mónica, estoy orgulloso de todos tus logros y espero servirte de inspiración para que puedas forjar tu propio futuro, que sé que así será.

Agradezco al resto de mi familia por todo el apoyo incondicional, el cual me ha ayudado a superar esta etapa.

Resumen

El presente estudio se enfoca en el desarrollo de un sistema integral de control de inventario y facturación, diseñado específicamente para el sector farmacéutico. A cargo de COFASA, empresa líder en la venta, comercialización y distribución de productos farmacéuticos en Costa Rica, busca implementar esta solución informática para ofrecerla a sus clientes, las farmacias.

El proyecto tiene como objetivo principal mejorar la eficiencia operativa y la gestión de inventarios de las farmacias mediante un sistema robusto y adaptado a las necesidades del sector. El sistema no solo facilitará la administración de stocks y la generación de facturas, sino que también optimizará los procesos de venta, control de caducidades y cumplimiento normativo, mejorando así la experiencia del cliente final.

Con un enfoque en la usabilidad, seguridad y adaptabilidad a las regulaciones locales e internacionales del sector farmacéutico, el sistema promete ser una herramienta indispensable para las farmacias que desean mejorar su gestión empresarial y garantizar un servicio eficiente a sus pacientes.

Esta investigación abordará tanto el desarrollo técnico del sistema como su implementación práctica, evaluando su impacto en la gestión operativa y financiera de las farmacias que lo adopten, y destacando los beneficios potenciales para la empresa desarrolladora y sus clientes.

El desarrollo de este sistema no solo representa un avance significativo en la oferta de servicios de COFASA, sino también una oportunidad para consolidar su liderazgo en el mercado farmacéutico costarricense a través de la innovación tecnológica y la mejora continua de sus productos y servicios.

Abastract

The present study focuses on the development of a comprehensive inventory control and billing system designed specifically for the pharmaceutical sector, led by COFASA, a company, leader in the sale, *marketing* and distribution of pharmaceutical products in Costa Rica, seeks implement this IT solution to offer it to its clients, the pharmacies.

The main objective of the project is to improve the operational efficiency and inventory management of pharmacies through a robust system adapted to the needs of the sector. The system will not only facilitate stock management and invoice generation, but will also optimize sales processes, expiration control and regulatory compliance, thus improving the end customer experience.

With a focus on usability, security and adaptability to local and international regulations in the pharmaceutical sector, the system promises to be an indispensable tool for pharmacies that wish to improve their business management and guarantee efficient service to their patients.

This research will address both the technical development of the system and its practical implementation, evaluating its impact on the operational and financial management of the pharmacies that adopt it, and highlighting the potential benefits for the developing company and its clients.

The development of this system not only represents a significant advance in the service offering of COFASA, but also an opportunity to consolidate its leadership in the Costa Rican pharmaceutical market through technological innovation and continuous improvement of its products and services.

Capítulo I. Introducción

Problema

¿Podría el desarrollo de un sistema informático que maneje la facturación y control de inventarios para los clientes de la COFASA aumentar sus ingresos?

Planteamiento del problema

Hay farmacias afiliadas a COFASA que, en la actualidad, utilizan un sistema informático para el control del inventario y la facturación, proporcionado por un proveedor externo. Sin embargo, con el apoyo de la empresa, como parte de un convenio con el proveedor, este recibe los pagos de las licencias por uso, mientras COFASA se encarga de ofrecer la instalación, capacitación y soporte, sin ningún costo para el cliente.

COFASA brinda estos beneficios como parte de un servicio al cliente y una extensión de la compra de medicamentos por parte de estos.

A raíz de un cambio de objetivos y en aras de generar un acercamiento más efectivo y tangible con los clientes, se toma la decisión de prescindir del contrato con el proveedor externo y crear un sistema informático propio que pueda comercializarse en beneficio de COFASA

Objetivos

Objetivo general

Desarrollar para la COFASA un sistema informático de facturación y control de inventario para farmacias.

Objetivos específicos

1. Analizar el sistema informático actual por medio de entrevistas y/u observaciones para el análisis del sistema informático.
2. Realizar un diseño por medio de la elaboración de casos de uso utilizando notación UML.
3. Programar el sistema con la utilización de las herramientas de desarrollo de *software* propuesta.
4. Ejecutar las pruebas correspondientes como parte del control de calidad del sistema.

Justificación

Todas las farmacias, como parte de sus obligaciones tributarias, deben llevar un control del inventario y generar facturas para sus clientes. El uso de un sistema informático para esta finalidad es una práctica común que facilita la entrega de facturas a los clientes y pacientes, así como un control más ordenado y efectivo de los artículos. COFASA se dio a la tarea de ofrecer a sus clientes el uso de un sistema informático, pero con la salvedad de que este pertenecía a un proveedor externo.

Se hizo un convenio con este proveedor, que ya tenía desarrollado un sistema informático que cumplía con la mayoría de las necesidades de las farmacias. Como parte del convenio, el cobro por licencias sería para el proveedor, mientras que la compra de medicamentos sería para COFASA. Además, la instalación, capacitación y soporte de nivel 1 serían ejecutados por COFASA

Esta metodología de trabajo se mantuvo durante varios años; sin embargo, en tiempos recientes, y debido a cambios en las políticas de la empresa, se solicitó al departamento de Tecnologías de la Información (TI) tener más control sobre el *software* para las farmacias. Se

conversó con el proveedor y se le pidió la posibilidad de entregar el código fuente, a lo cual se rehusó. Al no llegar a un consenso con el proveedor, se tomó la decisión de prescindir del convenio.

Se realizó un análisis de la situación actual y se valoraron las opciones disponibles en la empresa. Se determinó que se cuenta con el conocimiento humano y técnico para emprender un desarrollo interno y obtener un sistema informático propio.

COFASA brindó el respaldo con un presupuesto para cumplir con la tarea de desarrollar un sistema informático que permita a las farmacias llevar el control del inventario y facturar a sus clientes. A las farmacias se les hará un cobro por licencias del sistema, así como un cobro mensual por soporte, que contribuirá al mantenimiento del *software*.

El desarrollo de este proyecto no solo beneficiaría a las farmacias, permitiéndoles llevar un mayor y mejor control de su inventario, su presupuesto y sus ingresos en general, sino también a COFASA, al poder ofrecer un servicio adicional y un producto desarrollado exclusivamente para farmacias. El sistema generaría ingresos adicionales que servirían para el mantenimiento del sistema informático, así como para el financiamiento de otros proyectos del departamento de TI, entre los cuales se incluiría cubrir el salario de uno o dos técnicos.

Antecedentes

A continuación, se hace referencia a varios casos en los cuales se implementaron diversas metodologías de gestión de inventario con el fin de mejorar a través de su aplicación.

Antecedentes internacionales

Un primer trabajo corresponde a un modelo de gestión de inventarios para una empresa de productos alimenticios. En el año 2013 fue publicado en la revista Ingeniería Industrial (La Habana, Cuba), el artículo denominado “Un modelo de gestión de inventarios para una empresa de productos alimenticios” (Pérez-Vergara y otros, 2013). En este modelo presentado se realiza un análisis del comportamiento de la demanda del producto, y se propone el desarrollo de un sistema de revisión de inventarios de forma periódica. Además, permite, por sus características, una mayor flexibilidad tanto en la implementación como en el seguimiento. Este trabajo se relaciona con la investigación en curso, ya que propone una revisión de inventario de forma periódica, lo cual coincide con la propuesta de investigación que se va a desarrollar. Se pueden rescatar procesos que sirvan para un mejor control de costos de los productos y ventas.

En un segundo trabajo, Ramos (2016), en su tesis titulada *Inventario de Equipos Telemáticos del Departamento de Telemática de la Policía de Puno y Sistema de Información para el Control de Equipos Informáticos - 2016*, tuvo como objetivo principal controlar e inventariar equipos y/o accesorios informáticos (Ortiz Castillo, 2020). Este estudio utiliza la Metodología Ágil Extreme Programming (XP), que permitió acelerar y mantener versiones funcionales de manera reiterada, generando resultados estructurados y permitiendo correcciones modulares directamente con los empleados, hasta que se finalizó el sistema. El aporte de esta tesis a la actual investigación es el desarrollo e implementación de un *software* para inventario y el conocimiento de métodos ágiles para la gestión de inventarios. Utilizando estas técnicas, se podrán crear interfaces más accesibles a los usuarios, así como un manejo más sencillo de los módulos generados.

En un tercer trabajo, Guzmán Ortiz (2018), en su tesis titulada *Sistema de Control de Ventas Computarizado de Cuba SRL, una Compañía de Inversiones en la Ciudad de Chimbote*, tuvo como objetivo principal implementar un *software* de control de ventas computarizado para la Compañía de Inversión Cubana (Guzmán Ortiz, 2018). El método que se empleó es el descriptivo, con una orientación horizontal no experimental. Su población está formada por 18 trabajadores, y la muestra está formada por los mismos trabajadores (18). Se aplicó la metodología RUP para el diseño de las diferentes fases, lo que permitió el desarrollo del proyecto con una mayor fluidez. Este trabajo contribuye con la investigación actual definiendo metodologías para el diseño de diversas fases del desarrollo del sistema informático, así como el tipo de lenguaje a usar y la base de datos a seleccionar, que pueda brindar una mayor seguridad a las transacciones y al manejo de la información.

Antecedentes nacionales

Este proyecto corresponde a Méndez-Araya (2010), quien realizó una tesis con el nombre de *Sistema de Gestión de Ventas e Inventario, que se desarrolló para la empresa Ticofrut S.A., específicamente para el departamento de Tecnologías de Información (TI)* (Méndez-Araya, 2010). El problema que se soluciona es la implementación de un sistema informático que permita realizar la gestión de ventas de proyectos, el control de inventario y la generación de facturas por ventas. Este proyecto de graduación define el quehacer de la empresa y del mercado al que se dirige. Se analiza el problema especificando las necesidades que se desean cubrir, los beneficiarios del nuevo sistema, así como los objetivos y alcances logrados. Este trabajo se relaciona con la investigación en curso, ya que aborda un problema análogo al advertido por la empresa COFASA e indica puntos importantes a considerar, así como actividades a seguir que muestran paralelismos con el sistema propuesto.

Un segundo trabajo nacional es el de Guzmán Gutiérrez et al. (2018), titulado *Propuesta de un sistema de control interno para el mejoramiento de las cuentas por cobrar y el ciclo de compras e inventario de la farmacia Cavalei S.A.* (Bustamante Caballeros et al., 2018). Se trata de diseñar un sistema de control interno para el mejoramiento de la gestión de las cuentas por cobrar y del ciclo de compras e inventario de la farmacia Cavalei S.A., con el fin de apoyar la toma de decisiones para el logro de los objetivos de la empresa. Se logra identificar la necesidad de contar con un adecuado sistema de control interno que permita examinar los riesgos a los que se enfrenta la empresa y tomar decisiones para el logro de los objetivos. Este tema cobra importancia para la investigación actual porque, aunque no se va a abarcar cuentas por cobrar, está íntimamente relacionado con una farmacia e incluye el ciclo de compras e inventario, que también se tomará en cuenta al momento de desarrollar el proyecto. Se muestra cómo debe estructurarse el ciclo de compras desde una perspectiva de procesamiento de la información.

Este proyecto corresponde a Alpízar Rodríguez y García Bustamante (2019), quienes realizaron el trabajo titulado *Propuesta de Plan de Mercado para la Cadena de Farmacias Don Gerardo,* en el cual se plantea el contexto de las farmacias comunitarias en Costa Rica, así como las perspectivas teóricas referentes al plan de mercadeo, de forma que se puedan conocer los conceptos teóricos más relevantes (Alpízar Rodríguez y García Bustamante, 2019). Se presenta una propuesta de plan de mercadeo para farmacias Don Gerardo, que involucra las “7 Ps” del *marketing* de servicios, agregando a la mezcla tradicional tres elementos adicionales para la administración de la relación con los clientes. Los procesos se refieren a la manera en que una organización hace las cosas, el diseño del entorno físico en que los clientes reciben el servicio brindado, y el personal de servicio que interactúa directamente con los clientes. Estas metodologías de *marketing* influyen de manera considerable al momento de generar el plan de

implantación que se plantea en la investigación, ya que se refiere al mercado de farmacias, el mismo que abarca la investigación propuesta, y su enfoque puede considerarse como referencia al momento de crear las estrategias comerciales.

Proyecciones

En tiempos en que los ingresos y/o beneficios por servicios toman relevancia para todas las empresas, públicas o privadas, de cara a márgenes de intermediación cada día más bajos, este proyecto trae la posibilidad de incrementar el volumen transaccional que genera ingresos de facturación por parte de las farmacias hacia COFASA; además, permite una reducción en el costo operativo, cuya implementación en cuanto a infraestructura es mínima, ya que el desarrollo se haría internamente por parte del departamento de TI, contando con el personal capacitado para realizarlo.

Otro aspecto importante es que COFASA innovaría con este servicio para sus clientes y socios, lo que le permitiría una mayor presencia y compromiso hacia ellos en las distintas zonas del país.

Se abre una oportunidad de negocio, ampliando los servicios a las farmacias asociadas y a los posibles nuevos socios comerciales para el procesamiento de transacciones electrónicas, obteniendo un beneficio directo tanto la COFASA como los clientes que adquieran el servicio.

Las farmacias asociadas estarían aprovechando el uso de un sistema informático más adecuado al mercado farmacéutico, que les ofrecería información más actualizada para la toma de decisiones y un mejor manejo del inventario, así como proyecciones de ventas y compras de mercadería.

A continuación, se exponen varios aspectos que fortalecen esta iniciativa.

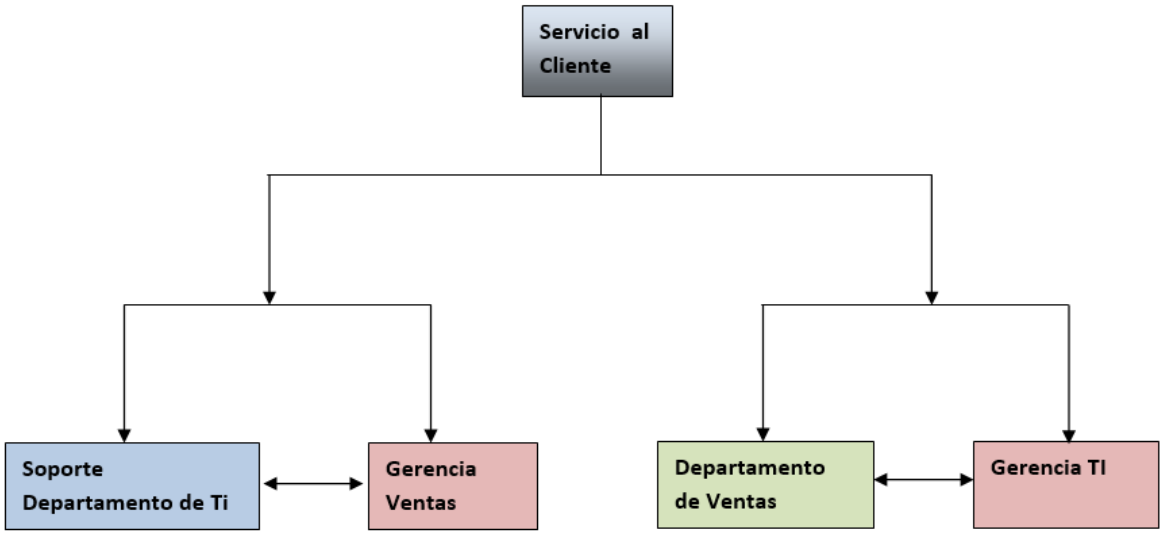
Antecedentes comerciales

1. Permite una mayor expansión y presencia comercial de COFASA en distintas zonas del país.
2. Crea más y mejores relaciones comerciales y financieras a corto, mediano y largo plazo con las farmacias afiliadas a COFASA
3. Les permite a las farmacias un mejor control del inventario y un punto de venta más dinámico, lo que conlleva una mayor y mejor atención de sus clientes.

Aspectos operativos

Figura 1

Estructura de servicio al cliente



Fuente:

Como es posible apreciar en la Figura 1 *Estructura de servicio al cliente* compañía Farmacéutica S.A. ya posee una estructura de servicio al cliente, el cual permitiría administrar este nuevo servicio.

Aspectos tecnológicos

1. El departamento de TI de COFASA dispone de un centro de cómputo principal y uno alterno, el cual existe como contingencia ante eventuales fallas, lo que le permitiría a la empresa la continuidad del negocio en casos de desastres naturales.
2. Se cuenta con personal calificado para el análisis, desarrollo e implementación del *software* para las farmacias.
3. El *hardware* necesario para el uso del *software* ya se encuentra en las farmacias que poseen un punto de venta. Si no lo tienen, se les brindará una lista de proveedores de *hardware* donde conseguirlo de forma rápida y económica.

Alcances

1. Con base en el objetivo n.° 1, se realizará una investigación que genere un informe donde se indiquen los requerimientos funcionales y no funcionales del sistema informático.
2. Con base en el objetivo n.° 2, se hará un análisis del sistema actual y se entregará un reporte que indique si es factible proseguir o desecharlo.
3. Con base en el objetivo n.° 3, se desarrollará el sistema propuesto.
4. Con base en el objetivo n.° 4, se realizarán las pruebas al sistema creado para garantizar su desempeño y su uso por parte del usuario final.

Limitaciones

1. El sistema se desarrollará en COFASA
2. No se desarrollará en esta etapa un manual de usuario final.
3. En esta etapa no se impartirá una capacitación.
4. Habrá acceso limitado a la información financiera de la empresa.
5. No hay presupuesto para la compra de licencias.
6. No existe presupuesto para la capacitación del personal.
7. Se requiere que las farmacias que quieran adquirir el sistema informático estén asociadas a COFASA
8. Se utilizará el personal interno para el desarrollo.
9. La aplicación solo abarcará el control de inventario y de facturación.
10. La investigación solo abarcará el período comprendido entre 2021 y 2022023.

Capítulo II. Marco teórico

Marco referencia de la institución

Reseña histórica

Según Suárez (2016), la empresa nació en el año 1940, frente a la antigua Embajada Americana, como iniciativa de un grupo de boticarios que se unieron para formar una cooperativa farmacéutica que les facilitara las importaciones, además de lograr una mejor economía para el grupo. En 1948, al darse un descontrol en el manejo de las mercaderías y los créditos, se contrató al Sr. Jorge Vanderlat, quien debió reestructurar la empresa y fundar Compañía Farmacéutica Ltda., ya que las leyes de la época no permitían llamarla cooperativa. Debido al crecimiento de la compañía, decidieron comprar una casa grande en avenida segunda entre calles 11 y 13 y se procedió, además de distribuir, a fabricar varios productos, tales como agua oxigenada, valeriana, entre otros. Alrededor de los años 60 y con el crecimiento de la producción, se decidió comprar un terreno en Guadalupe para instalar únicamente la producción en Laboratorio Cofala S.A., hoy LACOFA (Laboratorio Compañía Farmacéutica LC S.A.). Adicionalmente, se alquiló un local en Barrio La California para ubicar la Distribuidora COFASA y poder construir el edificio situado en avenida segunda entre las calles 11 y 13 en San José. En 1970, se trasladó al nuevo edificio en avenida segunda, lo que permitió un mayor servicio de la Distribuidora COFASA a todos sus clientes.

Grupo COFASA

Compañía Farmacéutica S.A. es una empresa que se dedica a la distribución de productos farmacéuticos y de cuidado personal. Comenzó operaciones hace más de 70 años. Compañía Farmacéutica S.A. está compuesta por tres pilares: la Distribuidora COFASA, que es la segunda

distribuidora en el país, con una participación superior al 14% del mercado; el LACOFA, que produce más de 110 fórmulas comerciales; y la Cadena de Farmacia COFASA con más de 300 afiliadas.

La Gerencia General manifestó que el desarrollo de la empresa inició con la representación de pocas casas farmacéuticas y, con el pasar del tiempo, se unieron otros laboratorios importantes. Gracias a ello, se manejan en la actualidad más de cuatro mil productos.

> Nuestro objetivo principal como empresa es lograr un crecimiento integral que incluya a los laboratorios representados, con el cumplimiento de sus metas; a los clientes, con condiciones competitivas en las que puedan trasladar parte de sus beneficios al consumidor final; a los accionistas, a través de utilidades que en su mayoría se capitalizan para reinvertirlas en la empresa; y a los colaboradores con salarios competitivos en un clima laboral agradable (fuente).

En los últimos 15 años, la compañía ha experimentado un crecimiento en las ventas muy superior al promedio del mercado, según datos de IMS (año), compañía encargada de tabular estadísticas del sector farmacéutico, consolidando cada vez más su potencial de desarrollo. De esta forma, muchos laboratorios han visto en Compañía Farmacéutica S.A. la mejor opción de distribución, porque el crecimiento no solo se ve en las ventas, sino también en los productos. Los crecimientos en las utilidades han sido superiores a los preestablecidos en el plan anual operativo.

La estrategia de mercado utilizada por Compañía Farmacéutica S.A. es obtener una sólida estructura comercial a través de beneficios equitativos para las farmacias pequeñas, medianas y grandes del país; entre estos se encuentran los descuentos comerciales y financieros, así como los servicios de valor agregado como COFACTIVA, en el que los clientes afiliados a la cadena de farmacias pueden aprovechar los descuentos adicionales que solo esta figura ofrece.

Los servicios que brinda COFASA han sido muy bien aceptados por sus clientes, con entregas en cualquier parte del territorio nacional, garantizando exactitud y plazos cada vez menores. Con los proveedores, se mantiene un intercambio constante de información en línea que permite reaccionar ante cambios del entorno. Además, se anticipan posibles exigencias con la aplicación de las buenas prácticas de almacenamiento y distribución de medicamentos en droguería, tales como la trazabilidad de productos por vencimiento, la garantía de condiciones ambientales como temperatura, humedad relativa y cadena de frío; la custodia de productos especiales como psicotrópicos y la estandarización de procesos que reducen la curva de aprendizaje del personal involucrado.

En el campo de responsabilidad social corporativa, COFASA maneja un programa de salud ocupacional, posee una planta de tratamientos de aguas residuales, realiza campañas de reciclaje y se ajusta a los lineamientos dictados por el Ministerio de Salud en lo vinculado con la recolección de productos vencidos.

Misión

Fabricar y distribuir productos farmacéuticos de uso humano y de cuidado personal de alta calidad, que satisfagan las necesidades de sus clientes y generen rentabilidad atractiva a sus

accionistas, así como estabilidad a sus trabajadores; fomentando el bienestar en el sector salud de la sociedad (fuente).

Valores

Servicio, Excelencia, Responsabilidad, Visión, Igualdad, Respeto con Integridad, consolidan día a día al Grupo COFASA como la empresa número uno en productos y servicios farmacéuticos de Costa Rica (fuente).

Infraestructura

La plataforma de distribución garantiza un servicio de entrega de pedidos en un plazo de una hora en el centro de San José, dos horas en el área metropolitana y un máximo de 24 horas en el resto del país, justo a tiempo.

En el área de ventas, existe una jefatura, un coordinador de área, 10 vendedores, 11 funcionarios de facturación y una encargada de servicio al cliente. Además, contamos con una central telefónica exclusiva para pedidos.

Los pedidos entran directamente a los funcionarios de teleoperadores, quienes los ingresan en el sistema de facturación de la compañía. Se generan los alistos de mercadería y se imprime la factura. Esta se recibe en el departamento de alistos, donde se seleccionan los productos; luego se revisa para garantizar que el pedido esté completo. Posteriormente, se hace la entrega de los artículos al cliente por medio de un vehículo de entrega o vía transportista, cuando el cliente se ubica fuera del área metropolitana.

Se cuenta con herramientas logísticas enfocadas en buenas prácticas y tecnologías de vanguardia, con sistemas como el WMS (*Warehouse Management System*), un sistema inteligente de gestión del almacén que cubre todos los procesos de ingreso y salida de mercadería de forma automática. Además, se tiene el TMS (*Transportation Management System*), que conecta en línea las áreas de transporte, ventas y cobro, y una plataforma web robusta para nuevos servicios.

El servicio al cliente es una herramienta vital, como lo ha sido la Cadena de Farmacias COFASA, donde el cliente puede adquirir mercadería desde una unidad con descuentos sostenibles superiores a los que ofrece el mercado.

El apoyo al recurso humano se considera una de las principales herramientas que han garantizado el éxito de la empresa; por ello, se invierte en capacitación y mejora continua tanto en mandos medios como operativos y ejecutivos, así como en otorgar un clima organizacional óptimo para el buen desempeño de las labores.

Distribución

En esta área se cuenta con una empresa outsourcing que brinda el servicio de entrega y distribución de los pedidos, principalmente en la Gran Área Metropolitana, Heredia, Alajuela, Cartago, en carretera y servicio de emergencia. La capacidad de entrega es la siguiente:

Tabla 1

Tiempo de entrega de pedidos

Ubicación	Tiempo
San José	2 horas
Alajuela	El mismo día
Cartago	El mismo día
Grecia	El mismo día
Heredia	El mismo día
Naranjo	El mismo día
Palmares	El mismo día
San Ramón	El mismo día
Sarchí	El mismo día

Cobertura

Todo el territorio nacional se encuentra cubierto para las entregas, con las siguientes frecuencias de visitas.

Tabla 2

Frecuencia de visitas

Ruta	Periocidad
San José	Todos los dias
Alajuela, Heredia, Cartago	Tres veces por semana
Carretera (Naranjo, Sarchi, Grecia, Palamres, San Ramón)	Una vez por semana
Zona Pacifica	Una vez por semana
San Carlos	Una vez por semana

Tiempo de entregas

La plataforma de distribución garantiza un servicio de entrega de pedidos desde dos horas en el centro de San José y Área Metropolitana, y máximo 24 horas en el resto del país, justo a tiempo. Además, se utiliza el transporte de encomiendas a todo el país, COFASA cubre este rubro como parte de su servicio a socios y clientes.

A continuación, se presenta un organigrama del departamento de TI.

Figura 2

Organigrama del departamento de TI

Fuente:

Breve historia de la programación

Figura 3

Historia de la programación

Fuente: (Díaz Garcés, 2023)

Según la página Consult (2023):

> Los lenguajes de programación son la herramienta principal con la que los programadores pueden crear soluciones informáticas. A lo largo de la historia, se han desarrollado diversos lenguajes de programación que han evolucionado para adaptarse a las necesidades de los programadores y las empresas.

En la década de 1970 se desarrollaron los primeros lenguajes de programación de bajo nivel, como el lenguaje de máquina y el lenguaje ensamblador. Su aprendizaje era muy difícil y se limitaba a un grupo de expertos.

En los años 50 llegaron los lenguajes de alto nivel, tales como Fortran y Cobol. Estos se enfocaban en sistemas científicos y financieros y se desarrollaron para simplificar la programación. Su aprendizaje era más sencillo y abarcó a un mayor número de personas.

Luego, en la década de 1960, aparecieron Pascal y C, que permitían mayor flexibilidad y facilitaban la programación al tener una mejor estructura y ser modulares. Esto ayudaba a la claridad y eficiencia de los programas que se creaban.

En la década de 1980 se crearon otros lenguajes de programación, como C++ y Java. La programación podía ser más modular y se reutilizaba el código fuente, lo que mejoró la eficiencia y la productividad de los programadores. En 1990 se desarrollaron lenguajes web como HTML, JavaScript y PHP, que permitieron crear aplicaciones web y la interacción con el usuario a través de estas. En la actualidad, hay una evolución constante en la programación, adaptándose a las necesidades de la inteligencia artificial, el internet de las cosas y la informática en la nube.

¿Qué es un lenguaje de programación?

Un lenguaje de programación es una serie de instrucciones que se le da a una computadora, las cuales son ejecutadas por esta misma, brindando al final un resultado o una salida.

¿Qué es la programación orientada a objetos?

La Programación Orientada a Objetos (también conocida como POO u OOP, por sus siglas en inglés) es un paradigma de programación que ofrece una nueva manera de obtener resultados.

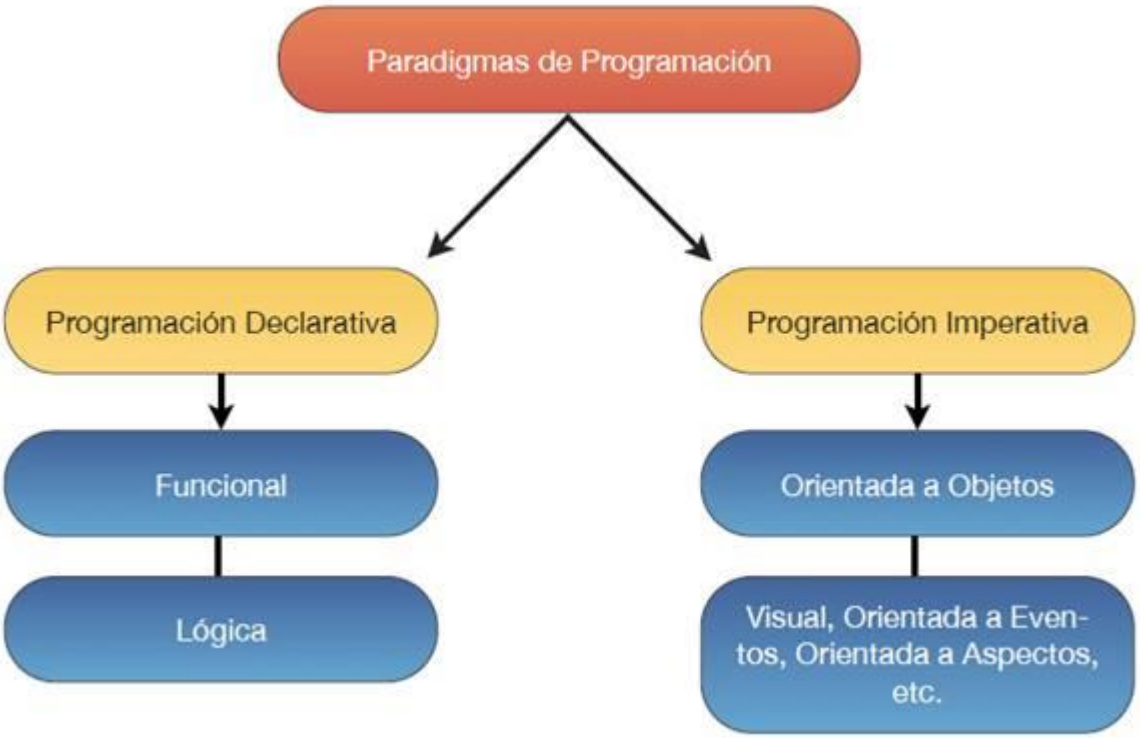
Paradigmas de la programación

¿Qué es un paradigma de programación?

De acuerdo con Roch Moraguez (2023) : “Los paradigmas de programación son el fundamento del desarrollo de *software* moderno. En esencia, son enfoques diferentes para programar.”

Figura 4

Paradigmas de programación



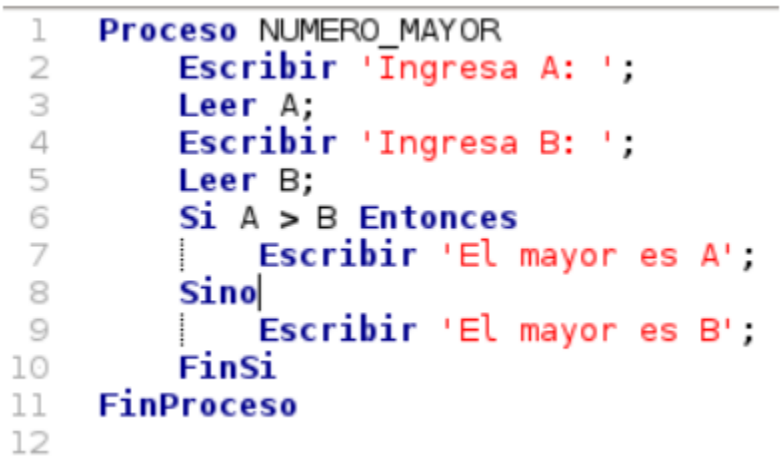
Fuente: **(Aprender a programar PRO, 2020)**

Programación declarativa

El principio fundamental de la programación declarativa radica en la descripción del resultado final que se busca, sin detallar el procedimiento para obtenerlo.

Figura 5

Ejemplo de lógica de programación



```
1   Proceso NUMERO_MAYOR
2       Escribir 'Ingresa A: ';
3       Leer A;
4       Escribir 'Ingresa B: ';
5       Leer B;
6       Si A > B Entonces
7           Escribir 'El mayor es A';
8       Sino
9           Escribir 'El mayor es B';
10      FinSi
11  FinProceso
12
```

Fuente: (Peraza, 2016)

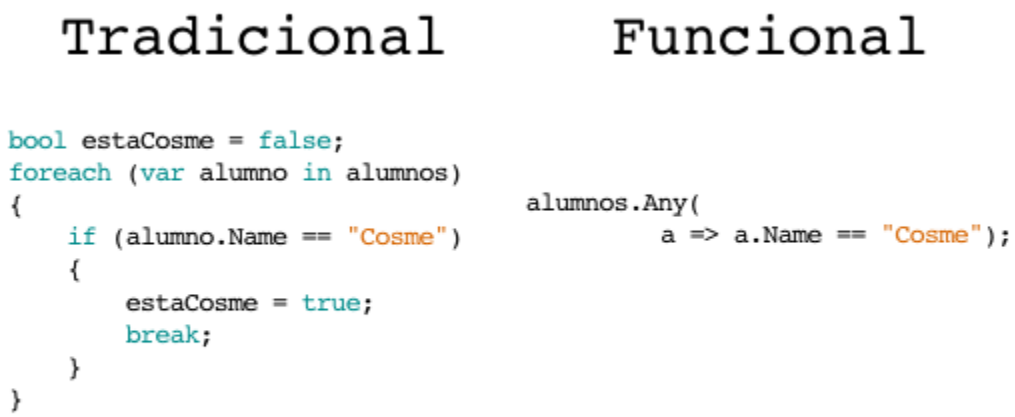
Programación lógica

Es un enfoque de programación que se basa en la lógica matemática. El programa se compone de reglas y hechos que se utilizan para inferir conclusiones lógicas. Esta programación se basa en dos principios fundamentales: la deducción y el no determinismo.

La deducción es la idea de que las conclusiones lógicas se pueden deducir a partir de las reglas y los hechos. El no determinismo es la idea de que el sistema de inferencia puede elegir entre varias posibilidades para llegar a la conclusión. Para poder utilizarlo, se requiere una alta capacidad de inferencia y razonamiento.

Figura 6

Ejemplo de programación funcional



Fuente: (Feregrino, 2017)

Programación funcional

Se basa en la idea de que el programa se compone de funciones que se aplican a los datos. Las funciones se tratan como valores y se pueden pasar como argumentos a otras funciones. Existen tres principios fundamentales: inmutabilidad, funciones puras y funciones de orden superior.

La inmutabilidad se refiere a la idea de que los datos no deben cambiar una vez que se han creado. Las funciones puras son aquellas que no tienen efectos secundarios y siempre producen el mismo resultado para los mismos argumentos. Las funciones de orden superior son funciones que toman otras funciones como argumentos.

Figura 7

Programación imperativa

Fuente:

Programación imperativa

Es el enfoque de programación más común. Los programas se componen de un conjunto de sentencias que cambian su estado. Son secuencias de comandos que ordenan acciones a la computadora. Se usa en aplicaciones de *software* que requieren un alto grado de control y precisión en la manipulación de datos.

Figura 8

Programación orientada a objetos

Fuente:

Programación Orientada a Objetos (POO)

Se trata de una técnica que aumenta considerablemente la sencillez del código. Se crean estructuras llamadas clases que permiten agrupar funcionalidades acotadas y bien definidas. Se suele adquirir mayor velocidad en el desarrollo de los programas gracias a los grados de reutilización del código. Lo más importante es que permite una mayor organización, lo cual facilita el desarrollo de programas más complejos.

Ventajas de la programación orientada a objetos

Según Programacion Pro (2023) las ventajas de la POO son:

> Reutilización de Código: El código y las clases creadas anteriormente se pueden reutilizar. Permite aprovechar el trabajo previo al crear nuevas clases y objetos basados en los existentes. Esto ahorra tiempo y esfuerzo, ya que no es necesario escribir el código desde cero cada vez que se necesita una funcionalidad similar. Además, facilita las actualizaciones y correcciones, ya que cualquier cambio realizado en una clase se reflejará en todas las instancias de esa clase.
>
> Modularidad y organización: Fomenta el modularidad, lo que significa que el código se divide en módulos independientes y bien definidos. Cada clase encapsula sus propios datos y comportamientos, lo que facilita su comprensión y mantenimiento. Además, la organización jerárquica de las clases en paquetes y módulos permite una estructura clara y coherente del proyecto, lo que facilita la colaboración entre los miembros del equipo de desarrollo.
>
> Abstracción y encapsulación: La abstracción es una característica central de estas herramientas, que permite representar conceptos complejos del mundo real en forma de objetos. Esto permite simplificar el diseño y la implementación del código, ya que solo se consideran los aspectos relevantes para el problema en cuestión. Por otro lado, la encapsulación asegura que los detalles internos de una clase estén ocultos al mundo exterior. Solo se exponen los métodos y propiedades necesarios para interactuar con el objeto, lo que proporciona una interfaz clara y reduce la complejidad.

Herencia y polimorfismo: La herencia es una característica poderosa de la POO que permite crear nuevas clases basadas en clases existentes. Esto facilita la creación de jerarquías de clases y promueve la reutilización de código de una manera estructurada. Además, el polimorfismo permite que un objeto pueda tomar diferentes formas o comportarse de diferentes maneras según el contexto. Esto ofrece flexibilidad y extensibilidad en el diseño y permite escribir código más genérico y flexible.

Mantenimiento y escalabilidad: Facilita el mantenimiento del código a largo plazo. Debido a su modularidad y organización, es más fácil identificar y solucionar problemas, así como realizar actualizaciones y mejoras. Además, la reutilización de código y la capacidad de extender clases existentes hacen que el código sea más escalable, ya que se pueden agregar nuevas funcionalidades sin modificar el código existente.

Descripción técnica

Plataforma

El proyecto se desarrollará con el paradigma de la Programación Orientada a Objetos, y el mecanismo de modelado del sistema POS utilizará la plataforma Microsoft Visual Studio, dentro de la cual se seleccionó el lenguaje de programación Visual Basic .NET, que es parte de esta herramienta, para el desarrollo del sistema de punto de venta, todo enmarcado en la versión Community (Microsoft, 2024).

Se pretende generar el proceso de análisis del sistema con POO (Programación Orientada a Objetos) y VB.NET, para que el proyecto pueda ser continuado por cualquier informático que maneje esos estándares.

Figura 9

Microsoft Visual Studio

Fuente: (MVP Cluster, 2017)

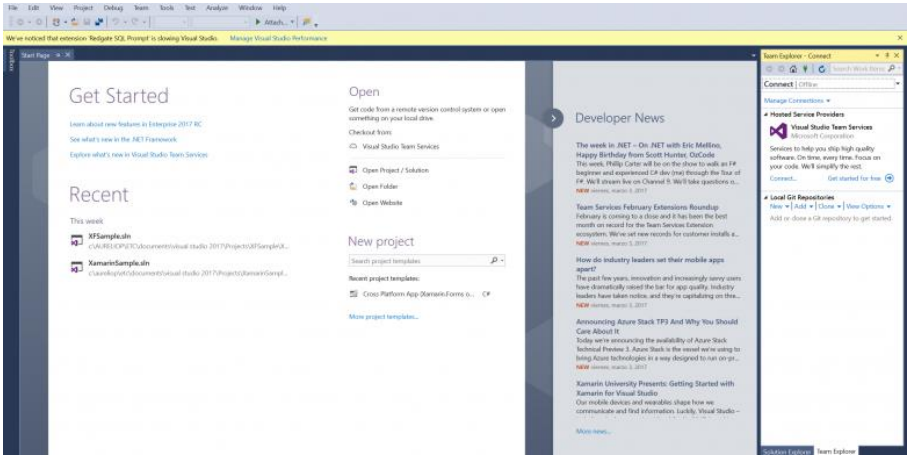
¿Qué es Microsoft Visual Studio?

Según la página web de Microsoft (2024), Visual Studio 2022 es

> […] una herramienta de desarrollo eficaz que permite completar todo el ciclo de desarrollo en un solo lugar. Es un entorno de desarrollo integrado (IDE) completo que puede usar para escribir, editar, depurar y compilar el código y, luego implementar la aplicación. Visual Studio 2022 es el IDE más completo para desarrolladores de .NET y C++ en Windows para crear aplicaciones web, en la nube, de escritorio, móviles, servicios y juegos.

Figura 10

IDE de Visual Studio



Fuente: (MVP Cluster, 2017)

¿Por qué usar Visual Studio 2022?

1. Porque es una plataforma basada en cargas de trabajo, es decir, se instala solo lo que se necesita.
2. Al ser una plataforma integrada, utiliza todas las herramientas y características necesarias para compilar las aplicaciones en un solo lugar.
3. Tiene compatibilidad con varios lenguajes, tales como C++, JavaScript, TypeScript, Python, VB.NET, etc.
4. Todo desarrollo que se realice es multiplataforma y puede usarse en cualquier plataforma.
5. Se puede integrar el control de versiones, lo que permite compartir el código entre los miembros del equipo.
6. Incluso puede usarse inteligencia artificial para escribir código de forma más eficaz con su ayuda.

Figura 11

Microsoft Visual Basic (VB.NET)

Fuente: (Vallaeta, 2024)

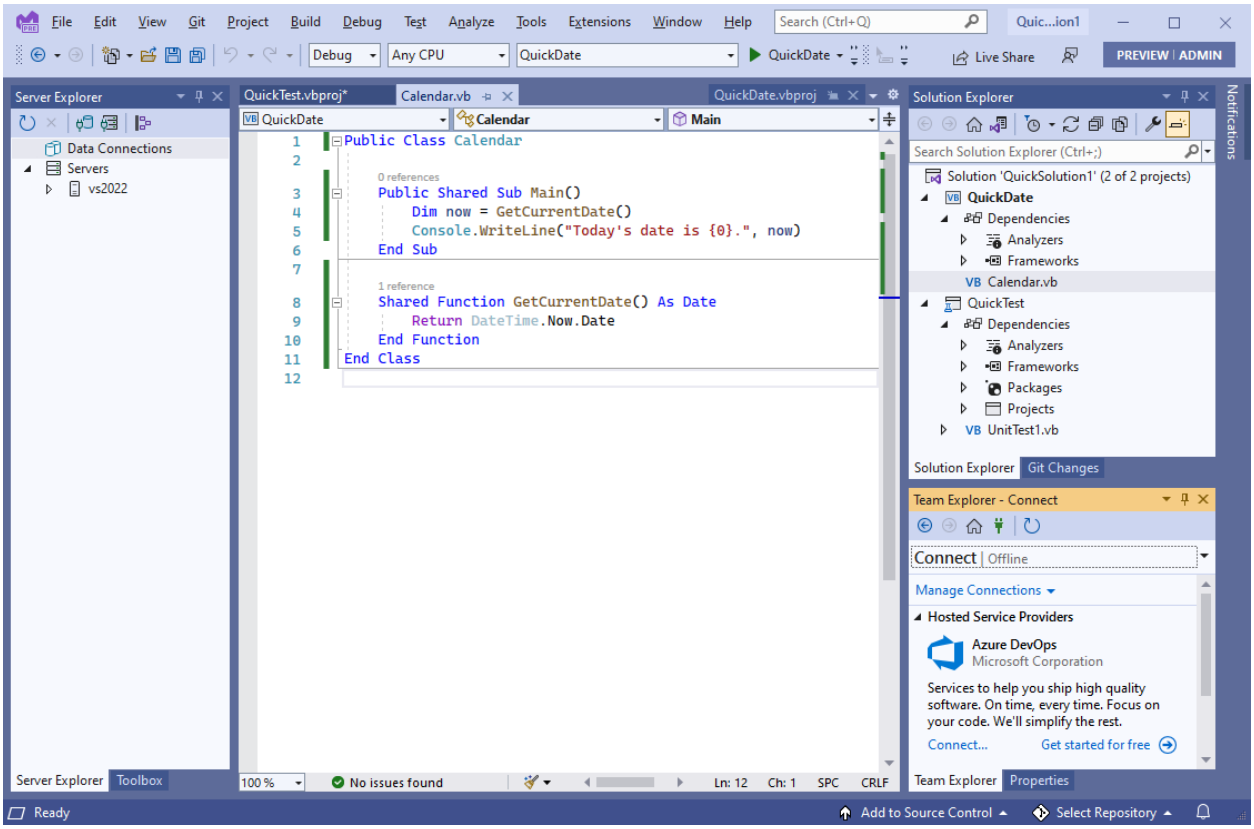
¿Qué es VB.NET?

Según la página Lov technology (Roch Moraguez, 2024):

Visual Basic (.NET) es un lenguaje de programación orientado a objetos desarrollado por Microsoft. Es descendiente del lenguaje Visual Basic original, publicado en 1991. Visual Basic (.NET) está diseñado para ser una herramienta versátil que permita crear aplicaciones tanto sencillas como complejas. Tiene una amplia gama de usos, desde la creación de aplicaciones de escritorio hasta la creación de soluciones de *software* de nivel empresarial.

Figura 12

Ejemplo de Ide de Vb.net



Fuente: (Microsoft, 2023)

Ventajas de utilizar Visual Basic (.NET)

Existen muchas ventajas en el uso de Visual Basic (.NET) para el desarrollo. Es un lenguaje relativamente fácil de aprender y utilizar, por lo cual es un candidato ideal para los desarrolladores que están empezando o que buscan un lenguaje con el que sea fácil trabajar, de manera que la curva de aprendizaje sea muy corta. Otra ventaja es que es un lenguaje muy flexible. Puede utilizarse desde sencillas aplicaciones de escritorio hasta complejas soluciones empresariales.

Visual Basic (.NET) utiliza una combinación de principios de programación orientada a objetos y el marco .NET. Se usa código en Visual Basic (.NET) para generar clases, objetos y métodos.

Cuando el código es compilado, se convierte a un lenguaje que el marco .NET puede entender. Luego de esto, el marco .NET toma el control y se encarga de aspectos como la gestión de la memoria.

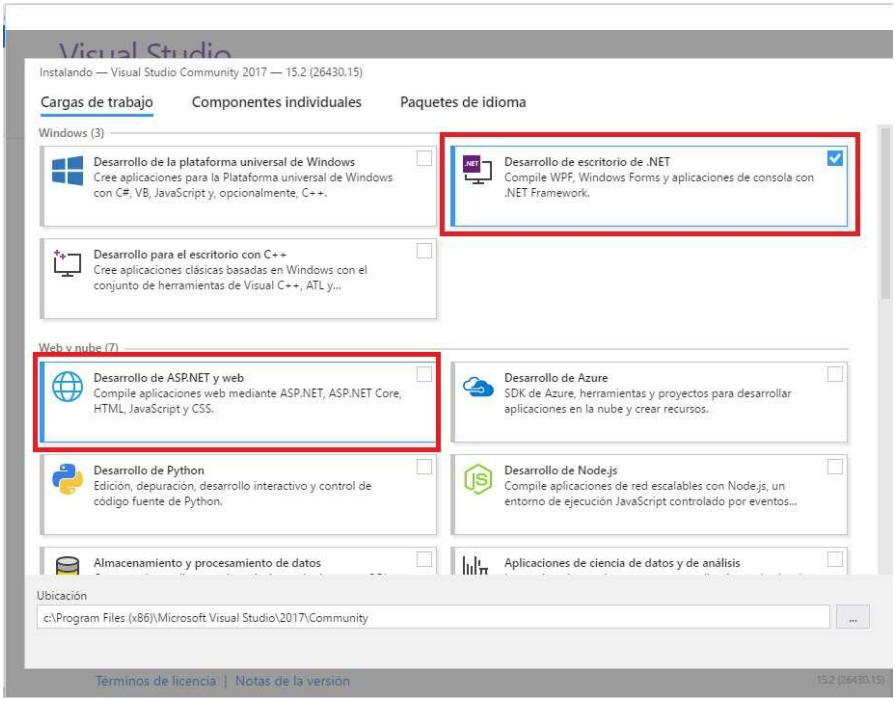
Usos de Visual Basic (.NET)

Como se ha indicado anteriormente, Visual Basic (.NET) es un lenguaje versátil que puede utilizarse para una amplia gama de aplicaciones. Algunos usos comunes del lenguaje incluyen:

1. Desarrollo de aplicaciones de escritorio.
2. Creación de aplicaciones web.
3. Creación de aplicaciones móviles.
4. Creación de soluciones de *software* para empresas.

Figura 13

Herramientas de desarrollo



Fuente: (Microsoft, 2024)

Herramientas de desarrollo de Visual Basic (.NET)

Hay muchas herramientas de desarrollo disponibles para trabajar con Visual Basic (.NET). Algunas de las herramientas más populares son:

1. Microsoft Visual Studio: un entorno de desarrollo integrado que proporciona a los desarrolladores una amplia gama de herramientas para crear aplicaciones con Visual Basic (.NET).
2. Xamarin: una plataforma de desarrollo para crear aplicaciones móviles para Android, iOS y, por supuesto, Windows, siempre utilizando Visual Basic (.NET).

3. Visual Basic Power Packs: un conjunto de herramientas y controles que pueden utilizarse para ampliar la funcionalidad de las aplicaciones Visual Basic (.NET).

Se utilizará VB.NET para el desarrollo de este proyecto por su versatilidad para usarse en una amplia gama de aplicaciones y entornos, que van desde aplicaciones de escritorio hasta aplicaciones web y empresariales, entre otros. Ya sea un desarrollador novato o uno experimentado, VB.NET ofrece una serie de herramientas y funciones que les servirán para crear aplicaciones en el menor tiempo posible, que sean potentes y escalables.

Base de datos

¿Qué es una base de datos?

> Según Amazon Web Services (2024), una base de datos es:
>
> […] una recopilación de datos sistemática y almacenada electrónicamente. Puede contener cualquier tipo de datos, incluidos palabras, números, imágenes, vídeos y archivos. Puede usar un *software* denominado sistema de administración de bases de datos (DBMS) para almacenar, recuperar y editar datos. En los sistemas informáticos, la palabra *base* de datos también puede referirse a cualquier DBMS, al sistema de base de datos o a una aplicación asociada con la base de datos.

Figura 14

Bases de datos

Fuente: **(Desarrolloweb6, 2024)**

Tipos de bases de datos

Existen diferentes clasificaciones o tipos de bases de datos:

1. Bases de datos relacionales: Este tipo de base de datos es muy utilizada en aplicaciones empresariales y transaccionales. Se utilizan tablas para organizar la información, con filas que representan registros y columnas que representan atributos.

   Ejemplos:

   1. MySql
   2. Microsoft SQL Server
   3. Oracle Database
   4. PostgreSQL
   5. IBM Db2

2. Bases de datos NoSQL: A diferencia de las bases de datos relacionales, no se usan tablas, si no que se guardan los datos en documentos, grafos o clave-valor. Están optimizadas para ofrecer escalabilidad horizontal y desarrollo ágil. Se utilizan en organizaciones que buscan almacenar datos no estructurados o semiestructurados.

   Ejemplos:

   1. MongDB
   2. Redis
   3. Apache Cassandra
   4. CoucheBase

3. Bases de datos en la nube (IaaS): Este tipo se entregan como un servicio desde la nube, su creación, mantenimiento y escalabilidad el corresponde al proveedor del servicio.

   Ejemplos:

   1. Google Firebase
   2. Microsoft Azure SQL Database
   3. Amazon Relational Database Service
   4. Oracle Autonomus Database

4. Base de datos en columnas: Se utilizan columnas en lugar de filas. Cuando se hace una consulta, se ignoran todos los datos que no se aplican a la consulta

especifica, porque solo se puede recuperar la información de las columnas que se desean.

Ejemplos:

1. Google BigQuery
2. Cassandra
3. Hbase
4. MariaDB
5. Azure SQL Data Warehouse

5. Bases de datos de columnas anchas (*wide column*): Tienen la ventaja de ser altamente escalables, puede manejar incluso petabytes de datos, lo que la convierte en la candidata ideal para aplicaciones de *big data* en tiempo real.

   Ejemplos:

   1. BitTable
   2. Apache Cassandra
   3. Scylla

6. Base de datos orientadas a objetos: Se basan en la POO, por lo que los datos y todos los atributos, están unidos como un objeto.

   Este tipo de base de datos se manejan mediante sistemas de gestión de bases de datos orientados a objetos (OODBMS [*Object Oriented Database Management*

System]). Estas bases de datos funcionan bien con lenguajes de programación orientados a objetos, tales como C++ y Java.

7. Bases de datos clave-valor (*key-value*): es un tipo de bases de datos NoSQL, se guardan los datos como un grupo de pares clave-valor formados por dos elementos de datos cada uno. Se puede manejar grandes volúmenes de tráfico y son muy escalables, ideales para procesos como la gestión de sesiones para aplicaciones web, juegos masivos en línea y carritos de compra.
8. Bases de datos jerárquicas: Se utilizan para soportar aplicaciones de alto rendimiento y alta disponibilidad.

   Ejemplos:

   1. Sistema de Gestión de información de IBM (IMS)
   2. Registro de Windows

9. Bases de datos documentales: Se les conoce como bases de datos orientadas a documentos (DODB), están diseñadas para almacenar y gestionar información orientada a documentos, también son conocidas como datos semiestructurados. Son sencillas y escalables, útiles para aplicaciones móviles.

   Ejemplos:

   1. MongoDB
   2. Amazon DocumentDB
   3. Apache CouchDB

10. Bases de datos graficas o de grafos (*graph*): Se utilizan para analizar relaciones entre puntos de datos heterogéneos, como en la prevención del fraude o para extracción de datos sobre los clientes de las redes sociales.

Ejemplos:

1. Datastax Enterprise Graph
2. Neo4J AuraDB

11. Bases de datos de series temporales (*time series*): Estas bases de datos están optimizadas para llevar una marca de tiempo o *timestamp*, lo que las hace útiles para monitoreo.

Ejemplo:

1. Druid
2. eXtremeDB
3. InfluxDB

Figura 15

SQL

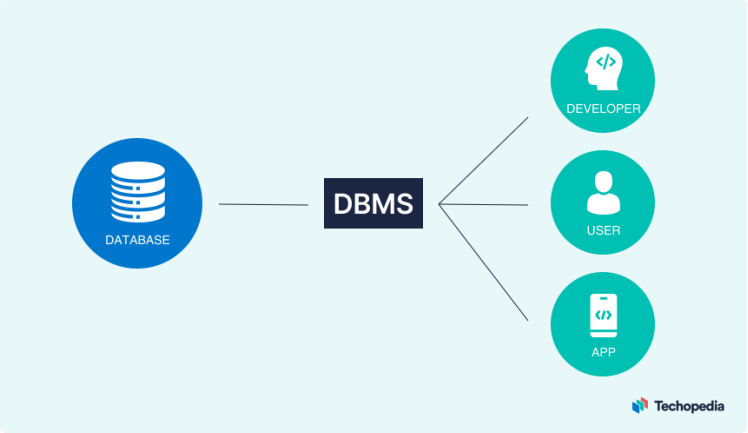
Fuente: (Tic Portal, 2023)

SQL

De acuerdo con la página Learn SQL (2023), SQL (*Structured Query Language* o Lenguaje de consulta estructurada) “es un lenguaje de programación que se utiliza para comunicarse con las bases de datos”. Se utiliza para extraer información y administrar sistemas de gestión de datos relacionales.

Figura 16

Sistema de Gestión de Bases de Datos (DBMS)



Fuente: (Rouse, 2024)

DBMS

Es un *software* de sistema para crear y administrar bases de datos. Proporciona a los usuarios y programadores una forma sistemática de crear, recuperar, actualizar y administrar datos. Esencialmente sirve como una interfaz entre la base de datos y los usuarios finales o programas de aplicación. Algunos de los sistemas de gestión de bases de datos más utilizados son los de ORACLE, Microsoft SQL Server, MySQL o PostgreSQL.

Figura 17

Arquitectura web

Fuente: (Delgado, 2022)

Arquitectura WEB

De acuerdo con el artículo de Pulido (2024) “La arquitectura web es la disciplina que engloba la organización de los contenidos e información de una web, incluyendo la jerarquía entre sus elementos y las relaciones entre los mismos”. Se encarga de diseñar los proyectos de desarrollo web y se enfoca en el análisis, planificación y estructuración del sitio.

Scrum

Francia Huambachano (2017) explica lo siguiente:

> Scrum es un proceso de gestión que reduce la complejidad en el desarrollo de productos para satisfacer las necesidades de los clientes. La gerencia y los equipos de trabajan juntos alrededor de requisitos y tecnologías para entregar productos funcionando de

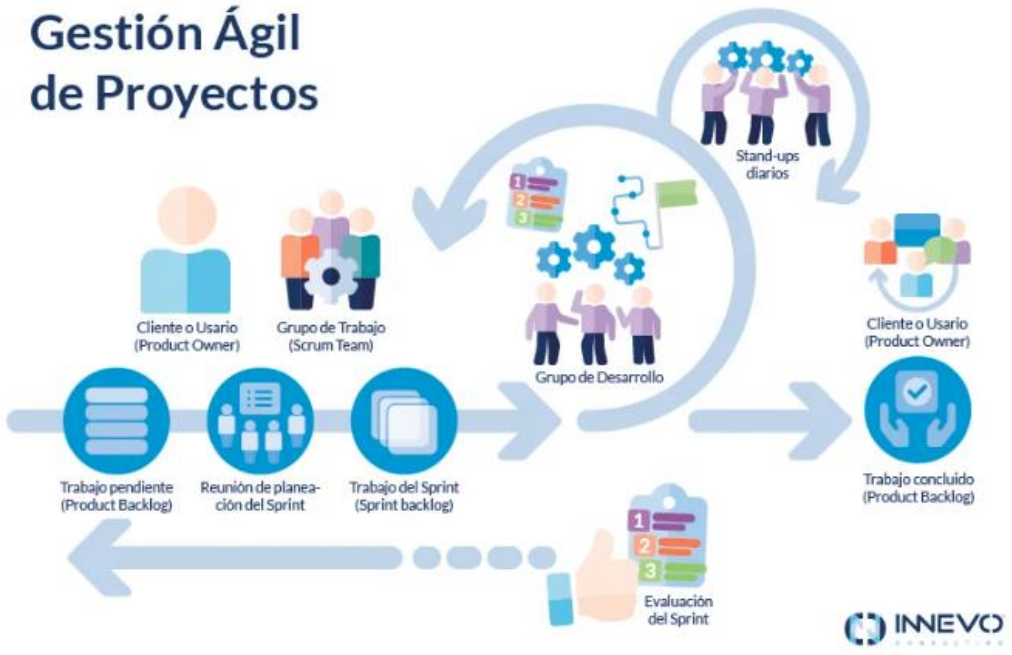
manera incremental usando el empirismo (el conocimiento viene de la experiencia y las decisiones que se toman en función de lo que se observa).

Es un marco de trabajo ágil utilizado en el desarrollo de *software* y otros proyectos complejos. Se basa en la colaboración de un equipo multidisciplinario y autoorganizado que trabaja en ciclos cortos e iterativos llamados *sprints*.

La metodología Scrum, fue diseñada originalmente para proyectos de desarrollo de *software*. Ahora este sistema centrado en la flexibilidad, el contacto con el cliente y la efectividad en el tiempo se aplica en proyectos de diferentes industrias.

Figura 18

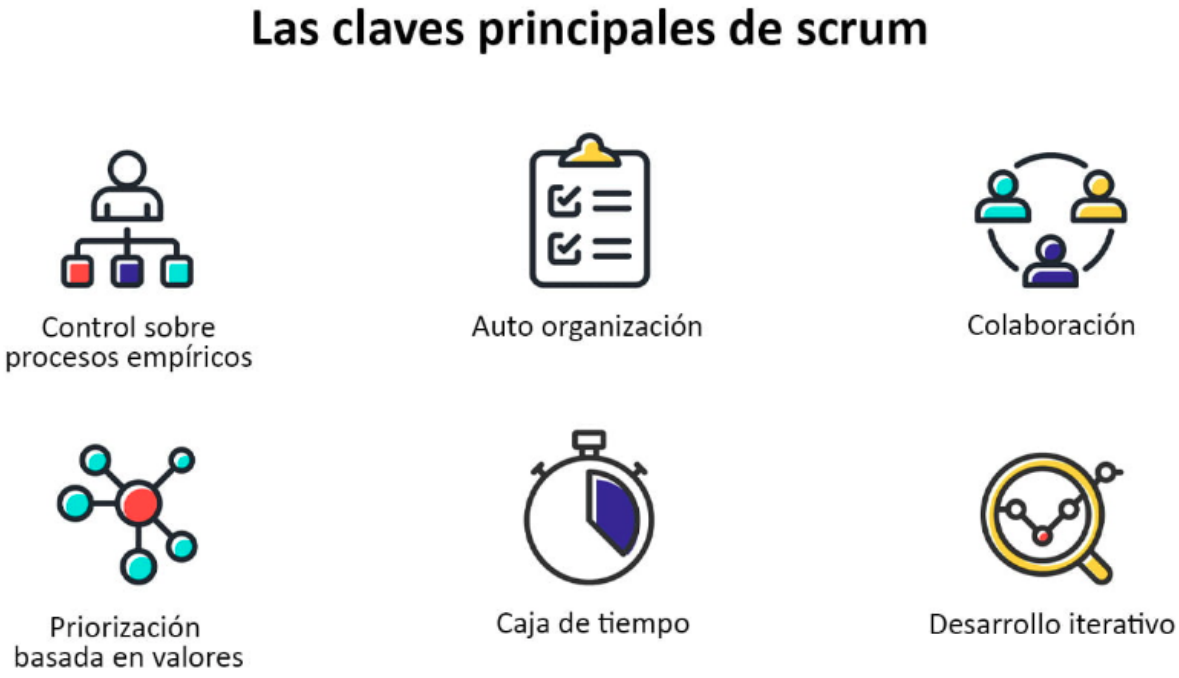
Gestión ágil de proyectos



Fuente: (Lomelí, 2023)

Figura 19

Principales claves de Scrum



Fuente: (Lucidspark, 2017)

Los miembros principales de un equipo Scrum son el *Product Owner*, el *Scrum Master* y el Equipo de Desarrollo.

El *Product Owner* es responsable de gestionar el backlog del producto, priorizar las tareas y tomar decisiones sobre qué funcionalidades se deben desarrollar. Su objetivo es maximizar el valor del producto y satisfacer las necesidades del cliente.

El *Scrum Master* es el encargado de garantizar que el equipo Scrum siga las prácticas y valores de Scrum, eliminando obstáculos y fomentando la colaboración y la mejora continua. Es un facilitador y *coach*.

El Equipo de Desarrollo es un equipo autogestionado y multidisciplinario que se encarga de desarrollar las funcionalidades del producto. Este equipo es responsable de planificar su trabajo, colaborar en la toma de decisiones y trabajar en conjunto para cumplir con los objetivos del sprint.

Se cuenta con varios componentes clave, tales como:

1. El *backlog* del producto, que contine todas las funcionalidades y tareas necesarias para el proyecto, priorizadas por el *Product Owner*.
2. El *backlog* del *sprint*, que contiene las tareas específicas que el equipo se compromete a completar durante el sprint.
3. Las reuniones diarias, donde el equipo se reúne brevemente para sincronizar su trabajo, identificar obstáculos y planificar el trabajo del día.
4. La revisión del sprint, donde se presentan las funcionalidades desarrolladas durante el sprint al *Product Owner* y otras partes interesadas para obtener retroalimentación.
5. La retrospectiva del sprint, donde el equipo reflexiona sobre su trabajó, identificar oportunidades de mejora y define acciones para aplicar en los futuros *sprints*.

Valores de Scrum

1. Coraje
2. Valentía para hacer lo correcto y trabajar en problemas complejos.
3. Foco

Todos los miembros del equipo deben enfocarse en el trabajo planificado en cada *sprint* que, en última instancia, permite cumplir los *sprint goals*.

1. Compromiso

   Cada miembro del equipo Scrum, es decir, todos, incluidos el *Scrum Master* y el *Product Owner*, harán el máximo esfuerzo posible y serán completamente transparentes sobre el progreso del Sprint.

2. Respeto

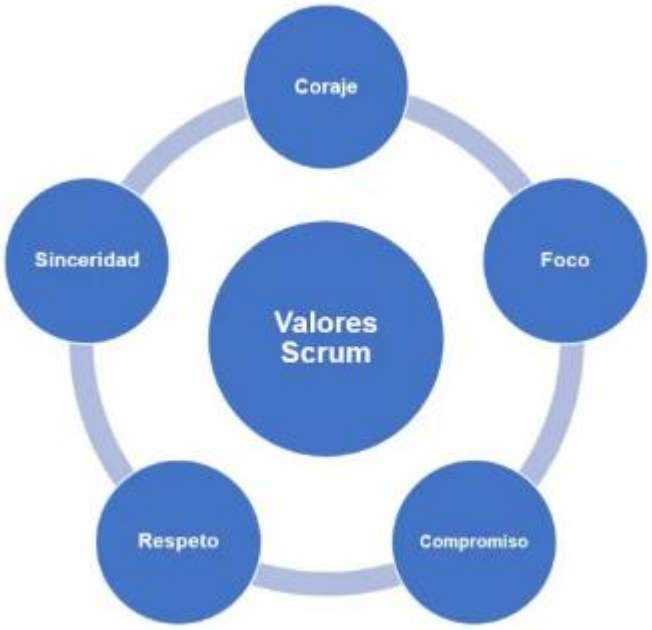
   Los miembros del equipo Scrum deben respetar el conocimiento, las habilidades y la experiencia profesional no solo del resto de los miembros del equipo, sino también de aquellas personas con las que se relacionan, sea de su organización o de otra.

3. Sinceridad

   El equipo debe ser transparente con el trabajo que realiza, con el progreso de este, y con el conocimiento que se adquiere (documentación).

Figura 20

Valores de Scrum



Fuente: (Rodríguez, 2020)

¿Qué son las historias de usuario en Scrum?

Son el elemento mínimo que conforma un proyecto en metodología ágil. Condensan los requisitos del cliente, lo define y ayudan a comprender al equipo, el porqué están construyendo.

Estructura de las historias de usuario

Según la página Productiviza (2024):

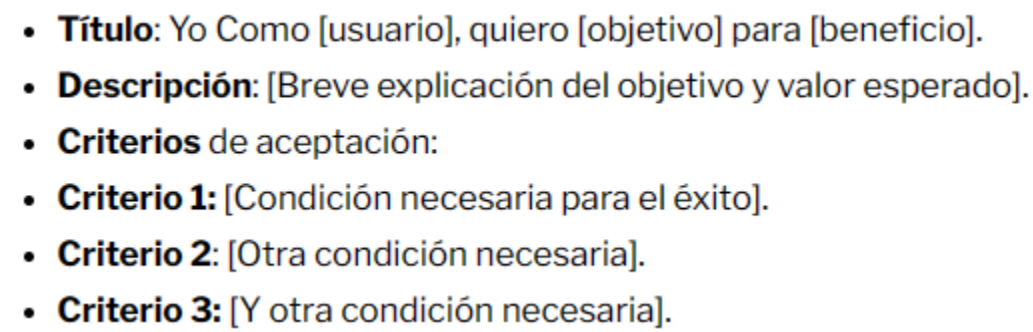
Una historia de usuario consta de tres elementos principales: el título, la descripción y los criterios de aceptación. El titulo debe ser descriptivo y centrado en el usuario, mientras que la descripción debe explicar brevemente el objetivo y el valor que se espera obtener.

Por otro lado, los criterios de aceptación definen las condiciones necesarias para que la historia de usuario se considere completa y exitosa.

Estas se redactan de forma clara y concisa, se sigue una estructura especifica o formato ya predefinido, lo cual ayuda a personalizar la funcionalidad y a entender mejor sus necesidades.

Figura 21

Ejemplo de estructura de una historia de usuario



Fuente: (Productiviza, 2024)

Figura 22

Ejemplo 1 Historia de usuario

Historia de usuario 1: Creación de la web

Descripción: Yo como nuevo usuario, quiero poder registrarme en el sitio web para acceder a contenido exclusivo.

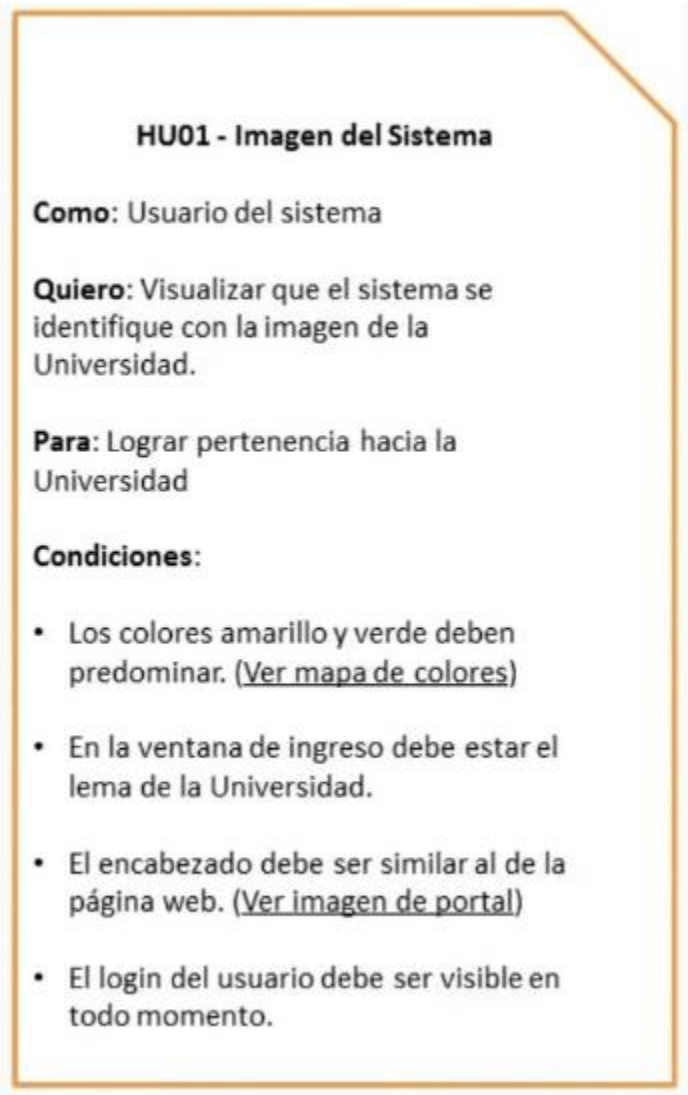
Criterios de aceptación:

- El usuario debe poder acceder al formulario de registro desde la página de inicio.
- El formulario de registro debe solicitar al usuario un nombre de usuario, correo electrónico y contraseña.
- El correo electrónico debe estar en un formato válido y no puede estar registrado previamente en el sistema.
- La contraseña debe tener al menos 8 caracteres y contener al menos una letra y un número.
- Al enviar el formulario, el usuario debe recibir un correo electrónico de verificación con un enlace para activar su cuenta.
- Después de hacer clic en el enlace de verificación, el usuario debe poder iniciar sesión en su cuenta correctamente.

Fuente: (Productiviza, 2024)

Figura 23

Ejemplo 2 de Historias de usuario



Fuente: (Mancuso, 2021)

Beneficios de las historias de usuario

1. Comunicación efectiva: Las historias de usuario son una forma clara y concisa de comunicar las necesidades y requerimientos del cliente al equipo de desarrollo. Esto facilita la compresión de lo que se espera y ayuda a evitar malentendidos.

2. Priorización: Permiten a los equipos priorizar de manera efectiva las tareas y funcionalidades que se deben implementar en cada sprint. Esto ayuda a enfocarse en las tareas más importantes y que generen más valor para el cliente.
3. Transparencia: Al tener las historias de usuario claramente definidas y visibles para todo el equipo, se fomenta la transparencia y la colaboración. Todos los miembros del equipo tienen claridad sobre lo que se está trabajando y cuál es el objetivo final.
4. Flexibilidad: Las historias de usuarios pueden ser modificadas o actualizadas según la retroalimentación del cliente o los cambios en los requerimientos. Esto permite adaptarse rápidamente a las necesidades del proyecto.
5. Entrega incremental. Al dividir las tareas en historias de usuario más pequeñas y manejables, se facilita la entrega incremental de funcionalidades y la obtención de retroalimentación temprana del cliente. Esto permite realizar ajustes durante el proceso de desarrollo y mejorar la satisfacción del cliente.

Las historias de usuario en Scrum proporcionan una forma efectiva de comunicar las necesidades del cliente, priorizar tareas, fomentar la transparencia y flexibilidad, y facilitar la entrega incremental de funcionalidades. Además de que ayudan a mantener el enfoque en lo que realmente importa y a obtener mejores resultados en el proyecto.

Git

Es un sistema de control de versiones distribuido que se utiliza comúnmente en proyectos de desarrollo de *software* en el marco de metodologías ágiles, tales como Scrum. En Scrum, Git se usa para gestionar y controlar las diferentes versiones de los archivos y código fuente que forman parte del proyecto. Permite a los equipos de desarrollo colaborar de manera eficiente,

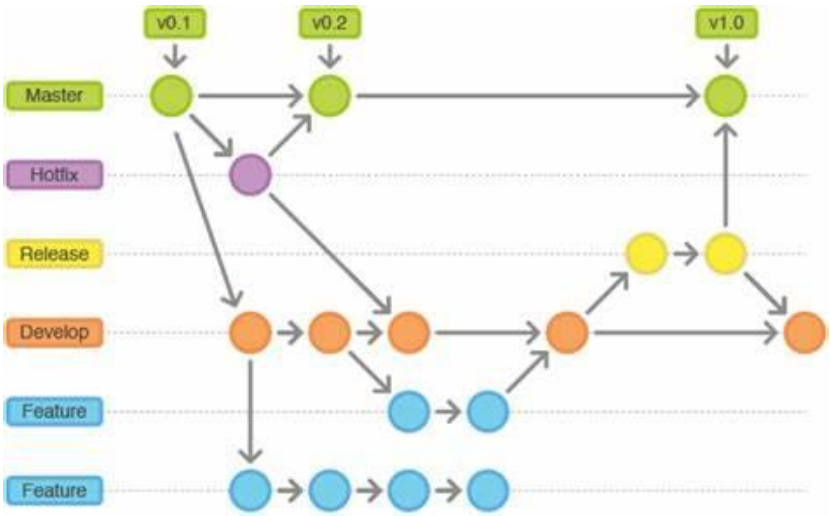
trabajar en paralelo en diferentes funcionalidades, realizar cambios en el código de forma controlada y revertir los cambios si es necesario.

Proporciona herramientas para crear ramas donde se pueden desarrollar nuevas características de forma aislada sin interferir con el código base; luego, se pueden fusionar estas ramas con la rama principal del proyecto. Además, se pueden realizar seguimientos de los cambios realizados en los archivos, identificar quién realizó cada cambio y en qué momento se realizó.

En el contexto de Scrum, Git facilita la colaboración entre los miembros del equipo, mejora la visibilidad y trazabilidad de los cambios realizados en el código y permite la entrega continua de valor al cliente. Al implementar Git en un proyecto de Scrum, se fomenta la transparencia, la comunicación efectiva y la entrega de productos de *software* de alta calidad y en constante evolución.

Figura 24

Ramas principales y de desarrollo Git



Fuente: (García, 2020)

Sistemas de información

Es un conjunto ordenado de personas, procesos y herramientas con el fin de administrar datos e información, de manera que se puedan recuperar y procesar de forma fácil y rápida. De acuerdo con Peiró (2020), “un sistema de información tiene como objetivo principal la gestión y administración de datos e información que lo componen. Lo más importante es poder recuperar siempre esos datos y que, además, se tenga un fácil acceso a ellos con total seguridad.”

Los componentes del sistema de información permiten una serie de procesos: la entrada de los datos, la gestión y el procesamiento de estos, el almacenamiento y la salida de todos aquellos interesados que deseen tener acceso a esta información.

De acuerdo con Comunidad Empresas (2024), los sistemas de información se pueden clasificar en 5 categorías:

1. Financieros: tienen relación con el dinero y los activos de una compañía.
2. Tecnológicos: cuando el vínculo es sobre temáticas especializadas y deben procesar automáticamente la información.
3. Humanos: aquellos sobre recursos humanos, relacionado con plantilla o trabajadores.
4. Materiales: más que todo, sistemas y soporte físico.
5. Administrativos: su nombre lo dice, pueden ser permisos, transacciones, entre otros.

Tipos de sistemas de información

1. Sistemas de Procesamiento de Transacciones (TPS): conocido también como EDP o Procesamiento Electrónico de Datos. Son básicos para el sistema empresarial, y son vistos como un sistema de gestión operativa, en las decisiones de los mandos de primer

nivel. Lo que hacen es recopilar la información que tiene que ver con las transacciones de la organización y su funcionamiento. Genera los datos de entrada a otros SI. Existen distintos TPS, como de pedidos, vendedores, ventas, compras, costos, gastos, ingresos, nóminas, entre otros.

2. Sistemas de Información Ejecutiva (EIS): es el acceso rápido a la información interna y externa de la empresa. Monitoriza las variables gerenciales de un área específica de la organización, con la posibilidad de entregar datos básicos más detallados si es necesario. Puede ser efectivo para tomar decisiones estratégicas en la compañía.
3. Sistemas de Información Gerencial (MIS): se enfoca en los mandos medios, brinda apoyo a la toma de decisiones. Se refiere a la gestión operativa rutinaria de la organización, contemplan la información general como un todo, entregando informes o estadísticas, con frecuencia en modo de panel de control. Se ven comúnmente en temáticas de *marketing*, producto, distribución, producción, aprovisionamiento, inventarios, costes, proveedores, tesorería, finanzas, inversiones, cotizaciones, recursos humanos, desempeño, remuneraciones, reclutamiento y más.
4. Sistemas de soporte de decisiones (DSS): foco en el procesamiento de información intra y extra organizacional, es un motor para la conducción de una empresa. Contribuye a la toma de decisiones y así poder resolver problemas concretos; son los que usa un directivo. Los problemas menos rutinarios se pueden resolver, se diferencia del anterior en que informa lo que podría suceder basándose en análisis de los datos.
5. Sistema de Gestión del Conocimiento (KMS): los procesos que tienen que ver con la captura, producción, almacenamiento y difusión del conocimiento de la compañía tiene

relación con este SI. Ideal para una organización con una comunicación poco fluida, se relaciona con otros como DMS, que veremos a continuación.

6. Sistema de Gestión Documental (DMS): gestiona gran cantidad de documentos, dando acceso, almacenamiento, seguridad, indexación y recuperación. Organiza todo en base a carpetas o categorías, tiene una búsqueda y acceso muy expedito y sencillo.
7. Sistema de Gestión de Contenidos (CMS): básicamente creación, administración y publicación de contenidos. Utilizado para sitios públicos corporativos, maneja el diseño, enlaces y el contenido como tal de la web correspondiente. Comparte cosas con el DMS, ya que tiene información valiosa para la organización.
8. Sistema de Automatización de Oficinas: las tareas básicas y rutinarias de una oficina se crean, recogen, almacenan, manipulan y comparten aquí. Sus tareas están en, por ejemplo, llamar, realizar documentos o copiar información de un lugar a otro.
9. Sistema de Planificación de Recursos Empresariales (ERP): forma parte del TPS, pero se diferencia en que permite todo lo que se hace en una empresa de manera global. La data queda centralizada en una única base de datos, a la cual pueden acceder desde distintas áreas de la compañía, con el objetivo de optimizar los procesos, compartir simplemente información, eliminar datos y operaciones innecesarias, soporte, visión global, control detallado y aprovechamiento del tiempo.

Beneficios de utilizar sistemas de información

Gracias a los sistemas de información, una empresa logra tener un control mucho más efectivo de las actividades que realiza, sus distintas áreas y, a la vez, mejorar la productividad de los procesos. El hecho de poder tener más y mejor información, en tiempo real, rompe ciertas barreras que pueden existir y ofrece ventajas competitivas, disminuye los errores, el tiempo y también los gastos de recursos gracias al cruce de datos con los objetivos propuestos, su evaluación y control. Se pueden dividir los beneficios generales de los sistemas de información en:

1. Control efectivo de las actividades de la empresa
2. Integrar nuevas tecnologías y herramientas claves para el negocio
3. Mejorar la efectividad de las operaciones
4. Ventajas competitivas
5. Más y mejor información, en tiempo real
6. Un mismo sistema, en distintos puntos y al mismo tiempo
7. Menos errores, menos gastos
8. Control cruzado de datos

Para finalizar, es valioso saber que cada tipo de sistema de información está compuesto por distintos recursos, que son convenientes según el propósito o fin para el que será utilizado, dependiendo del rubro o sus funciones específicas.

HTML

HTML es el lenguaje con el que se define el contenido de las páginas web. Básicamente, se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web, como imágenes, listas, vídeos, etc. El HTML es un lenguaje de marcación de elementos para la creación de documentos hipertexto, muy fácil de aprender, lo que permite que cualquier persona, aunque no haya programado en la vida, pueda enfrentarse a la tarea de crear una web.

Etiquetas en un archivo HTML

Las etiquetas HTML son bloques de código que dan formato, funcionalidad y estructura al contenido de las páginas web. Estos fragmentos sirven como indicadores o instrucciones para que un navegador muestre de forma adecuada la información contenida en tus documentos. Es importante mencionar que el HTML no se considera un lenguaje de programación, pues no crea funciones dinámicas. No obstante, con el apoyo de las etiquetas HTML, los desarrolladores pueden configurar diversas estructuras para sus sitios web; por ejemplo: secciones, tablas, párrafos, enlaces y atributos.

Partes básicas de las etiquetas HTML

1. Elemento
    1. Es el nombre de la etiqueta, aparece entre los signos de apertura. Por ejemplo: <title>Partes básicas de etiquetas HTML</title>, que indica que esa frase es un título.

2. Atributo
    1. Es lo que diferencia a un elemento y que le da ciertas características, como tamaño, color, tipo de fuente. Se escribe antes del signo de igual (=), junto al elemento de la etiqueta, únicamente en la apertura y antes de la variable. Por ejemplo: . Este texto aparecerá en tipografía Arial. El atributo es el tipo letra, es decir: face.
3. Contenido
    1. Es el texto que modifica las etiquetas, lo que aparece entre las etiquetas y atributos.
4. Variable
    1. Es la característica del atributo de una etiqueta: color, tamaño, tipo de letra, y se escribe generalmente entre comillas después del signo de igual. En el ejemplo de atributo, la variable es Arial.

Figura 25

Estructura básica de un archivo HTML

Fuente: **(Delgado, 2022)**

CSS

CSS son las siglas en inglés para «hojas de estilo en cascada» (***Cascading Style Sheets***). Básicamente, es un lenguaje que maneja el diseño y presentación de las páginas web, es decir, cómo lucen cuando un usuario las visita. Funciona junto con el lenguaje HTML que se encarga del contenido básico de los sitios. Se les denomina hojas de estilo «en cascada», porque puedes tener varias y una de ellas con las propiedades heredadas (o «en cascada») de otras.

Ventajas y desventajas de usar CSS

Ventajas de usar CSS

1. Separación de la estructura y la presentación. CSS permite separar el contenido HTML de su presentación visual. Es decir, te permite mantener el código HTML limpio y estructurado, mientras que el estilo se define en un archivo CSS separado. Este orden mejora la legibilidad del código y facilita el mantenimiento, así como la actualización de los estilos.
2. Consistencia y mantenibilidad. Al utilizar CSS, puedes aplicar estilos de manera consistente a un sitio o aplicación web. Los estilos se definen una vez y se aplican a múltiples elementos en las páginas, lo que favorece la renovación de la apariencia visual de todo el proyecto.
3. Eficiencia en el rendimiento. CSS permite cargar estilos externos en un archivo separado. Con esto, el navegador almacenará en caché los estilos y los aplicará a todas las páginas del sitio, lo que mejora el rendimiento al reducir la cantidad de datos que deben transferirse entre el servidor y el cliente.
4. Flexibilidad y control. Asimismo, ofrece una amplia gama de propiedades y selectores con los que se obtiene control preciso sobre el estilo de los elementos HTML. Se puede modificar, de forma sencilla, los colores, fuentes, márgenes, tamaños, diseños, etc. Gracias a esta característica, se puede personalizar y adaptar a varios dispositivos y tamaños de pantalla cualquier sitio web.

Desventajas de usar CSS

1. Curva de aprendizaje. CSS puede tener una curva de aprendizaje empinada, en particular, para los principiantes. Entender, por completo, todas las propiedades, selectores y conceptos avanzados puede llevar tiempo y práctica.
2. Compatibilidad entre navegadores. Aunque los estándares de CSS son de los más aceptados, algunos navegadores podrían interpretar y renderizar los estilos de manera diferente. Esto puede resultar en inconsistencias visuales y requerir pruebas o ajustes adicionales para garantizar la compatibilidad entre plataformas.
3. Especificidad y herencia. CSS utiliza reglas de especificidad y herencia para determinar qué estilos se aplican a los elementos. En ocasiones, el orden de las reglas y la jerarquía pueden generar resultados inesperados. Esto requiere una comprensión cuidadosa de cómo funcionan estas reglas para evitar conflictos y problemas de estilo.
4. Limitaciones en la maquetación. Aunque CSS ofrece una amplia gama de propiedades para el diseño y la maquetación, puede presentar limitaciones en ciertos casos más complejos. Algunos diseños específicos pueden requerir soluciones adicionales o el uso de técnicas más avanzadas para lograr el resultado deseado.

Diferencias entre HTML y CSS

HTML da estructura al contenido de un sitio web. Las siglas corresponden ***a HyperText Markup Language***, que significa «lenguaje de marcas de hipertexto» y hacen referencia al código que define el significado de las instrucciones dadas a una plataforma computacional. Estas instrucciones representan todos los enlaces (o hipertextos) que vinculan los contenidos de una página, por lo que HTML es la base de cualquier sitio web. En este lenguaje, es posible

incluir toda la información referente al contenido, así como las imágenes, audios y estilos. Sin embargo, su uso para estas tareas conlleva una mayor complejidad en el código fuente.

Para hacer más eficiente el uso de HTML, se han diseñado lenguajes computacionales que facilitan la gestión de los datos relacionados con el diseño visual de las plataformas. CSS es uno de los lenguajes más importantes que se utilizan para ordenar las instrucciones referentes a la apariencia de un sitio y presentar los contenidos de una página de forma atractiva.

De este modo, HTML se emplea para estructurar el contenido de un sitio, mientras que CSS, para estructurar su presentación.

Componentes de una hoja de estilo CSS

Las hojas de estilo CSS se componen de dos elementos: la primera parte que se llama declaración y la segunda denominada selector. A continuación, se muestra información más detallada al respecto:

1. Declaración

Es el elemento que indica lo qué se debe hacer. Está compuesta por una propiedad que puede ser el color, el tipo de fuente, tamaño, entre otros elementos.

2. Selector

Es la fase que indica a qué elemento se le aplica una declaración. En otras palabras, es la segunda fase de las reglas aplicadas y le señala al navegador cuáles son los elementos HTML a los que se le atribuye la regla. Existen varios tipos de selectores, estos son los básicos:

a. Universal: Se usa para elegir todos los elementos de una página web.
b. De tipo o etiqueta: Escoge todos los elementos de un sitio cuya etiqueta HTML tiene un mismo valor.
c. Descendente: Ayuda a que el selector etiqueta sea más preciso, porque elige a todos los elementos dentro de otros elementos.
d. De clase: Se utilizan para buscar secciones basadas en un atributo. Para evitar que se confunda con otros selectores, el valor se prefija con un punto.
e. De ID: Se emplean para seleccionar componentes en una página a los que se les haya dado como valor ID.

Bootstrap

De acuerdo con la página Rockcontent (2020), Bootstrap es un *framework* CSS desarrollado por Twitter en 2010 para estandarizar las herramientas de la compañía. Inicialmente se llamó Twitter Blueprint y, un poco más tarde, en 2011, se transformó en código abierto y su nombre cambió a Bootstrap.

El ***framework*** combina CSS y JavaScript para estilizar los elementos de una página HTML. Permite mucho más que simplemente cambiar el color de los botones y los enlaces.Esta herramienta proporciona interactividad en la página, ofreciendo una serie de componentes que facilitan la comunicación con el usuario, como menús de navegación, controles de página, barras de progreso y más. Además de todas las características que ofrece el *framework*, su principal objetivo es permitir la construcción de sitios web responsive para dispositivos móviles. Esto significa que las páginas están diseñadas para funcionar en desktop, tabletas y smartphones de una manera muy simple y organizada.

JavaScript

Según la página Coppola (2023), JavaScript es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, utilizado principalmente en el desarrollo web, pensado para agregar potencial de interacción y dinamismo a las páginas web. Es compatible con todos los navegadores modernos y se ejecuta del lado del cliente, lo que significa que se ejecuta en el navegador web del usuario final. Además de su uso en el desarrollo web, JavaScript también se utiliza en aplicaciones de servidor (con tecnologías como Node.js) y en el desarrollo de aplicaciones móviles y de escritorio.

JavaScript proporciona una amplia gama de funcionalidades, como manipulación del DOM (*Document Object Model*) para interactuar con elementos de una página web, manejo de eventos, comunicación con servidores a través de AJAX, creación de animaciones, validación de formularios y mucho más. Es un lenguaje flexible y dinámico que permite a los desarrolladores crear experiencias interactivas y ricas en contenido.

Es importante destacar que JavaScript no está relacionado con Java, a pesar de tener un nombre similar. Son dos lenguajes de programación diferentes con propósitos y características distintas.

¿Para qué sirve JavaScript?

En el contexto actual, JavaScript se utiliza para todo, gracias a la introducción de Node.js. Esta tecnología crea *software* robusto para empresas en todo el mundo. Además, organizaciones como LinkedIn y Medium lo implementan al construir plataformas para que los usuarios tengan acceso a sus servicios.

Lo que se puede hacer con JavaScript abarca diferentes tipos de *software*, como juegos, programas de computadora, aplicaciones web y hasta tecnologías de *blockchain*. JavaScript es posiblemente el lenguaje de programación más popular de la web. Por ejemplo, más de 125.000 empleos en LinkedIn buscan profesionales con habilidades en JavaScript.

JavaScript para desarrollo web

El uso más popular de JavaScript es para el desarrollo web y es una de las herramientas más poderosas que un desarrollador puede tener en sus manos. Los desarrolladores usan JavaScript en esta área para añadir interactividad y funciones que mejoren la experiencia del usuario y hagan que internet se disfrute mucho más.

JavaScript se ha expandido más allá del desarrollo de interfaz, que es donde comenzó. Recientemente, J**avaScript ha llegado al back-end, o dorsal** del **desarrollo web**. Esto quiere decir que los desarrolladores tienen acceso de interfaz a métodos CRUD (Create, Read, Update, Destroy; en español: Crear, Leer, Actualizar, Destruir) y hasta puede utilizarse en el motor de un sitio web.

Además, de acuerdo con W3techs, más del 90 % de todos los sitios web funcionan con JavaScript. Esto lo convierte en el líder primordial en tecnología de desarrollo web. Seguidamente, se muestran algunos usos específicos de JavaScript en desarrollo web:

1. Interactividad de interfaz o *front-end*: El desarrollo web mejora solamente por el aumento de la interactividad y funciones que JavaScript ofrece.

2. Aplicaciones web: Las aplicaciones web son similares a los sitios, pero en su lugar pueden empacarse en una caja más compacta, que mejora el control de la seguridad y otros aspectos.
3. Juegos de navegador: Los navegadores web actuales han cambiado mucho; los desarrolladores pueden crear juegos robustos que funcionan en ellos.
4. Desarrollo web dorsal o *back-end*: El desarrollo web se ha transformado tanto, que JavaScript puede utilizarse para gestionar el *back-end* de sitios y aplicaciones.

Figura 26

Uso de JavaScript

Fuente: (Coppola, 2023)

Responsive

En sus siglas en inglés, RWD significa diseño web adaptable; es una de las mejores prácticas que existen para lograr una excelente experiencia de usuario. Se trata de readaptar el diseño de una web para que pueda verse desde cualquier teléfono móvil o dispositivo; es decir, todos y cada uno de los elementos de un sitio web se adaptan a todo tipo de tamaños de pantalla: tabletas, ordenadores, *smartphones*... No importa desde qué dispositivo se ingrese; no será un problema si la web tiene un diseño responsive.

Cada dispositivo y cada terminal tiene un tamaño de pantalla distinto, por lo que el diseño responsive debe encargarse de **redimensionar y ubicar todos los elementos de una página web** de forma que su visualización sea la adecuada, garantizando, por lo tanto, una **mejor usabilidad al usuario**. Los contenidos (*layouts*), las imágenes, las tipografías, los vídeos, los *plugin*s y la usabilidad en general se vuelven fluidos gracias al código *media queries*.

Diseño web responsive vs. Diseño móvil

El diseño móvil y el diseño responsive son dos enfoques diferentes para adaptar un sitio web a dispositivos móviles.

El diseño móvil implica crear una versión completamente separada y específica del sitio web solo para dispositivos móviles. Este enfoque implica la implementación de una versión separada de la web para cada tamaño de pantalla de dispositivo móvil, lo que puede resultar en un desarrollo costoso y un mantenimiento complejo.

Para los profesionales del desarrollo web, el diseño responsive es ya un viejo amigo. En términos generales, "diseño web responsive" significa que el contenido del sitio web se adapta automáticamente al dispositivo desde el que se navega. Gracias al diseño responsive, la web se ajusta de manera fluida y dinámica para adaptarse a la pantalla del dispositivo, sin importar su tamaño.

Con la rápida difusión del uso de los smartphones, el diseño responsive se ha convertido en el diseño *mobile first* (Rigotti, 2023)

¿Qué es mobile first?

Según Developer Mozilla (2023), “es una forma de mejora progresiva, es un enfoque de desarrollo y diseño web que se enfoca en la priorización del diseño y el desarrollo para dispositivos móviles por encima del diseño y desarrollo par pantallas de escritorio”.

UML

El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semánticamente y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de *software* complejos, tanto en estructura como en comportamiento. UML tiene aplicaciones más allá del desarrollo de *software*, por ejemplo, en el flujo de procesos en la fabricación.

Es comparable a los planos usados en otros campos y consiste en diferentes tipos de diagramas. En general, los diagramas UML describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene.

UML no es un lenguaje de programación, pero existen herramientas que se pueden usar para generar código en diversos lenguajes usando los diagramas UML. UML guarda una relación directa con el análisis y el diseño orientados a objetos (Lucidchart, 2024).

Técnicas de levantamiento de requerimientos de *software*

El levantamiento de requerimientos se refiere a la identificación y documentación de los requerimientos de un sistema, a partir de los usuarios, clientes o interesados (*stakeholders*). A esta práctica también se le conoce como recopilación de requerimientos (PMO Informática, 2016).

1. Análisis de documentación
    1. Consiste en obtener la información sobre los requerimientos funcionales y requerimientos no funcionales de *software* a partir de documentos que ya están elaborados.
    2. Es útil cuando los expertos en la materia no están disponibles para ser entrevistados o ya no forman parte de la organización.
    3. Utiliza la documentación que sea relevante al requerimiento que se está levantando.
    4. Ejemplos de documentación: Planes de negocio, actas de constitución de proyecto, reglas de negocio, contratos, definiciones de alcance, memorándums, correos electrónicos, documentos de entrenamiento, entre otros.
2. Observación
    1. Consiste en estudiar el entorno de trabajo de los usuarios, clientes e interesados de proyecto (*stakeholders*).

2. Es una técnica útil cuando se está documentando la situación actual de procesos de negocio.

Puede ser de dos tipos, pasiva o activa.

1. En observación pasiva, el observador no hace preguntas, limitándose solo a tomar notas y a no interferir en el desempeño normal de las operaciones.
2. En observación activa, el observador puede conversar con el usuario.

3. Entrevistas
    1. Se realizan con los usuarios o interesados clave.
    2. Direccionan al usuario hacia aspectos específicos del requerimiento a levantar.
    3. Son útiles para obtener y documentar información detallada sobre los requerimientos y sus niveles de división.
    4. Pueden ser entrevistas formales o informales.
    5. Una clave es mantenerse enfocado en los objetivos de la entrevista.
    6. Las preguntas abiertas son útiles para identificar información faltante.
    7. Las preguntas cerradas son útiles para confirmar y validar información.
    8. El éxito de las entrevistas depende del grado de conocimiento del entrevistador y entrevistado, disposición del entrevistado de suministrar información, buena documentación de la discusión y en definitiva de una buena relación entre las partes.
4. Encuestas o cuestionarios
    1. Es una técnica útil para recopilar eficientemente los requerimientos de muchas personas.

2. La clave para el éxito es que tengan un propósito y audiencia claramente definida, establecer fechas topes para llenar la encuesta, con preguntas claras y concisas.
3. Deben enfocarse en los objetivos de negocio que se necesitan identificar.
4. Pueden apoyarse con entrevistas de seguimiento con usuarios individuales.
5. Pueden contener tanto preguntas cerradas como preguntas abiertas.

5. Mesas de trabajo (*workshops*)
    1. Es una técnica efectiva para obtener información rápidamente de varias personas.
    2. Es recomendable tener una agenda predefinida y preseleccionar a los participantes, siguiendo buenas prácticas para reuniones efectivas.
    3. Se puede utilizar un facilitador neutral y un transcriptor (que no sea el mismo facilitador).
    4. Se puede utilizar un material común sobre el cual enfocar la atención y conversar, por ejemplo, una presentación con un desglose del proceso que se está estudiando o un flujograma.
    5. Se pueden combinar con otras técnicas como pueden ser las entrevistas y cuestionarios.
6. Tormenta de ideas
    1. Es una sesión de trabajo estructurada orientada para obtener la mayor cantidad de ideas posibles.
    2. Es recomendable limitarlas en el tiempo, utilizar ayudas visuales y designar un facilitador.

3. Las reglas son importantes, por ejemplo, los criterios para evaluar ideas y asignarles un puntaje, no permitir las críticas a las ideas y limitar el tiempo de discusión.
4. En una primera fase, se deben identificar la mayor cantidad de ideas, para luego evaluarlas. Todas las ideas deben ser consideradas y deben limitarse que una idea se le ahogue o critique antes de tener tiempo de desarrollarla.

7. Historia del usuario
    1. Las historias de usuario son una aproximación simple al levantamiento de requerimientos de *software*, en la cual la conversación pasa a ser más importante que la formalización de requerimientos escritos.
    2. Es recomendable que sean escritas por el mismo cliente o interesado (con apoyo del facilitador si es necesario), con énfasis en las funcionalidades que el sistema deberá realizar.
    3. Al redactar una historia de usuario deben tenerse en cuenta describir el Rol, la funcionalidad y el resultado esperado de la aplicación en una frase corta.
    4. Las historias de usuario son una de las técnicas más difundidas para levantar requerimientos de *software* en metodologías ágiles.

Requerimientos no funcionales

Todos los servicios de TI, en algún punto de su ciclo de vida, necesitan considerar los requerimientos no funcionales y las pruebas asociadas a los mismos.

Para algunos proyectos, estos requerimientos implican una cantidad considerable de trabajo y esfuerzo, mientras que para otros no.

Los requerimientos no funcionales son los que especifican criterios para evaluar la operación de un servicio de tecnología de la información, en contraste con los requerimientos funcionales que especifican los comportamientos específicos.

Por lo general, el plan para implementar los requerimientos no funcionales se detalla en la arquitectura del sistema, mientras que el de los requerimientos funcionales se especifica en el diseño.

Requerimientos funcionales

Se refieren a cómo debe comportarse un sistema. Definen lo que el sistema debe hacer para satisfacer las necesidades o expectativas del usuario. Los requerimientos funcionales se pueden considerar como características que el usuario percibe. Son diferentes de los requerimientos no funcionales, que definen cómo debe funcionar internamente el sistema (por ejemplo, rendimiento, seguridad, etc.).

Los requerimientos funcionales se componen de dos partes: función y comportamiento. La función es lo que hace el sistema (por ejemplo, "calcular el impuesto sobre las ventas"). El comportamiento es cómo lo hace el sistema (por ejemplo, "El sistema calculará el impuesto sobre las ventas multiplicando el precio de compra por la tasa impositiva").

Al crear requerimientos funcionales, es importante tener en cuenta que deben ser específicos, medibles, alcanzables, relevantes y limitados en el tiempo. En otras palabras, sus requerimientos funcionales deben:

1. Sea específico sobre lo que debe hacer el sistema
2. Ser medible para que pueda saber si el sistema lo está haciendo.

3. Ser alcanzable dentro del marco de tiempo que ha establecido
4. Sea relevante para sus objetivos comerciales
5. Tener un límite de tiempo para que pueda seguir el progreso

Figura 27

Requisitos funcionales y no funcionales

Fuente: (Ken, 2023)

Metodologías de desarrollo de *software*

De acuerdo con Pérez (2016):

Existe la creencia extendida de que los programas *software* tienen que ver solamente con los ordenadores y las grandes computadoras.

Sin embargo, la ingeniería del *software* va mucho más allá. Se trata del proceso cuya finalidad es desarrollar productos o soluciones para un cliente o mercado en particular, teniendo en cuenta factores como los costes, la planificación, la calidad y las dificultades asociadas**.** A todo esto, es a lo que denominamos metodologías de desarrollo de *software*. Es decir, se trata del proceso que se suele seguir a la hora de diseñar una solución o un programa específico. Tiene que ver, por tanto, con la comunicación, la manipulación de modelos y el intercambio de información y datos entre las partes involucradas. O para ser más precisos, las metodologías de desarrollo de *software* son enfoques de carácter estructurado y estratégico que permiten el desarrollo de programas con base a modelos de sistemas, reglas, sugerencias de diseño y guías.

Modelo de desarrollo tradicionales

1. Cascada: La metodología de cascada es un enfoque lineal y secuencial en el que las etapas del desarrollo de *software* se llevan a cabo de forma secuencial, como una cascada. Cada fase depende de la finalización exitosa de la fase anterior y no permite cambios retrospectivos. Esta metodología es ideal para proyectos con requisitos bien definidos y estables desde el principio.
2. Prototipos: se basa en la construcción de un prototipo de *software* que se construye rápidamente para que los usuarios puedan probarlo y aportar *feedback.* Así, se puede arreglar lo que está mal e incluir otros requerimientos que puedan surgir. Es un modelo

iterativo que se basa en el método de prueba y error para comprender las especificidades del producto.

3. Espiral: es una combinación de los dos modelos anteriores, que añade el concepto de análisis de riesgo. Se divide en cuatro etapas: planificación, análisis de riesgo, desarrollo de prototipo y evaluación del cliente. El nombre de esta metodología da nombre a su funcionamiento, ya que se van procesando las etapas en forma de espiral. Cuanto más cerca del centro se está, más avanzado está el proyecto.
4. Incremental: en esta metodología de desarrollo de *software* se va construyendo el producto final de manera progresiva. En cada etapa incremental se agrega una nueva funcionalidad, lo que permite ver resultados de una forma más rápida en comparación con el modelo en cascada. El *software* se puede empezar a utilizar incluso antes de que se complete totalmente y, en general, es mucho más flexible que las demás metodologías.
5. Diseño rápido de aplicaciones (RAD): esta metodología permite desarrollar *software* de alta calidad en un corto periodo de tiempo. Los costos son mucho más altos y el desarrollo más flexible, aunque requiere una mayor intervención de los usuarios. Por otro lado, el código puede contener más errores, y sus funciones son limitadas debido al poco tiempo del que se dispone para desarrollarlas. El objetivo es iterar el menor número posible de veces para conseguir una aplicación completa de forma rápida.

Modelo de desarrollo de software agiles

Las metodologías ágiles de desarrollo de *software* son las más utilizadas hoy en día debido a su alta flexibilidad y agilidad. Los equipos de trabajo que las utilizan son mucho más productivos y eficientes, ya que saben lo que tienen que hacer en cada momento. Además, la

metodología permite adaptar el *software* a las necesidades que van surgiendo en el camino, lo que facilita la construcción de aplicaciones más funcionales.

Las metodologías ágiles se basan en la metodología incremental, en la que en cada ciclo de desarrollo se van agregando nuevas funcionalidades a la aplicación final. Sin embargo, los ciclos son mucho más cortos y rápidos, por lo que se agregan pequeñas funcionalidades en lugar de grandes cambios.

Este tipo de metodologías permite construir equipos de trabajo autosuficientes e independientes que se reúnen cada poco tiempo para poner en común las novedades. Poco a poco, se va construyendo y puliendo el producto final, a la vez que el cliente puede ir aportando nuevos requerimientos o correcciones, ya que puede comprobar cómo avanza el proyecto en tiempo real.

Las principales metodologías ágiles son:

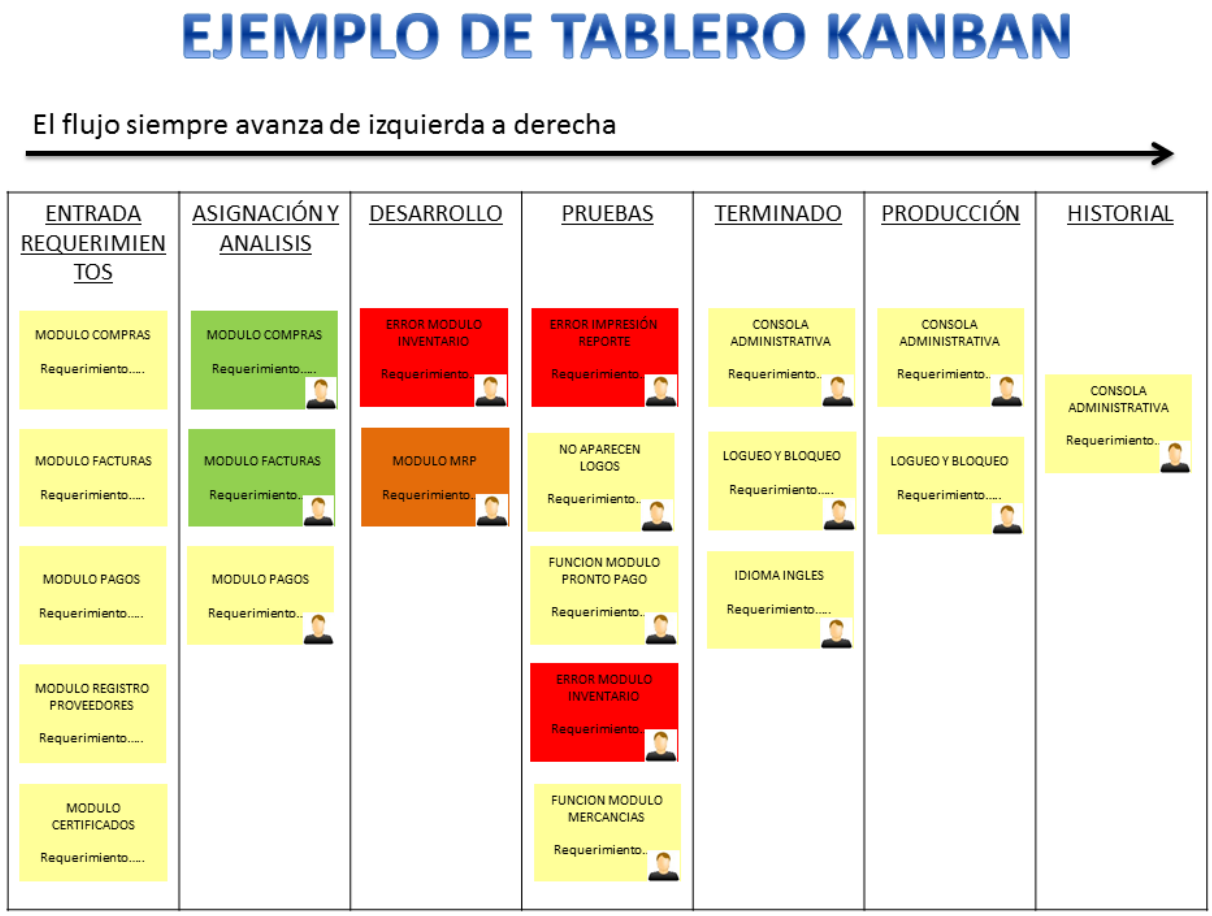
1. Kanban: metodología de trabajo inventada por la empresa de automóviles Toyota. Consiste en dividir las tareas en porciones mínimas y organizarlas en un tablero de trabajo dividido en tareas pendientes, en curso y finalizadas. De esta forma, se crea un flujo de trabajo muy visual basado en tareas prioritarias e incrementando el valor del producto.
2. Scrum: es también una metodología incremental que divide los requisitos y tareas de forma similar a Kanban. Se itera sobre bloques de tiempos cortos y fijos (entre dos y cuatro semanas) para conseguir un resultado completo en cada iteración. Las etapas son: planificación de la iteración (*planning sprint*), ejecución (*sprint*), reunión diaria (*daily*

meeting) y demostración de resultados (*sprint review*). Cada iteración por estas etapas se denomina también *sprint*.

3. Lean: está configurado para que pequeños equipos de desarrollo muy capacitados elaboren cualquier tarea en poco tiempo. Los activos más importantes son las personas y su compromiso, relegando así a un segundo plano el tiempo y los costes. El aprendizaje, las reacciones rápidas y potenciar el equipo son fundamentales.
4. Programación extrema (XP): es una metodología de desarrollo de *software* basada en las relaciones interpersonales, que se consideran la clave del éxito. Su principal objetivo es crear un buen ambiente de trabajo en equipo y que haya un *feedback* constante del cliente. El trabajo se basa en 12 conceptos: diseño sencillo, *testing*, refactorización y codificación con estándares, propiedad colectiva del código, programación en parejas, integración continua, entregas semanales e integridad con el cliente, cliente in situ, entregas frecuentes y planificación.

Figura 28

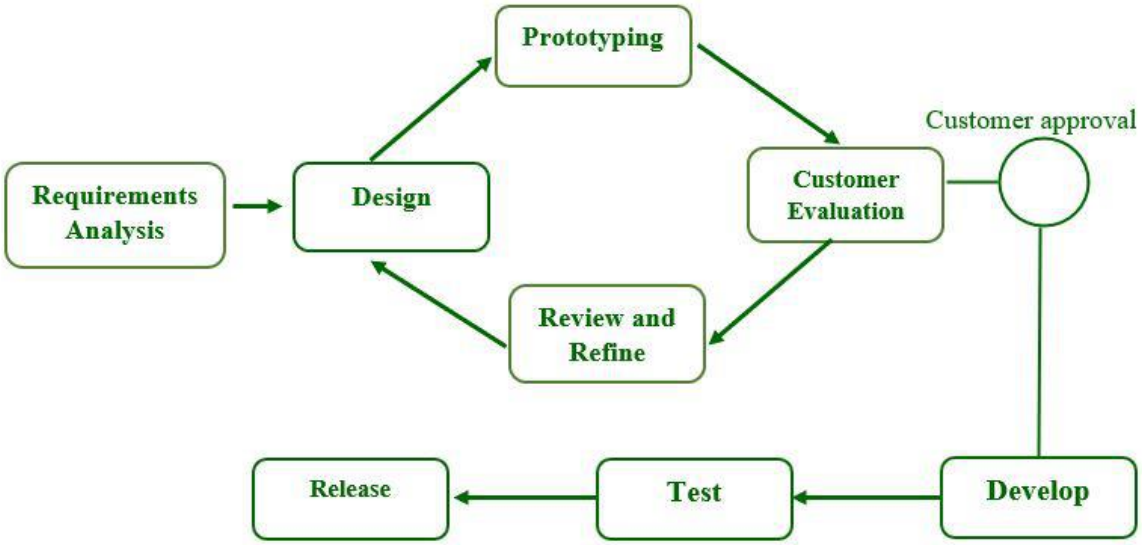
Modelo y fases de creación de programación kanban



Fuente:

Figura 29

Modelo y fases de creación de prototipos de software



Fuente:

Figura 30

Modelo de programación extrema (XP)

Fuente: **(Zoluxiones, 2022)**

Figura 31

Metodologías de desarrollo de software

Fuente: **(Universidades Santander, 2020)**

Algoritmos de programación

De acuerdo con Córdoba (2023), “Los algoritmos de programación son un conjunto de instrucciones ordenadas y estructuradas que permiten a una computadora realizar una tarea específica”. Algunas de sus características son:

1. Son secuenciales: Deben procesarse uno a la vez, comenzando por las primeras instrucciones y avanzando linealmente hasta las últimas.
2. Son precisos y específicos: Las instrucciones que los componen no puede ser ambiguas o subjetivas, sino directas, fáciles de seguir y lo menos generales posibles.
3. Son ordenados: Deben leerse en un orden específico para que tengan algún sentido. Colocar un algoritmo o un elemento del algoritmo en el lugar equivocado puede invalidar a los demás.
4. Son finitos: Tienen un inicio y un final determinados.
5. Son definidos: Un mismo algoritmo debe dar siempre los mismos resultados si es alimentado por los mismos elementos.

Partes de un algoritmo

1. *Input* o entrada: Contiene las instrucciones iniciales, en las que se ingresan los datos que el algoritmo necesita para operar.
2. Proceso o instrucciones: Este compuesto por las operaciones lógicas que el algoritmo emprenderá con lo recibido del input.
3. *Output* o salida: Son los resultados obtenidos luego del proceso, una vez terminada la ejecución del algoritmo.

Los algoritmos se definen a partir de varios criterios.

Según los procesos que se requieran:

1. Algoritmos computacionales: Son aquellos cuya resolución puede llevarse a cabo mediante una calculadora o computadora.

2. Algoritmos no computacionales: Son aquellos que no dependen del cálculo y no requieren de los procesos de una computadora para resolverse.

Según las secuencias que intervengan:

1. Algoritmos cualitativos: Son aquellos en cuya resolución no intervienen cálculos numéricos, sino secuencias lógicas y/o formales.
2. Algoritmos cuantitativos: Son aquellos que dependen de cálculos matemáticos para dar con su resolución.

Según su propósito:

1. Algoritmos de búsqueda: Son aquellos que permiten ubicar elementos de rasgos específicos dentro de un conjunto de datos.
2. Algoritmos de ordenamiento: Son aquellos que permiten organizar un conjunto de datos de acuerdo con un criterio especifico.
3. Algoritmos predictivos: Son aquellos que permiten hacer proyecciones lógicas futuras de un problema, es decir, buscar opciones probables de input.
4. Algoritmos probabilísticos: son aquellos que permiten obtener un resultado azaroso dentro de un conjunto de datos establecidos.
5. Algoritmos de optimización: Son aquellos que buscan hacer más eficiente un proceso determinado, y para lograrlo, buscan alternativas a los elementos de un conjunto de datos (Raffino, Equipo editorial, Etecé, 2024)**.**

Pruebas de *software*

Las pruebas de rendimiento son imprescindibles en todos los entornos de desarrollo y producción para garantizar que su sitio web o aplicación esté al día y pueda soportar la carga de usuarios esperada. Las pruebas funcionales deben realizarse con cada compilación para validar todos los cambios y funcionalidades con respecto a las especificaciones y requisitos. Las pruebas de integración deben realizarse al integrar un nuevo fragmento de código con algún otro módulo para asegurarse de que no haya conflictos y que ambos trabajen correctamente juntos. Las pruebas unitarias deben realizarse siempre que se termine de escribir cualquier código para validar la entrada y salida correctas.

Existen diferentes tipos de pruebas, como se muestra a continuación:

1. Pruebas unitarias

   Las pruebas unitarias son de muy bajo nivel y se realizan cerca de la fuente de la aplicación. Consisten en probar métodos y funciones individuales de las clases, componentes o módulos que usa tu *software*. En general, las pruebas unitarias son bastante baratas de automatizar y se pueden ejecutar rápidamente mediante un servidor de integración continua.

2. Pruebas de integración

   Las pruebas de integración verifican que los distintos módulos o servicios utilizados por tu aplicación funcionan bien en conjunto. Por ejemplo, se puede probar la interacción con la base de datos o asegurarse de que los microservicios funcionan bien en conjunto y

según lo esperado. Estos tipos de pruebas son más costosos de ejecutar, ya que requieren que varias partes de la aplicación estén en marcha.

3. Pruebas funcionales

Las pruebas funcionales se centran en los requisitos empresariales de una aplicación. Solo verifican el resultado de una acción y no comprueban los estados intermedios del sistema al realizar dicha acción.

A veces, se confunden las pruebas de integración con las funcionales, ya que ambas requieren que varios componentes interactúen entre sí. La diferencia es que una prueba de integración puede simplemente verificar que puedes hacer consultas en la base de datos, mientras que una prueba funcional esperaría obtener un valor específico desde la base de datos, según dicten los requisitos del producto.

4. Pruebas de extremo a extremo

Las pruebas integrales replican el comportamiento de un usuario con el *software* en un entorno de aplicación completo. Además, verifican que diversos flujos de usuario funcionen según lo previsto, y pueden ser tan sencillos como cargar una página web o iniciar sesión, o mucho más complejos, como la verificación de notificaciones de correo electrónico, pagos en línea, etc.

Las pruebas integrales son muy útiles, pero son costosas de llevar a cabo y pueden resultar difíciles de mantener cuando están automatizadas. Se recomienda tener algunas

pruebas integrales clave y depender más de pruebas de menor nivel (unitarias y de integración) para poder detectar rápidamente nuevos cambios

5. Pruebas de aceptación

   Las pruebas de aceptación son pruebas formales que verifican si un sistema satisface los requisitos empresariales. Requieren que se esté ejecutando toda la aplicación durante las pruebas y se centran en replicar las conductas de los usuarios. Sin embargo, también pueden ir más allá y medir el rendimiento del sistema y rechazar cambios si no se han cumplido determinados objetivos.

6. Pruebas de rendimiento

   Las pruebas de rendimiento evalúan el rendimiento de un sistema con una carga de trabajo determinada. Ayudan a medir la fiabilidad, la velocidad, la escalabilidad y la capacidad de respuesta de una aplicación. Por ejemplo, una prueba de rendimiento puede analizar los tiempos de respuesta al ejecutar un gran número de solicitudes, o cómo se comporta el sistema con una cantidad significativa de datos. Puede determinar si una aplicación cumple con los requisitos de rendimiento, localizar cuellos de botella, medir la estabilidad durante los picos de tráfico y mucho más.

7. Pruebas de humo

   Las pruebas de humo son pruebas básicas que sirven para comprobar el funcionamiento básico de la aplicación. Están concebidas para ejecutarse rápidamente, y su objetivo es

ofrecerte la seguridad de que las principales funciones de tu sistema funcionan según lo previsto.

Las pruebas de humo pueden resultar útiles justo después de realizar una compilación nueva para decidir si se pueden ejecutar o no pruebas más caras, o inmediatamente después de una implementación para asegurarse de que la aplicación funciona correctamente en el entorno que se acaba de implementar.

Las pruebas de *software* se realizan de acuerdo con ciertas circunstancias y dependiendo del objetivo o alcance de la prueba que necesitemos realizar, nos enfocaremos más en unas que en otras.

Así que son importantes si se quiere asegurar que los cambios en el código y todo lo que se desarrolla funcione tal y como se espera. Realizar pruebas para entender el comportamiento del sistema será siempre de utilidad.

Capítulo III. Marco metodológico

¿Qué es un marco metodológico?

Un marco metodológico es una estructura o conjunto de reglas y principios que guían el proceso de investigación. Es como un mapa que ayuda a los investigadores a planificar y llevar a cabo su estudio de manera organizada y sistemática. Algunos puntos clave son:

1. Diseño de investigación: El marco metodológico incluye el diseño general de la investigación, es decir, la manera en que se va a recopilar y analizar la información. Por ejemplo, si es un estudio experimental, observacional, de caso, etc.
2. Métodos de recopilación de datos: Describe las herramientas y técnicas que se utilizarán para recopilar la información. Esto puede incluir encuestas, entrevistas, experimentos, análisis de documentos, entre otros.
3. Población y muestra: Indica quiénes son los participantes o elementos de estudio (población) y cómo se seleccionará un subconjunto representativo (muestra) para realizar la investigación.
4. Procedimiento: Detalla los pasos específicos que se seguirán para llevar a cabo la investigación. Esto ayuda a asegurar la consistencia y replicabilidad del estudio.
5. Instrumentos de medición: Si se utilizan encuestas, *tests* u otros instrumentos, el marco metodológico describe cómo se desarrollarán y validarán para garantizar su fiabilidad.
6. Análisis de datos: Explica cómo se procesarán y analizarán los datos recopilados. Esto puede incluir el uso de *software* estadístico, métodos cualitativos, o una combinación de ambos, según el tipo de estudio.

7. Ética: Considera aspectos éticos, como la privacidad y el consentimiento de los participantes, para garantizar que la investigación se realice de manera ética y respetuosa.

Enfoque de la investigación

El enfoque de la investigación se refiere a la perspectiva o marco teórico desde el cual se abordará el tema de estudio. Es la forma en que se planifica y se organiza la investigación, estableciendo los objetivos, las preguntas de investigación, la metodología y el análisis de datos.

El enfoque de la investigación puede ser cualitativo, cuantitativo o mixto, dependiendo de la naturaleza del problema de investigación y de los objetivos de la tesis. También puede incluir un enfoque teórico, metodológico, empírico o práctico, según los elementos que se quieran destacar en la investigación.

El enfoque de la investigación es fundamental para definir la orientación y el alcance del trabajo de investigación, permitiendo una estructuración coherente y lógica de todos los elementos que la conforman.

El modelo de investigación que se utilizará en el desarrollo de este proyecto será el de investigación mixta, la cual combina tanto métodos cuantitativos como cualitativos en un mismo estudio. Esto significa que se utilizan tanto enfoques numéricos para recopilar y analizar datos, como enfoques descriptivos y subjetivos para comprender fenómenos en profundidad.

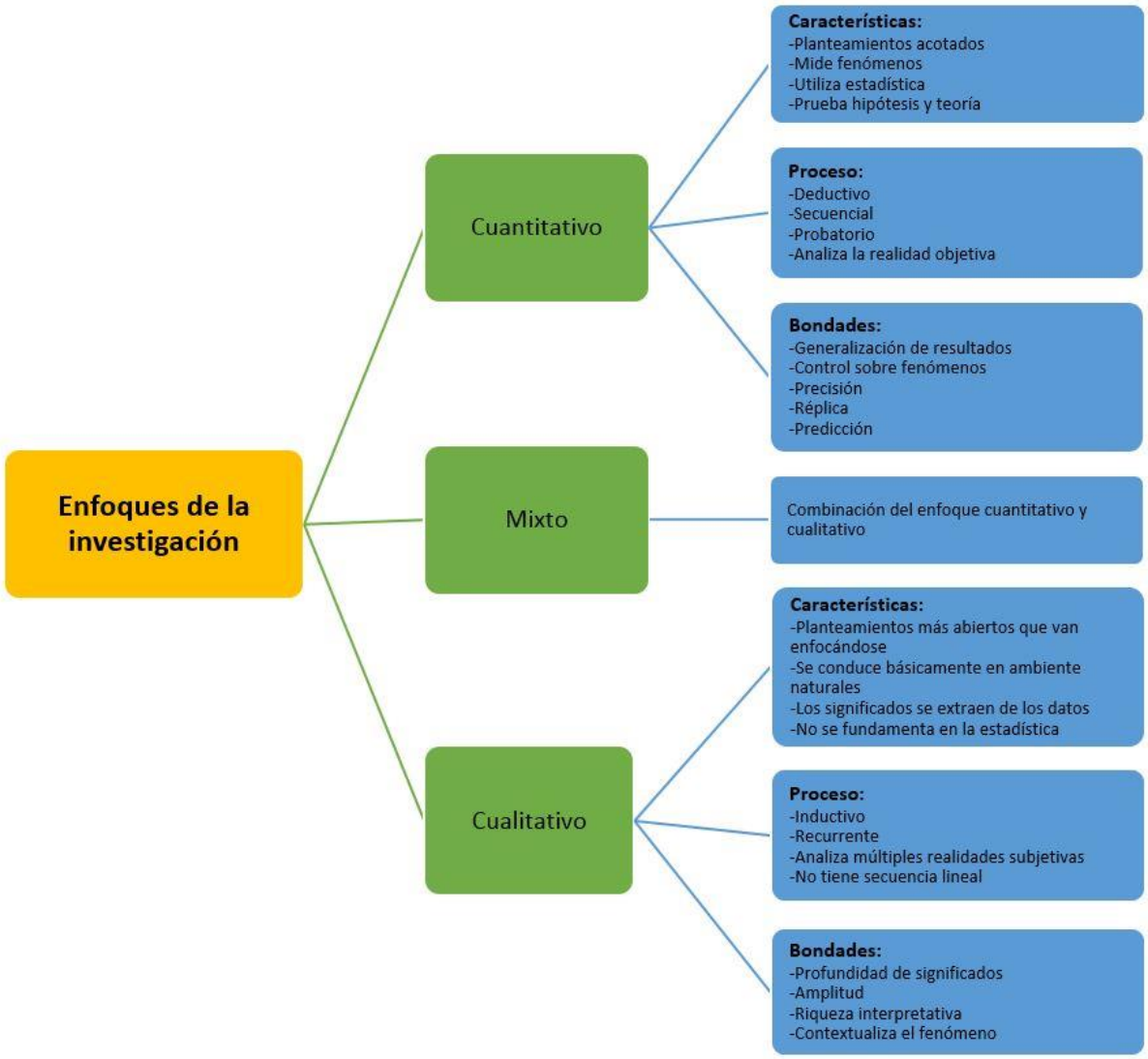
En la investigaSíción mixta, se pueden utilizar herramientas como encuestas, experimentos, análisis estadístico, entrevistas, grupos focales, observaciones, entre otros, con el

fin de obtener una perspectiva más amplia y completa sobre el tema de estudio. Esta metodología se utiliza cuando se quiere explorar un problema de investigación desde diferentes ángulos y se busca tanto la generalización de resultados como una comprensión más profunda de los aspectos cualitativos y subjetivos del fenómeno estudiado.

La investigación mixta ofrece la posibilidad de complementar y enriquecer los resultados obtenidos, al combinar la rigurosidad y objetividad de los métodos cuantitativos con la riqueza y profundidad de los métodos cualitativos. Es una metodología que combina lo mejor de ambos enfoques para proporcionar un panorama más completo y detallado del problema que se presenta en esta investigación.

Figura 32

Los enfoques cuantitativo y cualitativo de la investigación



Fuente: (Barriosó, 2024)

La investigación cualitativa se caracteriza por el uso de métodos como la observación participante, entrevistas en profundidad, grupos focales y análisis de contenido. Estos métodos permiten la recolección de datos en contextos naturales y la interpretación de los significados y experiencias de los participantes.

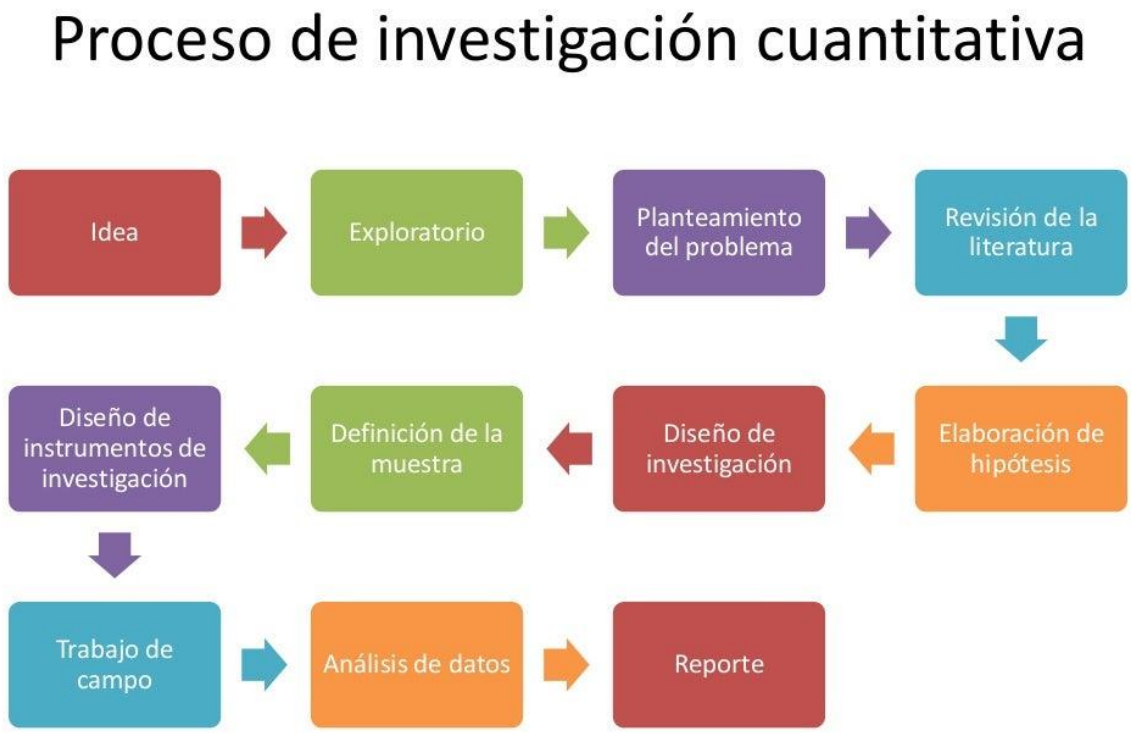
Por otro lado, la investigación cuantitativa se basa en la recolección de datos numéricos a través de encuestas, cuestionarios, pruebas y mediciones objetivas. Los métodos utilizados incluyen el análisis estadístico, la experimentación y la encuesta a gran escala. Ambas formas de investigación pueden complementarse entre sí, permitiendo una comprensión más completa de un fenómeno o problemática.

Replicar los resultados

1. Utiliza métodos de recolección de datos estructurados, como cuestionarios, encuestas y experimentos.
2. Busca establecer relaciones causales entre variables a través de análisis estadísticos.
3. Permite generalizar los resultados a poblaciones más amplias a partir de muestras representativas.
4. Se caracteriza por su rigurosidad y precisión en la recopilación y análisis de datos.
5. Busca identificar patrones y tendencias a partir de los datos recogidos.
6. Su diseño de investigación es deductivo, partiendo de hipótesis previas que se someten a prueba con los datos recopilados.

Figura 33

Proceso de investigación cuantitativa



Fuente: (UANL, 2024)

El enfoque cuantitativo tiene las siguientes características:

1. Utiliza datos numéricos y estadísticas para recopilar, analizar y presentar los resultados de la investigación.

2. Se centra en la objetividad y de poder

El enfoque cualitativo tiene las siguientes características:

1. Utiliza métodos de recolección de datos que permiten una comprensión detallada y profunda del fenómeno estudiado.

2. Se centra en la interpretación y comprensión de las experiencias, percepciones y significados de los participantes.

3. Se basa en la flexibilidad y la capacidad de adaptación durante la investigación.

4. Busca generar teorías emergentes a partir de los datos recopilados.

5. Se enfoca en contextos específicos y situaciones particulares.

6. Utiliza una aproximación inductiva en la obtención y análisis de datos.

7. Reconoce la subjetividad del investigador y la influencia de éste en el proceso de investigación.

8. Utiliza una gran variedad de técnicas de recolección de datos, como entrevistas, observación participante y análisis de documentos.

9. Busca dar voz a los participantes y entender su punto de vista desde una perspectiva holística.

10. Prioriza la validez interna y la credibilidad de los resultados obtenidos.

Figura 34

Métodos de investigación cualitativa

Fuente: (Infoupdate, s.f.)

El enfoque mixto tiene las siguientes características:

1. Integración de métodos: se utilizan tanto métodos cuantitativos como cualitativos, combinando datos numéricos y descriptivos para obtener una comprensión más completa del fenómeno estudiado.

2. Flexibilidad: este enfoque permite adaptarse a las necesidades de la investigación, ya que ofrece la posibilidad de modificar la estrategia de investigación en función de los resultados preliminares o del desarrollo del estudio.

3. Triangulación de datos: se busca fortalecer la validez de los hallazgos al comparar y contrastar los resultados obtenidos a partir de diferentes métodos de recolección de datos.

4. Profundidad y amplitud: el enfoque mixto permite explorar un problema de investigación desde distintas perspectivas, lo que facilita una comprensión más profunda y detallada del fenómeno estudiado.

5. Diseño secuencial o concurrente: se pueden realizar las fases de recolección y análisis de datos de manera secuencial, primero cuantitativa y luego cualitativa, o de manera concurrente, combinando ambos enfoques en el mismo estudio.

6. Validación cruzada: se pueden utilizar los resultados de un método para validar los hallazgos obtenidos con el otro método, lo que contribuye a la robustez de las conclusiones de la investigación.

7. Aportes complementarios: el enfoque mixto permite combinar las fortalezas de los métodos cuantitativos y cualitativos, lo que puede enriquecer la comprensión de un problema de investigación y generar un conocimiento más completo y significativo.

8. Enfoque holístico: al integrar diferentes enfoques metodológicos, se busca tener una visión global del fenómeno estudiado, considerando tanto los aspectos cuantitativos como cualitativos y sus interacciones.

Método de investigación

El método de investigación es un proceso organizado y sistemático para obtener información y conocimiento sobre un tema específico. Se basa en un conjunto de pasos y técnicas que permiten recolectar, analizar e interpretar datos de manera objetiva y confiable. El método de investigación se utiliza en diversas disciplinas académicas y científicas para abordar

preguntas de investigación, resolver problemas y llegar a conclusiones basadas en evidencias. El método de investigación que se aplicará en el presente proyecto sería el marco ágil Scrum.

Fuentes de información

Alrespecto, Ulate Soto y Vargas Morúa (2014) indican que:

> Se deben presentar de manera separadas todas las fuentes consultadas, clasificadas en fuentes primarias, secundarias y terciarias, junto con una breve descripción de cada una. Se presenta una explicación general de las fuentes, o bien de algún autor en particular por su relevancia para la investigación.

Las fuentes deben agruparse por afinidad:

1. Registros internos de la organización. Se incluyen todos aquellos documentos internos consultados. Por ejemplo: estados contables, registros de ventas, datos de producción, etcétera.
2. Periódicos y revistas. Revisión e incorporación de datos provenientes de este tipo de publicaciones.
3. Publicaciones de la organización y afines. Contempla la revisión de publicaciones hechas por la entidad sujeto de estudio, como memorias anuales, datos del sector, informes técnicos, entre otros.
4. Fuentes electrónicas. Describe los documentos o las páginas consultadas a través de internet.
5. Otras publicaciones. Incluye libros y otras publicaciones, tales como estudios médicos y bitácoras de ingeniería o de ventas.

Variables

Una variable de investigación o variable de estudio es un término que se utiliza para referirse a cualquier tipo de relación de causa y efecto. Representa un atributo medible que cambia a lo largo de un experimento comprobando los resultados. Estos atributos cuentas con diferentes medias, dependiendo tanto de las variables, del contexto del estudio o de los límites que los investigadores consideren.

Tabla 3

Cuadro de variables

Objetivo General	Objetivo Específico	Variable	Dimensión	Indicador	Definición Conceptual
Asegurar la calidad del software	Definir lo que el sistema debe hacer	Requerimientos Funcionales	Funciones específicas	Número de funciones implementadas	Especificaciones que describen las funciones y comportamientos que el sistema debe poseer para satisfacer las necesidades de los usuarios y cumplir con los objetivos del negocio.
			Procesos y flujos de trabajo	Número de procesos automatizados	
			Interacciones del usuario	Facilidad de uso de la interfaz	
			Validaciones y restricciones	Número de validaciones implementadas	
			Interoperabilidad	Número de integraciones con otros sistemas	
Mejorar la eficiencia y satisfacción del sistema	Establecer criterios de calidad y desempeño del sistema	Requerimientos No Funcionales	Rendimiento	Tiempo de respuesta promedio	Especificaciones que describen cómo debe comportarse un sistema en términos de calidad y desempeño, en lugar de lo que debe hacer.
			Fiabilidad	Tasa de disponibilidad	
			Usabilidad	Satisfacción del usuario	
			Escalabilidad	Capacidad máxima de usuarios concurrentes	
			Seguridad	Número de incidentes de seguridad	
			Mantenibilidad	Tiempo promedio de reparación de errores	
			Portabilidad	Número de plataformas soportadas	
			Interoperabilidad	Cumplimiento de estándares	
Facilitar la descripción de interacciones	Definir los escenarios de interacción entre usuarios y el sistema	Casos de Uso	Escenarios	Número de casos de uso documentados	Descripciones detalladas de cómo los usuarios interactúan con el sistema para lograr objetivos específicos, proporcionando un marco claro para el desarrollo y prueba del software.
			Actores	Número de actores identificados	
			Flujos alternativos	Número de flujos alternativos descritos	
			Precondiciones	Número de precondiciones definidas	
			Postcondiciones	Número de postcondiciones definidas	
Garantizar la validación exhaustiva del software	Planificar y ejecutar pruebas que aseguren la calidad del software	Plan de Pruebas	Tipos de pruebas	Número de pruebas unitarias, de integración, etc.	Una guía detallada para asegurar la calidad del producto final mediante la realización de pruebas exhaustivas y efectivas antes de su lanzamiento.
			Cobertura de pruebas	Porcentaje de cobertura de código	
			Casos de prueba	Número de casos de prueba documentados	
			Herramientas de prueba	Número de herramientas utilizadas	
			Resultados de pruebas	Tasa de éxito y fracaso de los casos de prueba	

Fuente:

Requerimientos funcionales

Los requerimientos funcionales de un sistema son aquellos que describen cualquier actividad que este deba realizar; en otras palabras, el comportamiento o función particular de un sistema o *software* cuando se cumplen ciertas condiciones.

Los requerimientos funcionales expresan las capacidades o cualidades que debe tener la solución para satisfacer los requerimientos de los interesados del proyecto. Se expresan en términos de cuál debe ser el comportamiento de la solución y qué información debe manejar. Deben proporcionar una descripción lo suficientemente detallada para permitir el desarrollo o implementación de la solución. Son los que más influyen en si la solución será aceptada o no por los usuarios.

Tipos de requisitos funcionales:

1. Regulaciones comerciales
2. Requisitos de certificación
3. Los requisitos de información
4. Funciones administrativas
5. Niveles de autorización
6. Seguimiento de auditoria
7. Interfaces externas
8. Administración de datos
9. Requisitos legales y reglamentarios

Los requerimientos funcionales deben ser específicos, medibles, alcanzables, relevantes y limitados en el tiempo (Visure Solutions, 2024).

Requerimientos no funcionales

Los requerimientos no funcionales, también conocidos como “requisitos no funcionales” o “restricciones de calidad”, desempeñan un papel fundamental en el desarrollo de *software* en general. Son las restricciones o los requisitos impuestos al sistema que definen sus atributos de calidad. Se ocupan de aspectos como la escalabilidad, la mantenibilidad, el rendimiento, la portabilidad, la seguridad, la confiabilidad y muchos más. Los requerimientos no funcionales son importantes porque ayudan a garantizar que el sistema satisfaga las necesidades del usuario.

Ventajas de los requisitos no funcionales

1. Garantizar que el sistema satisfaga las necesidades del usuario.
2. Garantizar que el sistema sea adecuado para su propósito.
3. Garantizar que el sistema sea escalable, seguro y confiable.
4. Garantizar que el sistema sea fácil de usar y mantener.

Desventajas de los requisitos no funcionales

1. Pueden ser difíciles de entender e implementar.
2. Pueden llevar mucho tiempo y ser costosos de probar.
3. Pueden afectar la funcionalidad del sistema si no se implementan correctamente.

Subclasificación de requisitos no funcionales

1. Interfaz del usuario
2. Fiabilidad
3. Seguridad
4. Rendimiento
5. Mantenimiento
6. Estándares

Algunos ejemplos de requisitos no funcionales son lo siguientes:

1. Seguridad: el sistema debe estar protegido contra el acceso no autorizado
2. Actuación: Debe poderse manejar el numero requerido de usuarios sin ninguna degradación en el rendimiento.
3. Escalabilidad: El sistema debe ser capaz de escalar hacia arriba o hacia abajo según sea necesario.
4. Disponibilidad: El sistema debe estar disponible cuando sea necesario.
5. Mantenimiento: El sistema debe ser fácil de mantener y actualizar.
6. Portabilidad: El sistema debe poder ejecutarse en diferentes plataformas con cambios mínimos.
7. Fiabilidad: El sistema debe ser confiable y cumplir con los requisitos del usuario.
8. Usabilidad: El sistema debe ser fácil de usar y comprender.
9. Compatibilidad: El sistema debe ser compatible con otros sistemas.
10. Cumplimiento: El sistema debe cumplir con todas las leyes y reglamentos aplicables.

Plan de pruebas

Según nos define TDM (2023), un plan de pruebas de *software* es una guía detallada que asegura la calidad del producto final mediante la realización de pruebas exhaustivas y efectivas antes de su lanzamiento.

En un mercado en constante cambio y con creciente competencia, la calidad del *software* y la creación de una excelente experiencia de usuario (UX) son fundamentales. El plan de pruebas se convierte en un paso indispensable para que un *software* destaque entre los demás.

Este proceso genera una mayor confianza tanto en los usuarios como en la empresa desarrolladora, ya que garantiza que se han tomado todas las medidas necesarias para entregar un producto final de alta calidad.

Casos de uso

Los casos de uso se crean para refinar un conjunto de requisitos basados en un rol o tarea. En lugar de la lista tradicional de requisitos que pueden no abordar directamente el uso de la solución, los casos de uso agrupan comunes basados en el tipo de rol u objetivo. Definen lo que los usuarios o roles están haciendo en la solución, un proceso de negocio define como realizan estas funciones.

Figura 35

Ejemplo de un caso de uso

CU002 Registrar Novedad de Nómina a un Empleado	
Descripción	Este caso de uso deberá permitir el registro de una novedad de nómina a un empleado para un periodo específico.
Requerimiento Funcional Asociado	RF-01 Crear Módulo de Nómina
Casos de Uso Asociados	CU001 Consultar empleado
Prioridad	Media
Precondiciones	• El empleado debe registrar en estado activo dentro de la nómina.
Post condiciones	• Se deberá presentar en pantalla la confirmación del registro de la novedad a nombre del empleado. • La novedad registrada deberá quedar almacenada dentro de las novedades asociadas al empleado.
Criterios de Aceptación	• El acceso a la funcionalidad de registro de novedades se dará solo para empleados activos en la nómina. • La funcionalidad deberá permitir seleccionar el tipo de novedad que será registrada. • De acuerdo con el tipo de novedad seleccionada deberán habilitarse los campos adicionales definidos para el registro de la información específica. • La funcionalidad deberá permitir visualizar los mensajes de información, confirmación o error a que haya lugar.
Documentos Anexos	No Aplica

Fuente: (Departamento Nacional de Planeación, Colombia, 2020)

Casos de uso de una empresa

Los casos de uso son una herramienta importante para diseñar procesos y sistemas de una empresa.

1. Diagrama de casos de uso: Es una representación visual de las diferentes maneras y posibles escenarios de uso de un sistema. Muestra cómo los usuarios interactúan con el sistema, ya sea en un sitio web o en una aplicación.

Algunas de las ventajas de usar diagramas de casos de uso son las siguientes:

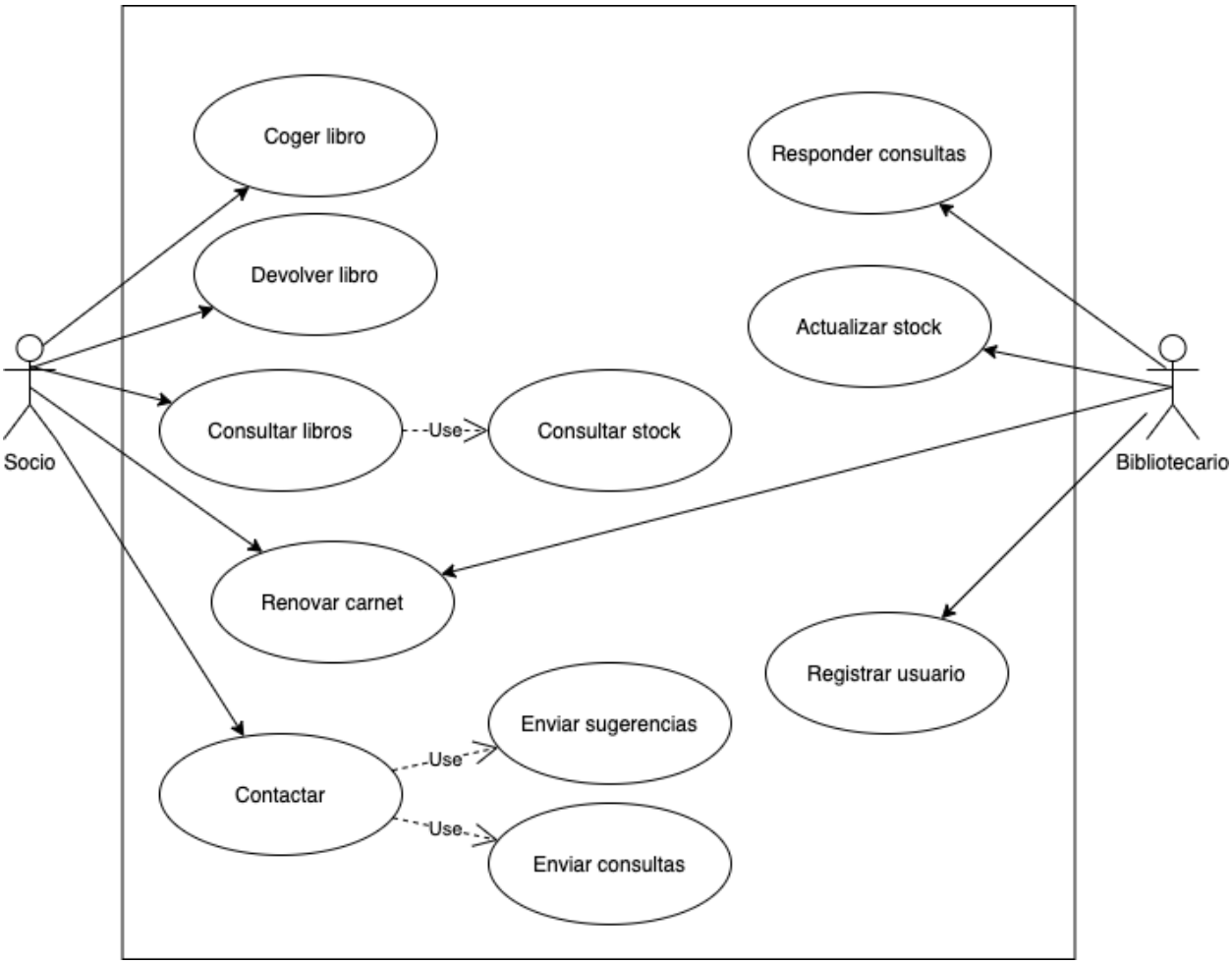
1. Guían el desarrollo: Ayudan a establecer el costo y la complejidad de los sistemas, especificando qué funciones deben desarrollarse.
2. Enfoque en el usuario: Escritos en lenguaje natural, los casos de uso son comprensibles para los usuarios y permiten una comunicación efectiva entre la empresa y sus clientes.
3. Simplifican las soluciones: Al desglosar las soluciones en funciones prácticas, los diagramas de casos de uso reducen la complejidad del problema que el sistema resuelve.

Seguidamente, se muestran algunos ejemplos de casos de uso empresariales, según Rodríguez Gómez (2018):

1. Caso IKEA: Diseño y venta de muebles y artículos para el hogar
2. Caso Nutella: Marketing y promoción de productos nutricionales
3. Caso Netflix: Cambio de reglas de la industria del entretenimiento global
4. Caso Tostao: Estrategia de marca cercana a los consumidores

Figura 36

Diagrama de casos de uso



Fuente: (Aabz Local, 2023)

Desarrollo de *software*

Según Stark cloud (2024):

El desarrollo de *software* es el proceso de crear y diseñar programas o aplicaciones informática mediante la escritura de código, diseño, pruebas y mantenimiento. Este

proceso implica una serie de pasos e instrucciones que los desarrolladores siguen para crear *software* funcional y de alta calidad.

El *software* es un conjunto de reglas o programas que le indican a la computadora qué acciones realizar.

Diferenciados de *software*

Para lograr sacar una diferencia de los competidores e implementar las innovaciones tecnológicas en los *softwares*, hay una serie de características que deberían estar presentes para lograr una mayor eficiencia, calidad de datos y procesos:

1. Inteligencia artificial (IA): La IA permite al *software* imitar la toma de decisiones de los humanos. Adquirir conocimiento o aprendizaje de estos temas permite al *software* tener una ventaja sobre el resto de la competencia. Esto se debe a la inclusión de la inteligencia artificial como complemento de las aplicaciones por medio de interfaces de programación de aplicaciones o API.
2. Desarrollo nativo en la nube: Posibilidad de crear aplicaciones para poder sacarle el mayor provecho y beneficio a los entornos de la nube. Una aplicación nativa contiene una serie de componentes discretos y reutilizables (microservicios), diseñados para que se puedan integrar a cualquier entorno en la nube. Debido a su diseño y arquitectura, estas aplicaciones pueden mejorar su rendimiento, escalabilidad, flexibilidad y extensibilidad.
3. Desarrollo basado en la nube: Los entornos de desarrollo de este tipo permiten una codificación, integración, diseño, pruebas y más funciones de desarrollo. También

brinda entrada a API, desarrollo de DevOps, microservicios y otras herramientas de servicios y experiencias.

Funcionamiento del desarrollo de software

Para lograr la correcta implementación y desarrollo de un *software* es necesario seguir y cumplir con una serie de requerimientos básicos:

Requisitos: El proceso comienza con la identificación de los requisitos del *software*, lo que implica comprender que busca el usuario, definir cuales características y funcionalidades debe tener la aplicación.

Diseño: Desarrolladores crean un diseño detallado del *software*, que incluye la arquitectura del sistema, interfaz de usuario, estructura de datos y otros aspectos técnicos. Esto determina la base inicial para la construcción y desarrollo del *software*.

Implementación: En esta fase, los programadores escriben el código fuente del *software* siguiendo el diseño previamente establecido y definido. Se utilizan lenguajes de programación y herramientas específicas para crear el *software* requerido.

Pruebas: Se somete el *software* a pruebas exhaustivas para identificar y corregir errores. Se asegura que funcione de acuerdo con los requisitos especificados en las etapas iniciales. Esto abarca pruebas de cohesión, pruebas de combinación y pruebas de aprobación del usuario.

Despliegue: Una vez que el *software* completa y pasa las pruebas requeridas, se lo considera listo para su uso y se implementa en un entorno de producción. Puede ser lanzado internamente en una empresa o públicamente para los usuarios.

Mantenimiento: El *software* se somete a un continuo mantenimiento para corregir errores, agregar nuevas características y mejorar el rendimiento. Esta fase no tiene límite de tiempo, depende de la vida útil del *software* y su funcionamiento.

Ejemplos de desarrollo de software

Existen una gran variedad de *softwares* y cada vez hay más para las distintas áreas o industrias. Estas llevan a un incremento en la eficiencia o mejora en las formas de trabajar dentro de las empresas:

Software de gestión empresarial: Los *softwares* permiten, en una única plataforma, gestionar, automatizar e integrar la variedad de procesos que suceden dentro de una empresa. Esto lleva a que puedan tomar decisiones eficientes con mejores y más relevantes datos e información en tiempo real.

Los sistemas de gestión más conocidos son:

1. CRM: administra datos de clientes y los potenciales, automatiza las ventas y *marketing*. Agiliza los procesos para todos los tamaños de empresas, incrementando los beneficios.
2. ERP: permite a empresas incrementar el flujo de datos e información que mejoran la productividad y eficiencia. Este *software* empresarial es adaptable a todos los tamaños de empresas, desde pymes hasta multinacionales.

Sistemas operativos: Estos sistemas operativos llevan a cabo la gestión de los recursos de *hardware* y permiten que los programas se ejecuten en una computadora o dispositivo móvil. Algunos ejemplos entre otros incluyen Microsoft Windows, macOS, Linux, Android y iOS.

Aplicaciones de productividad: Variedad de aplicaciones que permiten a los usuarios crear y gestionar documentos, hojas de cálculo y presentaciones. Los más conocidos y utilizados son Microsoft Office (Word, Excel y PowerPoint) y Google Workspace (anteriormente G Suite).

Aplicaciones web: Estas aplicaciones se ejecutan en navegadores web, permiten y proporcionan servicios en línea como redes sociales, comercio y correo electrónicos. Los ejemplos más conocidos son Facebook, Twitter, Amazon y Gmail.

Aplicaciones móviles: Aplicaciones que se ejecutan directamente en los dispositivos móviles y ofrecen una variedad de funciones, desde mensajería hasta entretenimiento y transporte. Como WhatsApp, Instagram, Uber y Spotify.

Videojuegos: Programas de *software* diseñados para entretenimiento y ocio para ser utilizados en una diversidad de dispositivos. Los juegos de video más populares como Fortnite, Minecraft, Call of Duty, Candy Crush y muchos más.

Software científico: Programas utilizados en investigación y ciencia, como *software* de simulación, análisis de datos y modelado matemático.

El desarrollo de *software* de sistema es un campo amplio y diverso que abarca una gran variedad de aplicaciones y tecnologías. Los ejemplos anteriores ilustran algunos de los tipos de *software* más comunes.

Instrumentos

En el presente caso, se utilizará una mezcla entre entrevistas a los usuarios involucrados en los procesos y tomas de decisiones, así como observaciones y cuestionarios. Todo esto con el fin de obtener la información más detallada posible. Adicionalmente, se elaborará un documento para cada requerimiento identificado, con el fin de establecer información requerida, validaciones y diseño de pantallas.

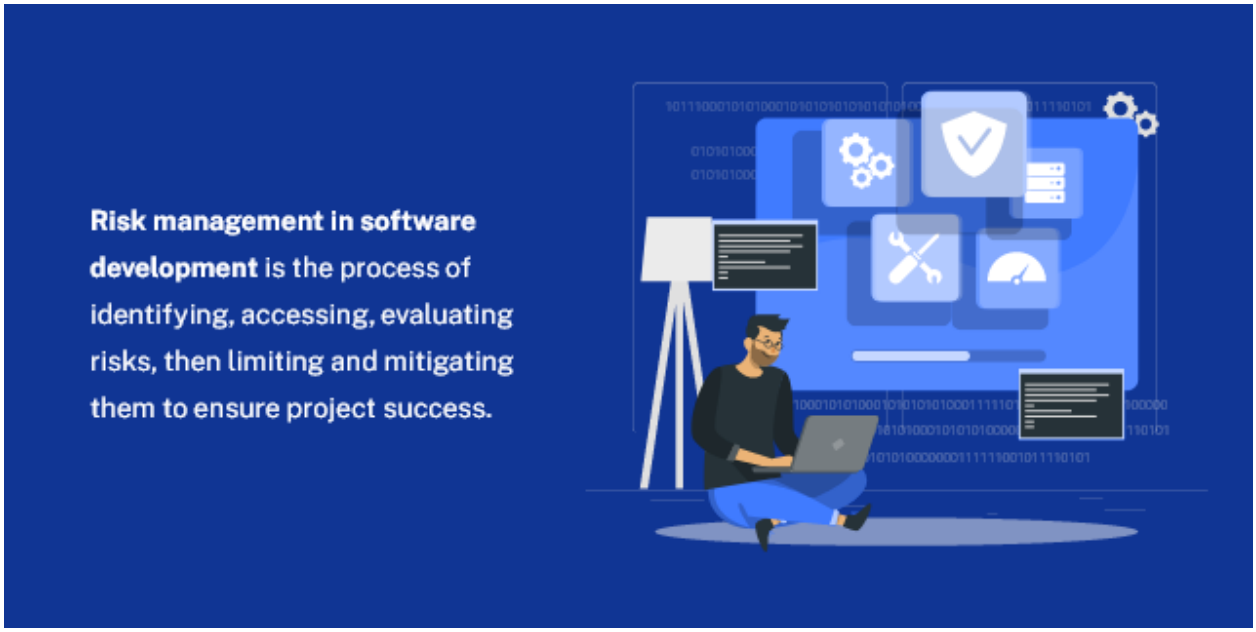
Capítulo IV. Análisis de resultados

Plan de gestión de riesgos

Un plan de gestión de riesgos es un documento que identifica, evalúa, monitorea y controla los riesgos potenciales que pueden afectar un proyecto, negocio u organización. El objetivo de un plan de gestión de riesgos es minimizar los impactos negativos de posibles riesgos y maximizar las oportunidades de éxito. Este plan incluye una serie de estrategias y acciones para manejar los riesgos, asignando responsabilidades específicas a los miembros del equipo y estableciendo un proceso para la comunicación y mitigación de riesgos.

Figura 37

Gestión de riesgos en el desarrollo de software



Fuente: (Gottdiener, 2022)

Tareas de la gestión de riesgos

1. Identificar

    1. Reconocer las amenazas potenciales, incluida la gestión avanzada del riesgo de los datos en la ingeniería de *software* y la gestión global del riesgo.
    2. Definir un riesgo en relación con la gestión de proyectos de *software* y evaluar su importancia.
2. Clasificar y priorizar
    1. Clasificar los riesgos, aplicando la gestión de riesgos en la gestión de proyectos de *software*.
    2. Priorizar los riesgos identificados del proyecto de *software* para centrarse en los de mayor impacto.
3. Desarrollar un plan de acción
    1. Cree una plantilla de plan de acción para la mitigación de riesgos.
    2. Abordar aplicaciones específicas de la gestión de riesgos en la gestión de proyectos de *software*, adaptando las estrategias en consecuencia.
4. Supervisar continuamente
    1. Implantar la supervisión continua en la gestión de riesgos del ciclo de vida del *software*
    2. Fomentar la comunicación constante dentro del equipo para identificar y abordar nuevas amenazas.
5. Aplicar planes de acción
    1. Actuar conforme al plan de acción si se materializa alguna amenaza identificada.
    2. Gestionar los riesgos de desarrollo de productos de forma proactiva mediante una evaluación eficaz de los riesgos del *software*.

Matriz de riesgos

Según (Team Asana, 2024):

> Una matriz de riesgos es una herramienta de análisis de riesgos que sirve para evaluar la probabilidad y la gravedad del riesgo durante el proceso de planificación del proyecto. Una vez que es evaluada la probabilidad y la gravedad de cada riesgo, puede ubicarse en la matriz para calcular la calificación del impacto de cada riesgo. Estas calificaciones ayudarán al equipo a determinar que prioridad asignar a los riesgos del proyecto y a gestionarlos de manera efectiva.

Tabla 4

Matriz de riesgos

Probabilidad ----------> \ Gravedad ---------->	**1 Insignificante**	**2 Menor**	**3 Moderada**	**4 Importante**	**5 Catastrófica**
5 Muy probable	5 Ausencia por enfermedad de un miembro del equipo	10	15	20	25
4 Probable	4 Corte de luz temporario	8	12	16	20
3 Posible	3	6 Extensión del tiempo programado para entregar el proyecto	9 Falta de presupuesto para desarrollar el proyecto	12	15
2 No es probable	2	4	6	8 Perdida importante de personal	10 Inundacion o incendio graves
1 Muy improbable	1	2	3	4	5 Falta prolongada de internet

Fuente:

Estudio de factibilidad

Quirós (2024) indica que “es un estudio de factibilidad es el que hace una empresa para determinar la posibilidad de poder desarrollar un negocio o un proyecto que espera implementar”. Algunos puntos clave se mencionan a continuación:

1. Ayuda a las empresas a tomar mejores decisiones, prediciendo el potencial éxito o fracaso de un proyecto.
2. Revela posibles desafíos que el proyecto podría enfrentar, permitiendo a la empresa prepararse adecuadamente.
3. Proporciona una base para desarrollar estrategias efectivas que aumenten la probabilidad de éxito del proyecto.

Análisis de factibilidad

Por medio de este proceso, se analizarán los objetivos del proyecto actual, de manera que su implementación sea viable para la empresa, demostrándose que se tiene el recurso humano apropiado, así como la infraestructura necesaria para la implementación de este. Es importante destacar que la factibilidad se refiere al grado de viabilidad o posibilidad de llevar a cabo una acción, mediante la evaluación de todos los elementos que integran el proyecto. Como se ha señalado previamente en este trabajo, los beneficios que se obtendrían al llevar a cabo el proyecto incluirían:

1. Optimización de los recursos.
2. Seguridad en el manejo de la información.
3. Automatización en la generación de reportes del inventario.
4. Control más eficiente de existencias para la toma de decisiones.

Factibilidad técnica

La factibilidad técnica es un proceso mediante el cual una empresa lleva a cabo estudios y análisis para determinar si es posible implementar un proyecto o negocio relacionado con un determinado producto o servicio. Durante este proceso, se analizan las estrategias y métodos necesarios para alcanzar resultados positivos.

El estudio de factibilidad resulta fundamental para comprender tanto las ventajas como las desventajas de establecer un negocio, comercio o empresa. Además, permite identificar posibles problemáticas y proporciona soluciones para superarlas. En lo referente al *hardware* que se necesita para el presente proyecto, el equipo que se utilice como servidor o terminal debe tener los siguientes requerimientos mínimos:

Tabla 5

Recursos del equipo para desarrollo y mantenimiento del sistema

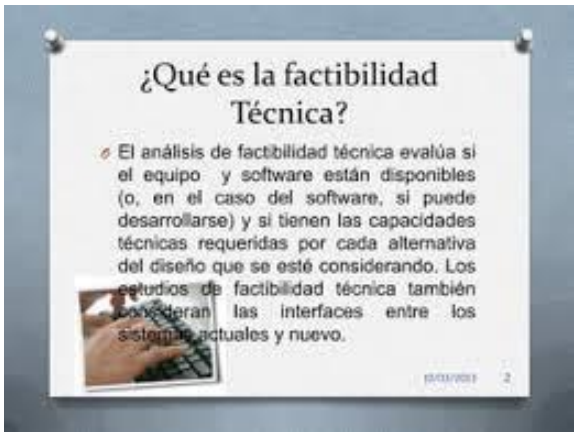
Caracteristicas	Especificaciones
Sistema Operativo	Windows 10 / 11 Windows Server 2012R2 Windows Server 2016 Windows Server 2019 Windows Server 2022
Procesador	Intel Core i3 en Adelante AMD Athlon II en adelante
Almacenamiento	512 Gb en adelante
Memoria RAM	16 Gb
Marca	Cualquiera

Fuente:

Al evaluar el *hardware* actual y considerar la configuración mínima necesaria, la empresa no necesita realizar ninguna inversión, ya que dispone de los equipos que cumplen con las características mencionadas para la correcta ejecución de la aplicación que se desarrollara.

En cuanto al *software*, contamos con todas las aplicaciones utilizadas para el desarrollo y funcionamiento del proyecto, como SQL Server Community y Visual Studio 2022. El sistema operativo de los equipos es Windows 11, lo que significa que no es necesario realizar ninguna inversión adicional para adquirir otros programas. Se les solicitara a los clientes que adquieran el *software*, como parte de los requisitos para usarlo, que las estaciones utilicen Windows, con las características antes mencionadas.

Figura 38
¿Qué es factibilidad técnica?



Fuente: (Lacayo Saballos, 2013)

Factibilidad económica

Tesis y Masters (2024) definen que:

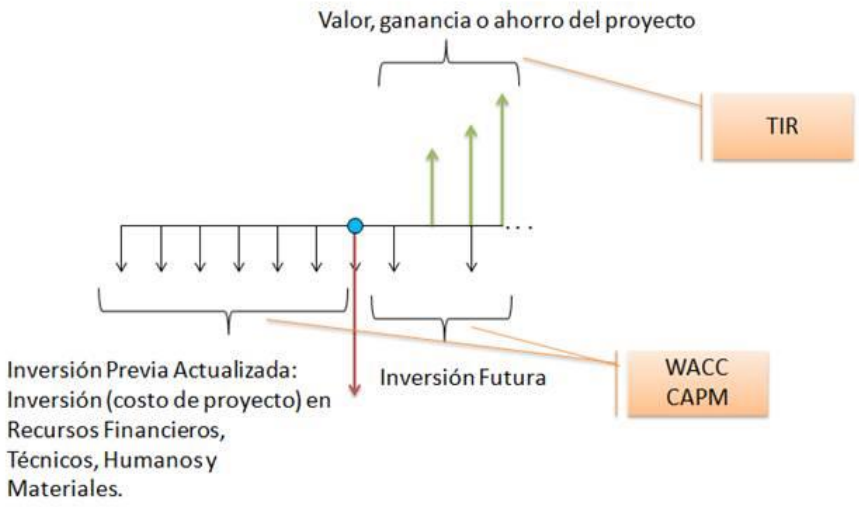
La factibilidad económica es el estudio que analiza los posibles costos e ingresos que una empresa podría obtener al lanzar un nuevo producto o servicio al mercado, determinando

si es rentable hacerlo. Este análisis se basa en el principio de costo-beneficio de un proyecto en particular. Se evalúa el potencial de dicho proyecto para tomar decisiones fundamentadas sobre su viabilidad en el mercado, considerando sus fortalezas, debilidades, riesgos y oportunidades.

Debido a que el sistema está siendo desarrollado por un empleado de la empresa, no implicará un gasto adicional, ya que las horas dedicadas a este proyecto forman parte del costo anual. El salario del empleado ya está incluido en la nómina que la empresa calcula anualmente. Sin embargo, para estimar el costo del proyecto, se calculará teniendo en cuenta el valor de las horas de un profesional en informática. Según el análisis efectuado, se describen los costos económicos para el desarrollo del proyecto informático:

Profesión	Salario
Desarrollador	900,000.00
Analista	1,100,000.00
Total	**2,000,000.00**

Figura 39
Ejemplo de factibilidad económica



Fuente: (Valdés Souto, 2013)

Factibilidad operativa

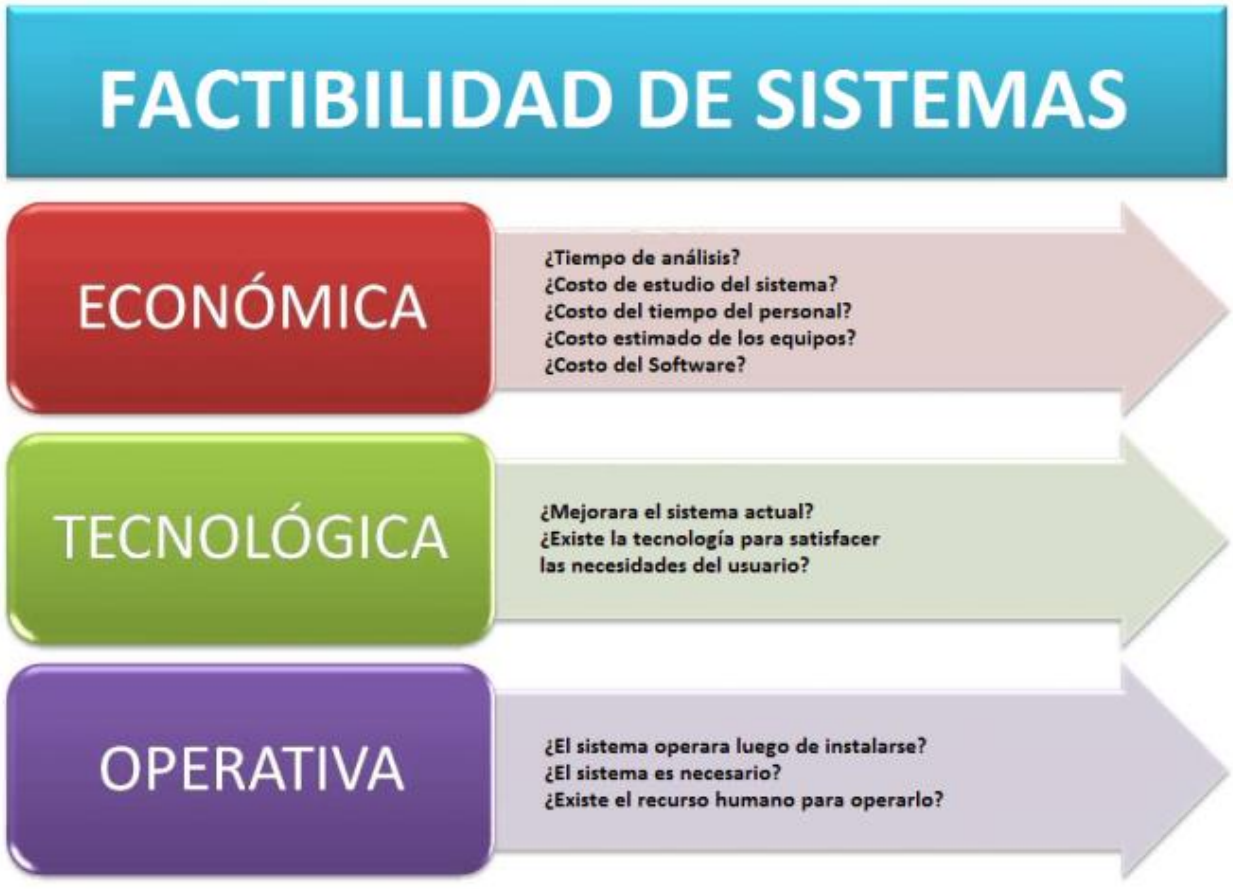
La factibilidad operativa en el desarrollo de *software* se refiere a la evaluación de si un proyecto de desarrollo de *software* puede ser implementado y operado de forma efectiva en el entorno tecnológico y organizacional existente. Esto implica analizar aspectos como la disponibilidad de recursos técnicos y humanos, la infraestructura tecnológica necesaria, la compatibilidad con sistemas existentes, la capacidad de mantenimiento y soporte, entre otros.

Este tipo de factibilidad se encarga de determinar si un proyecto de desarrollo de *software* es viable y sostenible en términos operativos, teniendo en cuenta todos los factores que puedan afectar su implementación y éxito a largo plazo.

Con el fin de garantizar el éxito del proyecto, es indispensable seguir una serie de pasos que nos permitan desarrollar de manera exitosa el sistema, entre ellos se pueden mencionar:

1. Desarrollar planes de acción para reducir los riesgos.
2. Diseñar un plan para la implementación de la producción.
3. Controlar las versiones y realizar pruebas.
4. Realizar el mantenimiento y seguimiento del proyecto.
5. Diseñar un plan de capacitación para los usuarios finales.
6. Crear un plan de contingencia para situaciones imprevistas.
7. Obtener retroalimentación y evaluar el rendimiento del proyecto.

Figura 40
Factibilidad de sistemas



Fuente: (Castellanos, 2009)

Capítulo V. Conclusiones y recomendaciones

Conclusiones

1. Identificación de la limitante principal: El análisis del departamento de TI de COFASA reveló que la principal limitante para el manejo correcto y seguro de la información referente a inventarios y facturación era la ausencia de una aplicación informática consolidada y que perteneciera enteramente a la empresa. Esta carencia impedía brindar a los clientes un *software* que les diera acceso a información en tiempo real sobre el estado de los artículos, así como sobre sus ventas y compras. La implementación del prototipo logró satisfacer las necesidades expuestas al inicio, representando una mejora sustancial respecto al proceso actual.
2. Objetivo del sistema: El desarrollo de este sistema tiene como objetivo proporcionar a los clientes un mejor control de los inventarios, así como brindar mayor seguridad y una mejor administración de la información, permitiendo la generación de reportes sobre las ventas y las existencias disponibles.
3. Recomendaciones futuras: Dado que se trata de un *software* en fase inicial, es necesario realizar una segunda etapa y darle seguimiento al desarrollo del sistema para mejorar los procesos actuales, así como crear nuevos, dependiendo de las necesidades de los clientes.
4. Implementación técnica: Se logró establecer una base de datos segura y confiable utilizando SQL Server, lo que garantiza un crecimiento normalizado y controlado de los datos. Esta implementación proporciona una base sólida para futuras expansiones y mejoras del sistema.
5. Cumplimiento de los objetivos: A nivel de planificación, el proyecto cumplió con los alcances y las necesidades expuestas por la Jefatura de TI de COFASA para el desarrollo de un *software* propio que controle los inventarios y que sirva como punto de venta que

se pueda ofrecer a los clientes actuales y futuros. La implementación del sistema propuesto ha mejorado la gestión de negociación con los clientes para su compra e instalación, alineándose con los objetivos y expectativas establecidos al inicio del proyecto.

Recomendaciones

1. Mejora continua del sistema: Se recomienda seguir implementando nuevas características en el sistema, basadas en el análisis realizado por la Jefatura del departamento de TI, para asegurar que el sistema sea completamente funcional y eficiente para la compañía.
2. Fomento del uso del aplicativo: Es fundamental fomentar el uso del aplicativo entre los usuarios y clientes que lo adquieran para que se familiaricen con su funcionamiento. Esto permitirá disponer de información clave, actualizada y accesible sobre los artículos y sus respectivas estadísticas.
3. Copias de seguridad y respaldo: Se sugiere programar copias de seguridad automáticas de la base de datos del sistema a intervalos regulares. Esto es importante para evitar posibles pérdidas de información y asegurar la integridad de los datos.
4. Expansión del uso del aplicativo: Finalmente, se recomienda seguir desarrollando y ofreciendo este tipo de aplicativos a los clientes de la empresa. Esto proporcionará acceso a información de manera más efectiva y eficiente, facilitando una toma de decisiones más adecuada a las necesidades y exigencias de cada farmacia. Además, ayudará a mantener el control y el orden sobre los inventarios y sus movimientos.
5. Migración a un sistema web: Se recomienda migrar a un sistema web que permita utilizarlo desde cualquier parte del mundo, ampliando así la cantidad de clientes potenciales a los cuales se tendría acceso.

Capítulo VI. Análisis de requerimientos

Análisis de requerimientos

A continuación, se observa los tres requerimientos que sirve para obtener un correcto funcionamiento en el prototipo que se desarrolla en COFASA:

1. Requerimientos del usuario.
2. Requerimientos funcionales.
3. Requerimientos no funcionales.

Requerimientos del usuario

Los requisitos de usuario (RU o URS en inglés) son las expectativas que una organización tiene con respecto al sistema para cumplir con sus necesidades ya sea de negocio o de cumplimiento regulatorio. Es el “qué y para qué”, la justificación del sistema (QbDGroup, 2022).

A continuación, se muestran las tablas con los requisitos del usuario de acuerdo con las solicitudes realizadas, los cuales son fundamentales para el correcto desarrollo del sistema.

Tabla 6

Requerimiento de usuario RU01

Responsable: Wilberth Chacón			
Id:	RU01	**Prioridad:**	1
Descripción:			
Diseñar un sistema de facturación y control de inventario para farmacias.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguna			

En la Tabla 6, se observa el requerimiento generado por la Gerencia del departamento de TI de COFASA.

Tabla 7

Requerimiento de usuario RU02

Responsable: Wilberth Chacón			
Id:	RU02	**Prioridad:**	1
Descripción:			
Desarrollar un módulo para el inicio de sesión e ingreso al sistema.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguna			

En la Tabla 7, se observa el requerimiento de la Gerencia de TI para el ingreso al sistema.

Tabla 8

Requerimiento de usuario RU03

Responsable: Wilberth Chacón			
Id:	RU03	**Prioridad:**	1
Descripción:			
Desarrollar un módulo para el registro de Usuarios.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguna			

En la Tabla 8, se observa el requerimiento de la Gerencia de TI para el registro de usuarios que puedan usar el sistema.

Tabla 9

Requerimiento de usuario RU04

Responsable: Wilberth Chacón			
Id:	RU04	**Prioridad:**	1
Descripción:			
Desarrollar un módulo para el registro de Perfiles para los Usuarios.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguna			

En la Tabla 9, se observa el requerimiento de la Gerencia de TI para el registro de perfiles que se usarán con los usuarios.

Tabla 10

Requerimiento de usuario RU05

Responsable: Wilberth Chacón			
Id:	RU05	**Prioridad:**	1
Descripción:			
Desarrollar un módulo para el registro de articulos			
Fuente:			
Gerente TI			
Dependencias:			
Ninguna			

En la Tabla 10, se observa el requerimiento de la Gerencia de TI para el registro de artículos.

Tabla 11

Requerimiento de usuario RU06

Responsable: Wilberth Chacón			
Id:	RU06	**Prioridad:**	1
Descripción:			
Desarrollar un módulo para el registro de proveedores			
Fuente:			
Gerente TI			
Dependencias:			
Ninguna			

En la Tabla 11, se observa el requerimiento de la Gerencia de TI para el registro de proveedores.

Tabla 12

Requerimiento de usuario RU07

Responsable: Wilberth Chacón			
Id:	RU07	**Prioridad:**	1
Descripción:			
Desarrollar un módulo para el registro de Clientes			
Fuente:			
Gerente TI			
Dependencias:			
Ninguna			

En la Tabla 12, se observa el requerimiento de la Gerencia de TI para el registro de clientes.

Tabla 13

Requerimiento de usuario RU08

Responsable: Wilberth Chacón			
Id:	RU08	**Prioridad:**	1
Descripción:			
Desarrollar un módulo para la generación de Reportes			
Fuente:			
Gerente TI			
Dependencias:			
Ninguna			

En la Tabla 13, se observa el requerimiento de la Gerencia de TI para la generación de reportes.

Tabla 14

Requerimiento de usuario RU09

Responsable: Wilberth Chacón			
Id:	RU09	**Prioridad:**	1
Descripción:			
Desarrollar un módulo que permita parametrizar el sistema.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguna			

En la Tabla 14, se observa el requerimiento de la Gerencia de TI para crear un módulo que permita parametrizar el sistema.

Tabla 15

Requerimiento de usuario RU10

Responsable: Wilberth Chacón			
Id:	RU10	**Prioridad:**	1
Descripción:			
Desarrollar un modulo de Dashboard para mostrar estadisticas de ventas, compras y otros datos .			
Fuente:			
Gerente TI			
Dependencias:			
Ninguna			

En la Tabla 15, se observa el requerimiento de la Gerencia de TI para desarrollar un módulo de *dahsboard* que permita mostrar estadísticas.

Tabla 16

Requerimiento de usuario RU11

Responsable: Wilberth Chacón			
Id:	RU11	**Prioridad:**	1
Descripción:			
Desarrollar un módulo para el registro de Compras.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguna			

En la Tabla 16, se observa el requerimiento de la Gerencia de TI para implementar un módulo de registro de compras.

Tabla 17

Requerimiento de usuario RU12

Responsable: Wilberth Chacón			
Id:	RU12	**Prioridad:**	1
Descripción:			
Desarrollar un módulo para el registro de ventas.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguna			

En la Tabla 17, se observa el requerimiento de la Gerencia de TI para desarrollar un módulo de facturación o ventas.

Requerimientos funcionales

Los requerimientos funcionales son especificaciones detalladas de las funciones y capacidades que un sistema, *software* o producto debe cumplir para satisfacer las necesidades y expectativas de los usuarios. Estos requerimientos describen cómo debe comportarse el sistema en diferentes situaciones y qué funciones debe desempeñar (Sommerville, 2011). A continuación, se detalla la lista de requisitos funcionales que el sistema debe cumplir para asegurar su correcto funcionamiento.

Tabla 18

Requerimiento funcional RF01

Responsable: Wilberth Chacón			
Id:	RF01	**Prioridad:**	1
Descripción:			
El sistema permite crear, modificar, eliminar y actualizar registros de usuarios, siendo responsable el administrador del sistema.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguno			

En la Tabla 18, se observa el requerimiento funcional de la gestión de crear, modificar, eliminar y actualizar registros de usuarios, siendo responsable el administrador del sistema.

Tabla 19

Requerimiento funcional RF02

Responsable: Wilberth Chacón			
Id:	RF02	**Prioridad:**	1
Descripción:			
El sistema permite crear, modificar, eliminar y actualizar registros de perfiles de los usuarios, siendo responsable el administrador del sistema.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguno			

En la Tabla 19, se observa el requerimiento funcional de la gestión de crear, modificar, eliminar y actualizar registros de perfiles de los usuarios, siendo responsable el administrador del sistema.

Tabla 20

Requerimiento funcional RF03

Responsable: Wilberth Chacón			
Id:	RF03	**Prioridad:**	1
Descripción:			
El sistema permite crear, modificar, eliminar y actualizarlos permisos y roles de acceso al sistema, siendo responsable el administrador del sistema.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguno			

En la Tabla 20, se observa el requerimiento funcional de la gestión de crear, modificar, eliminar y actualizarlos permisos y roles de acceso al sistema, siendo responsable el administrador del sistema.

Tabla 21

Requerimiento funcional RF04

Responsable: Wilberth Chacón			
Id:	RF04	**Prioridad:**	1
Descripción:			
El sistema permite crear, modificar, eliminar y actualizar registros de articulos, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguno			

En la Tabla 21, se observa el requerimiento funcional de la gestión de crear, modificar, eliminar y actualizar registros de artículos, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.

Tabla 22

Requerimiento funcional RF05

Responsable: Wilberth Chacón			
Id:	RF05	**Prioridad:**	1
Descripción:			
El sistema permite crear, modificar, eliminar y actualizar registros de clientes, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguno			

En la Tabla 22, se observa el requerimiento funcional de la gestión de crear, modificar, eliminar y actualizar registros de clientes, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.

Tabla 23

Requerimiento Funcional RF06

Responsable: Wilberth Chacón			
Id:	RF06	**Prioridad:**	1
Descripción:			
El sistema permite crear, modificar, eliminar y actualizar registros de proveedores, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguno			

En la Tabla 23, se observa el requerimiento funcional de la gestión de crear, modificar, eliminar y actualizar registros de proveedores, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.

Tabla 24

Requerimiento Funcional RF07

Responsable: Wilberth Chacón			
Id:	RF07	**Prioridad:**	1
Descripción:			
El sistema permite crear, modificar y /o eliminar registros de compras a proveedores, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguno			

En la Tabla 24, se observa el requerimiento funcional de la gestión de crear, modificar y /o eliminar registros de compras a proveedores, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.

Tabla 25

Requerimiento funcional RF08

Responsable: Wilberth Chacón			
Id:	RF08	**Prioridad:**	1
Descripción:			
El sistema permite generar reportes en pantalla y/o impresora, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguno			

En la Tabla 25, se observa el requerimiento funcional de la gestión de generar reportes en pantalla y/o impresora, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.

Tabla 26

Requerimiento funcional RF09

Responsable: Wilberth Chacón			
Id:	RF09	**Prioridad:**	1
Descripción:			
El sistema permite mostrar una pantalla de estadisticas, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguno			

En la Tabla 26, se observa el requerimiento funcional de la gestión de mostrar una pantalla de estadísticas, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.

Tabla 27

Requerimiento funcional RF10

Responsable: Wilberth Chacón			
Id:	RF10	**Prioridad:**	1
Descripción:			
El sistema permite crear facturas de ventas a clientes, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.			
Fuente:			
Gerente TI			
Dependencias:			
Ninguno			

En la Tabla 27, se observa el requerimiento funcional de la gestión de crear facturas de ventas a clientes, siendo responsables tanto el administrador del sistema como los usuarios habilitados para este fin.

Requerimientos no funcionales

Los requerimientos no funcionales son aquellos que no describen una funcionalidad específica del sistema, sino que se enfocan en atributos de calidad como la usabilidad, performance, seguridad, entre otros. Estos requerimientos suelen ser más difíciles de medir y verificar que los funcionales (Pressman, 2015). A continuación, se detalla la lista de requisitos no funcionales que el sistema debe cumplir para asegurar su correcto funcionamiento.

Tabla 28

Requerimiento no funcionales del sistema

ID	Descripción
RNF001	Seguridad
RNF002	Usabilidad
RNF003	Utilidad
RNF004	Accesibilidad

En la Tabla 28, se describen los requisitos no funcionales que deben ser implementados en el sistema para garantizar su correcto funcionamiento.

Tabla 29

Requerimiento no funcional RNF001

Responsable: Wilberth Chacón			
Id:	RNF001	**Prioridad:**	1
Descripción:			
El sistema validara el ingreso de los usuarios que cuenten con clave y un rol a registrado por el administrador del sistema			
Fuente:			
Gerente TI			

En la Tabla 29, se asignará un rol a cada usuario por parte del administrador del sistema para que puedan acceder de acorde al puesto y función que desempeñen. Esto permite el acceso al sistema de manera segura, ya que se requiere que el administrador haya brindado el acceso al usuario.

Tabla 30

Requerimiento no funcional RNF002

Responsable: Wilberth Chacón			
Id:	RNF002	**Prioridad:**	1
Descripción:			
El sistema ha sido diseñado para que los usuarios puedan acceder y utilizarlo sin dificultades.			
Fuente:			
Gerente TI			

En la Tabla 30, se observa el requerimiento no funcional del desarrollo de una interfaz de fácil acceso y comprensión para la utilización del sistema.

Tabla 31

Requerimiento no funcional RNF003

Responsable: Wilberth Chacón			
Id:	RNF003	**Prioridad:**	1
Descripción:			
No hay límite de registro de información en el Sistema, lo que permite acceder a la información necesaria y registros de manera indefinida.			
Fuente:			
Gerente TI			

En la Tabla 31, se observa el requerimiento no funcional que permite que el sistema no tenga límite de acceso a la información registrada.

Modelos UML

Casos de uso

A continuación, se presentan los actores que forman parte del sistema:

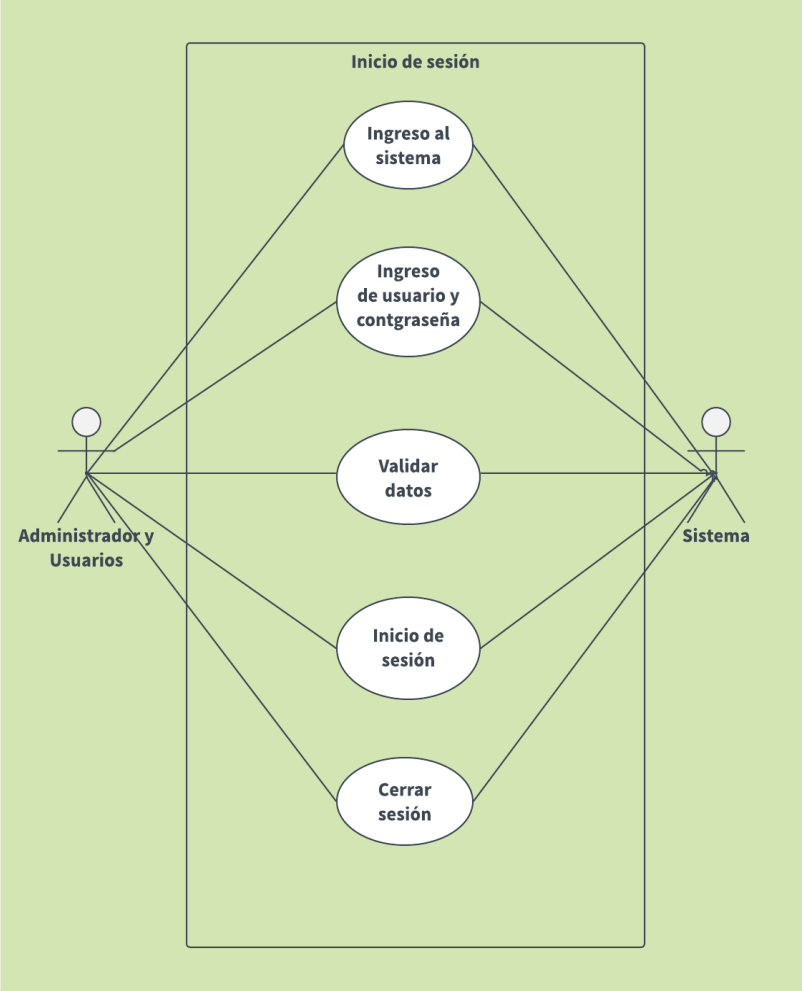
Administrador: Personal encargado de llevar un registro preciso de los usuarios que van a poder hacer uso del sistema.

Usuario: Es la persona que puede agregar, modificar o eliminar información, pero que no tiene privilegios para eliminar usuarios o asignar roles a otros usuarios.

Caso de uso 1. Inicio de sesión

Figura 41

Diagrama de caso de uso - Inicio de sesión



En la Figura 41 se observa el diagrama de caso de uso correspondiente al inicio de sesión del sistema de facturación y control de inventario para COFASA

Tabla 32

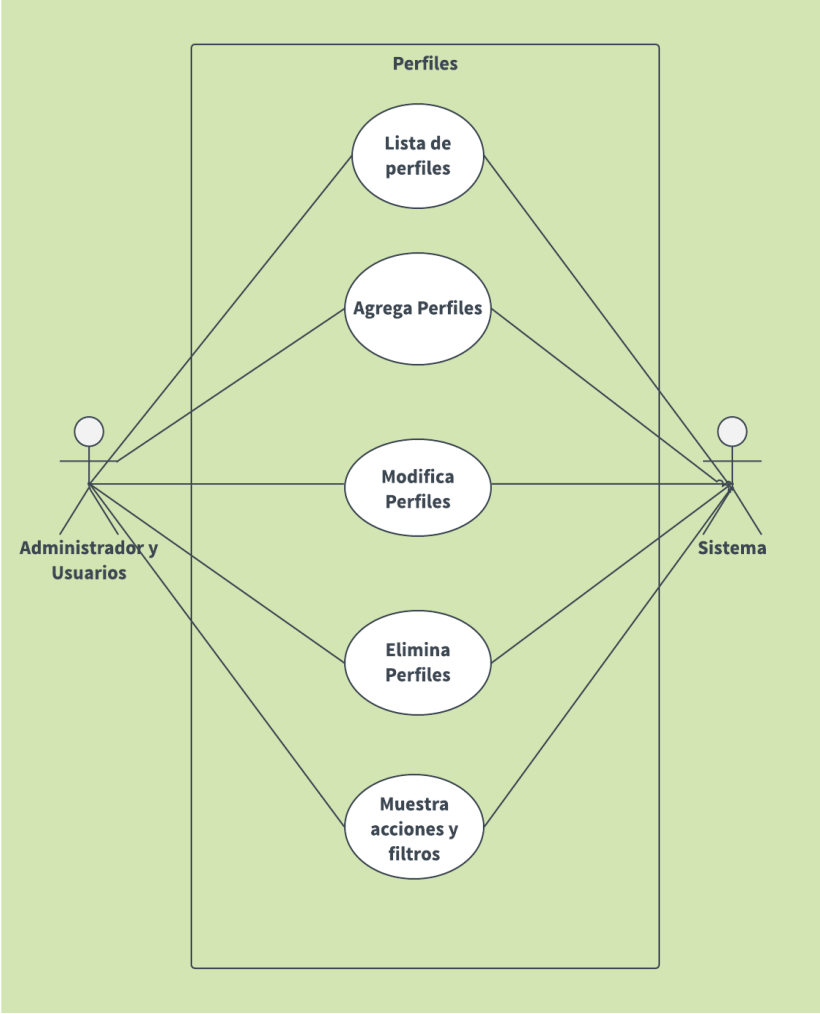
Inicio de sesión

Casos de uso: inicio de sesión
Autor/a Cesar Ruiz
Actores:
1. Usuario: (administradores) 2. Sistema
Objetivos:
1. Ingresar datos para el inicio de sesión. 2. Cerrar sesión.
Descripción:
1. Acceder al sistema. 2. Ingresar usuario y clave en el módulo de inicio de sesión. 3. Se le da clic al botón de ingresar. 4. Al validar el ingreso a la aplicación, se visualizará el menú principal del sistema. 5. Al finalizar las tareas para salir de la aplicación deberá dar clic al botón de Salir y así se cerrará la sesión.
Observaciones:
Los usuarios no autorizados, no tendrán acceso al sistema. En caso de olvido de la contras se le deberá solicitar al administrador que la restablezca.
Pantalla de errores:(Sistema)
1. Se muestra error al momento de ingresar una contraseña invalida. 2. Se muestra error al momento de ingresar un usuario erróneo o una contraseña invalida. 3. Se muestra error al momento de ingresar un usuario no registrado.

Caso de uso 2. Registro de perfiles

Figura 42

Caso de uso - Perfiles



En la Figura 42 se observa el diagrama de caso de uso correspondiente al registro de perfiles.

Figura 43

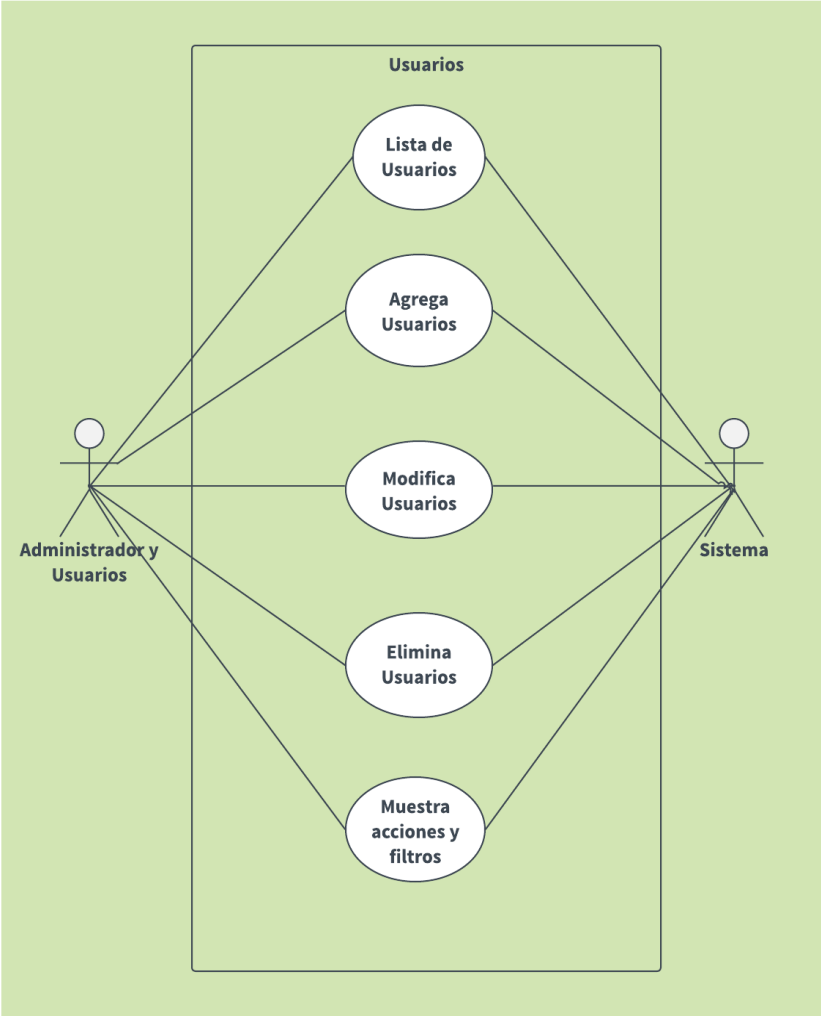
Registro de perfiles

Casos de uso: Registro de perfiles
Autor/a Cesar Ruiz
Actores:
1. Usuario: (administradores) 2. Sistema
Objetivos:
Registrar, actualizar y/o eliminar perfiles.
Descripción:
1. Acceder al sistema. 2. Seleccionar opción del menú correspondiente a perfiles 3. Si es un perfil nuevo dar clic al boton de la derecha de agregar, si ya existe solo darle doble clic para seleccionar 4. Ingresar datos del perfil 5. Guardar los datos
Observaciones:
1. Agregar, modificar y/o eliminar registros de perfiles.
Pantalla de errores:(Sistema)
1. Se muestra error si no se cargan los datos 2. Se muestra error si no guardan las modificaciones

Caso de uso 3. Registro de usuarios

Figura 44

Caso de uso - Usuarios



En la Figura 44 se observa el diagrama de caso de uso correspondiente al registro de usuarios.

Figura 45

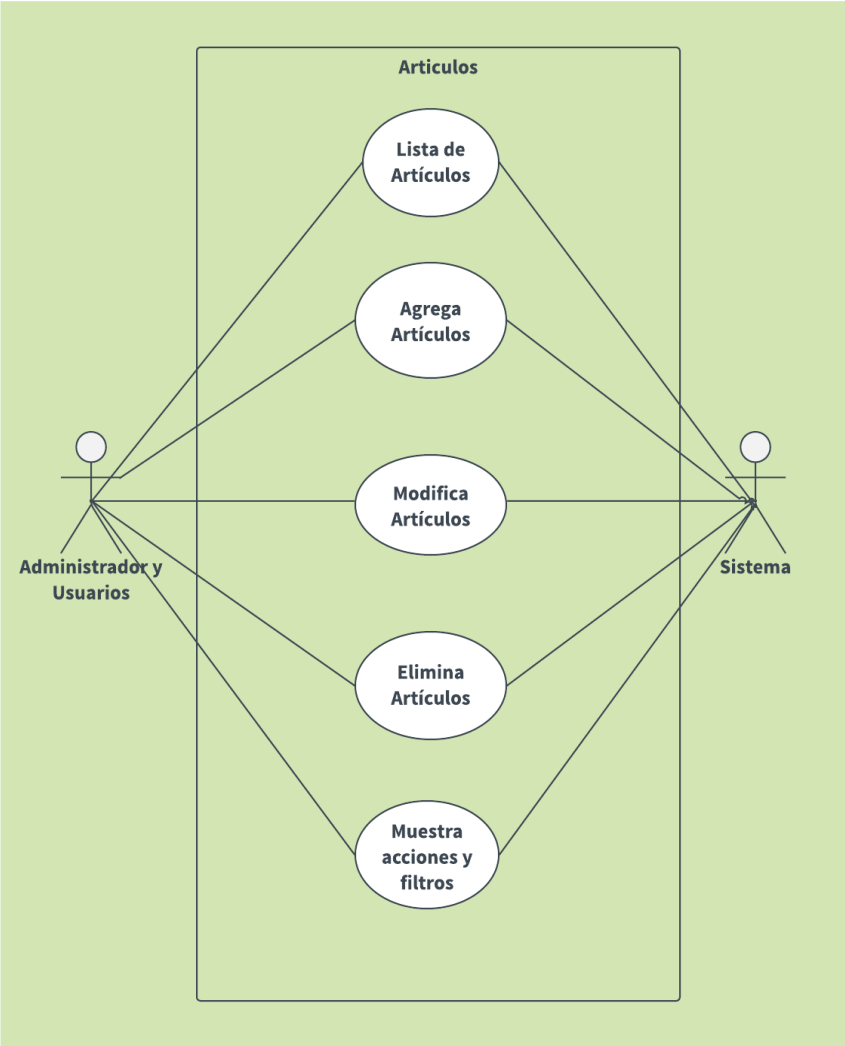
Registro de usuarios

Casos de uso: Registro de Usuarios
Autor/a Cesar Ruiz
Actores:
1. Usuario: (administradores) 2. Sistema
Objetivos:
Agregar, actualizar y/o eliminar registro(s).
Descripción:
1. Acceder al sistema. 2. Seleccionar opción del menú correspondiente 3. Si es un nuevo registro dar clic al boton de la derecha de agregar, si ya existe dar doble clic para seleccionar 4. Ingresar datos. 5. Guardar los datos
Observaciones:
1. Agregar, modificar y/o eliminar registros
Pantalla de errores:(Sistema)
1. Se muestra error si no se cargan los datos 2. Se muestra error si no guardan las modificaciones

Caso de uso 4. Registro de artículos

Figura 46

Caso de uso - Artículos



En la Figura 46 se observa el diagrama de caso de uso correspondiente al registro de artículos.

Figura 47

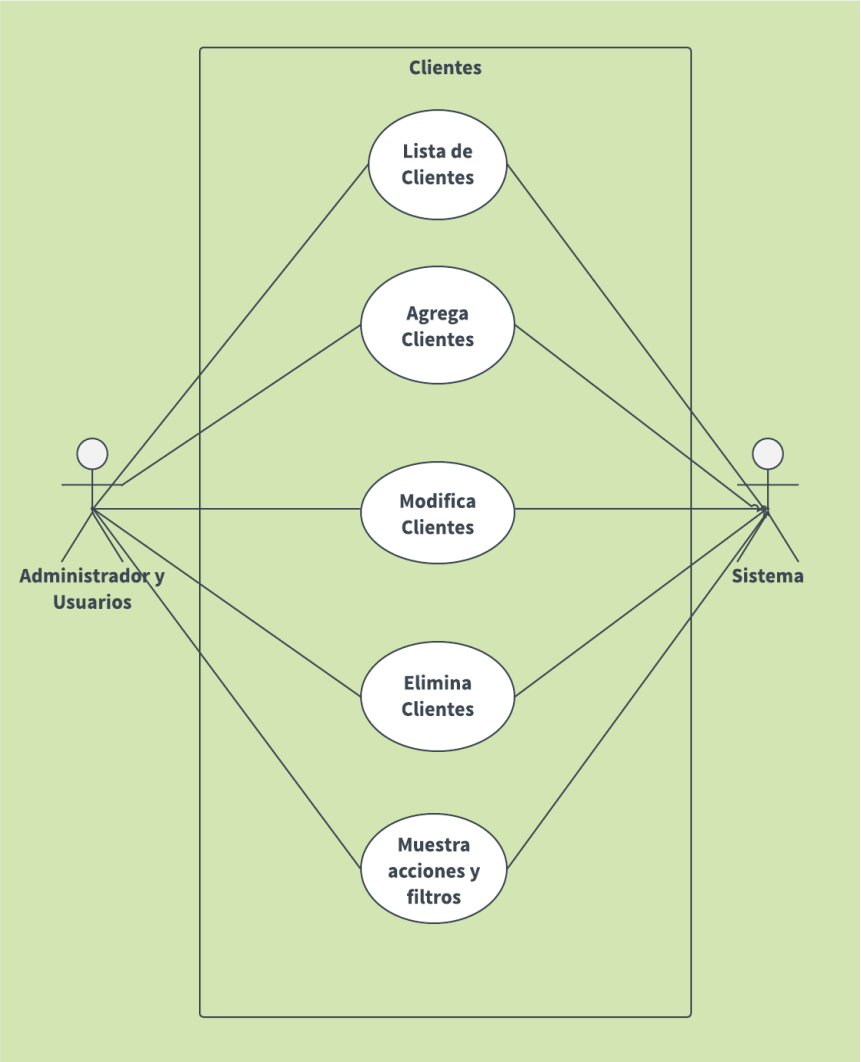
Registro de artículos

Casos de uso: Registro de articulos
Autor/a Cesar Ruiz
Actores:
1. Usuario: (administradores) 2. Sistema
Objetivos:
Agregar, actualizar y/o eliminar registro(s).
Descripción:
1. Acceder al sistema. 2. Seleccionar opción del menú correspondiente 3. Si es un nuevo registro dar clic al boton de la derecha de agregar, si ya existe dar doble clic para seleccionar 4. Ingresar datos. 5. Guardar los datos
Observaciones:
1. Agregar, modificar y/o eliminar registros
Pantalla de errores:(Sistema)
1. Se muestra error si no se cargan los datos 2. Se muestra error si no guardan las modificaciones

Caso de uso 5. Registro de clientes

Figura 48

Caso de uso - Clientes



En la Figura 48 se observa el diagrama de caso de uso correspondiente al registro de clientes

Figura 49

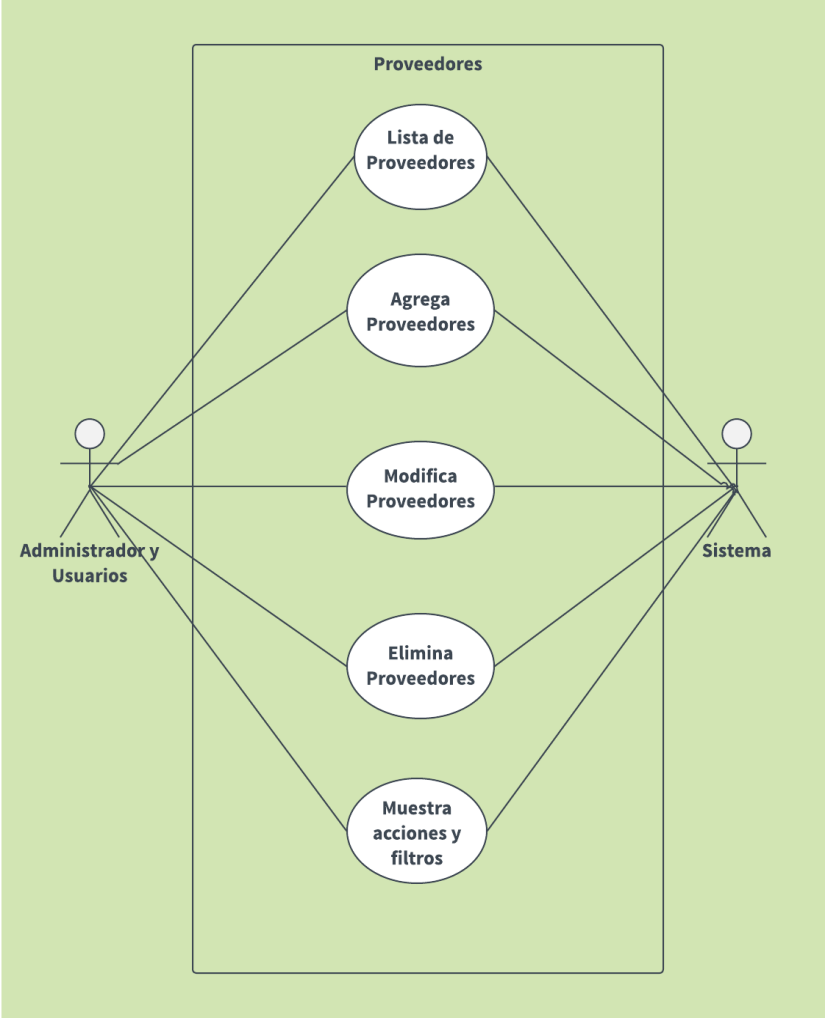
Registro de clientes

Casos de uso: Registro de Clientes
Autor/a Cesar Ruiz
Actores:
1. Usuario: (administradores) 2. Sistema
Objetivos:
Agregar, actualizar y/o eliminar registro(s).
Descripción:
1. Acceder al sistema. 2. Seleccionar opción del menú correspondiente 3. Si es un nuevo registro dar clic al boton de la derecha de agregar, si ya existe dar doble clic para seleccionar 4. Ingresar datos. 5. Guardar los datos
Observaciones:
1. Agregar, modificar y/o eliminar registros
Pantalla de errores:(Sistema)
1. Se muestra error si no se cargan los datos 2. Se muestra error si no guardan las modificaciones

Caso de uso 6. Registro de proveedores

Figura 50

Caso de uso - Proveedores



En la Figura 50 se observa el diagrama de caso de uso correspondiente al registro de proveedores

Figura 51

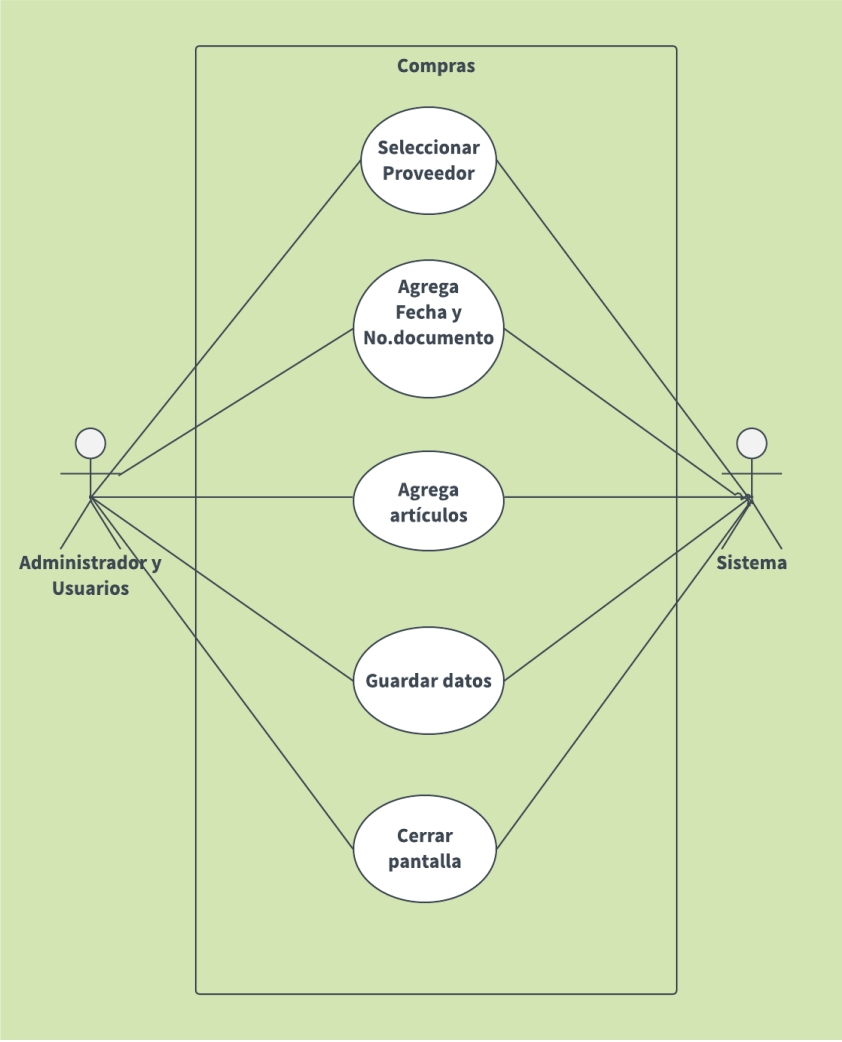
Registro de proveedores

Casos de uso: Registro de Proveedores
Autor/a Cesar Ruiz
Actores:
1. Usuario: (administradores) 2. Sistema
Objetivos:
Agregar, actualizar y/o eliminar registro(s).
Descripción:
1. Acceder al sistema. 2. Seleccionar opción del menú correspondiente 3. Si es un nuevo registro dar clic al boton de la derecha de agregar, si ya existe dar doble clic para seleccionar 4. Ingresar datos. 5. Guardar los datos
Observaciones:
1. Agregar, modificar y/o eliminar registros
Pantalla de errores:(Sistema)
1. Se muestra error si no se cargan los datos 2. Se muestra error si no guardan las modificaciones

Caso de uso 7. Registro de compras

Figura 52

Caso de uso - Compras



En la Figura 52 se observa el diagrama de caso de uso correspondiente al registro de compras.

Figura 53

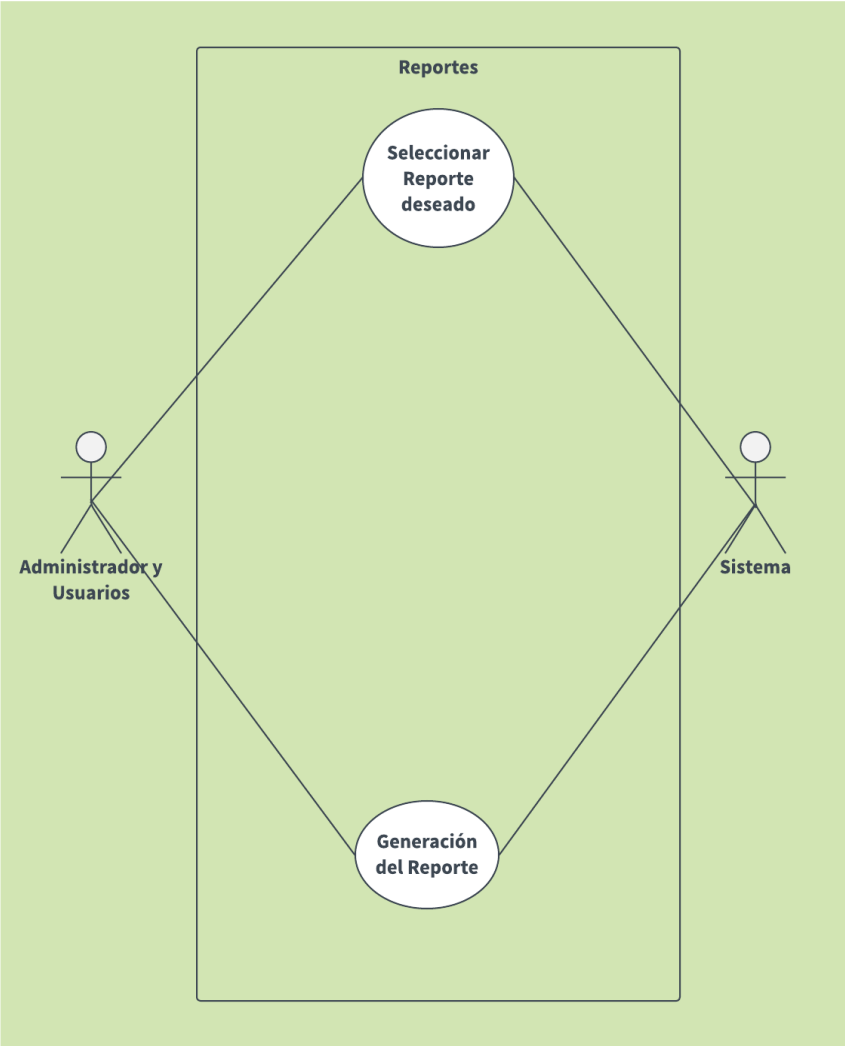
Registro de compras a proveedores

Casos de uso: Registro de Compras
Autor/a Cesar Ruiz
Actores:
1. Usuario: (administradores) 2. Sistema
Objetivos:
Agregar, actualizar y/o eliminar registro(s).
Descripción:
1. Acceder al sistema. 2. Seleccionar opción del menú correspondiente 3. Si es un nuevo registro dar clic al boton de la derecha de agregar, si ya existe dar doble clic para seleccionar 4. Ingresar datos. 5. Guardar los datos
Observaciones:
1. Agregar, modificar y/o eliminar registros
Pantalla de errores:(Sistema)
1. Se muestra error si no se cargan los datos 2. Se muestra error si no guardan las modificaciones

Caso de uso 8. Generación de reportes

Figura 54

Caso de uso - Generación de reportes



En la Figura 54 se observa el diagrama de caso de uso correspondiente a la generación de reporte de artículos.

Figura 55

Generación de reporte de artículos

Casos de uso: Generación de Reporte de Articulos
Autor/a Cesar Ruiz
Actores:
1. Usuario: (administradores) 2. Sistema
Objetivos:
1. Generar un reporte de articulos.
Descripción:
1. Visualizar los articulos que existen.
Observaciones:
Visaulizar Reportes de Articulos
Pantalla de errores:(Sistema)
1. Se muestra error si no se cargan los datos

Diagrama de secuencia

Diagrama de secuencia 1. Inicio de sesión

Figura 56

Diagrama de secuencia - Inicio de sesión

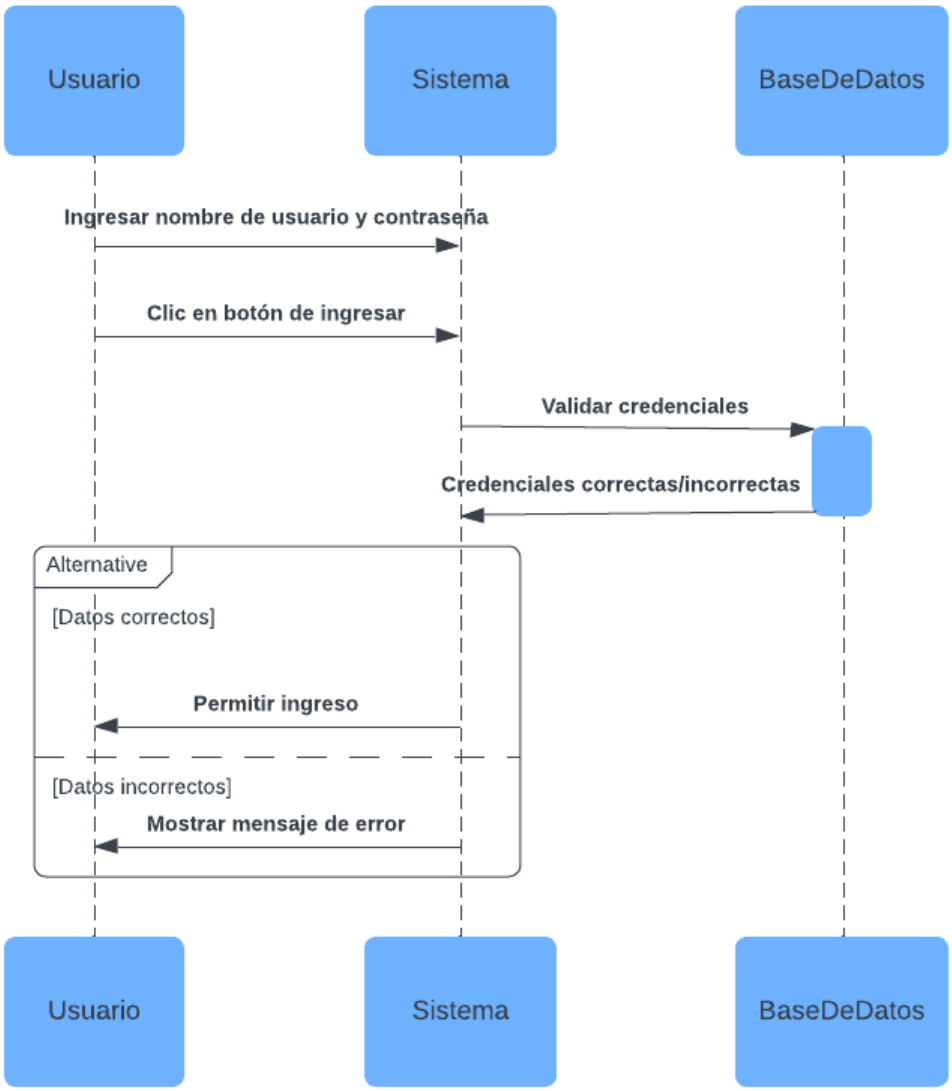


Diagrama de secuencia 2. Menú principal

Figura 57

Diagrama de secuencia - Menú principal

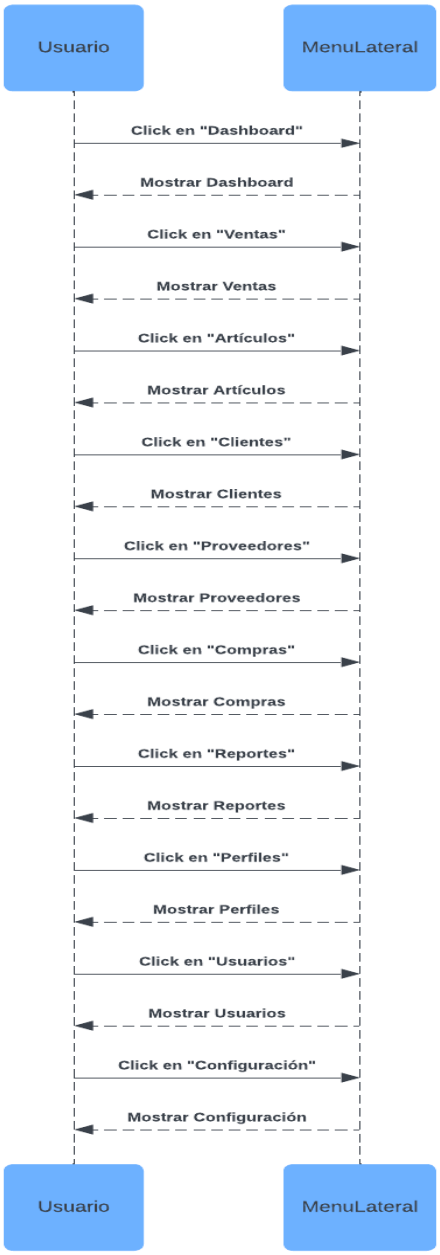


Diagrama de secuencia 3. Ventas

Figura 58

Diagrama de secuencia - Factura de venta

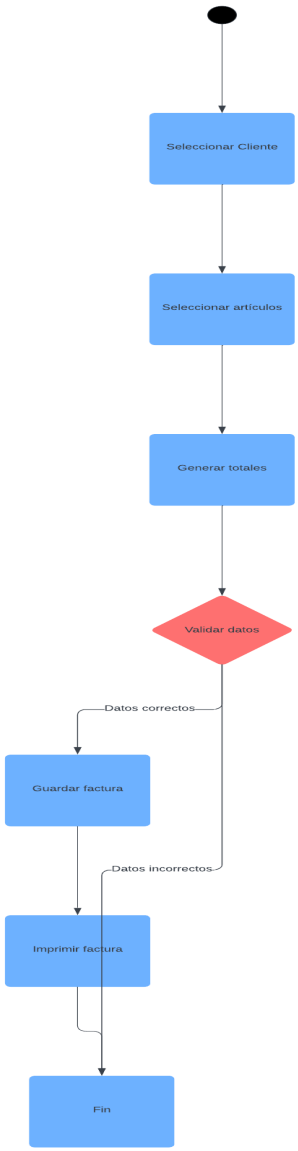


Diagrama de secuencia 4. Perfiles

Figura 59

Diagrama de secuencia - Perfiles

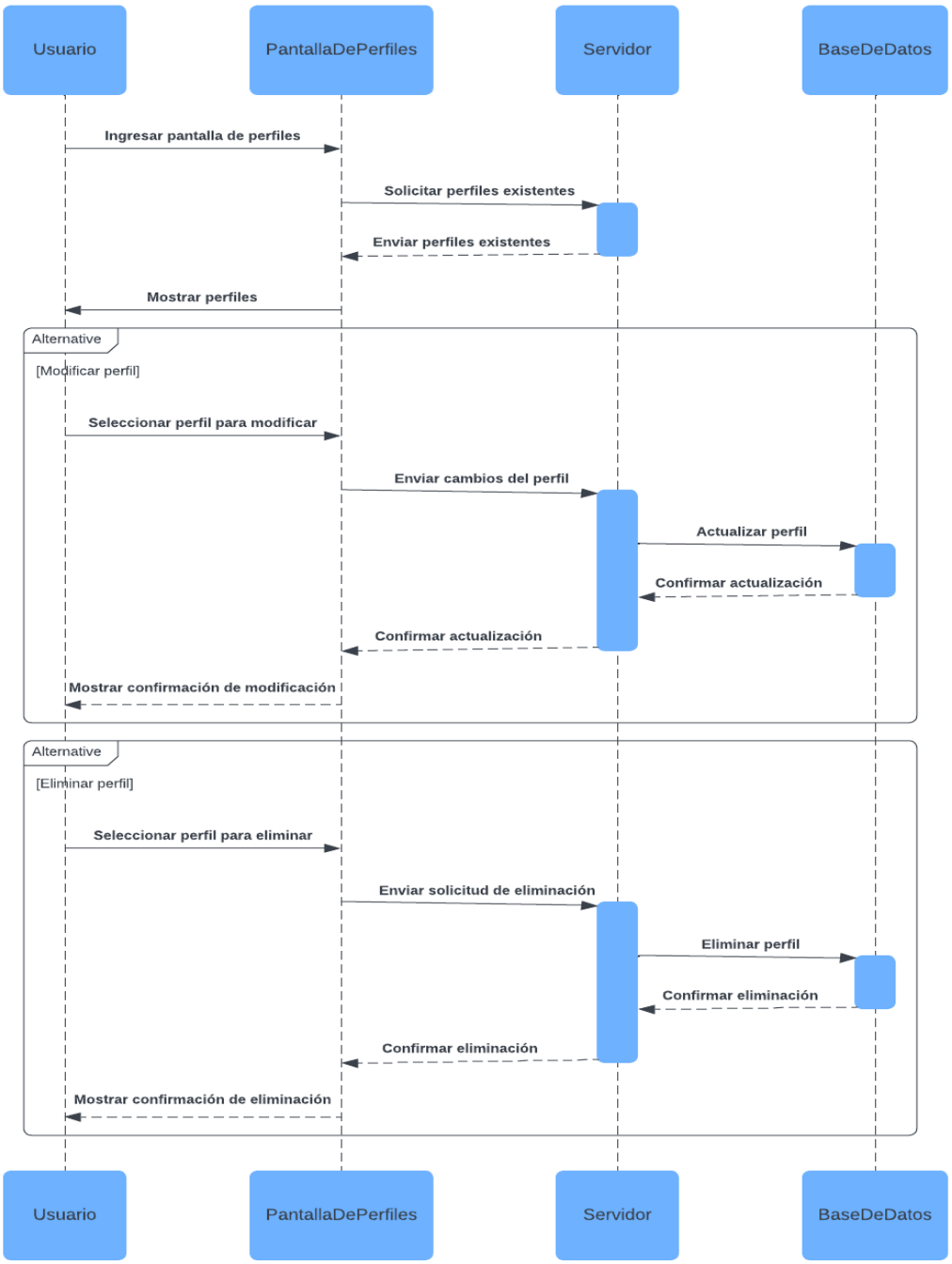


Diagrama de secuencia 5. Usuarios

Figura 60

Diagrama de secuencia - Usuario

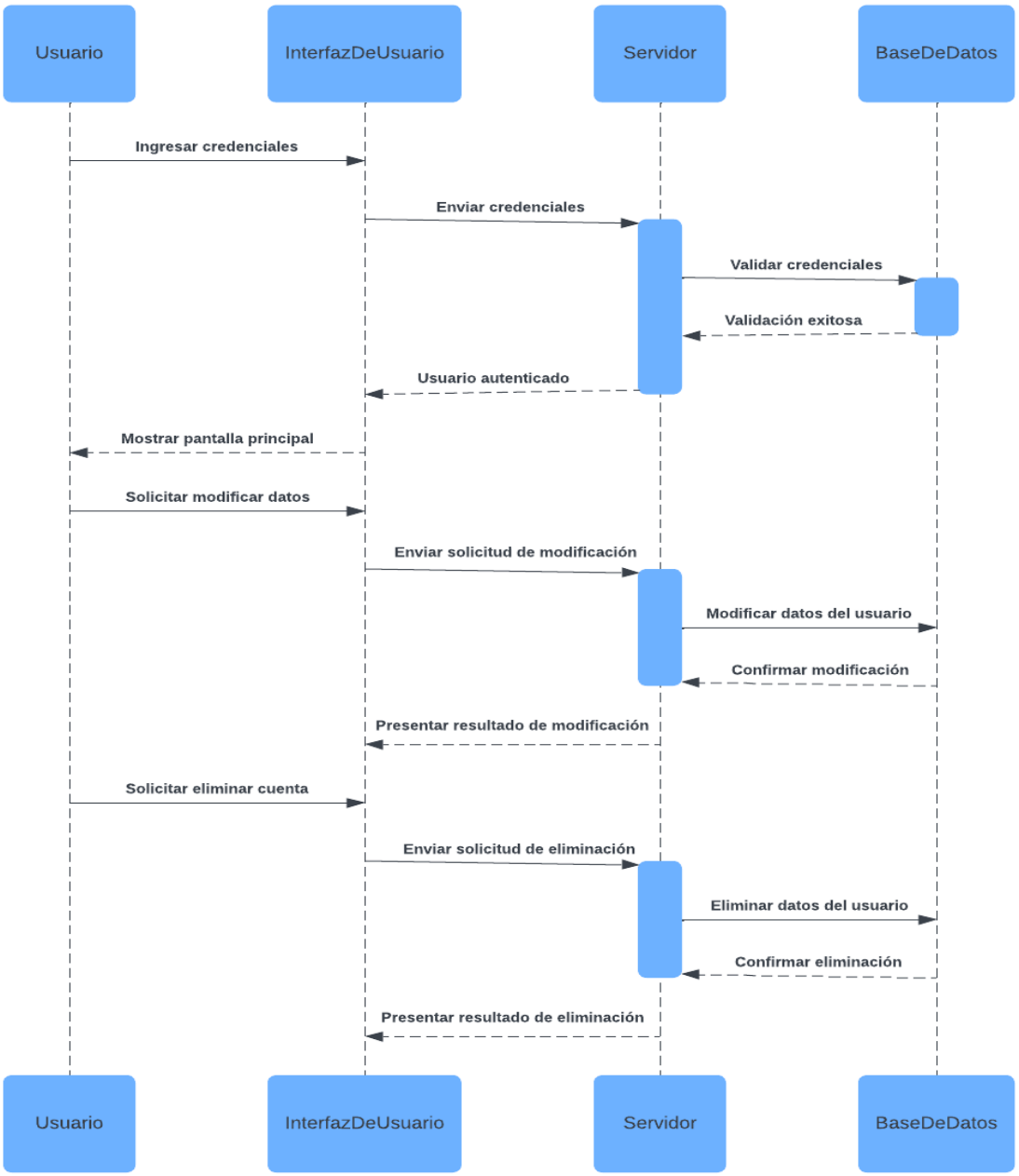


Diagrama de secuencia 6. Artículos

Figura 61

Diagrama de secuencia - Mantenimiento de artículos

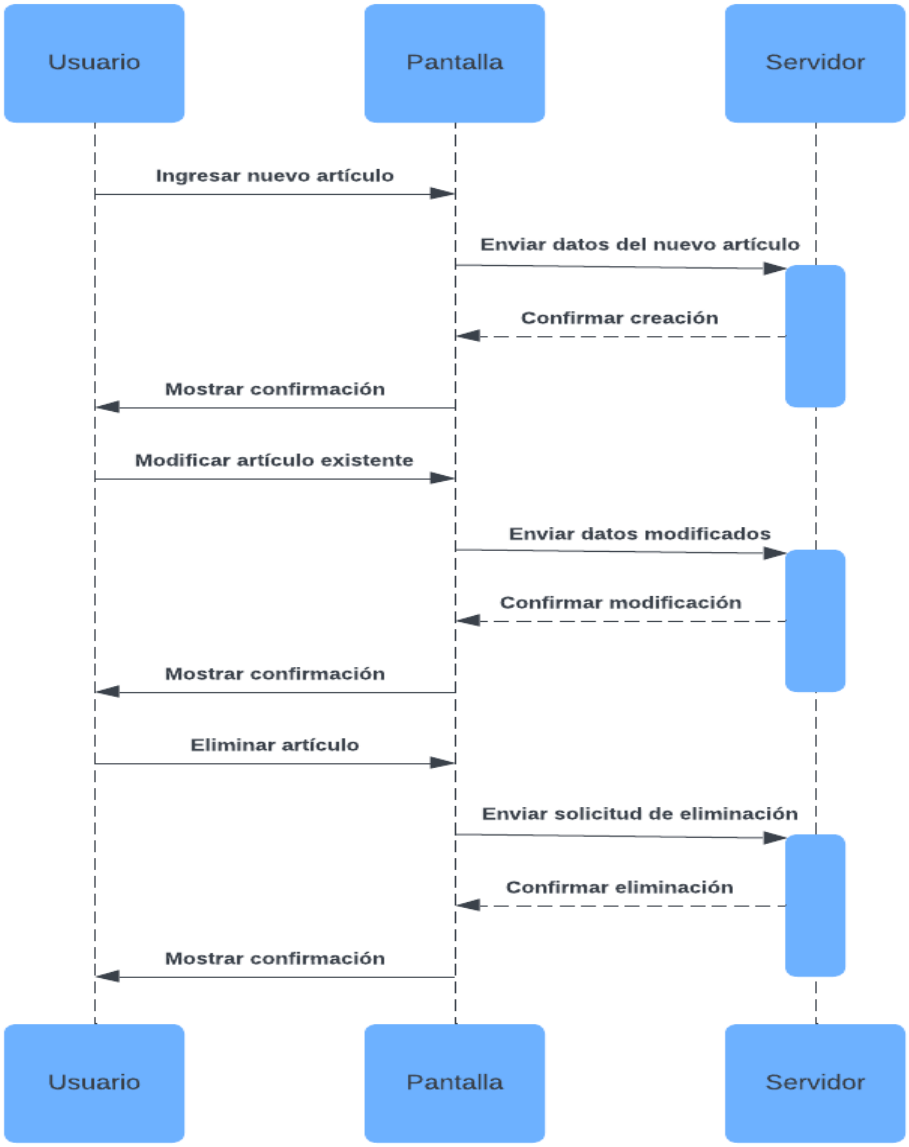


Diagrama de secuencia 7. Clientes

Figura 62

Diagrama de secuencia - Mantenimiento de clientes

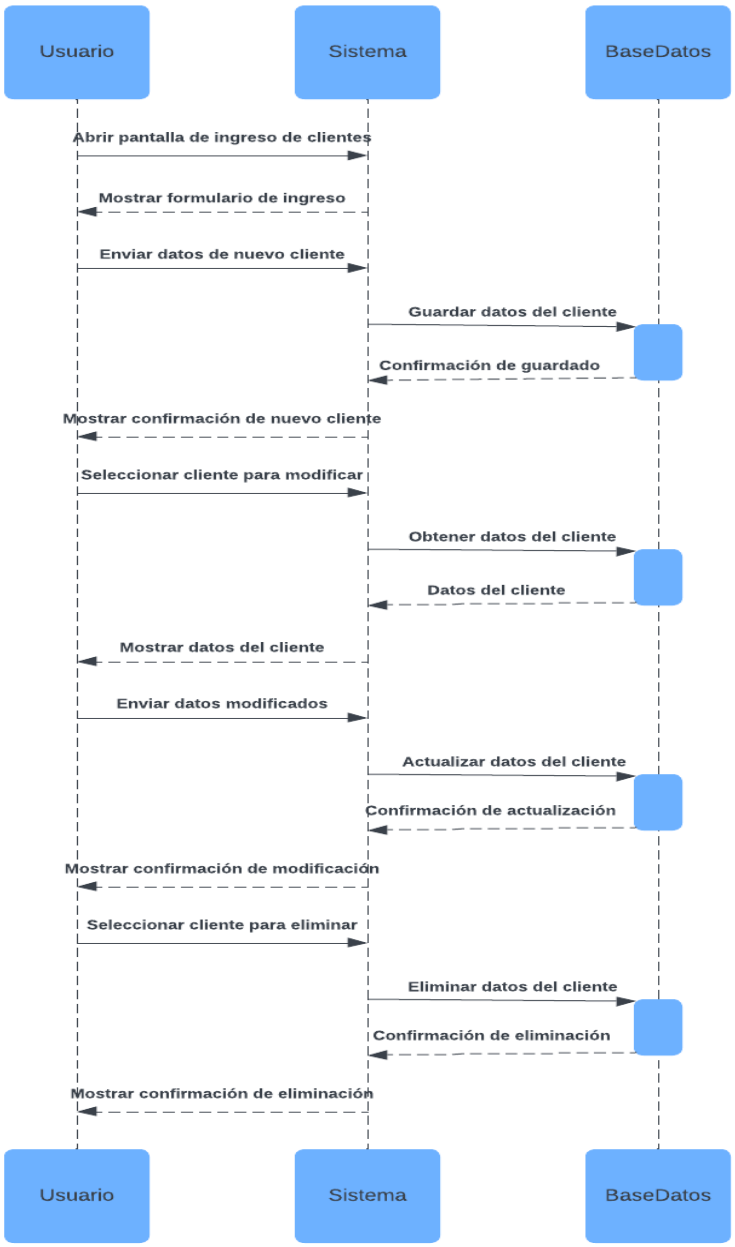


Diagrama de secuencia 8. Proveedores

Figura 63

Diagrama de secuencia – Mantenimiento de proveedores

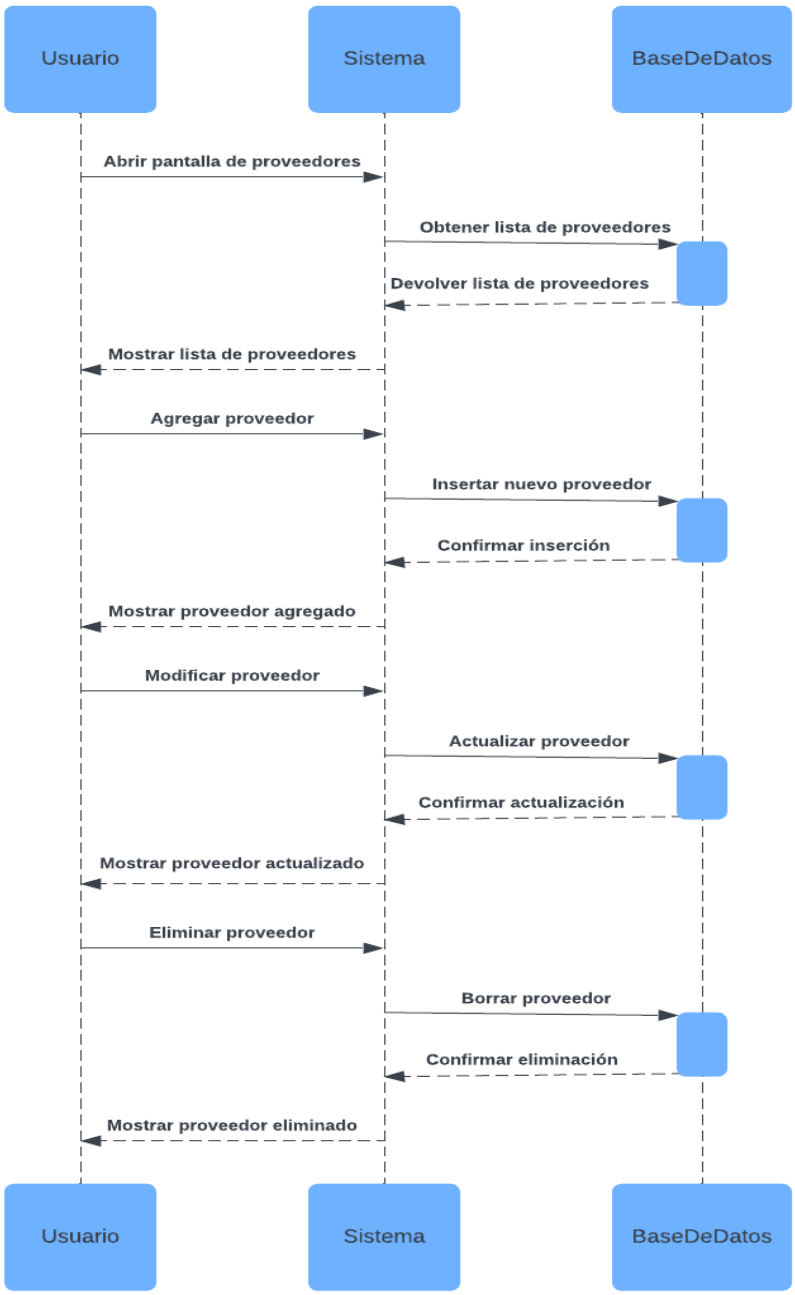


Diagrama de secuencia 9. Compras

Figura 64

Diagrama de secuencia - Compras

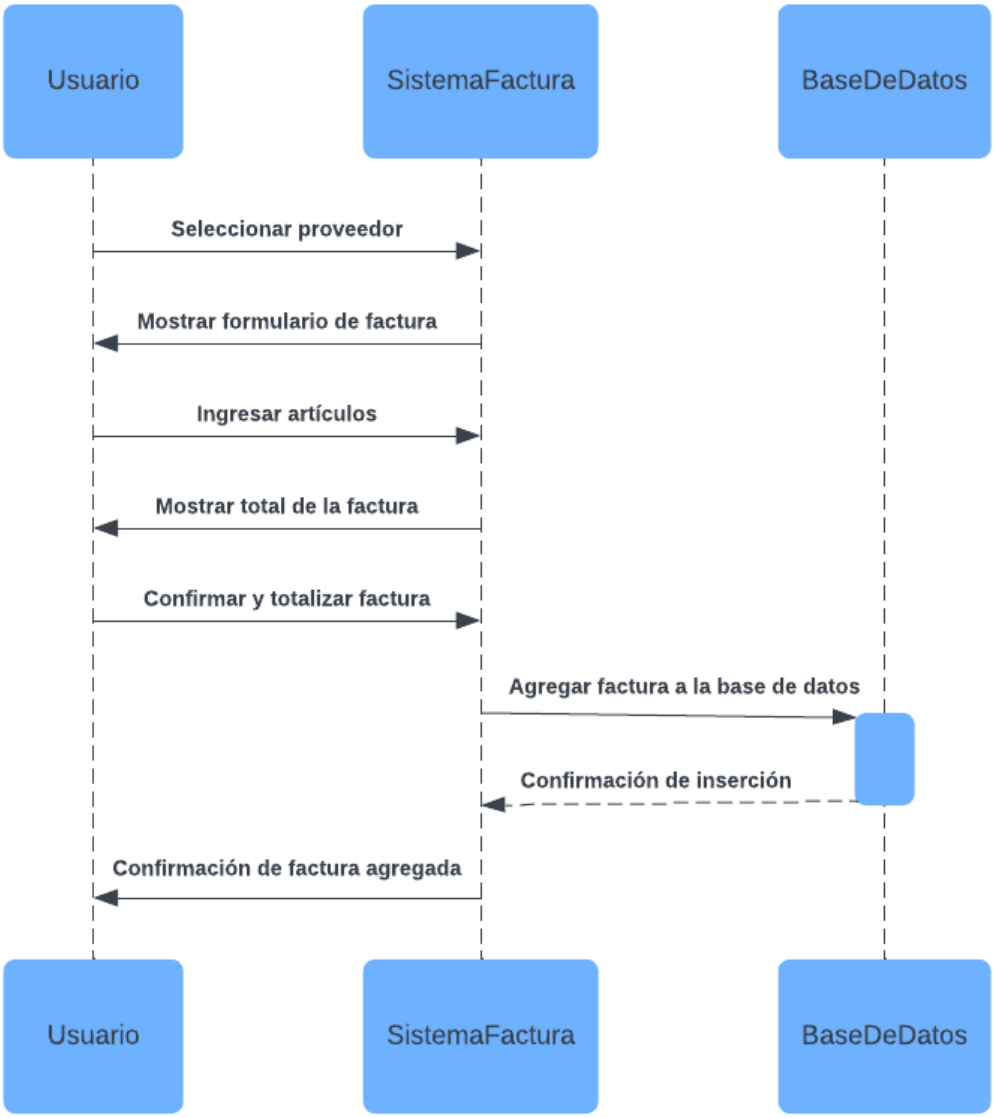


Diagrama de secuencia 10. Reportes

Figura 65

Diagrama de secuencia - Generación de reportes

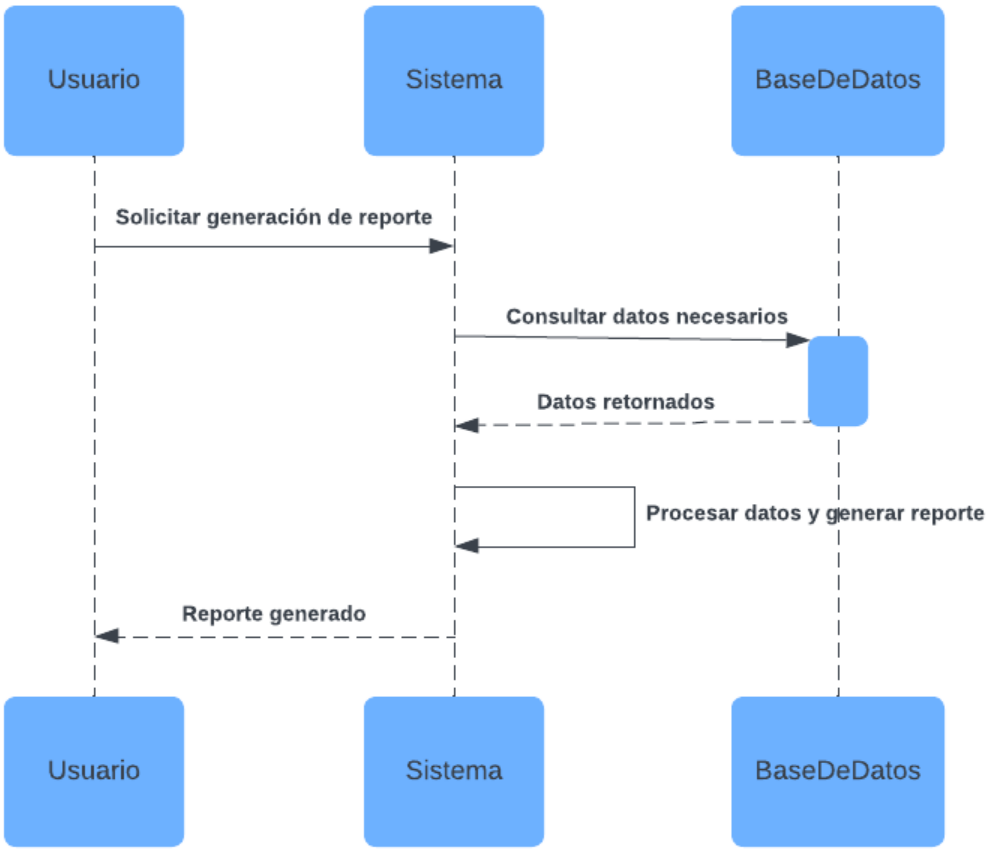


Diagrama de actividades

Diagrama de actividad 1. Inicio de sesión

Figura 66

Diagrama de actividad - Inicio de sesión

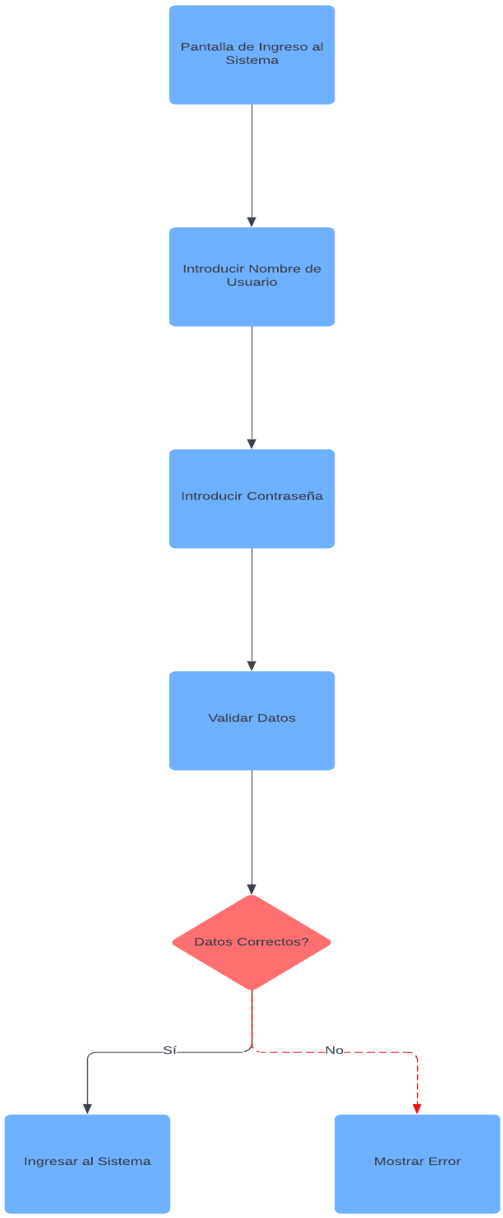


Diagrama de actividad 2. Ventas

Figura 67

Diagrama de actividad - Ventas

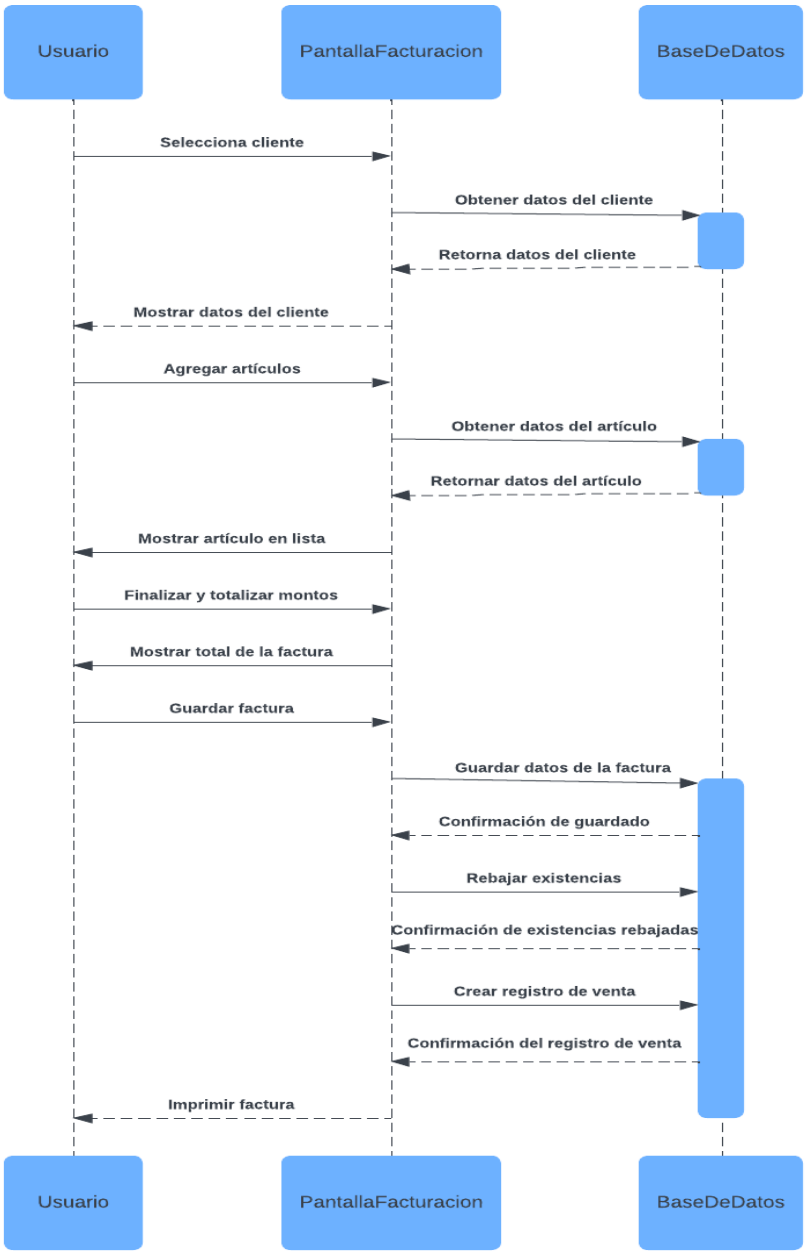


Diagrama de actividad 3. Perfiles

Figura 68

Diagrama de actividad - Perfiles

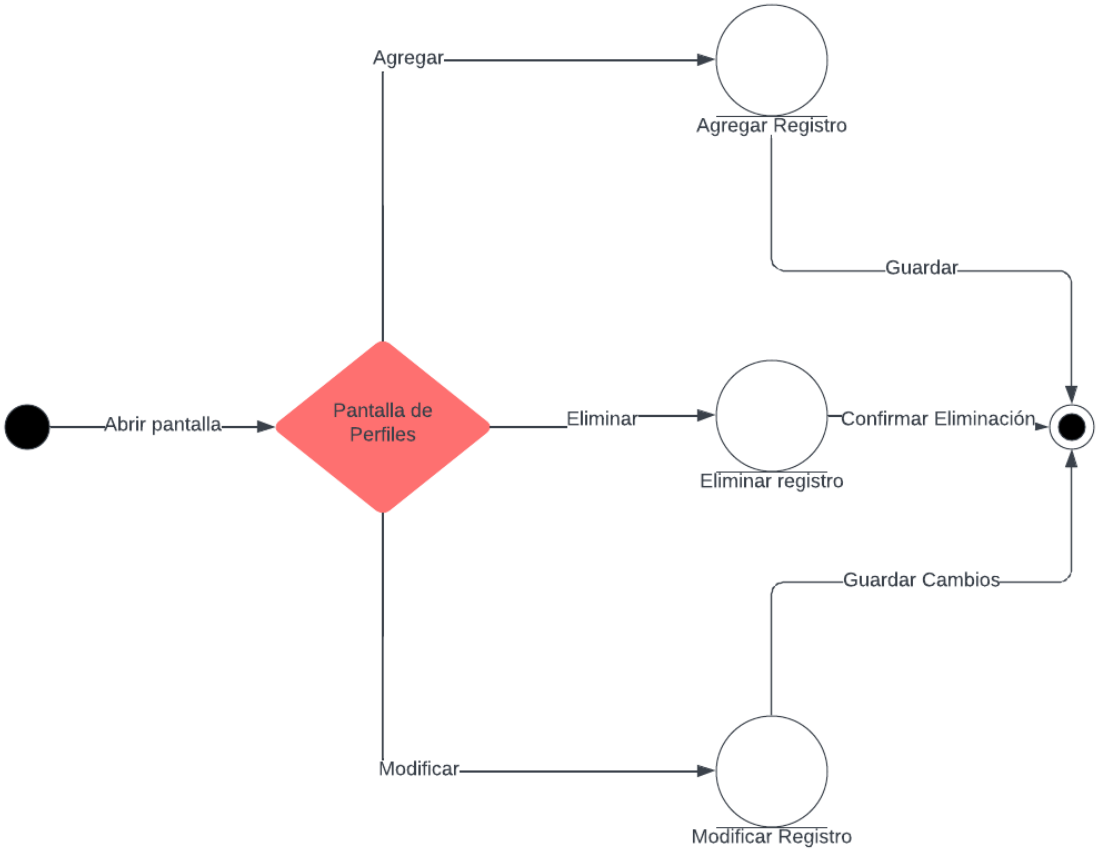


Diagrama de actividad 4. Usuarios

Figura 69

Diagrama de actividad - Usuarios

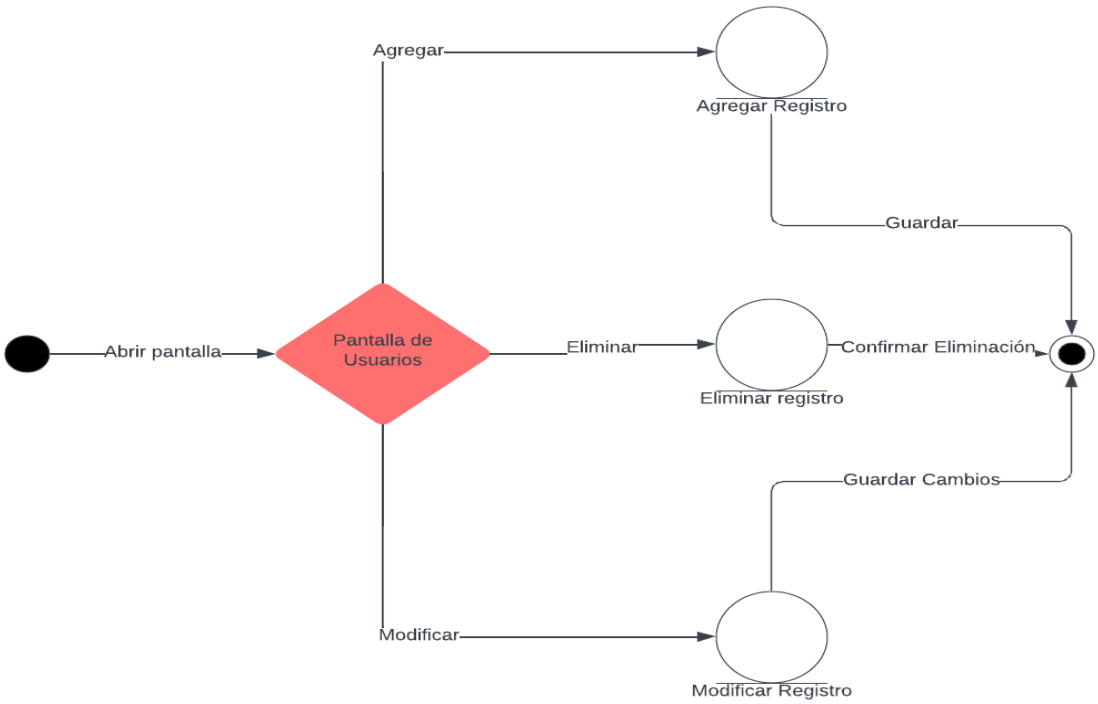


Diagrama de actividad 5. Artículos

Figura 70

Diagrama de actividad - Mantenimiento de Artículos

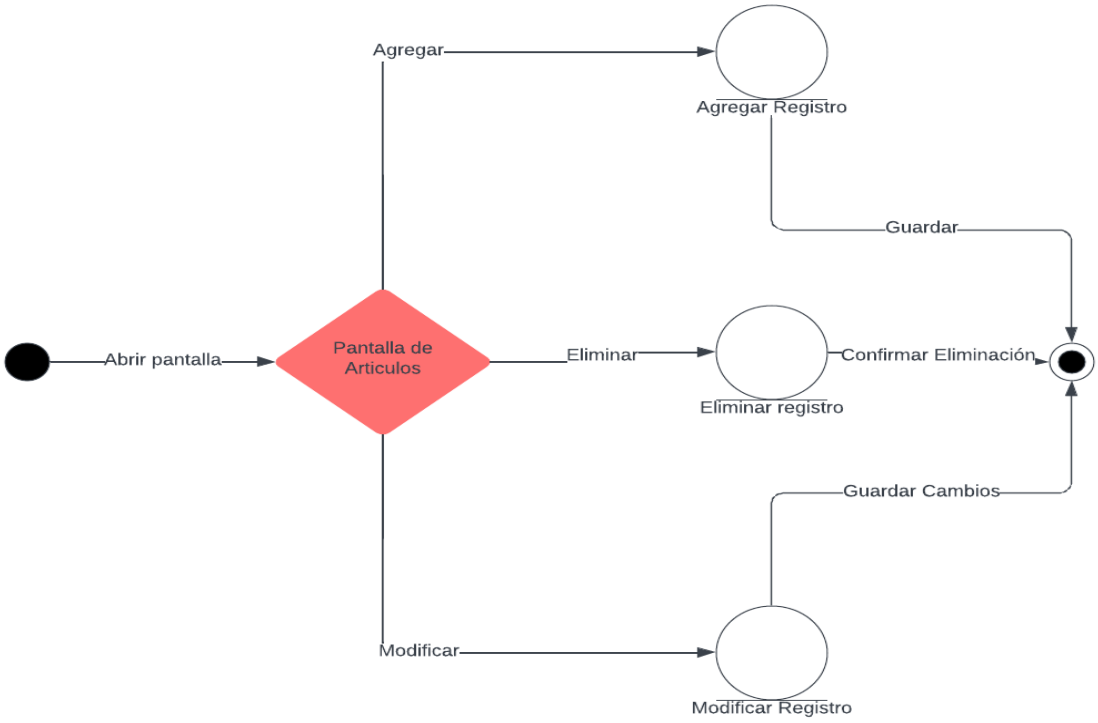


Diagrama de actividad 6. Clientes

Figura 71

Diagrama de actividad - Clientes

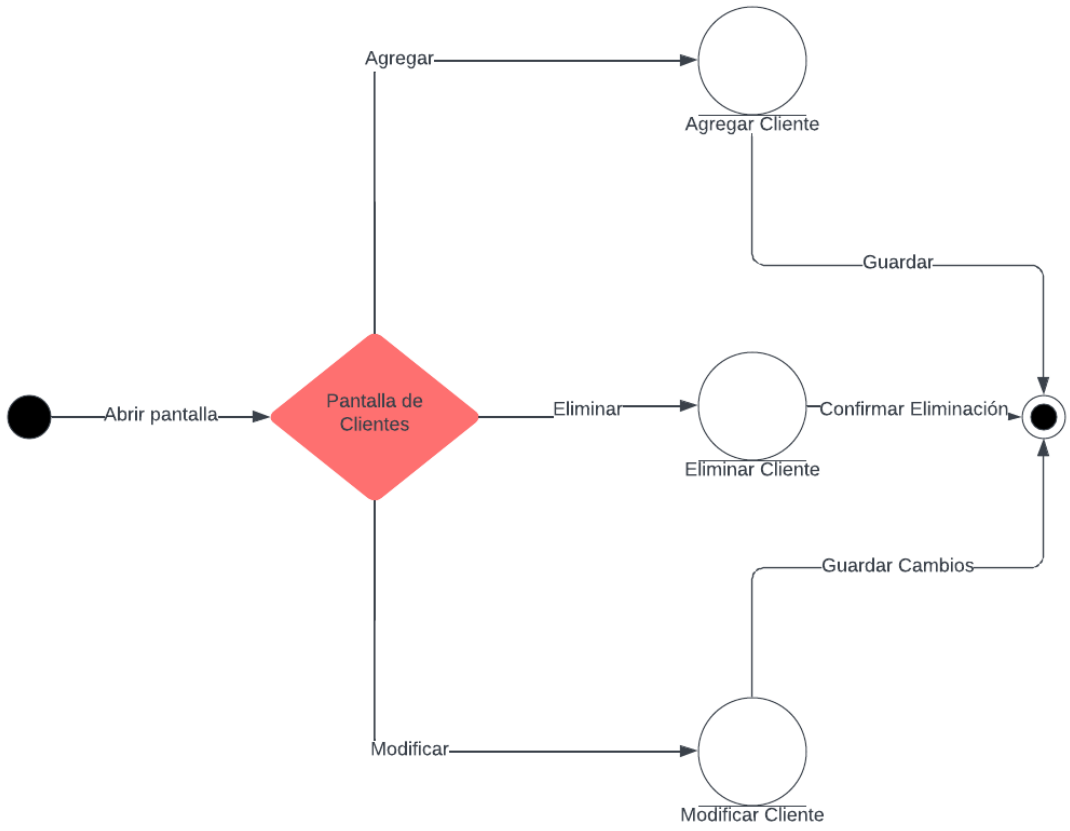


Diagrama de actividad 7. Proveedores

Figura 72

Diagrama de actividad - Mantenimiento de Proveedores

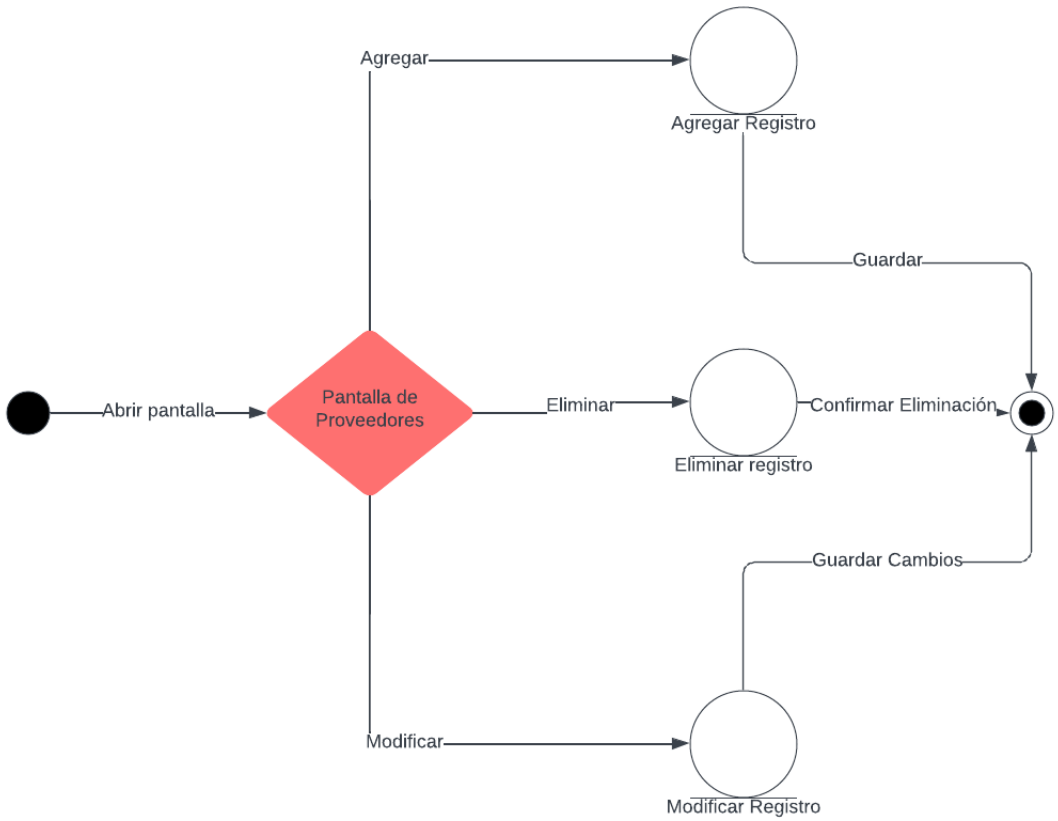


Diagrama de actividad 8. Compras

Figura 73

Diagrama de actividad - Compras

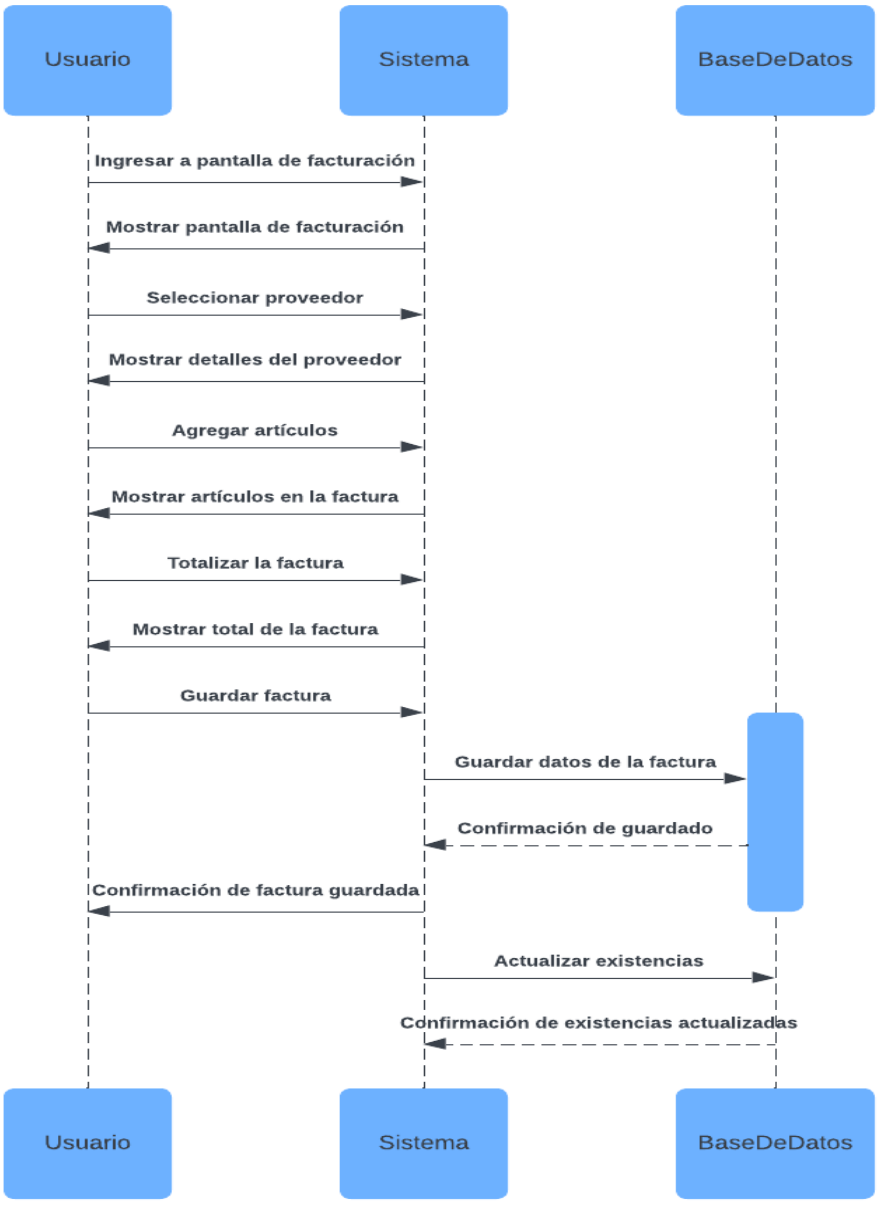


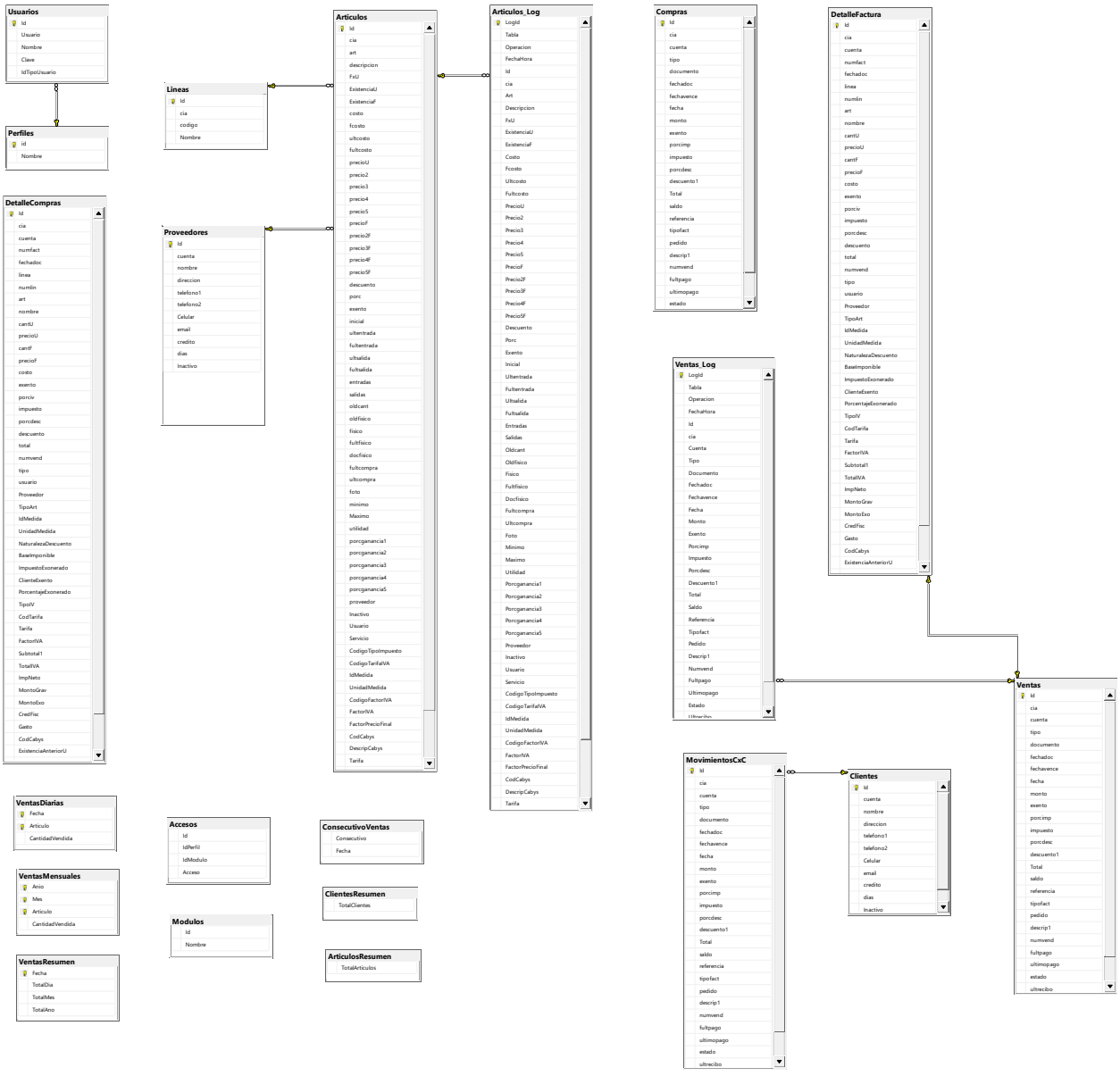
Diagrama de actividad 9. Reportes

Figura 74

Diagrama de actividad - Reportes

Modelo entidad relación

Modelo relacional



Diccionario de datos

Tabla 33

Tabla de compras

Nombre de la tabla	COMPRAS		
Campo	**Tipo de Dato**	**Nulo**	**Llave Primaria**
Id	Int	No	PK
cuenta	char(20)	Si	
tipo	char(2)	Si	
documento	char(20)	Si	
fechadoc	datetime	Si	
fechavence	datetime	Si	
fecha	datetime	Si	
monto	numeric(15,4)	Si	
exento	numeric(14,4)	Si	
porcimp	numeric(10,2)	Si	
Impuesto	numeric(15,5)	Si	
porcdesc	numeric(10,2)	Si	
descuento1	numeric(15,5)	Si	
Total	numeric(15,5)	Si	
saldo	numeric(15,5)	Si	
referencia	char(20)	Si	
tipofact	numeric(1,0)	Si	
pedido	char(20)	Si	
descrip1	char(100)	Si	
numvend	char(10)	Si	
fultpago	datetime	Si	
ultimopago	numeric(14,4)	Si	
estado	char(1)	Si	
ultrecibo	char(10)	Si	

Tabla 34

Tabla de accesos a los sistemas

Nombre de la tabla	ACCESOS		
Campo	**Tipo de Dato**	**Nulo**	**Llave Primaria**
Id	Int	No	PK
IdPerfil	Int	Si	
IdModulo	int	Si	
Acceso	bit	Si	

Tabla 35

Tabla de Módulos del Sistema

Nombre de la tabla	MODULOS		
Campo	**Tipo de Dato**	**Nulo**	**Llave Primaria**
Id	Int	No	PK
Nombre	nvarchar(50)	Si	

Tabla 36

Tabla de consecutivo de facturas de clientes

Nombre de la tabla	CONSECUTIVOVENTAS		
Campo	**Tipo de Dato**	**Nulo**	**Llave Primaria**
Consecutivo	int	No	PK
Fecha	datetime	SI	

Tabla 37

Tabla del registro de ventas a clientes

Nombre de la tabla	VENTAS		
Campo	**Tipo de Dato**	**Nulo**	**Llave Primaria**
Id	int	No	PK
cia	char(2)	Si	
cuenta	char(20)	Si	
tipo	char(2)	Si	
documento	char(20)	Si	
fechadoc	datetime	Si	
fechavence	datetime	Si	
fecha	datetime	Si	
monto	numeric(15, 5)	Si	
exento	numeric(14, 4)	Si	
porcimp	numeric(10, 2)	Si	
impuesto	numeric(15, 5)	Si	
porcdesc	numeric(10, 2)	Si	
descuento1	numeric(15, 5)	Si	
Total	numeric(15, 5)	Si	
saldo	numeric(15, 5)	Si	
referencia	char(20)	Si	
tipofact	numeric(1, 0)	Si	
pedido	char(20)	Si	
descrip1	char(100)	Si	
numvend	char(10)	Si	
fultpago	datetime	Si	
ultimopago	numeric(14, 4)	Si	
estado	char(1)	Si	
ultrecibo	char(10)	Si	
IdCliente	int	Si	

Tabla 38

Tabla de detalle de las facturas de clientes

Nombre de la tabla	DETALLEFACTURA		
Campo	**Tipo de Dato**	**Nulo**	**Llave Primaria**
Id	int	No	PK
cia	char(2)	Si	
cuenta	char(20)	Si	
numfact	char(20)	Si	
fechadoc	datetime	Si	
linea	char(3)	Si	
numlin	numeric(10, 0)	Si	
art	char(100)	Si	
nombre	text	Si	
cantU	numeric(15, 2)	Si	
precioU	numeric(15, 5)	Si	
cantF	numeric(15, 2)	Si	
precioF	numeric(15, 5)	Si	
costo	numeric(15, 5)	Si	
exento	bit	Si	
porciv	numeric(14, 4)	Si	
impuesto	numeric(15, 5)	Si	
porcdesc	numeric(6, 2)	Si	
descuento	numeric(15, 5)	Si	
total	numeric(15, 5)	Si	
numvend	char(10)	Si	
tipo	char(2)	Si	
usuario	char(8)	Si	
Proveedor	char(10)	Si	
TipoArt	char(2)	Si	
IdMedida	numeric(10, 0)	Si	
UnidadMedida	char(15)	Si	
NaturalezaDescuento	char(80)	Si	
BaseImponible	numeric(18, 5)	Si	
ImpuestoExonerado	numeric(18, 5)	Si	
ClienteExento	bit	Si	
PorcentajeExonerado	numeric(5, 2)	Si	
TipoIV	char(2)	Si	
CodTarifa	char(2)	Si	
Tarifa	numeric(4, 2)	Si	
FactorIVA	numeric(5, 4)	Si	
Subtotal1	numeric(18, 5)	Si	
TotalIVA	numeric(18, 5)	Si	
ImpNeto	numeric(18, 5)	Si	
MontoGrav	numeric(18, 5)	Si	
MontoExo	numeric(18, 5)	Si	
CredFisc	numeric(18, 5)	Si	
Gasto	numeric(18, 5)	Si	
CodCabys	nvarchar(200)	Si	
ExistenciaAnteriorU	numeric(10, 2)	Si	
ExistenciaActualU	numeric(10, 2)	Si	
ExistenciaAnteriorF	numeric(10, 2)	Si	
ExistenciaActualF	numeric(10, 2)	Si	

Tabla 39

Tabla de artículos

Nombre de la tabla	ARTICULOS		
Campo	**Tipo de Dato**	**Nulo**	**Llave Primaria**
Id	int	No	PK
cia	char(2)	Si	
art	char(100)	Si	
descripcion	text	Si	
FxU	numeric(10, 0)	Si	
ExistenciaU	numeric(15, 2)	Si	
ExistenciaF	numeric(15, 2)	Si	
costo	numeric(15, 2)	Si	
fcosto	datetime	Si	
ultcosto	numeric(14, 4)	Si	
fultcosto	datetime	Si	
precioU	numeric(15, 2)	Si	
precio2	numeric(15, 2)	Si	
precio3	numeric(15, 2)	Si	
precio4	numeric(15, 2)	Si	
precio5	numeric(15, 2)	Si	
precioF	numeric(15, 2)	Si	
precio2F	numeric(15, 2)	Si	
precio3F	numeric(15, 2)	Si	
precio4F	numeric(15, 2)	Si	
precio5F	numeric(15, 2)	Si	
descuento	numeric(15, 2)	Si	
porc	numeric(14, 4)	Si	
exento	bit	Si	
inicial	numeric(15, 2)	Si	
ultentrada	numeric(15, 2)	Si	
fultentrada	datetime	Si	
ultsalida	numeric(15, 2)	Si	
fultsalida	datetime	Si	
entradas	numeric(15, 2)	Si	
salidas	numeric(15, 2)	Si	
oldcant	numeric(14, 4)	Si	
oldfisico	numeric(14, 4)	Si	
fisico	numeric(14, 4)	Si	
fultfisico	datetime	Si	
docfisico	numeric(15, 0)	Si	
fultcompra	datetime	Si	
ultcompra	numeric(14, 4)	Si	
foto	varchar(200)	Si	
minimo	numeric(15, 2)	Si	
Maximo	numeric(10, 0)	Si	
utilidad	numeric(14, 4)	Si	
porcganancia1	numeric(14, 4)	Si	
porcganancia2	numeric(14, 4)	Si	
porcganancia3	numeric(14, 4)	Si	
porcganancia4	numeric(14, 4)	Si	
porcganancia5	numeric(14, 4)	Si	
proveedor	char(10)	Si	
Inactivo	bit	Si	
Usuario	char(100)	Si	
Servicio	bit	Si	
CodigoTipoImpuesto	char(2)	Si	
CodigoTarifaIVA	char(2)	Si	
IdMedida	int	Si	
UnidadMedida	char(10)	Si	
CodigoFactorIVA	char(2)	Si	
FactorIVA	numeric(10, 3)	Si	
FactorPrecioFinal	numeric(10, 3)	Si	
CodCabys	nvarchar(200)	Si	
DescripCabys	text	Si	
Tarifa	numeric(10, 2)	Si	
IdProveedor	int	Si	
IdLinea	int	Si	
linea	char(3)	Si	
ubicacion	char(20)	Si	

Tabla 40

Tabla de líneas

Nombre de la tabla	LINEAS		
Campo	**Tipo de Dato**	**Nulo**	**Llave Primaria**
Id	int	No	PK
cia	nvarchar(2)	Si	
codigo	nvarchar(50)	Si	
Nombre	nvarchar(200)	Si	

Tabla 41

Tabla de proveedores

Nombre de la tabla	PROVEEDORES		
Campo	**Tipo de Dato**	**Nulo**	**Llave Primaria**
Id	int	No	PK
cuenta	char(20)	Si	
nombre	char(200)	Si	
direccion	text	Si	
telefono1	char(20)	Si	
telefono2	char(20)	Si	
Celular	char(20)	Si	
email	char(100)	Si	
credito	numeric(14, 2)	Si	
dias	numeric(10, 0)	Si	
Inactivo	bit	Si	

Tabla 42

Tabla de movimientos de clientes

Nombre de la tabla	MOVIMIENTOSCXC		
Campo	**Tipo de Dato**	**Nulo**	**Llave Primaria**
Id	int	No	PK
cia	char(2)	Si	
cuenta	char(20)	Si	
tipo	char(2)	Si	
documento	char(20)	Si	
fechadoc	datetime	Si	
fechavence	datetime	Si	
fecha	datetime	Si	
monto	numeric(15, 5)	Si	
exento	numeric(14, 4)	Si	
porcimp	numeric(10, 2)	Si	
impuesto	numeric(15, 5)	Si	
porcdesc	numeric(10, 2)	Si	
descuento1	numeric(15, 5)	Si	
Total	numeric(15, 5)	Si	
saldo	numeric(15, 5)	Si	
referencia	char(20)	Si	
tipofact	numeric(1, 0)	Si	
pedido	char(20)	Si	
descrip1	char(100)	Si	
numvend	char(10)	Si	
fultpago	datetime	Si	
ultimopago	numeric(14, 4)	Si	
estado	char(1)	Si	
ultrecibo	char(10)	Si	
IdCliente	int	Si	

Tabla 43

Tabla de clientes

Nombre de la tabla	CLIENTES		
Campo	**Tipo de Dato**	**Nulo**	**Llave Primaria**
Id	int	No	PK
cuenta	char(20)	Si	
nombre	char(200)	Si	
direccion	text	Si	
telefono1	char(20)	Si	
telefono2	char(20)	Si	
Celular	char(20)	Si	
email	char(100)	Si	
credito	numeric(14, 2)	Si	
dias	numeric(10, 0)	Si	
Inactivo	bit	Si	

Tabla 44

Tabla de usuarios

Nombre de la tabla	USUARIOS		
Campo	**Tipo de Dato**	**Nulo**	**Llave Primaria**
Id	int	No	PK
Usuario	nvarchar(20)	Si	
Nombre	nvarchar(100)	Si	
Clave	nvarchar(200)	Si	
IdTipoUsuario	int	Si	

Tabla 45

Tabla de perfiles

Nombre de la tabla	PERFILES		
Campo	**Tipo de Dato**	**Nulo**	**Llave Primaria**
id	int	No	PK
Nombre	nvarchar(50)	Si	

Diseño de pantallas del sistema

Pantalla de ingreso al sistema

Figura 75

Pantalla de ingreso al sistema

Pantalla principal del sistema

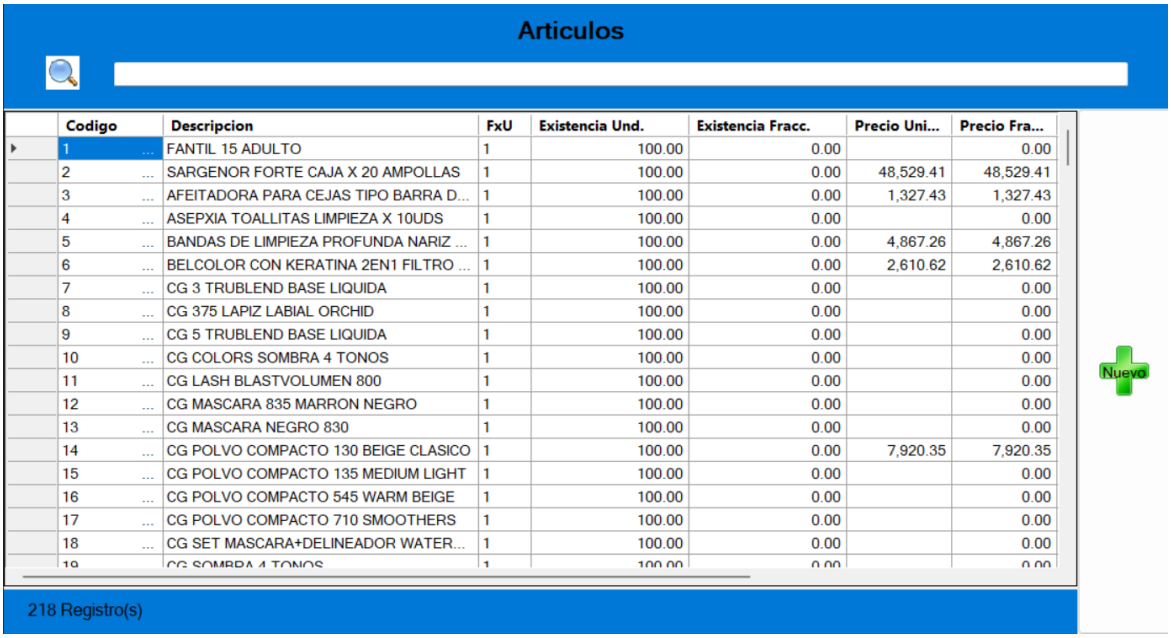
Figura 76

Pantalla principal del sistema

Pantalla de búsqueda de artículos

Figura 77

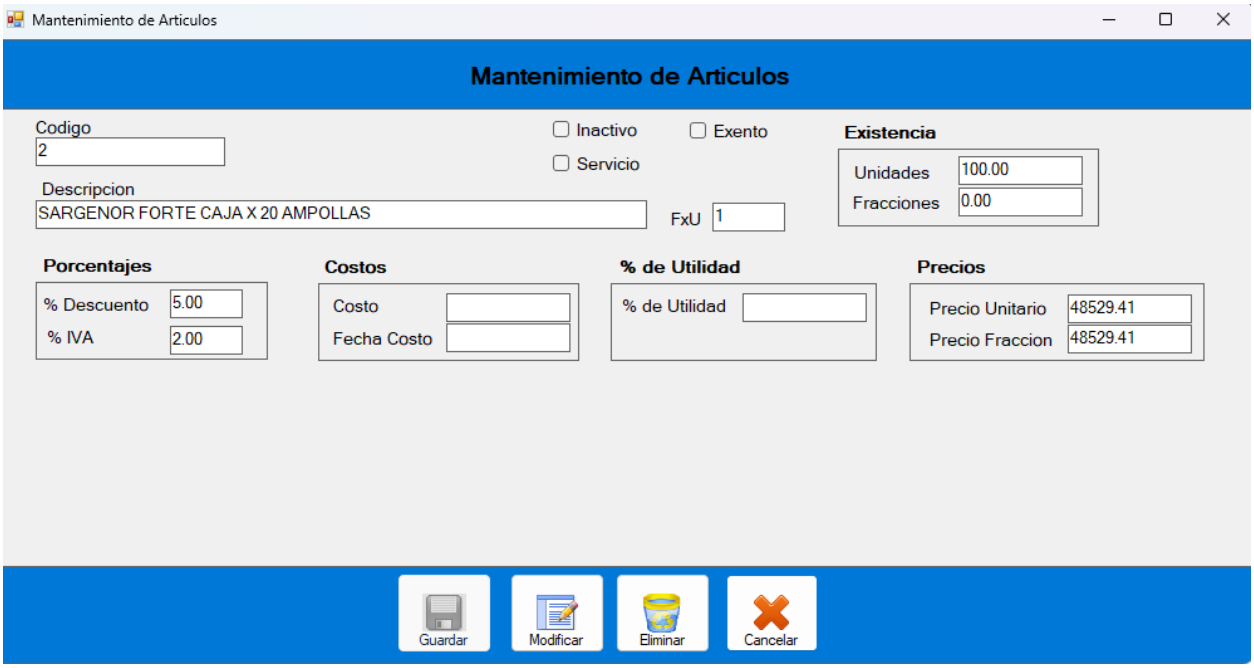
Pantalla de búsqueda de artículos



Pantalla de mantenimiento de artículos

Figura 78

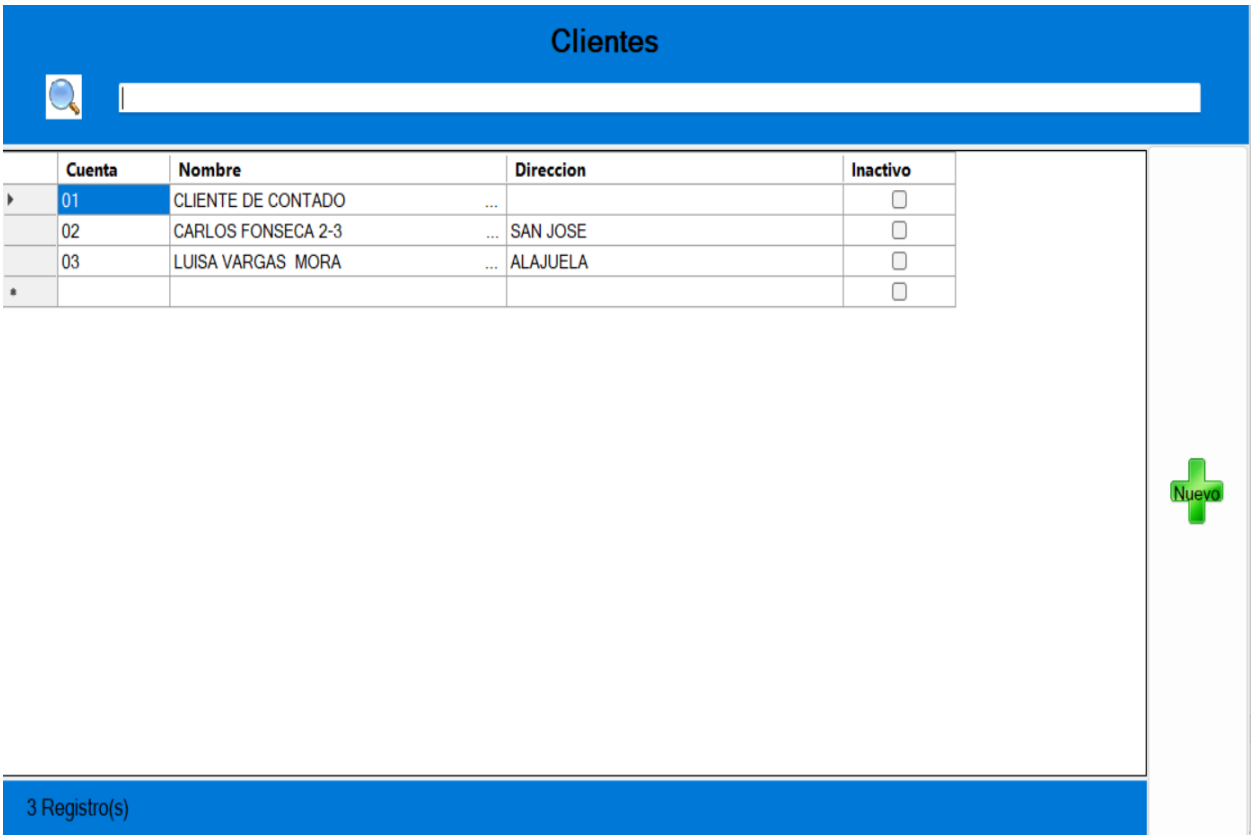
Mantenimiento de artículos



Pantalla de búsqueda de clientes

Figura 79

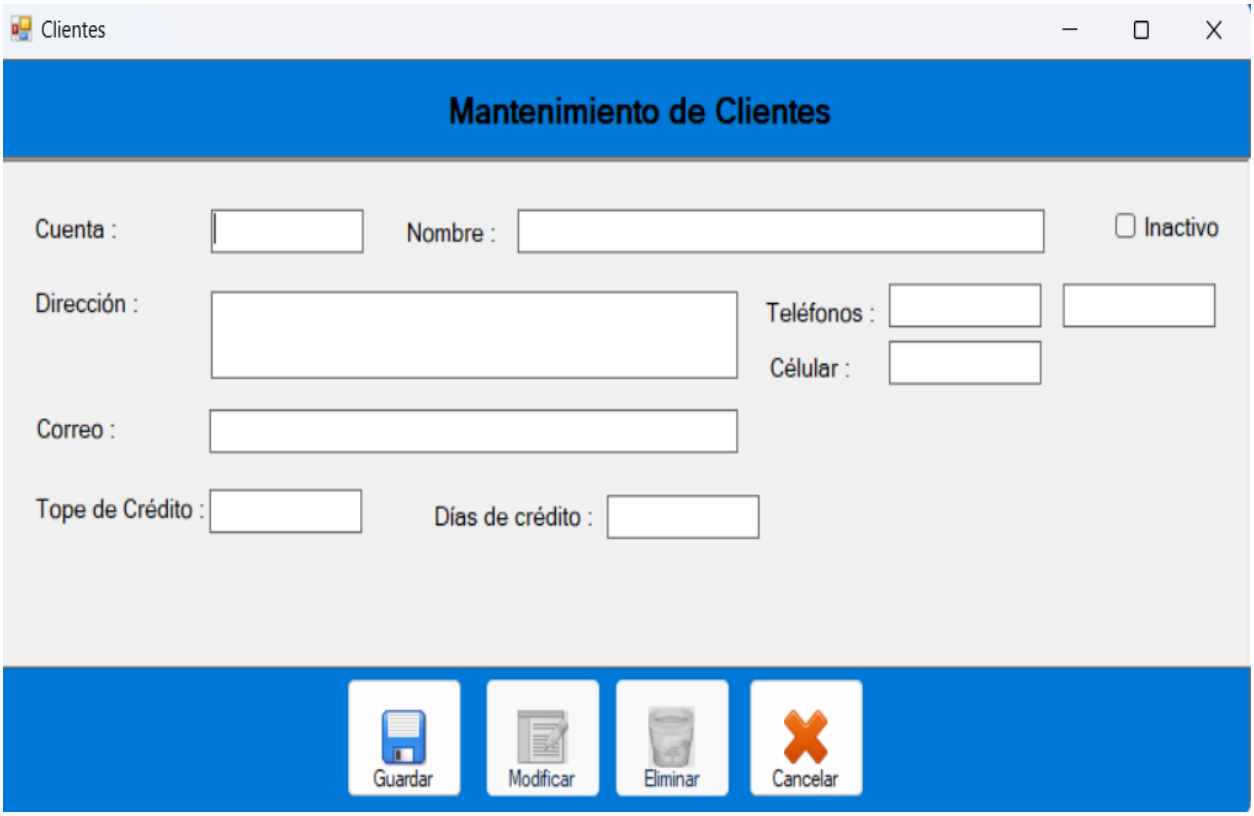
Pantalla de búsqueda de clientes



Pantalla de mantenimiento de clientes

Figura 80

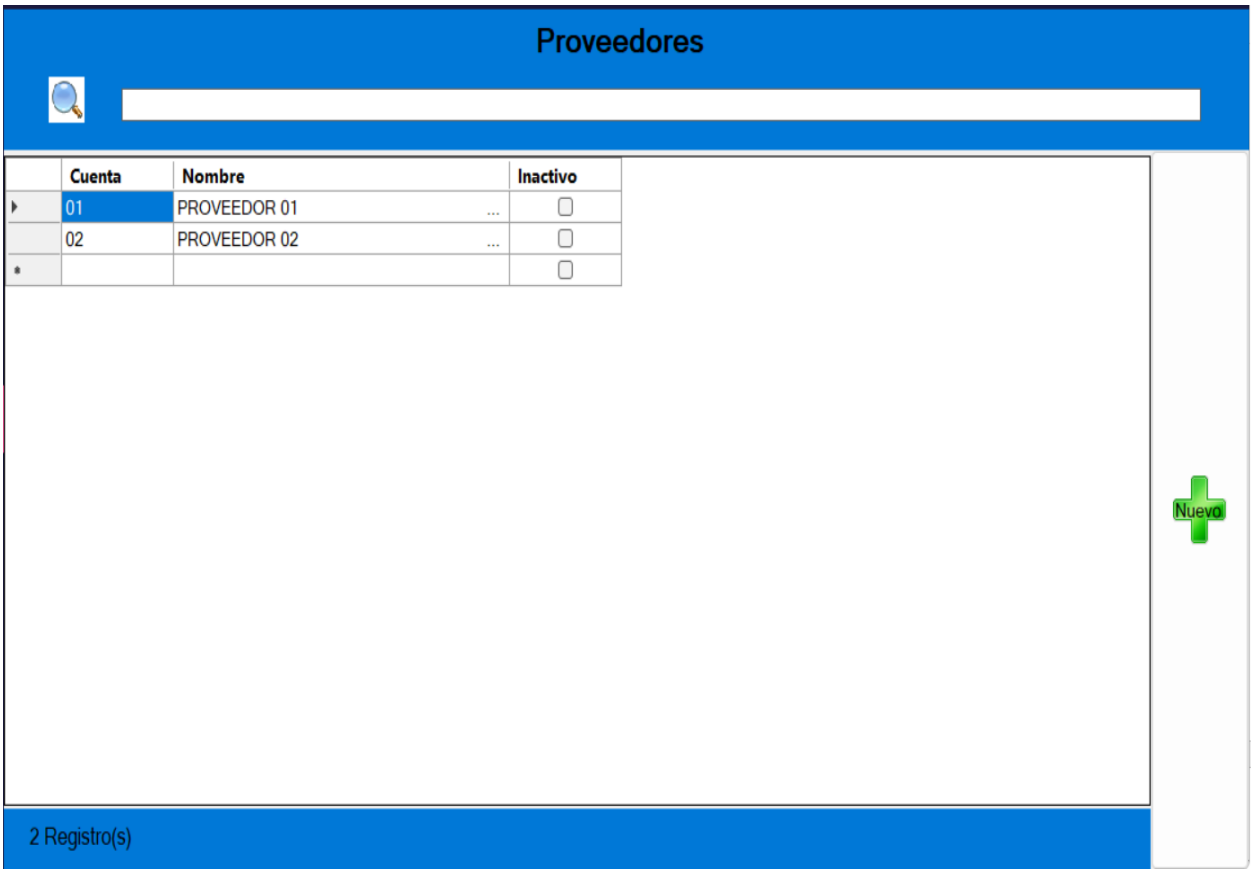
Pantalla de mantenimiento de clientes



Pantalla de búsqueda de proveedores

Figura 81

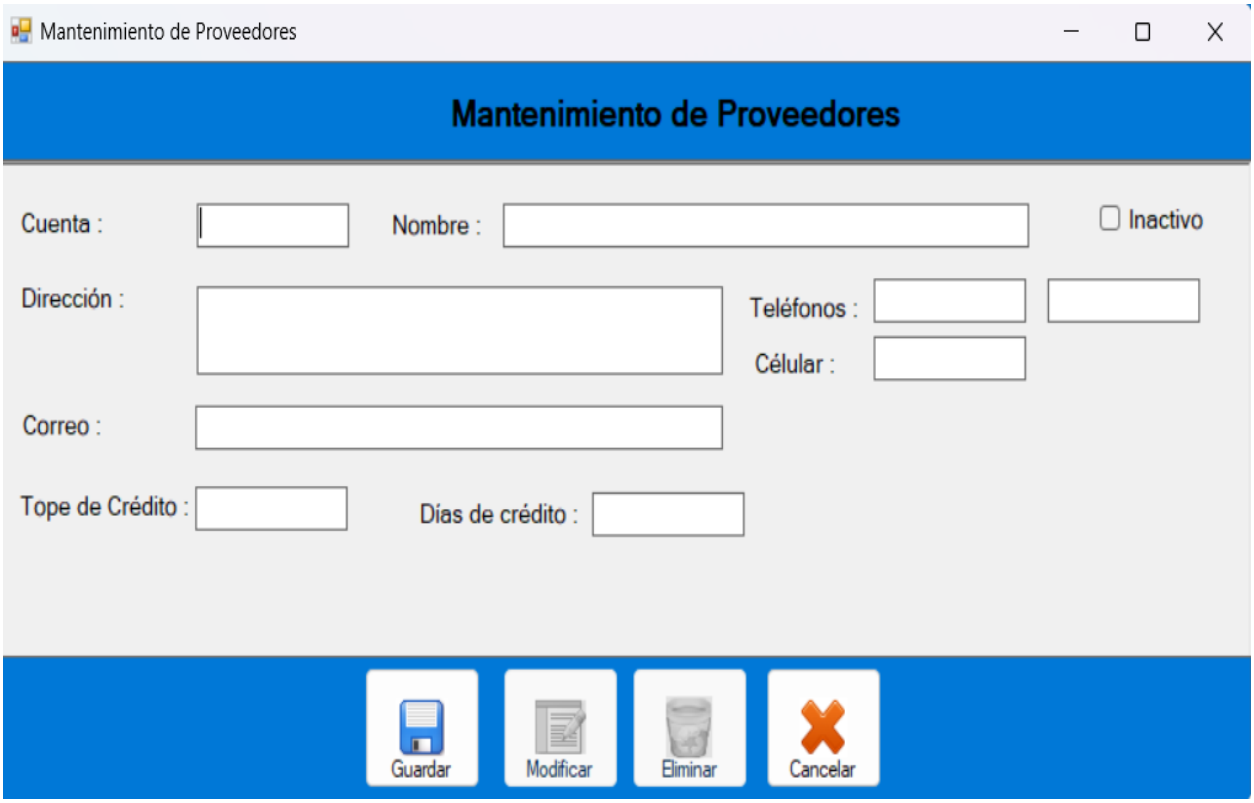
Pantalla de búsqueda de proveedores



Pantalla de mantenimiento de proveedores

Figura 82

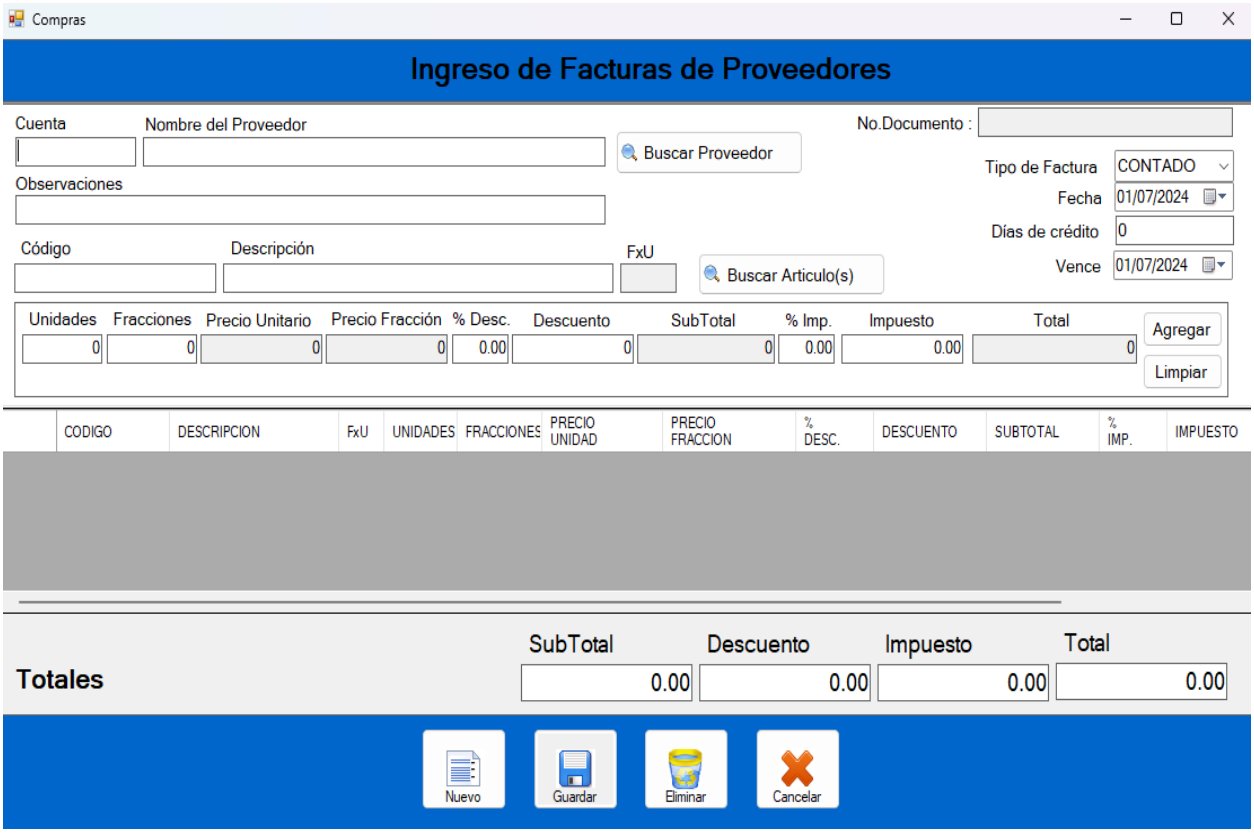
Pantalla de mantenimiento de proveedores



Pantalla de ingreso de compras

Figura 83

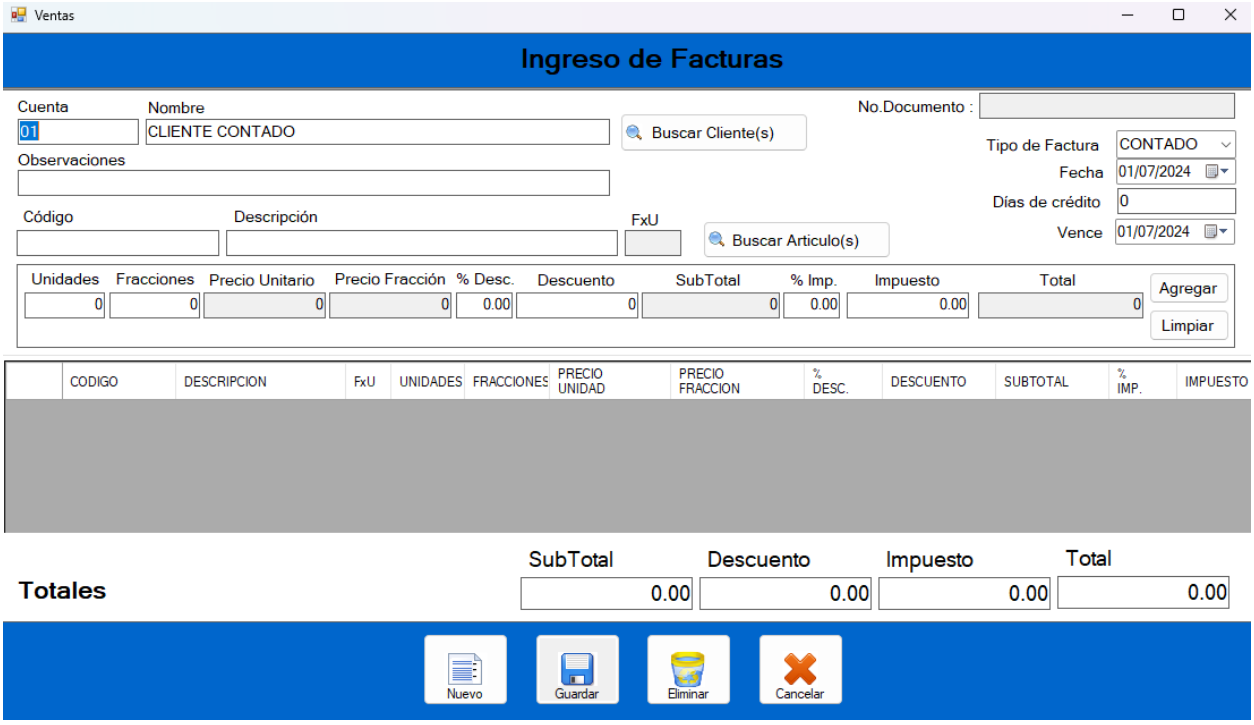
Ingreso de compras de proveedores



Pantalla de ventas

Figura 84

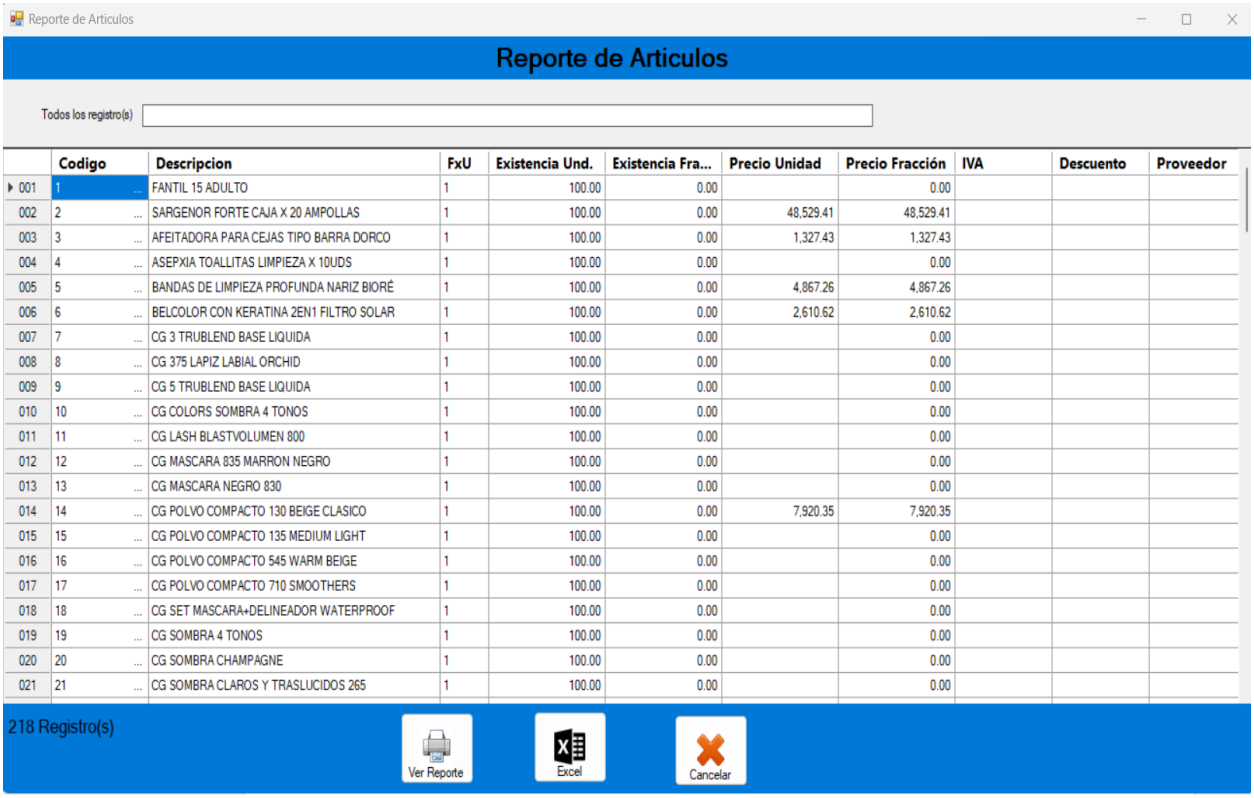
Pantalla de ventas



Pantalla de reporte de artículos

Figura 85

Pantalla de reporte de artículos



	Codigo	Descripcion	FxU	Existencia Und.	Existencia Fra...	Precio Unidad	Precio Fracción	IVA	Descuento	Proveedor
001	1	FANTIL 15 ADULTO	1	100.00	0.00		0.00			
002	2	SARGENOR FORTE CAJA X 20 AMPOLLAS	1	100.00	0.00	48,529.41	48,529.41			
003	3	AFEITADORA PARA CEJAS TIPO BARRA DORCO	1	100.00	0.00	1,327.43	1,327.43			
004	4	ASEPXIA TOALLITAS LIMPIEZA X 10UDS	1	100.00	0.00		0.00			
005	5	BANDAS DE LIMPIEZA PROFUNDA NARIZ BIORÉ	1	100.00	0.00	4,867.26	4,867.26			
006	6	BELCOLOR CON KERATINA 2EN1 FILTRO SOLAR	1	100.00	0.00	2,610.62	2,610.62			
007	7	CG 3 TRUBLEND BASE LIQUIDA	1	100.00	0.00		0.00			
008	8	CG 375 LAPIZ LABIAL ORCHID	1	100.00	0.00		0.00			
009	9	CG 5 TRUBLEND BASE LIQUIDA	1	100.00	0.00		0.00			
010	10	CG COLORS SOMBRA 4 TONOS	1	100.00	0.00		0.00			
011	11	CG LASH BLASTVOLUMEN 800	1	100.00	0.00		0.00			
012	12	CG MASCARA 835 MARRON NEGRO	1	100.00	0.00		0.00			
013	13	CG MASCARA NEGRO 830	1	100.00	0.00		0.00			
014	14	CG POLVO COMPACTO 130 BEIGE CLASICO	1	100.00	0.00	7,920.35	7,920.35			
015	15	CG POLVO COMPACTO 135 MEDIUM LIGHT	1	100.00	0.00		0.00			
016	16	CG POLVO COMPACTO 545 WARM BEIGE	1	100.00	0.00		0.00			
017	17	CG POLVO COMPACTO 710 SMOOTHERS	1	100.00	0.00		0.00			
018	18	CG SET MASCARA+DELINEADOR WATERPROOF	1	100.00	0.00		0.00			
019	19	CG SOMBRA 4 TONOS	1	100.00	0.00		0.00			
020	20	CG SOMBRA CHAMPAGNE	1	100.00	0.00		0.00			
021	21	CG SOMBRA CLAROS Y TRASLUCIDOS 265	1	100.00	0.00		0.00			

Pantalla de reporte de clientes

Figura 86

Pantalla de reporte de clientes

Pantalla de reporte de proveedores

Figura 87

Pantalla de reporte de proveedores

Pantalla de exportación de datos a Excel

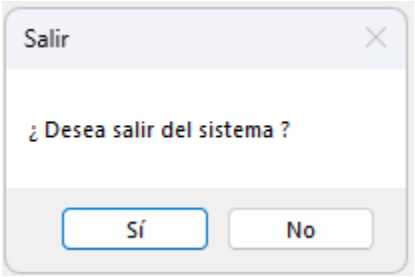
Figura 88

Pantalla de exportación de datos a Excel

Pantalla de Salir del Sistema

Figura 89

Pantalla de salida del sistema



Referencias bibliográficas

Aabz Local. (2023). *Top 84+ imagen ejemplos de diagramas de casos de uso resueltos*. https://abzlocal.mx/top-84-imagen-ejemplos-de-diagramas-de-casos-de-uso-resueltos/

Alpízar Rodríguez, A. y García Bustamante, S. (2019). *Propuesta de Plan de Mercadeo para la Cadena de Farmacias Don Gerardo*. doi:https://www.kerwa.ucr.ac.cr/bitstream/handle/10669/79891/Propuesta%20de%20Plan%20de%20Mercadeo%20para%20la%20cadena%20de%20Farmacias%20DG.pdf?sequence=1&isAllowed=y

Amazon Web Services. (2024). *¿Qué es una base de datos?* https://aws.amazon.com/es/what-is/database/

Aprender a programar PRO. (2020). *¿Qué son los paradigmas de la programación?* https://aprenderaprogramarpro.blogspot.com/2017/05/paradigmas-de-programacion.html

Asesorías. (s.f.). *Historias de Usuario Scrum: Definición, ejemplos y plantillas*. https://asesorias.com/empresas/modelos-plantillas/historias-usuario-scrum/

Barriosó, R. (2024). *Los enfoques cuantitativo y cualitativo de la investigación científica*. Quizlet. https://quizlet.com/gt/290827773/los-enfoques-cuantitativo-y-cualitativo-de-la-investigacion-cientifica-diagram/

Bustamante Caballeros, A. B., Leiva Pérez, D. P., Vargas Espinoza, M. y Vargas Rosales, K. (2018). *Propuesta de un sistema de control interno para el mejoramiento de las cuentas por cobrar y el ciclo de compras e inventario de la farmacia Cavalei S.A.* doi:http://repositorio.sibdi.ucr.ac.cr:8080/jspui/handle/123456789/7273

Castellanos, L. (2009). *Estudio de Factibilidad*. Desarrollo de sistemas. https://desarrollodesistemas.wordpress.com/2009/07/05/estudio-de-factibilidad/

Comunidad Empresas. (2024). *¿Qué es y cuáles son los principales tipos de sistemas de información?* https://ce.entel.cl/articulos/principales-tipos-de-sistemas-de-informacion/

Consult, C. (2023). *Lenguajes de programación: Historia y evolución*. Historia sobre. https://historiasobre.com/lenguajes-de-programacion-historia-y-evolucion/

Coppola, M. (2023). *Qué es JavaScript, para qué sirve y cómo funciona*. Hubspot: https://blog.hubspot.es/website/que-es-javascript

Córdoba, A. (2023). *Algoritmos de programación*. Aprende informáticas. https://aprendeinformaticas.com/algoritmos-de-programacion/

Delgado, H. (2022). *Arquitectura de la Información* . Disenowebakus. https://disenowebakus.net/arquitectura-de-la-informacion.php

Delgado, H. (2022). *Estructura básica de una página Web - html, head y body*. Disenowebakus. https://disenowebakus.net/domine-html-y-dhtml-primeros-pasos.php

Departamento Nacional de Planeación, Colombia. (2020). *Guía para la Elaboración y Presentación de Casos de Uso.* https://colaboracion.dnp.gov.co/CDTI/Oficina%20Informatica/Sistemas%20de%20informaci%C3%B3n/Gu%C3%ADas%20Formatos%20Plantillas/Gu%C3%ADa%20para%20la%20Elaboraci%C3%B3n%20y%20Presentaci%C3%B3n%20de%20Casos%20de%20Uso.pdf?

Desarrolloweb6. (2024). *Bases de datos*. https://desarrolloweb.com/home/bases-de-datos

Developer Mozilla. (2023). *Mobile First*. https://developer.mozilla.org/es/docs/Glossary/Mobile_First

Díaz Garcés, J. (23 de Octubre de 2023). *Historia de la programación*. CIPSA. https://cipsa.net/historia-de-la-programacion/

Enciclopedia de Humanidades. (s.f.). Sistema de información. https://humanidades.com/sistema-de-informacion/

Espinola, J. P. (2022). Paradigma. *Concepto.de*. https://concepto.de/que-es-paradigma/

Feregrino, A. (2017). *La Progamación funcional*. That C# guy. https://thatcsharpguy.com/tv/funcional/

Francia Huambachano, J. (25 de Septiembre de 2017). *Qué es scrum*. https://www.scrum.org/resources/blog/que-es-scrum

García, A. (2020). *Integración continua de Software: GIT y GIT FLOW*. Castor. https://castor.com.co/integracion-continua-de-*software*-git-y-git-flow/

Gonçalves, L. (2024). *Qué es la metodología ágil.* Adapt methodology. https://adaptmethodology.com/es/blog/que-es-la-metodologia-agil/

Gonzalez, D. (2013). *Responsive web design: diseño multidispositivo para mejorar la experiencia de usuario*. https://doi.org/10.1344/BiD2014.31.19

Gottdiener, Z. (2022). *Gestión de riesgos en el desarrollo de software: 7 riesgos comunes. https://www.door3.com*. Door 3: https://www.door3.com/es/blog/understanding-risk-management-in-*software*-development-7-common-risks

Guzmán Ortiz, R. L. (2018). *Sistema Informático de control de ventas para la empresa inversiones Cuba SRL de la ciudad de Chimbote.* Universidad San Pedro (USP). http://repositorio.usanpedro.edu.pe/handle/USANPEDRO/4397

Indeed (2024). *Ventajas y desventajas importantes de la metodología Scrum.* https://es.indeed.com/orientacion-laboral/desarrollo-profesional/ventajas-desventajas-metodologia-scrum

Infoupdate. (s.f.). *Que Es Un Enfoque Cualitativo En Una Investigacion*. Infoupdate: https://infoupdate.org/que-es-un-enfoque-cualitativo-en-una-investigacion/

Ionox. (2020). *Paradigmas de programación: principios básicos de programación.* https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/paradigmas-de-programacion/

Ken, A. (2023). *Requisitos no funcionales: ¿Por qué son importantes?* Gluo. https://www.gluo.mx/blog/requisitos-no-funcionales-por-que-son-importantes

Lacayo Saballos, G. (10 de Marzo de 2013). *Factibilidad Técnica y Económica*. Slides Share. https://es.slideshare.net/slideshow/factibilidad-tcnica-y-econmica/17081507

Learn SQL. (2023). *Qué es SQL*. https://learnsql.es/blog/que-es-sql/

Lomelí, L. (26 de Mayo de 2023). *Metodología srcum*. Innevo. https://blog.innevo.com/metodologia-scrum

Lucidchart. (2024). *Qué es el lenguaje unificado de modelado (UML)*. https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml

Lucidspark. (2017). *Introducción Metodología Scrum: ¿Qué es y cómo funciona?* https://lucidspark.com/es/blog/introduccion-metodologia-scrum

Mancuso, G. (2021). *Historias de usuario de Scrum: Plantilla y Ejemplos*. Compara *software*. https://blog.comparasoftware.com/historias-de-usuario-de-scrum-plantilla-y-ejemplos/

Méndez-Araya, J. (2010). *Sistema de Gestión de Ventas e Inventario*. doi:https://hdl.handle.net/2238/4001

Microsoft. (2023). *Le damos la bienvenida al IDE de Visual Studio | Visual Basic.* https://learn.microsoft.com/es-es/visualstudio/get-started/visual-basic/visual-studio-ide?view=vs-2022

Microsoft. (2024). *¿Qué es Visual Studio?* https://learn.microsoft.com/es-es/visualstudio/get-started/visual-studio-ide?view=vs-2022#Informaci%C3%B3n%20general

Microsoft. (2024). *Desarrollo de aplicaciones .NET*. https://visualstudio.microsoft.com/es/vs/features/net-development/

Microsoft. (2024). *Visual Studio*. https://visualstudio.microsoft.com/es/vs/

MVP Cluster. (2017). *Microsoft presenta Visual Studio 2017 para facilitar el trabajo de los desarrolladores*. https://mvpcluster.com/noticia/visual-studio/

Ortiz Castillo, K. L. (2021). *Sistema informático de gestión de inventario para la Empresa de Servicios y Soluciones Tecnológicas – Sonda 2020.* Universidad San Pedro (USP). https://publicaciones.usanpedro.edu.pe/bitstream/handle/20.500.129076/23056/Tesis_77329.pdf?sequence=1&isAllowed=y

Peiró, R. (2020). *Sistema de información*. Economipedia. https://economipedia.com/definiciones/sistema-de-informacion.html

Peraza, T. (20 de Marzo de 2016). *Lógica Computacional*. Lógica Computacional. https://tperaza.wordpress.com/2016/03/20/logica-computacional/

Pérez, A. (2016). *¿Qué son las metodologías de desarrollo de software?* OBS Business School. https://www.obsbusiness.school/blog/que-son-las-metodologias-de-desarrollo-de-*software*

Pérez-Vergara, I., Cifuentes-Laguna, M., Vásquez-García, C. y Marcela-Ocampo, D. (2013). Un modelo de gestión de inventarios para una empresa de productos alimenticios. *Ingenieria Industrial, XXXIII*(2). doi:https://www.redalyc.org/pdf/3604/360433580012.pdf

PMO Informática. (2016). *7 Técnicas de levantamiento de requerimientos software*. https://www.pmoinformatica.com/2016/08/tecnicas-levantamiento-requerimientos.html

Pressman, R. (2015). *Ingenieria del Software.* Grupo Editorial Patria.

Productiviza. (2024). *Historias de Usuario en SCRUM: EJEMPLOS y PLANTILLA*. https://www.productiviza.com/historias-de-usuario-en-scrum-aplicalas-asi-con-ejemplos/

Programacion Pro. (2023). *Ventajas de la programación orientada a objetos*. https://programacionpro.com/ventajas-de-la-programacion-orientada-a-objetos/

Pulido, A. (2024). *Arquitectura web: definición y ejemplos*. Online Zebra. https://onlinezebra.com/blog/arquitectura-web-definicion-y-ejemplos/

QbDGroup. (2022). *Life Sciences Insights*. https://qbdgroup.com/es-es/blog/validacion-de-sistemas-informatizados-requisitos-de-usuario-urs/#:~:text=Los%20Requisitos%20de%20Usuario%20(RU,%E2%80%9D%2C%20la%20justificaci%C3%B3n%20del%20sistema.

Quirós, M. (2024). *Estudio de factibilidad: Qué es y qué tipos hay*. Economipedia. https://economipedia.com/definiciones/estudio-de-factibilidad.html

Raffino, Equipo editorial, Etecé. (2024). *Algoritmo en informática*. https://concepto.de/algoritmo-en-informatica/

Red Hat. (2022). *¿Qué es la metodología ágil?* https://www.redhat.com/es/topics/devops/what-is-agile-methodology

Redacción APD. (2024). *Cómo aplicar la metodología Scrum y qué es el método Scrum.* https://www.apd.es/metodologia-scrum-que-es/#:~:text=La%20metodolog%C3%ADa%20Scrum%20es%20un,resultado%20de%20un%20proyecto%20determinado.

Rigotti, T. (2023). *Qué es el diseño responsive y por qué debes hacerlo así desde YA*. Rodanet. https://rodanet.com/diseno-responsive/

Roch Moraguez, E. (2023). *¿Qué son los paradigmas de programación: Cómo trabajan y para qué sirven?* Lov technology. https://lovtechnology.com/que-son-los-paradigmas-de-programacion-como-trabajan-y-para-que-sirven/

Roch Moraguez, E. (2024). *¿Qué es Visual Basic (.NET)?* LovTechnology.com. https://lovtechnology.com/que-es-visual-basic-net/

Rockcontent. (2020). *Bootstrap: guía para principiantes de qué es, por qué y cómo usarlo*. https://rockcontent.com/es/blog/bootstrap/

Rodríguez Gómez, J. (2018). *Casos Empresariales*. Marketing Inteli. https://www.marketinginteli.com/casos-empresariales/

Rodríguez, S. (2020). *Valores Scrum: cómo ponerlos en práctica*. Scrumio. https://www.scrumio.com/blog/valores-scrum-como-ponerlos-en-practica/

Rouse, M. (2024). *Sistema de gestión de bases de datos (DBMS).* Techopedia. https://www.techopedia.com/es/definicion/sistema-gestion-bases-datos-dbms

Sommerville, I. (2011). *Ingenieria de Software.* Addison-Wesley.

Stark cloud. (2024). *¿Qué es el Desarrollo de Software?* https://www.starkcloud.com/starkcloud-blog/cloud/que-es-el-desarrollo-de-*software*

Suarez M., J. (2016). *Costa Rica Antigua e Inédita.*

TDM. (2023). *Plan de pruebas de software: 8 pasos para realizarlas*. Icaria Technology. https://icariatechnology.com/plan-de-pruebas-de-*software*/

Team Asana. (2024). *Matriz de riesgos: cómo evaluar los riesgos para lograr el éxito del proyecto (incluye ejemplos)*. Asana. https://asana.com/es/resources/risk-matrix-template

Tesis y Masters. (2024). *Factibilidad económica*. https://tesisymasters.cl/factibilidad-economica/

Tic Portal. (2023). *Base de datos SQL*. https://www.ticportal.es/glosario-tic/base-datos-sql

UANL. (2024). *Los enfoques cuantitativo y cualitativo en la investigación*. Slide Share. https://es.slideshare.net/slideshow/los-enfoques-cuantitativo-y-cualitativo-en-la-investigacin/30112318#7

Ulate Soto, I. y Vargas Morúa, E. (2014). *Metodología para elaborar una tesis.* Universidad Estatal a Distancia.

Universidades Santander. (2020). *Metodologías de desarrollo de software: ¿qué son?* https://www.santanderopenacademy.com/es/blog/metodologias-desarrollo-*software*.html

Valdés Souto, F. (2013). *Evaluación de factibilidad de los proyectos de TI*. https://franciscovaldessouto.wordpress.com/2013/05/14/evaluacion-de-factibilidad-de-los-proyectos-de-ti/

Vallaeta. (2024). *Framework.* Cleanpng. https://www.cleanpng.com/png-visual-basic-net-net-framework-asp-net-visual-basi-3181312/

Vargas Encalada, E. E., Rengifo Lozano, R. A., Guizado Oscco, F. y Sánchez Aguirre, F. M. (2019). Sistemas de información como herramienta para reorganizar procesos de manufactura. *Revista Venezolana de Gerencia*, *24*(85). https://www.redalyc.org/journal/290/29058864015/html/#redalyc_29058864015_ref6

Visure Solutions. (2024). *Qué son los requisitos funcionales: ejemplos, definición, guía completa*. https://visuresolutions.com/es/blog/functional-requirements/

Wikipedia. (s.f.). *Paradigma de programación*. https://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n

Zoluxiones. (2022). *Programación Extrema o Metodología XP*. [Publicación de Facebook]. https://www.facebook.com/zoluxiones/photos/zoluxionesxp-la-programaci%C3%B3n-extrema-o-metodolog%C3%ADa-xp-es-una-metodolog%C3%ADa-de-marc/2898822906863300/