Desarrollo de una aplicación de escritorio para el control de inventarios de la papelería Vargos en la ciudad de Bogotá.

Sebastián Antonio Rodríguez Rivera

Universitaria Agustiniana
Facultad de Ingenierías
Programa de Tecnología en Desarrollo de Software
Bogotá D. C.
2022

Desarrollo de una aplicación de escritorio para el control de inventarios de la papelería vargos en la ciudad de Bogotá,

Sebastián Antonio Rodríguez Rivera

Director:
Duver Rene Acosta Orjuela

Trabajo de grado para optar al título de Desarrollador de Software

Universitaria Agustiniana
Facultad de Ingenierías
Programa de Tecnología en Desarrollo de Software
Bogotá D. C.
2022

Quiero expresar mi gratitud a Dios que siempre está a mi lado y a mi mamá por su apoyo en cada instante de mi vida

Agradecimiento

Mi más sincero agradecimiento para mi tía Lucy Rodríguez y mi madre Gloria Rivera que me brindaron el apoyo económico suficiente para culminar mi carrera.

Resumen

El objetivo central del proyecto es el desarrollo de una aplicación de escritorio para el control de inventarios específicamente para la papelería vargos en la ciudad de Bogotá, esto con el propósito de gestionar de manera correcta y organizada el stock de sus productos los cuales son comercializados por esta papelería, ya que anteriormente se llevaba el control manual por medio de libretas o archivos en Excel ya que nunca desde que empezó a operar esta papelería ha contado con una herramienta tecnológica, esta aplicación tendrá el inicio de sesión del usuario el cual será el administrador del negocio y el vendedor con restricciones limitadas el cual solo realizará las ventas de la papelería, el software contara con cinco módulos principales los cuales son: Inventario de productos, control de usuarios, proveedores, ventas e historial de ventas esto para la optimización de procesos dentro de la misma. Para el desarrollo del documento se referenciaron cinco trabajos similares que implementaron una aplicación para optimizar procesos, y cubrir esta necesidad.

Palabras claves: Control, stock, manual, archivos.

Abstract

The main objective of the project is the development of a desktop application for inventory control specifically for the stationery vargos in the city of Bogota, this in order to manage correctly and organized the stock of their products which are marketed by this stationery, This application will have the user login which will be the administrator of the business and the seller with limited restrictions which will only make the sales of the stationery, the software will have five main modules which are: Product inventory, user control, suppliers, sales and sales history for the optimization of processes within it. For the development of the document, five similar works were referenced that implemented an application to optimize processes and cover this need.

Keywords: Control, stock, manual, archivos.

Tabla de Contenidos

Lista de tablas

Lista de figuras

Introducción

Un sistema de control de inventarios es una herramienta que permite gestionar y registrar grandes cantidades de productos que existen dentro de un negocio, ya sean pequeñas o medianas empresas. En la actualidad es indispensable contar con un sistema de control de inventarios, el cual el sistema le indica que artículos están disponibles en un momento dado y que artículos faltan (posiblemente agotados). También para determinar el nivel de las ventas de determinado producto como e identificar productos cercanos. Es por esto que desde mis conocimientos previos en tecnología se propone crear un software que permita a la papelería vargos darles un orden a sus productos de venta comercial por medio de la implementación de un sistema de control de inventarios, que permita

Para esto se realiza una aplicación de escritorio que permita, al sistema acceder a una base de datos y de la cantidad costo y precio de comercialización de cada producto, que conforma el inventario de la compañía. Comúnmente los productos se les da una identificación única, para el respectivo registro del producto. Con el fin de administrar adecuadamente los inventarios y llevar de manera eficiente cada movimiento de sus productos, es decir entradas y salidas de mercancías, para el adecuado control de la información.

Cuando no se le da el respectivo control y manejo de inventarios en un alto porcentaje es posible que se generen perdidas de dinero y a futuro se vean más reflejadas las pérdidas económicas de su negocio, en definitiva, para dar reducción a estos riesgos es imprescindible tener claridad, cuáles son sus necesidades y la de sus clientes para llegar a tomar la mejor decisión para su negocio, para permitir implementar un sistema de control de inventarios con altos índices de calidad.

Las entradas y salidas de productos exigen llevar un control de las existencias, ya que es el registro de los bienes resguardados en un almacén, este control es de vital importancia porque permite conocer le existencia real de los productos.

1. Título del proyecto de software

Desarrollo de una aplicación de escritorio para el control de inventarios de la papelería vargos en la ciudad de Bogotá,

2. Planeación del proyecto

2.1. Objetivos del proyecto

2.1.1. Objetivo General

Desarrollar una aplicación de escritorio para el control de inventarios de la papelería vargos en la ciudad de Bogotá, con el fin de gestionar de manera organizada los productos que son comercializados por esta papelería.

2.1.2. Objetivos Específicos

- Establecer la base de datos de mongodb para la conexión de la aplicación de escritorio.
- Definir dentro del programa de escritorio un módulo de ventas para los productos que son comercializados por la papelería vargos ubicada en Bogotá en la localidad de Fontibón barrio zona franca.
- Aplicar la metodología ágil Extreme programming, o metodología XP en el desarrollo del sistema de control de inventarios para la papelería vargos ubicada en Bogotá en el barrio zona franca en la localidad de Fontibón.
- Diseñar dentro de la aplicación cinco módulos principales, productos, control de usuarios, proveedores, ventas e historial de ventas, con el fin de que el usuario pueda registrar un dato editarlo o simplemente eliminarlo de la base de datos.

2.2. Planteamiento del problema y/o necesidad

En la mayoría de los casos no se cuenta con un sistema de control de inventarios de alta calidad o simplemente hay pymes que siguen utilizando el método convencional para llevar el control de su negocio, sin contar con ninguna clase de registro. Como es el caso de la papelería vargos que, desde sus inicios en el año 2009 hasta la actualidad, ha llevado el control de sus inventarios por medio físico, en el último tramo del 2020 ha visto reflejado que la manera convencional de llevar el control de sus productos lo ha llevado a bajar sus ganancias comerciales.

Llevar el mal control del inventario genera costos elevados en su negocio, es decir tener demasiado inventario, tiene como finalidad ser destruido o dañado, por circunstancias que están fuera de control, si no se cuenta con un sistema que elimine, el inventario que este defectuoso. Se puede determinar si se está gastando en inventario adicional del que en realidad no se está requiriendo, debido al no saber en realidad cuanta mercancía tiene a disposición, esto por consecuencia lleva a un desperdicio de dinero innecesario.

La universidad de los andes de Mérida, Venezuela (Duran, 2012) la administración de los inventarios es un tema de vital importancia para evitar problemas financieros en las organizaciones, ya que es un componente fundamental en la productividad de una empresa ya que es el activo corriente de menor liquidez que manejan y que además contribuye a generar rentabilidad.

Por otra parte, una planeación deficiente, seda a partir de no realizar el respectivo seguimiento de los inventarios, por ende, esta mala planificación con lleva a no tener claridad de cuantos productos hay en el stock y por consecuencia hay una alta probabilidad de no tener un producto que el cliente este necesitando, y esto genera bajo rendimiento en las ventas, y perdidas de dinero a corto o largo plazo para su negocio.

Por ende, el mal manejo de los inventarios genera múltiples malestares en una organización, para esto es necesario implementar estrategias para el respectivo control de los inventarios.

Con el propósito de dar solución a esta necesidad se implementará un sistema de control de inventarios para el control de la mercancía de la papelería Vargos en la ciudad de Bogotá, la cual el software de escritorio contará con seis módulos entre ellos el login de usuario del administrador el cual tendrá acceso a todos los módulos del sistema, en caso contrario el vendedor tendrá acceso al sistema, pero únicamente al módulo de ventas, después de validar las credenciales, el usuario encontrara cinco pestañas la primera de ellas será la pestaña de usuarios, seguido de la pestaña de inventarios donde se registraran todos los productos comercializados por la papelería vargos, en cuarto lugar encontramos la pestaña de proveedores donde el usuario visualizara un formulario donde se podrán registrar todos los datos correspondientes a los proveedores, en quinto lugar tendremos la pestaña o el módulo de ventas, y por último el usuario encontrara la pestaña de historial de ventas donde podrá visualizar las ventas que ha realizado durante los días, cada vez que se registra un producto nuevo en el formulario de los inventarios automáticamente el producto se almacenara en una lista para proceder a realizar la venta, el usuario podrá cancelar cualquier producto que el cliente ya no dese llevar, todos los registros serán almacenados en la base de datos de MongoDB.

2.3. Alcance del proyecto

El software Inventarios Vargos se desarrolló con el fin de ayudar a la papelería a llevar el inventario de sus productos para gestionar y controlar su stock de manera eficaz y más eficiente. El Software brindara, control de las existencias reales de los productos, donde se almacenarán previamente en la base datos, teniendo la oportunidad de agregar un producto, editarlo o respectivamente eliminarlo, entre las opciones del sistema se encontrarán las pestañas inventario de productos, proveedores y el modulo para gestionar las ventas de la papelería.

2.4. Metodología de desarrollo de software

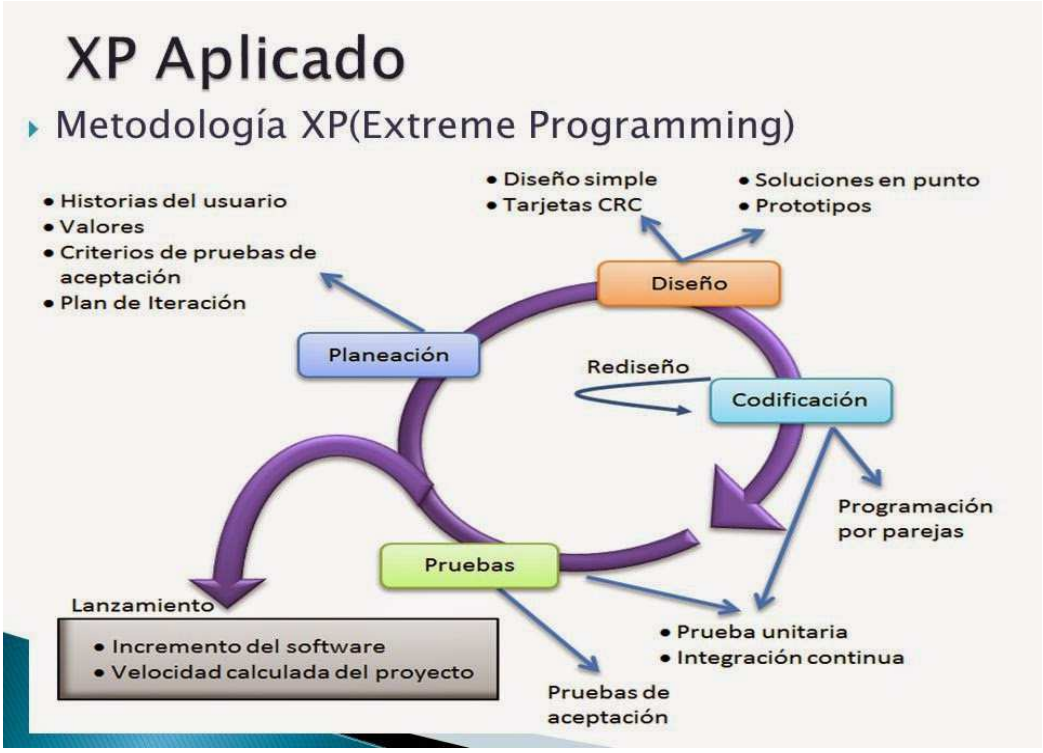


Figura 1. Metodología XP – (Google).

Para el presente proyecto se toma como base la metodología descriptiva con el propósito de recolectar información clave y veras del inventario de los productos que son comercializados por esta papelería Vargos de la ciudad de Bogotá, de tal manera que se pueda detallar los problemas que hay se presentan.

La metodología aplicada para cumplir los objetivos del presente proyecto, es la metodología ágil extreme programing o programación extrema (XP), con el objetivo de automatizar, su papelería vargos, implementando un Software de control de inventarios, mediante esta metodología, la cual se basa en las necesidades del cliente, en el diseño y la planificación que permite la ejecución del proyecto sea más rápida y dinámica.

XP nace oficialmente en aproximados de marzo de 1996 en un proyecto desarrollado por Kent Beck en DaimlerChrysler, después de haber trabajado varios años con Ward Cunningham en busca de una nueva aproximación al problema del desarrollo de software que hiciera las cosas más simples de lo que nos tenían acostumbrados los métodos existentes. Para muchos, XP no es más que sentido común. Kent definió cuatro grandes tareas a realizar en el desarrollo de todo proyecto: planificación, diseño, desarrollo y pruebas; teniendo siempre presente las cuatro características básicas que debe reunir un programador XP: simplicidad en el desarrollo, comunicación entre las partes implicadas, realimentación para poder reutilizar y coraje. (sergioalbertoc, 2015)

Con el propósito de ir construyendo un producto a la medida y se adapte a los requerimientos del cliente, ya que esta metodología se acopla de gran manera al proyecto en desarrollo, porque es una metodología flexible, cuenta con un diseño simple, ya que aparte de ser una de las metodologías más exitosas de la actualidad se usa para el desarrollo de proyectos cortos, como base esta metodología posee cinco valores que establecen el fundamento para todo proyecto realizado como parte de XP, los cuales son la comunicación, simplicidad, retroalimentación, valentía, respeto.

Telassim Gómez resalta esta metodología entre las más adoptadas para la realización de proyectos, “XP es una de las llamadas metodologías ágiles de desarrollo de software más exitosas de los tiempos recientes. La metodología propuesta en XP está diseñada para entregar el software que los clientes necesitan en el momento en que lo necesitan. XP alienta a los desarrolladores a responder a los requerimientos cambiantes de los clientes, aún en fases tardías del ciclo de vida del desarrollo”. (Gomez, 2018, pág. 19)

Para muchos teóricos esta metodología se aproxima a tener una calidad óptima del software, esto debido a que el ciclo de vida del software, van surgiendo cambios a medida del transcurso del desarrollo estos cambios son naturales, entre más cambios, se puede estar más cerca de obtener

mejores resultados, y otorgar la satisfacción del cliente, teniendo como uno de sus objetivos minimizar los riesgos tiempo, costo, calidad, alcance.

Esta programación extrema posee un conjunto de reglas y buenas practicas, que se dan en contexto a partir de cuatro actividades estructurales que son planeación. Diseño, codificación y pruebas.

Planificación: Se escucha y se recrea las historias de los usuarios, las dos partes involucradas en el proceso, (Usuario – Cliente). Con la finalidad de recabar los requerimientos necesarios para el desarrollo del software, planteando los objetivos, para posteriormente alcanzarlos, sin entrar mucho en detalle, es decir partir de una necesidad en el cual posteriormente se le da un orden de cómo se realizará el proyecto, para entender el contexto del negocio del software (Rodriguez, 2014, pág. 8).

Para la realización de esta fase, se llevó a cabo una serie de lineamientos, los cuales se desarrollaron a través de tres procesos fundamentales para el inicio de la realización del Software, en el primer proceso se involucra el dueño de la Papelería Vargos, el cual determino las funcionalidades que desea en el sistema de información, En el segundo proceso como desarrollador del proyecto, se debe determinar los procesos que se utilizan y se conlleva a la realización de un prototipo el cual dictamina la validez del desarrollo del proyecto, que en esta oportunidad es un proyecto viable para su desarrollo. Para el tercer proceso es fundamental, estar en constante comunicación con el usuario para evitar fallos en el sistema a futuro por mal levantamiento de la información.

Codificación: En la codificación el cliente hace parte vital del equipo de desarrollo, es importante su presencia en las distintas fases de X, P, ya que el cliente debe supervisar que se cumpla la funcionalidad específica, es importante destacar que el código se escribe partiendo de unos estándares establecidos ya que tienen como finalidad facilitar su comprensión y escalabilidad, se debe tener en cuenta la codificación primero de la prueba antes que el código esto debido a que XP no contiene una respuesta clara a esta inquietud (Rodriguez, 2014).

En la fase de la codificación del proyecto, se tuvo constante acompañamiento por parte del Administrador de la papelería Vargos el cual, desde su experiencia presento las especificaciones

que desea tener en el Software, estableciendo estándares de calidad, con el fin de que el Software de inventarios, sea de fácil compresión dentro del sistema.

Diseño: El diseño de esta metodología está basado en la simplicidad, es decir en la sencillez de su diseño, la recomendación XP se debe invertir el tiempo justo y necesario para la elaboración de diagramas y diseños de interfaz gráfica, el cual costara menor tiempo de ejecución y esfuerzo en el desarrollo (Rodriguez, 2014).

Para el cumplimiento de esta fase dentro del desarrollo del software, el diseño se basó en la simplicidad, y en establecer los menores tiempos de ejecución en el desarrollo del diseño de la interfaz, siendo un diseño sencillo, evitando complicaciones futuras en esta fase siguiendo las recomendaciones que establece esta metodología.

Prueba: Uno de los pilares de esta metodología dentro de esta fase, es el proceso el cual podemos estar probando constantemente el funcionamiento del código, las veces que sea necesario, nos permite aumentar más la calidad del sistema disminuyendo así el número de errores no detectados y disminuyendo de esta manera que se generen errores a lo largo del tiempo del desarrollo, a partir de esto también se puede evitar malestares al momento de realizar alguna modificación que se requiera (Rodriguez, 2014).

En la ejecución de esta fase, dentro del desarrollo del software, se puede destacar esta fase como una de las más importantes en el proceso del desarrollo, el cual reflejamos dentro del sistema, haciendo pruebas constantes para dictaminar el funcionamiento del código y previniendo errores a lo largo del proceso de ejecución del proyecto, y tener capacidad de reacción si se debe hacer alguna modificación dentro del código.

3. Marco teórico y Estado del arte

Este proyecto tiene como objetivo implementar un sistema de control de inventarios esto con el propósito de llevar el orden y control de las existencias de los productos, este desarrollo permite tomar como referencia proyectos con un desarrollo similar.

Desde los tiempos de la antigüedad, el ser humano siempre ha tenido la necesidad de contar con sistema contable que le permita llevar el orden de los procesos, especialmente en el ámbito económico, al comienzo se utilizaban recursos muy esenciales de acuerdo a la época. Los sistemas para administrar procesos nacieron con el propósito de que el ser humano pudiera, organizar, administrar, y registrar los productos, para poder conocer cuánto fueron las ganancias o pérdidas de sus actividades comerciales, durante muchos años el abaco, los folios, el lápiz y las libretas, fueron los acompañantes desde los inicios, para llevar los procesos contables, con el riesgo que todo ejercicio o actividad manual implicaba (Aguirre, 2018).

Al principio de las civilizaciones se establecían según sus habilidades aritméticas con métodos rudimentarias de sumas y restas, lo cual para la época era entendible por las herramientas que había a disposición y sobre todo el nivel de conocimiento del ser humano. El principal factor que promovió el desarrollo de un sistema contable, fue la creación de la moneda como instrumento, para generar valor a un intercambio de productos de bienes y servicios, fueron estas actividades las cuales fueron el centro de motivación para las personas pensar cómo se debería llevar sus procesos contables, ya que la memoria no era la mejor alternativa (Aguirre, 2018).

Con el auge de los intercambios comerciales en la edad antigua está en su punto más alto, se fue acreditando los intereses de contar las transacciones lo que en la actualidad se conoce como factura. Entre los años 5.400 a 3.200 A.C, nacen los primeros vestigios de una organización bancaria, en el templo de rojo de babilonia, se entregaba los depósitos y las ofrendas que se presentaban con intereses, y en Grecia se estipularon algunas leyes que obligaban a los comerciantes a que llevaran libretas en las que pudieran anotar sus operaciones. Todo esto fue motivado por la aparición de la escritura. Quizá fueron los romanos, con la creación de un **sistema contable** que constaban de dos libros, el Adversaria y el Codex, los que dieron ejemplo de implantar un orden en sus ejercicios comerciales, en los cuales anotaban sus ingresos y gastos**,** durante la edad media y la edad moderna

estas prácticas se mantuvieron de la misma manera (Aguirre, 2018).

Con los avances que presenta la tecnología fueron surgiendo en el mundo desde la primera mitrad del siglo XX. La contabilidad encontró un método fundamental para facilitar estas tareas manuales, el surgimiento de los sistemas de control de procesos, o un ordenador electrónico el cual fue realmente importante para complementar el trabajo manual, el cual ofrecía un panorama muy limitado para el control de procesos contables, anteriormente este gran avance no se evidenciaba como una necesidad dado al desconocimiento de la industria tecnológica, con el transcurso del tiempo esta necesidad fue tomando más fuerza con la evolución de la tecnológica, y softwares más completos para facilitarle la vida a los usuarios (Aguirre, 2018).

De acuerdo con Yuly Castañeda y Diego Vargas, quienes desarrollaron un sistema de control de inventarios para la compañía (Melaxa), estudiantes egresados de la universidad libre, resaltan que "para comprender el control y almacenamiento adecuado de los materiales es necesario comprender el comportamiento del inventario tanto como elemento tangible y físico manteniendo dentro de la instalación (Cuando se la hace un conteo y control clasificado como vida real o conteo de estante), elemento intangible que existe en los registros de una compañía (vida en el papel o conteo de registros)" . (Castañena Ramirez, 2013, pág. 31)

Según los autores Sandra Martínez y Sara Rocha, quienes implementaron de un sistema de inventario en la ferretería Benjumea en el municipio de córdoba, los cuales hacen un enfoque en las bases teóricas relacionados con el tema del trabajo de investigación, "cabe destacar que desde la antigüedad las sociedades han implementado el resguardo de alimentos y suministros que le permitieran a los ancestros sobrevivir en tiempos de carencia alimentaria, es aquí donde nace la importancia de la implementación de los inventarios que reduzcan la problemática de la escasez o desabastecimiento, asegurando la subsistencia de vida, el desarrollo de las actividades cotidianas y por ende las operaciones económicas de la empresa". (Martinez Sandra, 2019, pág. 21)

Inventario:

Hace referencia a la relación detallada, ordenada y valorada de aquellos elementos que componen el patrimonio de una compañía o persona en un momento determinado, mas espeficamente es el registro de los bienes resguardados en una empresa. (Siigo, 2018).

Control de Inventario:

El control de los inventarios es de vital importancia ya que permite conocer la existencia real de los productos en su totalidad, el objetivo central del control del inventario es recolectar información de las entradas y salidas de los productos, con el fin de obtener un ahorro en los costos, en base a esto se puede llegar a la toma de decisiones, para encontrar estabilidad en los productos resguardados esto para que el producto este el menor tiempo posible inmovilizado.

Sistema de Inventario:

Un sistema de inventario es una herramienta la cual tiene como finalidad gestionar empleando registros y cantidades de mercancías existentes en una empresa, como también para determinar el costo de los productos que ya han sido vendidos, gracias a un sistema de control de inventarios podemos saber con exactitud cuanta mercancía se tiene a disposición en el instante y que producto se está agotando. La funcionalidad en los sistemas de inventarios es diferente depende del software, es decir, lo inicial que nos permite hacer un sistema de control de inventarios es acceder a la base de datos con el fin de registrar las cantidades, los costos y determinar el precio de venta de cada producto que componen el stock de una compañía, se debe tener presente la actualización de los inventarios en tiempo real precisamente su funcionalidad es determinar los registros de venta que se encuentra asociado al inventario, por ende cada vez que un producto sea vendido el sistema automáticamente lo debe sacar del stock.

Esneider Romero resalta que, a nivel empresarial y comercial, es importante tener un software para controlar los procesos, las entradas y salidas de los productos, las cuales se almacenan en una base de datos y son la lista de los bienes de la entidad mediante el cual se usa para desarrollar su actividad económica. Mantener actualizado este inventario es importante para tener claridad sobre qué elementos o artículos tiene disponible la empresa, las empresas que se dedican a la

comercialización de artículos o productos deben tener un inventario lo más preciso posible. De acuerdo a las políticas de inventario que manejan, deberán preparar el flujo de caja para la compra de más productos o materias primas, supliendo las existencias de inventario según sea las necesidades del mercado o la estrategia comercial de la fuerza de venta. (Esneider, 2019).

Al llevarse el control de los inventarios de manera física, conlleva que a determinado tiempo se vean las consecuencias debido a que va ser más complejo llevar el control de los productos de manera eficiente y generando buen ambiente financiero para la compañía, es por esto que los sistemas de inventarios están para facilitarle la vida al usuario dejando atrás esos métodos convencionales de llevar el control de sus productos.

Es importante destacar que cuando no se lleva el inventario de manera organizada la empresa o compañía, presentan inconvenientes en sus actividades comerciales, debido a que al no tener mercancía suficiente se pueden generar pérdidas para la empresa, al llegar al no poder cubrir la demanda actual en el mercado. (Esneider, 2019).

Según Paola Milena Cantor y Pilar Andrea Torres Pulido Los inventarios son necesarios para dar un buen servicio al cliente, para hacer funcionar la planta más eficientemente, manteniendo la producción en cuotas más uniformes y mantener lotes de producción razonablemente grandes. No obstante, mientras cierta inversión en inventarios es necesaria, mucha inversión en inventarios es perjudicial. Muchas empresas cuentan con recursos limitados y por ello el dinero invertido en inventarios podría ser útil para dar mantenimiento a la planta, para el desarrollo de nuevos productos. (Cantor Milena, 2013).

Estado del arte

La gestión de los inventarios ha evolucionado con el paso del tiempo esto se remota a los tiempos de los egipcios, los pueblos de la antigüedad almacenaban alimentos, para poder subsistir en tiempos de crisis, de esta manera surge la necesidad de crear los inventarios la cual era una forma de enfrentar los escases que se vivía en esa época, asegurando la permanencia del negocio y el desarrollo de sus actividades. De esta manera, nace la importancia de los inventarios con el propósito de asegurar el buen manejo y abastecimiento de los productos, debido al gran volumen de alimentos en ese entonces, el cual conlleva al paso del tiempo a lo que hoy en día llamamos

como inventarios, debido al costo que presentaban, con el fin de ser administrados de manera correcta, dando un orden e implementando mecanismos, simples, pero que con el transcurso del tiempo se han hecho más sofisticados de tal medida que fueron aumentando de gran demanda en cantidad y variedad los inventarios.

En la administración de los inventarios existen varias etapas fundamentales en este proceso las cuales son la planeación, Organización, Dirección, y control.

Planeación: En este elemento consiste en determinar los objetivos y estrategias, en este proceso administrativo se atribuye a la toma de decisiones, con el fin de seleccionar la mejor estrategia o alternativa, determinando el paso a seguir, fijando unos principios con el propósito de orientar.

Organización: En este proceso se establece la estructura técnica con el fin de determinar y enumerar las actividades necesarias para poder cumplir los objetivos generales de una compañía.

Dirección: En este proceso se pone en acción, para impulsar guiar y coordinar esfuerzos establecidos por los miembros de una organización.

Control: Este proceso hace referencia al establecimiento de sistemas con el fin de llegar a medir los resultados proyectados y evaluar si se han cumplido los objetivos.

Para Ricardo Rivera Cuando se habla de seguimiento y control de inventarios no es simplemente un tema de comprobación de valores, se trata de algo mucho más amplio, que contempla las acciones que se deben realizar/ejecutar para garantizar la gestión eficaz de un almacén. También se considera la retroalimentación confiable de datos para lograr una buena gestión por parte de la organización, involucrando desde el mismo control de inventarios hasta las compras, trámite de pedidos y facturación. (Rivera, 2012, pág. 33).

Diseño e implementación de un software de registro y control de inventarios

Este Software contribuyo a la implementación de un sistema de inventarios para el respectivo control y registro de mercancías, la cual adoptaron como principal herramienta al área administrativa las cuales se adaptaron de acuerdo a la necesidad que posee el área operativa, por ende, mediante el uso del software, permite analizar y controlar de manera adecuada las cantidades

exactas de mercancía, con el fin de contribuir al mejoramiento y dinamismo de las operaciones de la empresa, y tener información clara de los productos del supermercado en la ciudad de Huila. (Suarez, 2012, pág. 9).

Para el desarrollo del sistema de información utilizaron la metodología espiral la cual se basa, y tiene como regla fundamental poder retroceder durante el desarrollo, en cualquier punto del proceso, como es el caso de las etapas de pruebas. Esta metodología se adapta si el proyecto en desarrollo es grande y posteriormente si los requisitos son muy poco claros y complejos. (Suarez, 2012).

Figura 2. Supermercado centro sur. (2012).

Sistema de control de compra, venta e inventarios para la empresa Protec

Mónica Mendoza autora del Sistema de control de compra, venta e inventarios en el año 2017 en la paz Bolivia.

Este proyecto se desarrolló a partir de la necesidad de la empresa Protec de sistematizar la compra y venta de sus productos, esta empresa comercializa, tubos de oxígeno, reguladores medicinales y entre otros materiales de seguridad industrial. Anteriormente esta compañía llevaba la compra y venta de sus productos de forma manual la cual en su momento resultaba insuficiente, y podría a ver ocasionado pérdidas por una mala administración, ante las necesidades de la empresa, haciendo uso de la metodología de desarrollo ágil XP en las diferentes fases dentro del desarrollo, (Mendoza, 2017, pág. 2).

Implementación de un sistema de inventarios para el área de soporte técnico en la empresa comercializadora Arturo Calle S.A.S

David Latorre autor del desarrollo e implementación de un sistema de inventarios para el área de soporte técnico para la compañía Arturo Calle en el año 2017 en la ciudad de Bogotá.

Para este desarrollo se basó en la necesidad de la empresa Arturo calle para el área de soporte técnico de la misma, la implementación de este sistema dio como solución a problemas evidentes que se venían presentando con el transcurso del tiempo, la cual la manera de como llevaban el registro de los dispositivos tecnológicos de la empresa, contribuían a llevar información errónea, perdida de datos, falta de control que afectaban los procesos en la gestión de incidencias en el área de soporte técnico. Para la ejecución de este proyecto se realizó con base a una serie de lineamientos de la metodología de software Extreme Programming o programación extrema XP, y fue desarrollado en el ciclo de cuatro fases que presenta esta metodología (exploración, planificación, iteración y puesta en producción) (Pelaez, 2017, pág. 14).

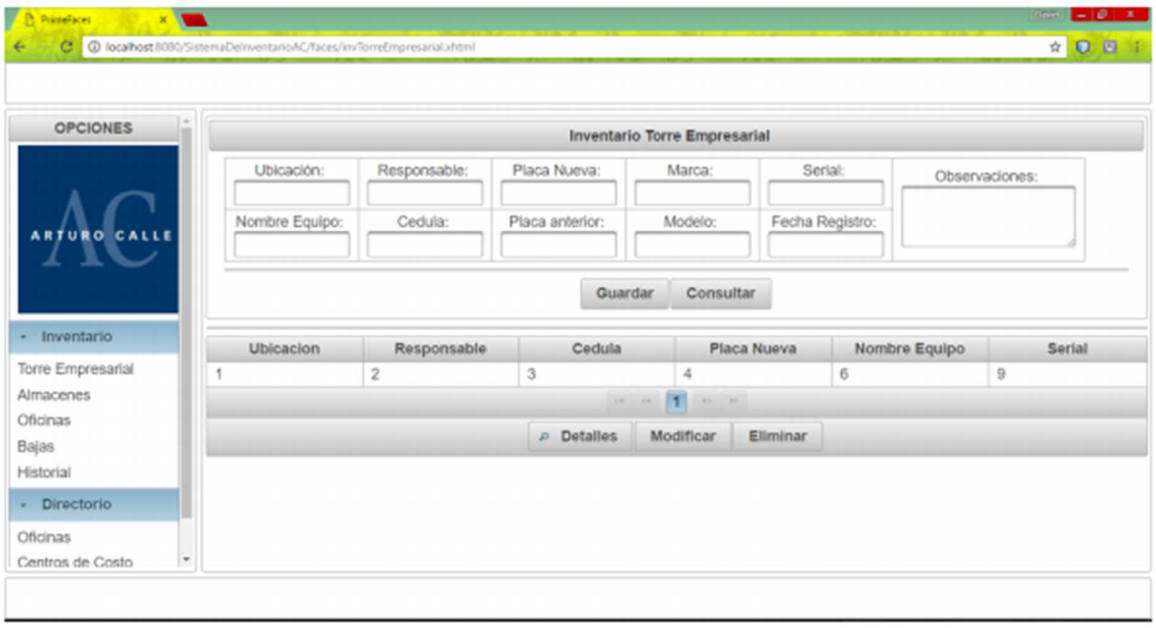


Figura 3. Inventario torre empresarial (2017).

Desarrollo de un sistema de puntos de ventas para micro mercados Adrián

Andrés Guzmán implemento un desarrollo de un sistema de ventas para micro mercados Adrián en el año 2008.

La construcción de este sistema de ventas, se desarrolló para dar solución a los mercados adrián, para llevar los procesos de venta de los productos de manera adecuada, ya que anteriormente no contaban con un sistema de puntos de venta y las tareas se hacían manualmente desde el inventario hasta el proceso de facturación. En la actualidad las empresas buscan los recursos que les permitan obtener ventas competitivas y seguir creciendo en este campo (Avila, 2008, págs. 5,6).

Para llevar a cabo este software se utilizaron las mejores prácticas de la metodología ágil XP (Extreme Programming) siendo esta una de las metodologías más seleccionadas por jefes de proyectos, ya que esta metodología permite un desarrollo rápido para pequeños y medianos proyectos, no se recomienda la utilización de esta metodología para proyectos de larga duración (Avila, 2008, pág. 5).

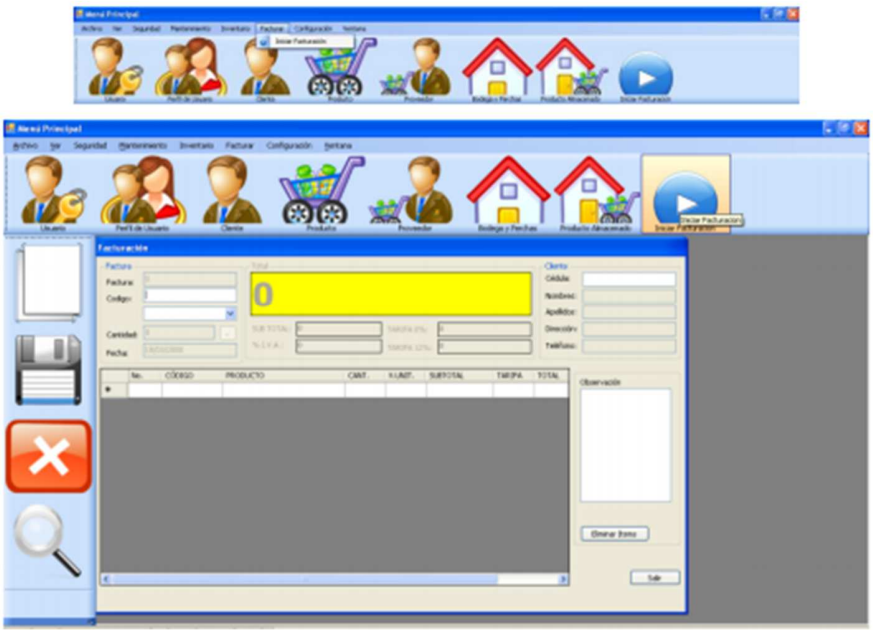


Figura 4. Mercados Adrián (2008).

Desarrollo de un sistema de control de inventario para pymes comercializadoras aplicando la metodología personalizada de XP

Saúl Apaza y Johnny Peralta quienes contribuyeron a realizar un sistema de control de inventario para pymes comercializadoras para la Universidad Peruana unión en el año 2020.

Este software se realizó con el fin de ayudar las pymes pequeñas o medianas empresas a llevar el respectivo control de inventarios y registros del área comercial, esta necesidad de implementar un sistema de inventario nace a partir de llevar los procesos manuales que retrasan las tareas de cualquier pyme comercializadora de productos. Para llevar a cabo este desarrollo se basaron en la metodología ágil Extreme Programing (XP) para el análisis y diseño, así como la revisión y evaluación de otros documentos para recabar información (Apaza, 2020).

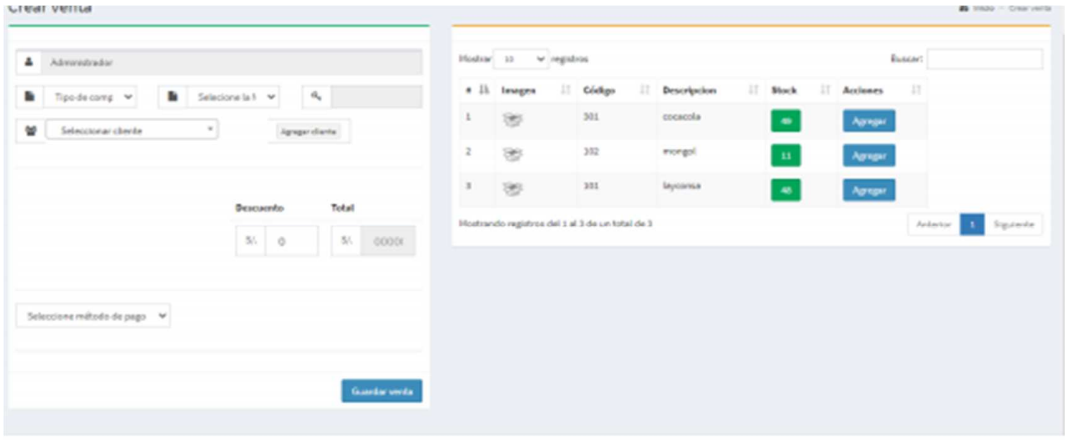


Figura 5 Inventario para pymes comercializadoras.(2020).

4. Especificación de Requisitos de Software (IEEE 830)

4.1. Perspectiva del Producto

Se proyecta implementar un software de escritorio para el control de los inventarios en la papelería vargos de la ciudad de Bogotá esto con el propósito de llevar de manera organizada y controlada todos los registros de sus productos, con un apartado para poder categorizar los productos, para llevar el seguimiento necesario de los inventarios en tiempo real. El cual por medio de un CRUD podemos actualizar y eliminar datos. Para que sea de fácil acceso este software se va permitir ver desde la versión de Windows XP hasta Windows 10 pro.

4.2. Funcionalidad del Producto

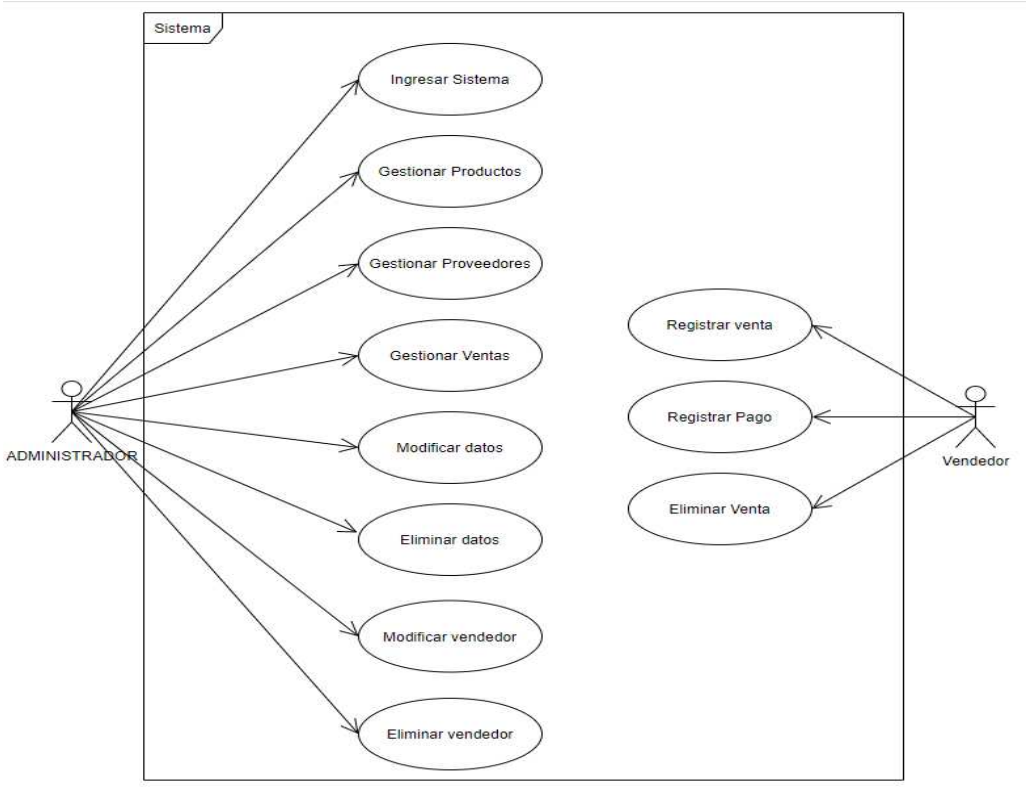


Figura 6. Funcionalidad del producto (propia).

4.3. Características de los Usuarios

Este software de control de inventarios tendrá dos actores, uno de ellos será el administrador el cual tiene la función y la capacidad de modificar tanto los productos del inventario, proveedores y las ventas podrá eliminar editar o actualizar los datos de la base de datos y velar que el sistema esté funcionando correctamente, el segundo actor será el vendedor el cual solo podrá tener acceso al módulo de ventas y realizar los procesos de la misma tales como registrar ventas, borrar los artículos que el cliente ya no desee llevar o en su defecto eliminar la venta de la base de datos.

TABLA 1.
Tipo de usuario Administrador

Tipo de usuario	Administrador
Actividades	Crear, editar y eliminar stock de la base de datos, velar que el sistema funcione correctamente.

TABLA 2.
Tipo de usuario Vendedor

Tipo de usuario	Vendedor
Actividades	Registrar las ventas en el sistema, como registrar el pago o en su defecto cancelar la venta.

4.4. Restricciones

Las restricciones principales para la aplicación de escritorio del sistema de control de inventarios para esta papelería son las siguientes que se mencionan a continuación:

- ✓ La aplicación debe tener una conexión de internet estable.
- ✓ El sistema debe permitir la iteración con bases de datos de MongoDB.
- ✓ La aplicación es exclusivamente Desktop para Windows.
- ✓ La aplicación debe ser de uso muy práctico para el administrador
- ✓ En la aplicación la interface de usuario debe ser intuitiva y de entorno amigable para el usuario en este caso para el administrador.
- ✓ Para ingresar al sistema se deben validar la credencial de aseso.

4.5. Suposiciones y Dependencias

El presente Software está diseñado con el propósito que funcione en un ordenador, esto con unas características específicas las cuales se debe contar con un sistema operativo como la versión de Windows XP, hasta versiones superiores de Windows 8.

4.6. Requisitos Específicos

4.6.1. Actores/Roles

El administrador es aquel que mantiene el sistema y asegura el respectivo funcionamiento del mismo.

Es aquel que colabora con las ventas en el sistema

4.6.2. Requisitos Funcionales

- ✓ El sistema debe permitir iniciar sesión.
- ✓ La aplicación deberá permitir buscar la información en la base de datos el nombre de determinado producto y mostrarlo en pantalla.
- ✓ La aplicación deberá permitir en la base de datos añadir un nuevo producto o si es el caso eliminarlo de la base de datos.
- ✓ El sistema debe restar productos del Stock al realizarse una venta.
- ✓ El sistema debe permitir registrar productos.
- ✓ El sistema debe permitir editar los productos.

- ✓ El sistema debe permitir eliminar los productos.
- ✓ El sistema deberá tener un buscador el cual tendrá como fin ayudar al administrador a encontrar la información de un producto más rápido.
- ✓ El sistema deberá permitir dar acceso al vendedor, únicamente para registrar ventas no tiene más permisos.

4.6.3. Diagrama de casos de uso

Administrador

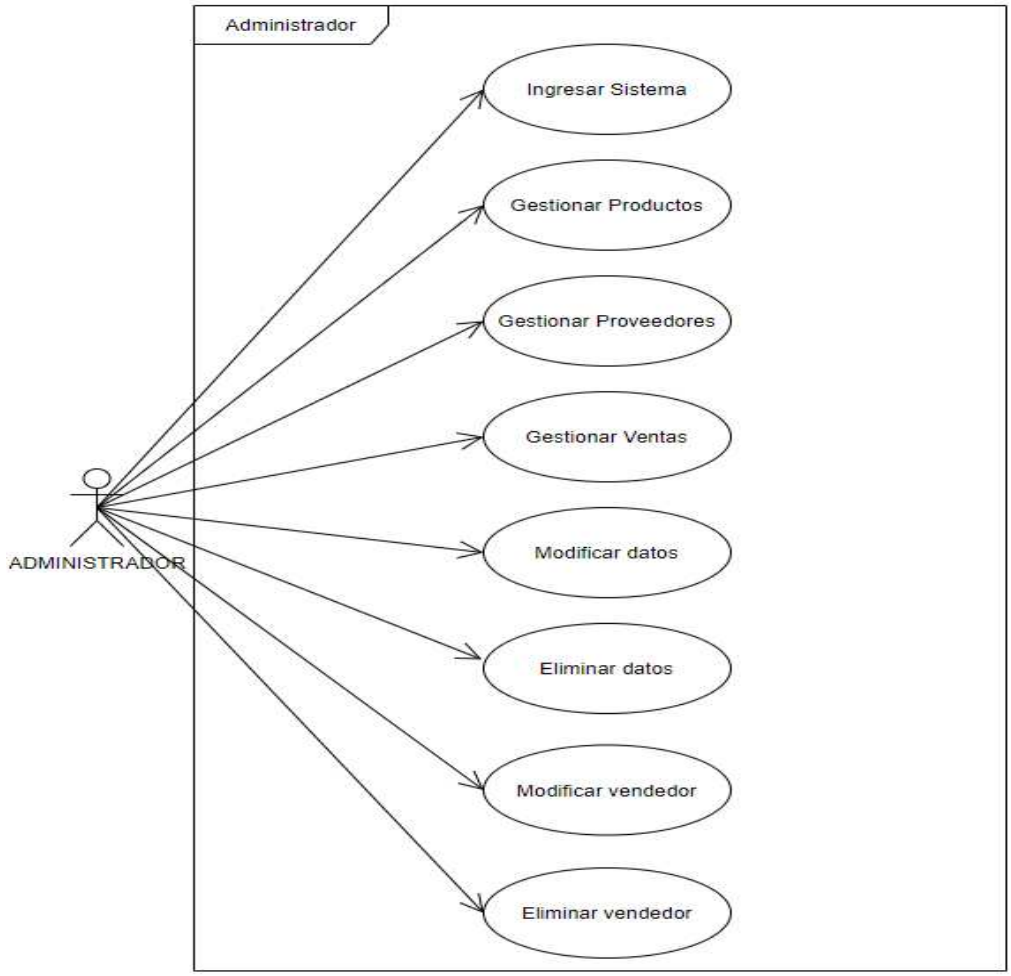


Figura 7. Diagrama Casos de Uso - Administrador (propia).

Vendedor

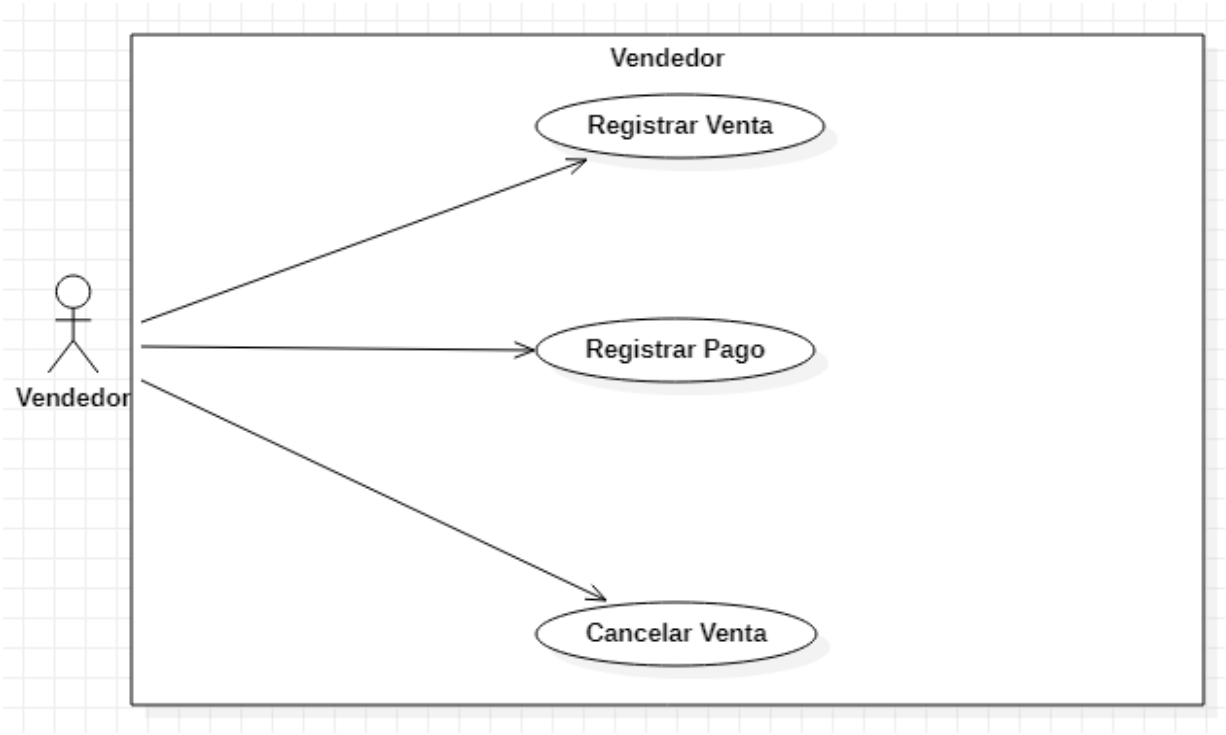


Figura 8. Diagrama Casos de Uso - Vendedor (propia).

4.6.4. Especificaciones de Casos de uso

Tabla 3. Especificación C.U Administrador- acceso al sistema: RF01

Identificación del requisito:	RF01
Nombre del Requisito:	Acceso al sistema
Características:	El administrador deberá tener un identificación de usuario y una contraseña para poder acceder al sistema de inventarios.
Descripción del requisito:	El sistema solo podrá ser consultado por el administrador.
Prioridad del requisito: Alta	

Tabla 4. Especificación C.U Administrador-sistema: RF02

Identificación del requisito:	RF02
Nombre del Requisito:	Apartados del sistema
Características:	El administrador podrá acceder a las funciones del sistema.
Descripción del requisito:	El sistema mostrara seis pestañas
Prioridad del requisito: Alta	

Tabla 5. Especificación C.U Vendedor-sistema: RF03

Identificación del requisito:	RF03
Nombre del Requisito:	Login Vendedor
Características:	El vendedor también tendrá sus credenciales para poder ingresar al sistema, únicamente en el apartado de ventas.
Descripción del requisito:	Solo tendrá acceso al apartado de ventas.
Prioridad del requisito: Alta	

Tabla 6. Especificación C.U Inventario productos-sistema: RF04

Identificación del requisito:	RF04
Nombre del Requisito:	Sistema Inventario
Características:	En el módulo del inventario se almacenarán todos los registros de los productos, comercializados por esta papelería, con el fin de eliminar, editar o actualizar la información del stock.

Descripción del requisito:	Modulo inventario almacenamiento del stock.
Prioridad del requisito: Alta	

Tabla 7. Especificación C.U Usuarios-sistema: RF05

Identificación del requisito:	RF05
Nombre del Requisito:	Modulo Usuarios
Características:	El sistema almacenara la información de las credenciales de usuario donde el administrador podrá cambiar el usuario y la contraseña, como también asignar estas mismas credenciales al vendedor solo para el acceso a realizar ventas.
Descripción del requisito:	El sistema deberá almacenar y mostrar, la información de la tabla de usuarios.
Prioridad del requisito: Alta	

Tabla 8. Especificación C.U Proveedores-sistema: RF05

Identificación del requisito:	RF05
Nombre del Requisito:	Modulo Proveedores
Características:	El sistema almacenara la información de los proveedores.
Descripción del requisito:	El sistema deberá almacenar y mostrar, la información de la tabla de proveedores,

	teniendo así el usuario la información a la mano.
Prioridad del requisito: Alta	

Tabla 9. Especificación C.U Ventas-sistema: RF06

Identificación del requisito:	RF06
Nombre del Requisito:	Modulo ventas
Características:	El sistema podrá realizar ventas de los productos, con la información almacenada en el inventario, cada vez que se realice una venta, se resta el producto en el inventario.
Descripción del requisito:	Se podrá realizar la venta o en su defecto cancelar algún producto de la lista que el usuario ya no dese llevar.
Prioridad del requisito: Alta	

4.6.5 Modelo de vistas

4.6.5.1 Vista logica

4.6.5.1.1 Diagrama de clase

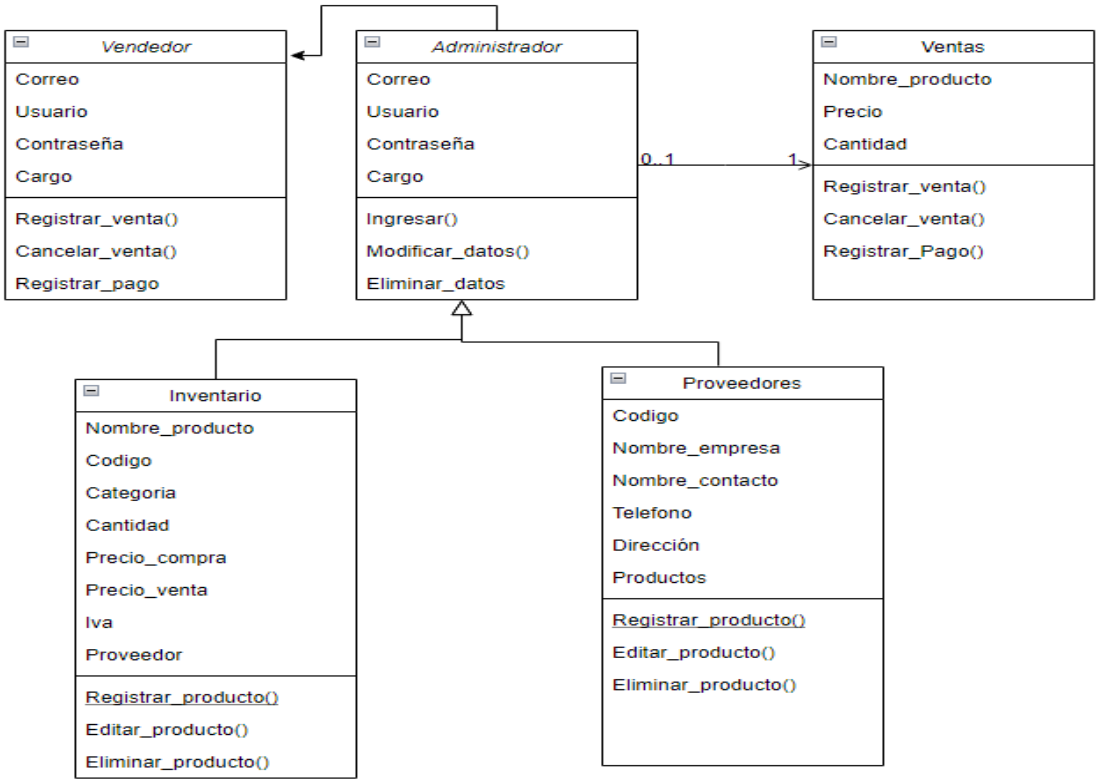


Figura 9. Diagrama de clase (propia).

4.6.5.1.2 Diagrama de comunicación

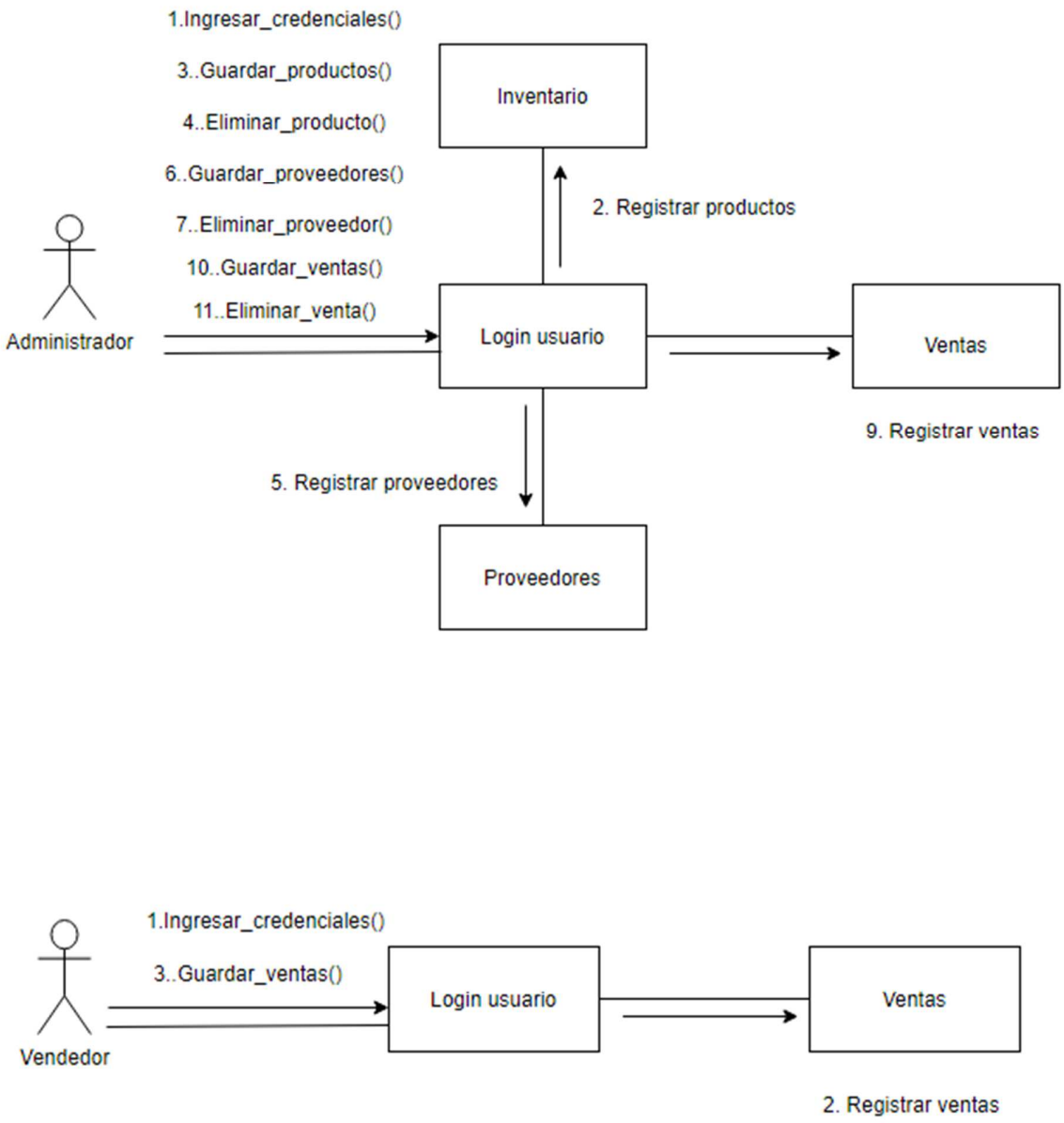
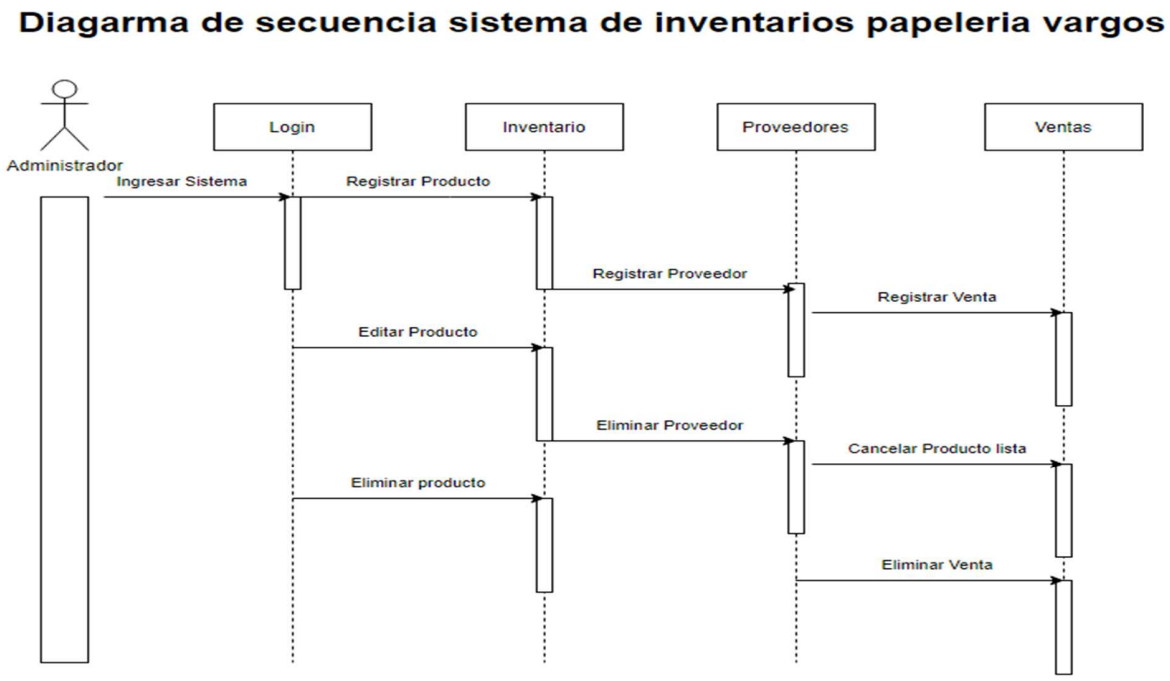


Figura 10. Diagrama de comunicación - (propia).

4.6.5.1.3 Diagrama de secuencia



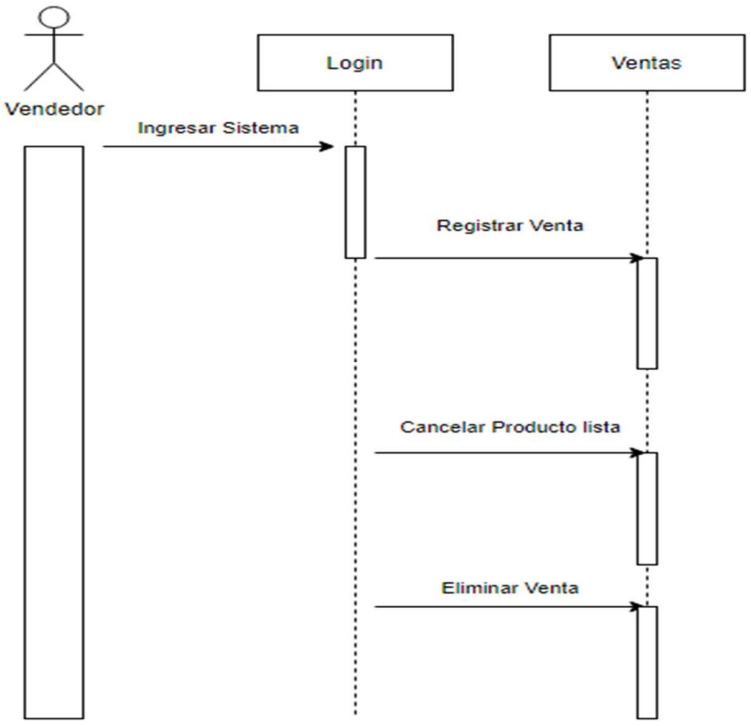


Figura 11. Diagrama de Secuencia - (propia).

4.7. Requisitos de Rendimiento

- ✓ La aplicación debe ser de escritorio para las versiones de Windows.
- ✓ El sistema debe permitir la interacción con bases de datos de MongoDB.
- ✓ El sistema debe tener una conexión muy estable de internet.
- ✓ Las interfaces deben cumplir con el propósito de ser amigables e intuitivas para el usuario.
- ✓ La aplicación debe ser de uso fácil.
- ✓ La aplicación tendrá que registrar y eliminar información de la base de datos.

4.8. Restricciones de Diseño

La papelería Vargos es una micro empresa encargada de comercializar productos para después ser distribuidos al público, desea un sistema de inventarios para su respectivo control y orden de las entradas y salidas de sus productos.

El desarrollo de este software parte de unas restricciones en el cual se lleva el proceso de diseño, a continuación, se mostrarán las restricciones relacionadas con el diseño.

- ✓ El lenguaje de programación que se utiliza para el desarrollo de la aplicación del sistema de inventarios es Electron que nos permite desarrollar aplicaciones para escritorio con JavaScript en acompañamiento de HTML y CSS.
- ✓ La aplicación tiene como requisito de que su navegabilidad debe ser de uso facil para los usuarios involucrados en este sistema.
- ✓ Las interfaces de esta aplicación de escritorio deden ser amigables e intuitivas para el usuario.

4.9. Atributos del Software del Sistema

- **Escalabilidad**

En el trayecto del desarrollo de este documento se ha mencionado en varias ocasiones que la aplicación ha sido desarrollada para el sistema operativo Windows XP hasta la versión más reciente de Windows.

-**Inicio Sesión**

El administrador después de haber iniciado sesión el usuario podrá ver las opciones que le muestra el software y navegar dentro de la misma.

-**Seguridad**

El administrador del software de inventarios vargos tiene la responsabilidad de velar por la seguridad de la aplicación, por lo cual puede añadir o eliminar datos en la base de datos en el sistema, es por esto que el administrador requiere de unas credenciales que en este caso son la identificación de usuario y una contraseña.

-**Permisos**

El administrador permitirá el acceso al vendedor restringiendo algunos apartados dentro del sistema es decir solo podrá acceder a la parte de ventas (Registrar venta, registrar pago y cancelar venta si es necesario).

5. Diseño del software (ISO -12207-1)

5.1. Diseño de la Arquitectura de software

Diagrama de componentes

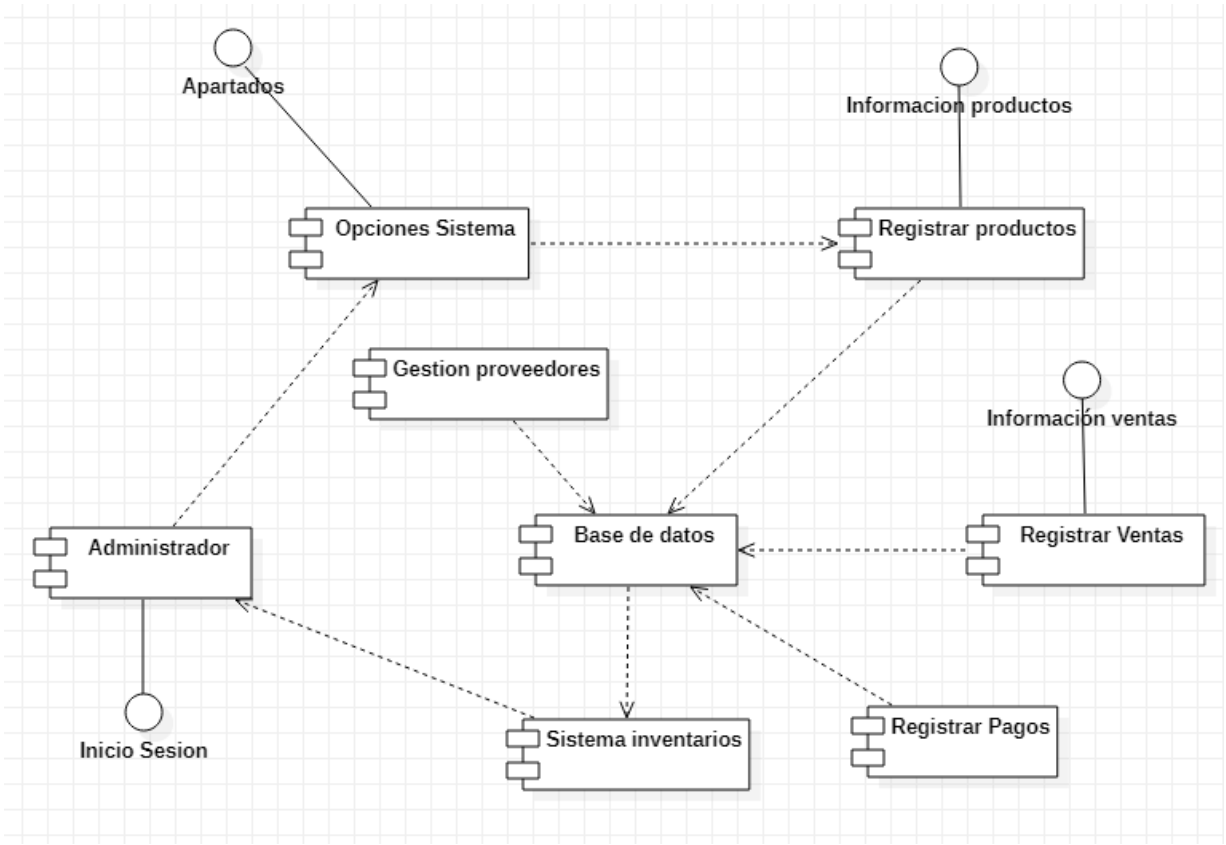


Figura 12. Diagrama de componentes - (propia).

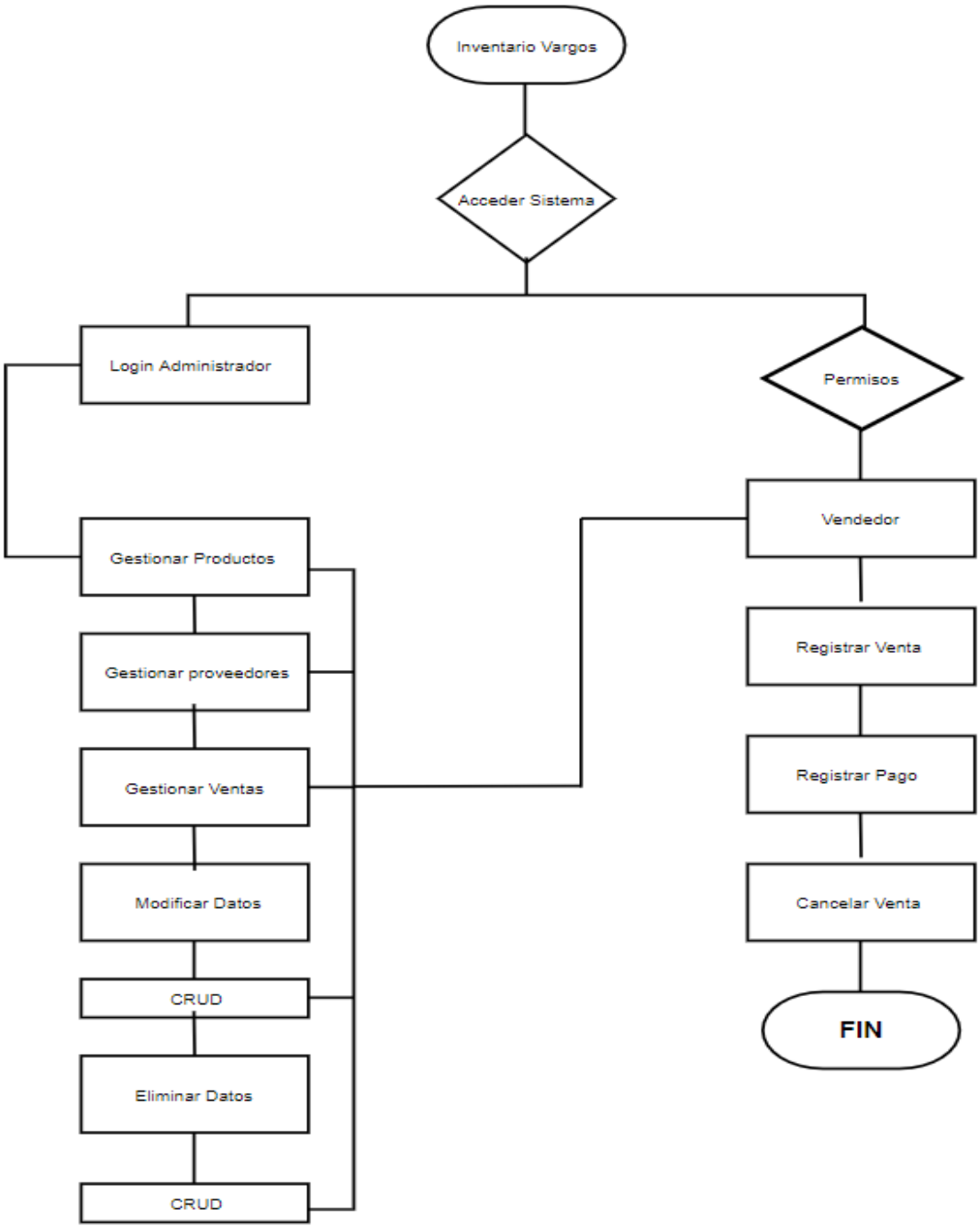


Figura 13. Diagrama de arquitectura inventarios vargos (propia).

5.2. Diseño detallado del software

5.2.1. Diagrama de clases

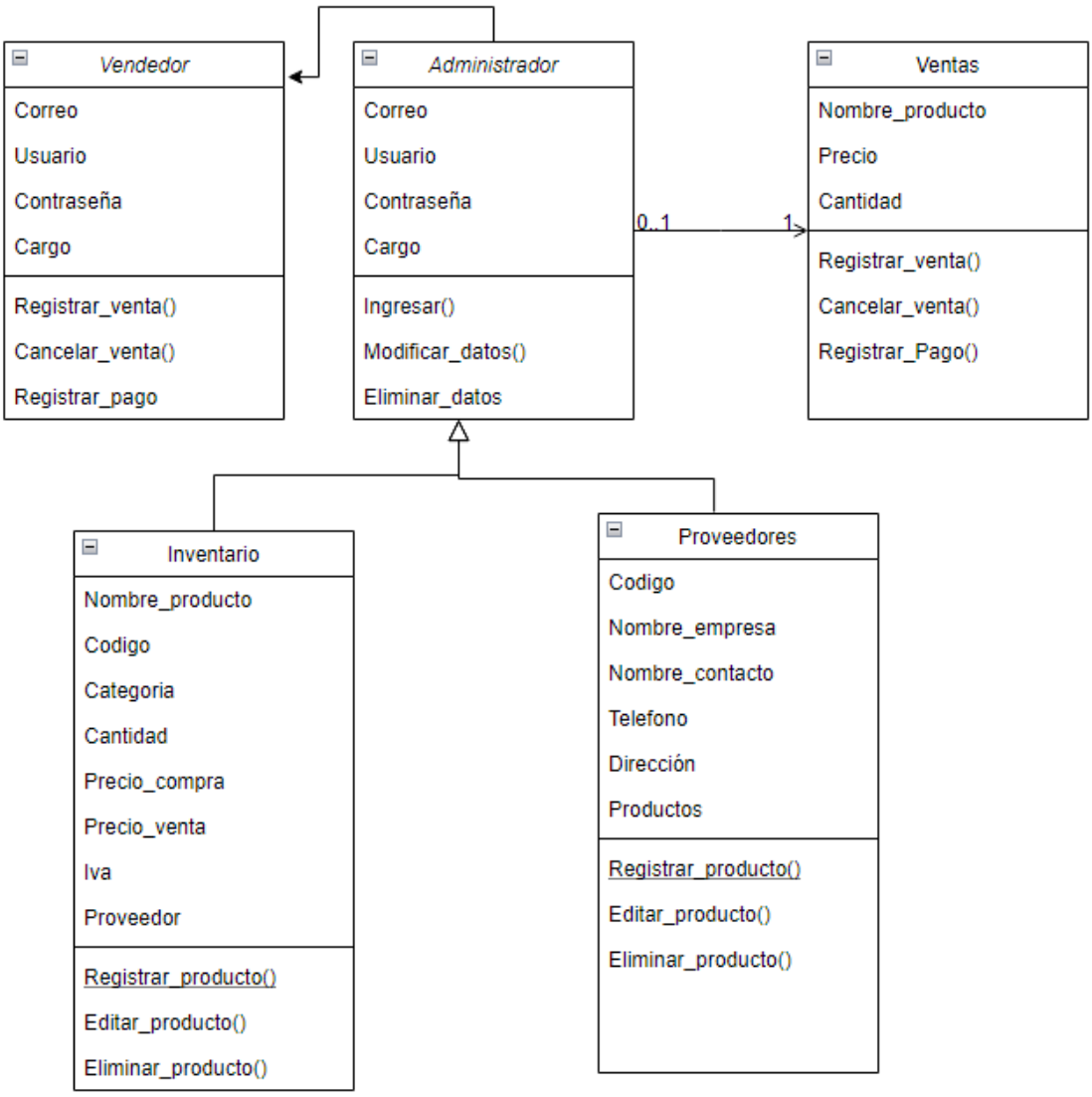


Figura 14. Diagrama de clases (propia).

5.2.2. Diagrama de paquetes

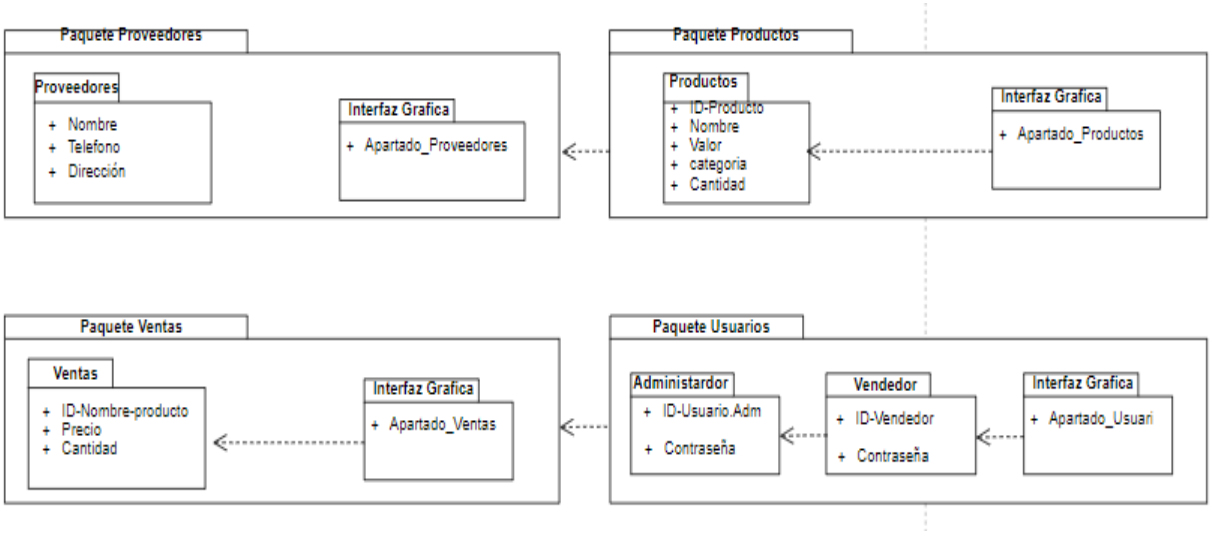


Figura 15. Diagrama de paquetes (propia).

5.2.3. Diagrama de despliegue

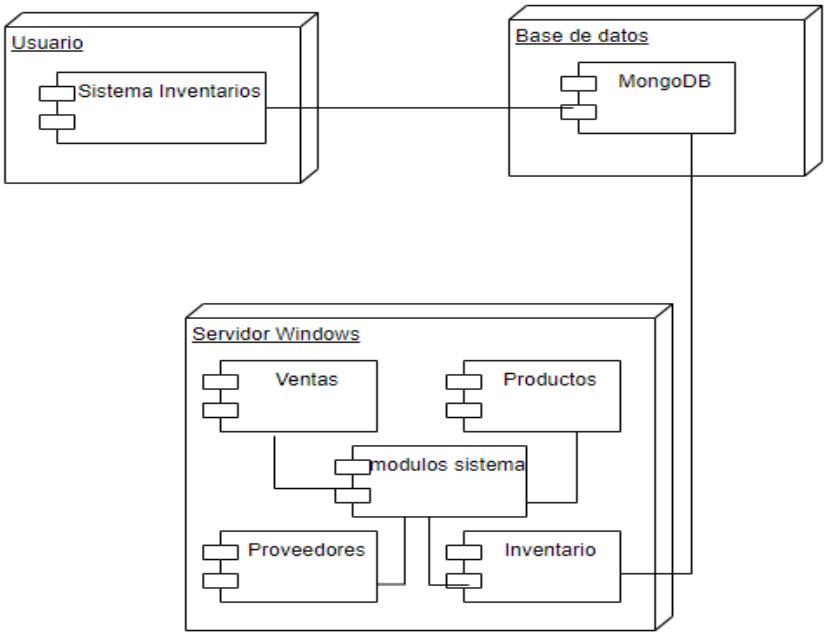


Figura 16. Diagrama de despliegue (propia).

5.2.3. Diagramas de despliegue

5.3. Diseño de la Interfaz

5.3.1. Interfaz Gráfica de Usuario

Figura 17. Interfaz – Login de usuario (propia).

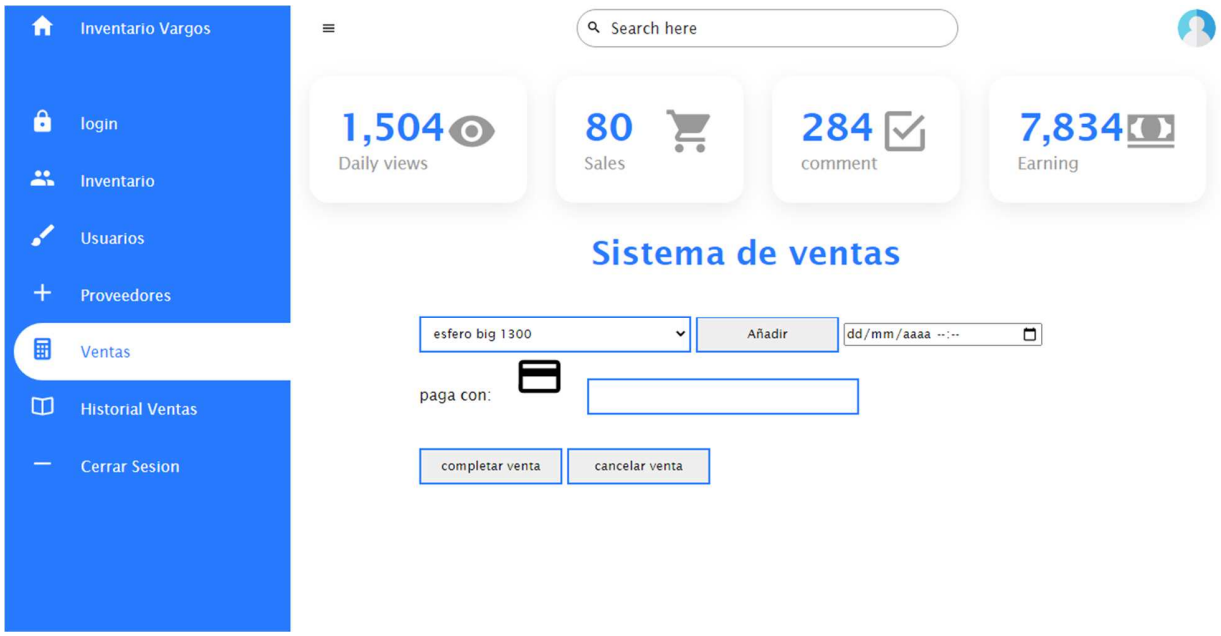


Figura 18. Interfaz – Apartado ventas (propia)

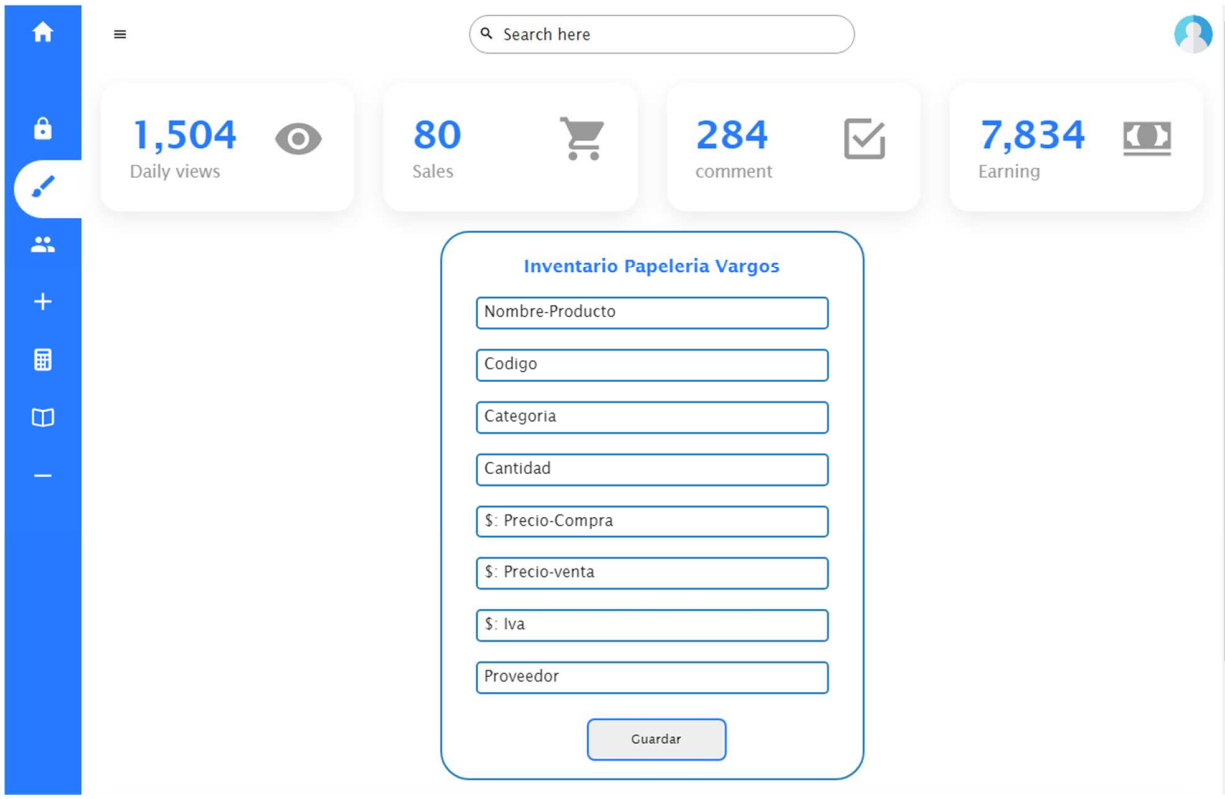


Figura 19. Interfaz – Apartado Productos (propia)

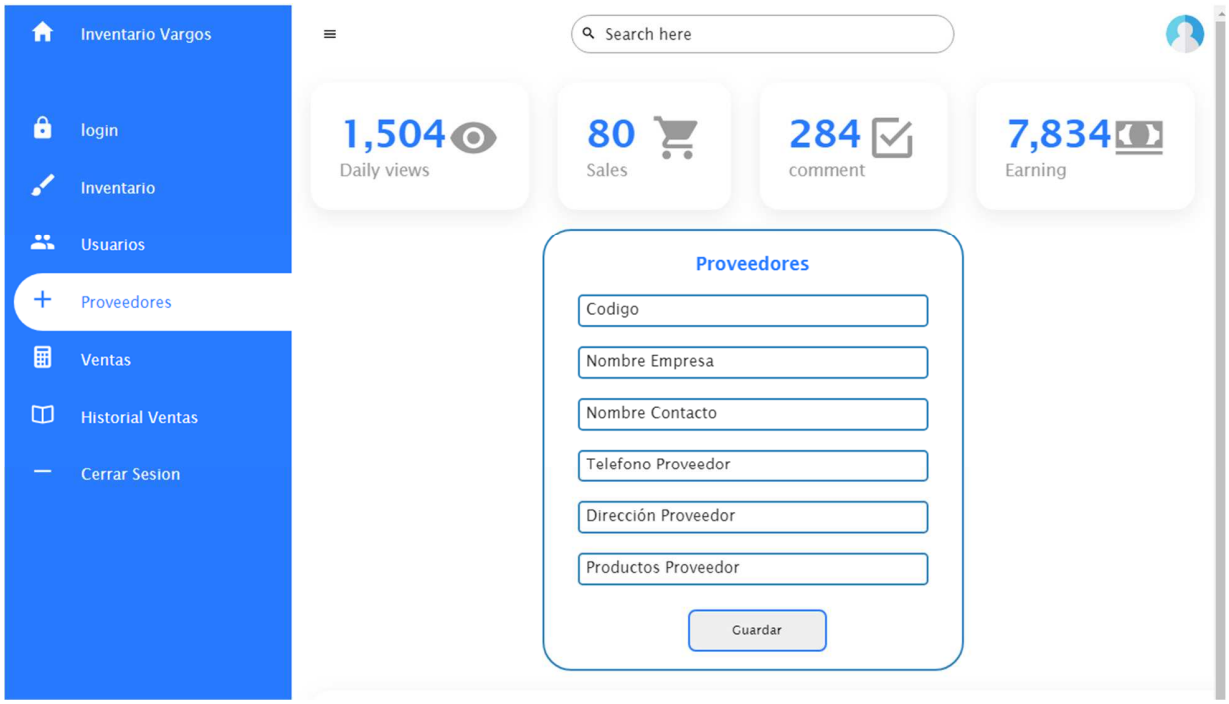


Figura 20. Interfaz – Apartado Proveedores (propia).

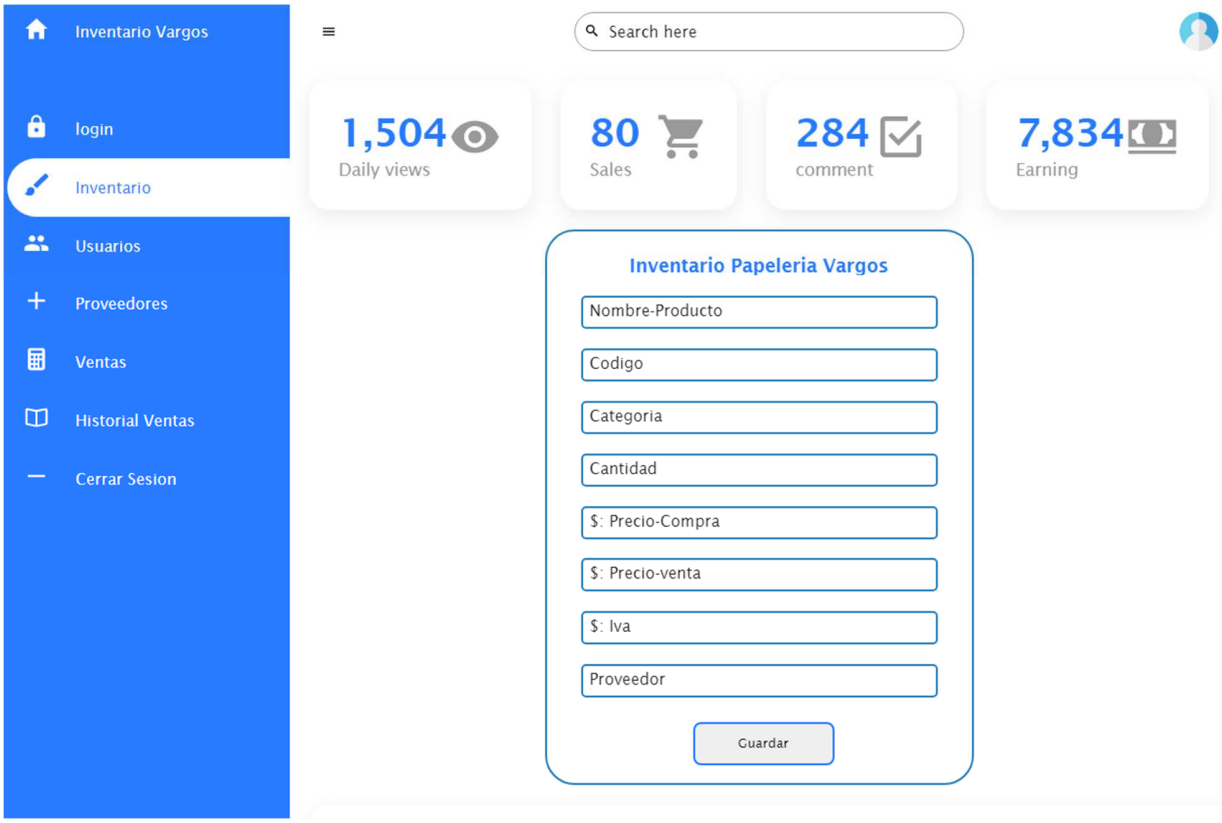


Figura 21. Interfaz – Apartado Inventario (propia).

5.3.2. Interfaces de Entrada

Las interfaces de entrada se reflejan en las imágenes las cuales en la mayoría del sistema nos permiten ingresar información a través de formularios.

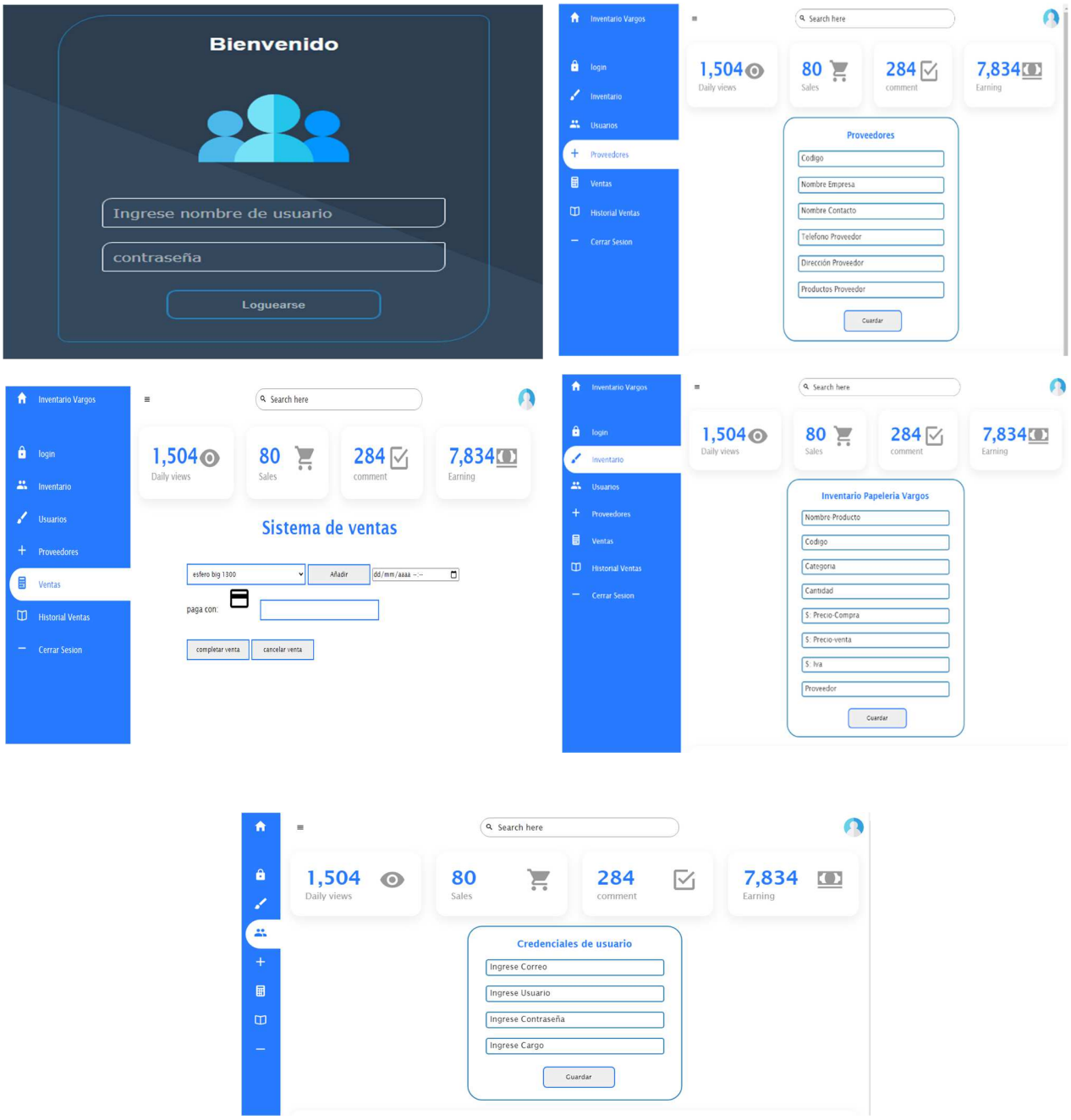


Figura 22. Interfaz de Entrada (propia).

5.3.3. Interfaces de Salida

Las interfaces de salida se encuentran en todas las imágenes de la interfaz del usuario ya que todas las páginas muestran datos de la base de datos. Los módulos muestran a través de tablas los datos almacenados en la base de datos.

Figura 23. Interfaz de salida productos-inventario (propia).

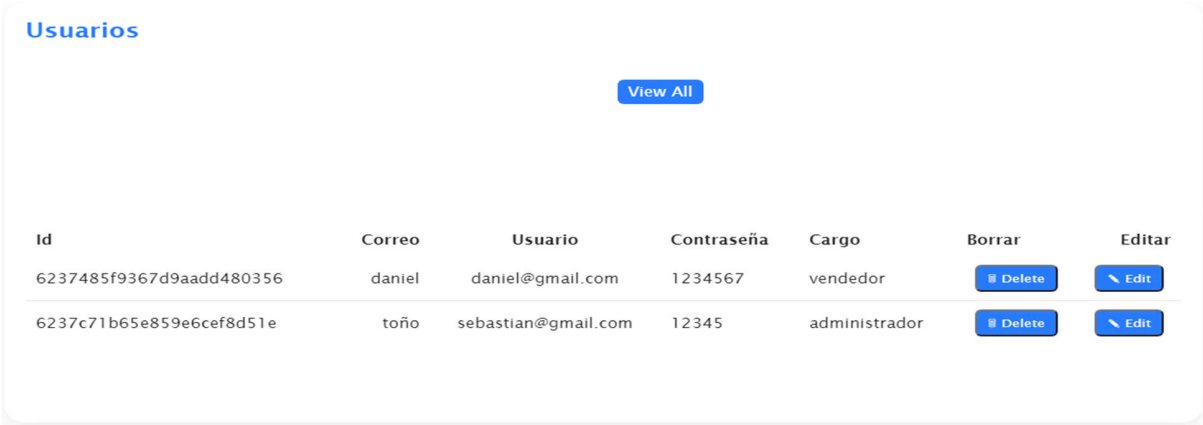


Figura 24. Interfaz de salida usuarios (propia).

Figura 25. Interfaz de salida proveedores (propia).

Figura 26. Interfaz de salida historial de ventas (propia).

6. Implementación

6.1. Plataformas de desarrollo

El programa fue desarrollado, a partir del framework de electrón el cual permite la construcción de aplicaciones multiplataforma con el fin de ejecutarse en cualquier sistema operativo ya sea macOs, Windows, y Linux, utilizando tecnologías tales como (html, css, y Javascript), en colaboración con Nodejs, el cual es una plataforma que está diseñada para la construcción de aplicaciones de red, que tiene Javascript como lenguaje destinado. Estos frameworks permiten la realización de programas de escritorio utilizando patrones que comúnmente se utilizan para la creación y construcción de aplicaciones web. Windows es el sistema operativo más utilizado y más común entre los servidores, esta aplicación tiene como gestor de base de datos MongoDB y Javascript como lenguaje de programación, por último, los programas empleados para el desarrollo del programa fueron vidual studio como editor de código.

En base al diseño, se utilizó el programa Illustrator CC, con el cual fueron desarrolladas las interfaces y los elementos gráficos que estan contenidos en este trabajo. Tambien es importante destacar la tipografía con la ayuda de google-fonts.

6.2. Base de datos

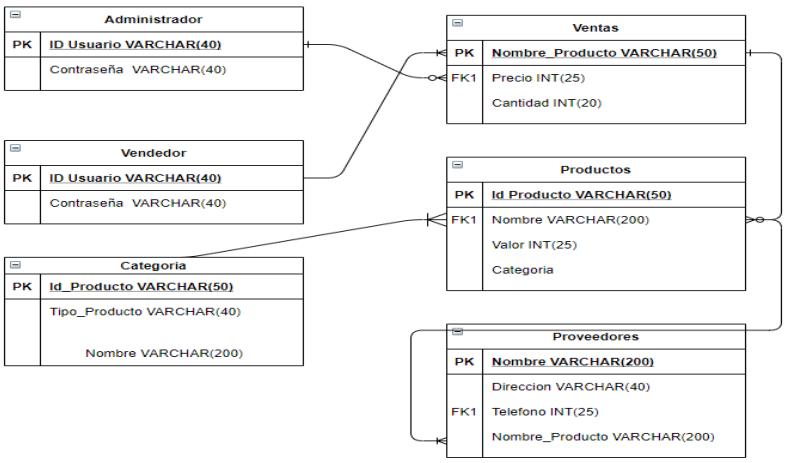


Figura 27. Base de datos (propia).

6.3 Infraestructura de hardware y redes

Esta aplicación de sistema de inventarios se desarrolla para la papelería Vargos en la ciudad de Bogotá, el cual se basa en el modelo Cliente/Servidor, se desarrolla bajo la utilización del lenguaje JS y utilizando un framework de JS llamado electrón, el cual nos permite, desarrollar aplicaciones de escritorio, mediante el uso de tecnologías web, electrón nos brinda una ventaja ya que el desarrollo de la aplicación se realiza utilizando HTML, CSS, JS, esto nos brinda una mayor escalabilidad y menor costo, esto requiere tener una conexión estable a internet para otorgar un buen funcionamiento de la aplicación.

Electrón js funciona a través de dos tipos de procesos en el primero de ellos encontramos el main y en segundo lugar encontramos el proceso renderer, El primer proceso es un proceso de Node Js el cual es el proceso principal, esto permite tener acceso a varias Api.

7. Pruebas del software

7.1 Pruebas del software

Tabla 10. Pruebas de software

Requisitos Funcionales	Cumple	Observaciones
El sistema debe permitir iniciar sesión.	**X**	
La aplicación deberá permitir buscar la información en la base de datos el nombre de determinado producto y mostrarlo en pantalla.		Falta implementar, pero se va realizar a futuro.

La aplicación deberá permitir en la base de datos añadir un nuevo producto o si es el caso eliminarlo de la base de datos.	**X**	
El sistema debe restar productos del Stock al realizarse una venta.	**X**	
El sistema debe permitir registrar productos.	**X**	
El sistema debe permitir editar los productos.	**X**	
El sistema debe permitir eliminar los productos.		
El sistema deberá tener un buscador el cual tendrá como fin ayudar al administrador a encontrar la información de un producto más rápido.		Implementación a futuro.

El sistema deberá permitir dar acceso al vendedor, únicamente para registrar ventas no tiene más permisos.	**X**	

Requisitos no funcionales	**Cumple**	**Observaciones**
La aplicación únicamente funcionara para ordenadores de escritorio.	**X**	
Los permisos de usuario serán únicamente manipulados por el administrador.	**X**	
El usuario solo podrá cerrar sesión si así lo requiere.	**X**	

Tabla 10. Pruebas de software

7.2 Pruebas de Usabilidad

Para llevar a cabo la prueba de usabilidad fue necesario contar con el administrador y el vendedor del negocio los cuales probaron el software para darnos a conocer su opinión al respecto por medio de este test de usabilidad

Administrador:

1. ¿Cree usted que como administrador y dueño de la papelería vargos utilizaría este sistema de control de inventarios frecuentemente?
2. ¿El sistema le resulto complejo para su navegabilidad?
3. ¿Cree usted que el sistema es bastante fácil de utilizar?
4. ¿Cree que necesitaría ayuda de un tercero para utilizar este sistema de control de inventarios?
5. ¿Cree que los módulos del sistema se encuentran bien integrados?
6. ¿En su opinión cree que hubo muchas inconsistencias en el sistema?
7. ¿Se sintió algo incómodo al utilizar este sistema?
8. ¿Se sintió seguro al utilizar este sistema?
9. ¿Cree que otras personas aprenderían a utilizar este sistema fácilmente?
10. ¿Cree que como administrador del sistema necesita aprender otras cosas antes de utilizar debidamente o correctamente el sistema?
11. ¿Cree usted que la interfaz de usuario es amigable e intuitiva?
12. ¿Qué tan probable seria que usted le recomiende este software a un familiar o un conocido que tenga un negocio en una escala de 1 a 10?

13. ¿Qué tan satisfecho esta con este software de control de inventarios en una escala de 1 a 10?

14. Cree que los colores que se utilizaron para la aplicación de escritorio son los adecuados.

Vendedor:

1. ¿Usted como vendedor de la papelería vargos y encargado del módulo de ventas de la papelería cree que es complejo su navegabilidad?

   Respuesta:

2. ¿Cree usted que dentro de este módulo es fácil la realización de las ventas de la papelería?

   Respuesta:

3. ¿Se siente incómodo al utilizar este módulo para la realización de las ventas?

   Respuesta:

4. ¿En una escala de 1 a 10 que tan complejo es realizar una venta?

   Respuesta:

5. ¿Cree usted que hay inconsistencias con este módulo para la realización de las ventas de la papelería vargos?

complejo para su navegabilidad

Conclusiones

- De acuerdo con los objetivos planteados al inicio del desarrollo se puede resaltar que se cumplen con las funcionalidades del producto.

- Se concluye que toda pyme pequeña o mediana empresa debe tener un sistema de control de inventarios que lleve de manera exacta y organizada el stock de sus productos, dando así una solución fácil y efectiva a los malos procesos administrativos dentro de cualquier organización.

- Contar con un sistema de control de inventarios en una compañía es importante para lograr alcanzar competitividad y sobre todo una mejor rentabilidad para el negocio.

- Con este desarrollo

- Con el desarrollo de la aplicación de escritorio se logró automatizar la papelería vargos con el propósito de mejorar las entradas y salidas de los productos que son comercializados por esta papelería.

Bibliografía

Aguirre. (2018). *Evolución del software para la implementación de un sistema contable*. Obtenido de https://loggro.com/blog/articulo/evolucion-del-software-para-la-implementacion-de-un-sistema-contable/

Apaza, P. (2020). *Desarrollo de un sistema de control de inventario para pymes comercializadoras aplicando la metodología personalizada de XP*.

Avila, G. (2008). *Desarrollo de un sistema de puntos de ventas para micro mercados Adrián.* Obtenido de http://repositorio.espe.edu.ec/bitstream/21000/848/1/T-ESPE-021852.pdf

Cantor Milena, T. P. (2013). *SISTEMA DE INFORMACION PARA EL CONTROL DE INVENTARIOSFACTURACION Y PROVEEDORES DE LA EMPRESA HIPERCERAMICA SANTA LUCIA*.

Castañena Ramirez, S. V. (2013). *IMPLEMENTACIÓN DE UN SISTEMA DE GESTIÓN DE INVENTARIOS EN MELEZA S.A.* Obtenido de https://repository.unilibre.edu.co/bitstream/handle/10901/9430/DOCUMENTO%20FINAL.pdf?sequence=1&isAllowed=y

Duran, Y. (1 de junio de 2012). *Administración del inventario: elemento clave para la optimización de las utilidades en las.* Obtenido de http://www.redalyc.org/articulo.oa?id=465545892008

Esneider, R. (Febrero de 2019). *Diseño e implementación de sistema de inventarios para el almacén de pinturas y ferretería ferrocolor.* Obtenido de https://repository.ucc.edu.co/bitstream/20.500.12494/8557/3/2019_Dise%C3%B1o_implementaci%C3%B3n_sistema.pdf

Gomez. (2018).

Gomez. (2018). *"SISTEMA WEB DE CONTROL DE INVENTARIOS, VENTAS DE PRODUCTOS Y MANTENIMIENTO DE MAQUINARIA.* Obtenido de https://repositorio.umsa.bo/bitstream/handle/123456789/17508/T-3426.pdf?sequence=1&isAllowed=y

Gomez. (2018). SISTEMA WEB DE CONTROL DE INVENTARIOS.

Martinez Sandra, R. S. (2019). *Implementación de un sistema de control de inventario en la empresa Ferretería Benjumea & Benjumea ubicada en el municipio de CereteCórdoba.* Obtenido de https://repository.ucc.edu.co/bitstream/20.500.12494/7593/1/2019_implementacion_sistema_control.pdf

Mendoza, S. (2017). *Sistema de control de compra, venta e inventarios para la empresa Protec.*

Pelaez, L. (2017). *IMPLEMENTACIÓN DE UN SISTEMA DE INVENTARIOS PARA EL ÁREA DE SOPORTE TECNICO.* Obtenido de https://repository.ucatolica.edu.co/bitstream/10983/14503/1/DocumentoTrabajoDeGrado.pdf

Rivera, R. (2012). *Mejoramiento de la gestion de inventarios en el almacen de repuestos de empresa andina de herramientas.*

Rodriguez, B. (Marzo de 2014). *Metodologia XP.* Obtenido de

https://luismejias21.files.wordpress.com/2018/03/metodologia-xp.pdf

sergioalbertoc. (Septiembre de 2015). *Programación eXtrema*.

Siigo. (2018). Obtenido de https://www.siigo.com/blog/contador/que-es-un-inventario/

Suarez, C. (2012). *Diseño e impelentación de un sistema de control de invenatrios.*

Vermorel, J. (Marzo de 2020). *LOKAD QUANTITATIVE SUPPLY CHAIN*. Obtenido de Analisis ABC (INVENTARIO): https://www.lokad.com/es/definicion-analisis-abc-(inventario)

Villarig, A. (17 de Julio de 2018). *UPAD*. Obtenido de https://www.upadpsicologiacoaching.com/teoria-inteligencias-multiples-howard-gardner/

Anexos

Modelo de encuesta software usabilidad:

¿Cree usted necesario que en la actualidad las pymes pequeñas o medianas, deben tener un software para controlar los procesos de su negocio o se puede optar por seguir utilizando herramientas como Excel o en su defecto el método manual convencional?

- Si
- No

¿Cree que es fácil la navegación del software de escritorio para el control de inventario?

- Si
- No

¿La interfaz gráfica de usuario es sencilla e intuitiva para el usuario?

- Si
- No

¿Cree usted que el software cubre las necesidades básicas de un sistema de control de inventario?

- Si
- No

¿Cree que las diferentes funciones del sistema se encuentran muy bien integradas?

- Si
- No

¿En ámbitos generales, los colores y el diseño de todo el recurso son adecuados?

- Si
- No

¿Los iconos empleados en la navegación del software ayudan aclarar los contenidos al usuario?

- Si
- No

¿Cómo calificaría el sistema de control de inventarios?

- Bueno
- Muy bueno
- Malo
- Muy malo
- Regular

¿Si tuviera un local comercial implementaría este sistema de inventarios básico para llevar un control de manera organizada de los productos?

- Si
- No

¿Usted como administrador de su negocio estaría satisfecho con este sistema de inventarios?

- Si
- No

Manual de usuario

Aplicación de escritorio para el control de inventarios para la papelería vargos en la ciudad de Bogotá

1. **Ingreso al sistema**

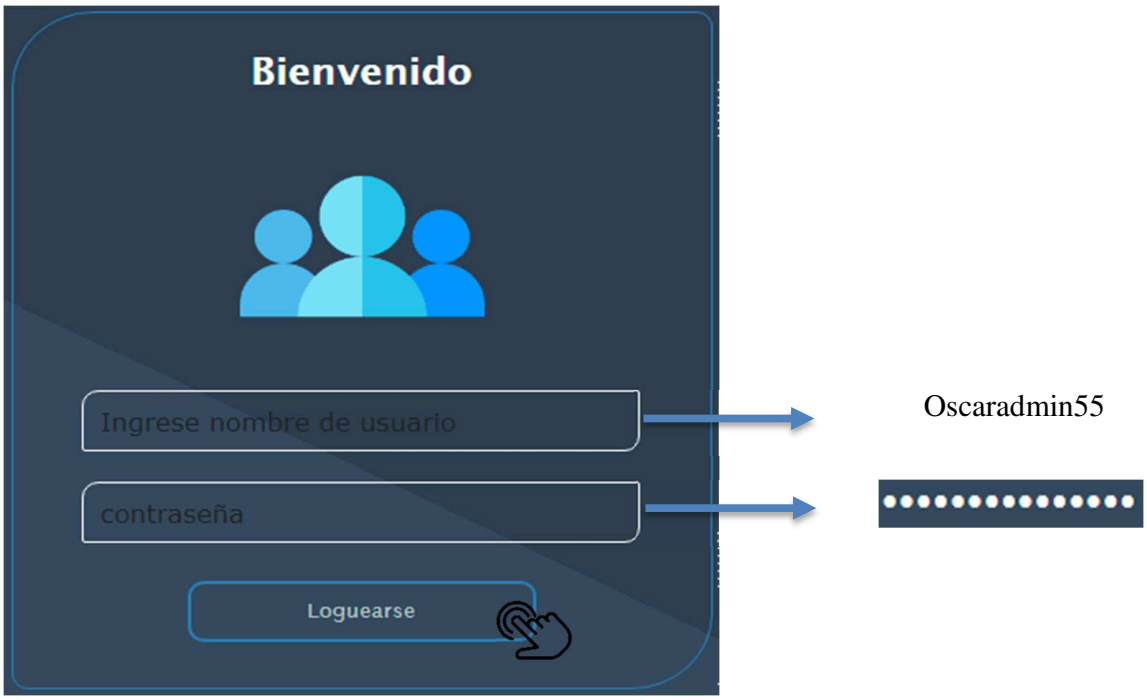
   Administrador: El administrador ingresa al sistema con el usuario y la contraseña que asigno para el ingreso de la aplicación y es el único el cual tiene acceso a todo el software de inventarios.

   Vendedor: El vendedor ingresa al sistema con el usuario y la contraseña que el administrador le asigno, y tendrá acceso únicamente para la realización de ventas.

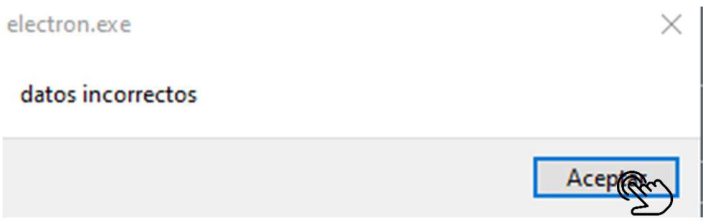
- Darle click en el espacio donde dice ingrese nombre de usuario el cual asigno previamente. (Vargosadmin55).

- Darle click en el espacio donde dice contraseña e ingrese su credencial de acceso al sistema.

- Darle click en el botón de loguearse e ingrese y disfrute de la aplicación.



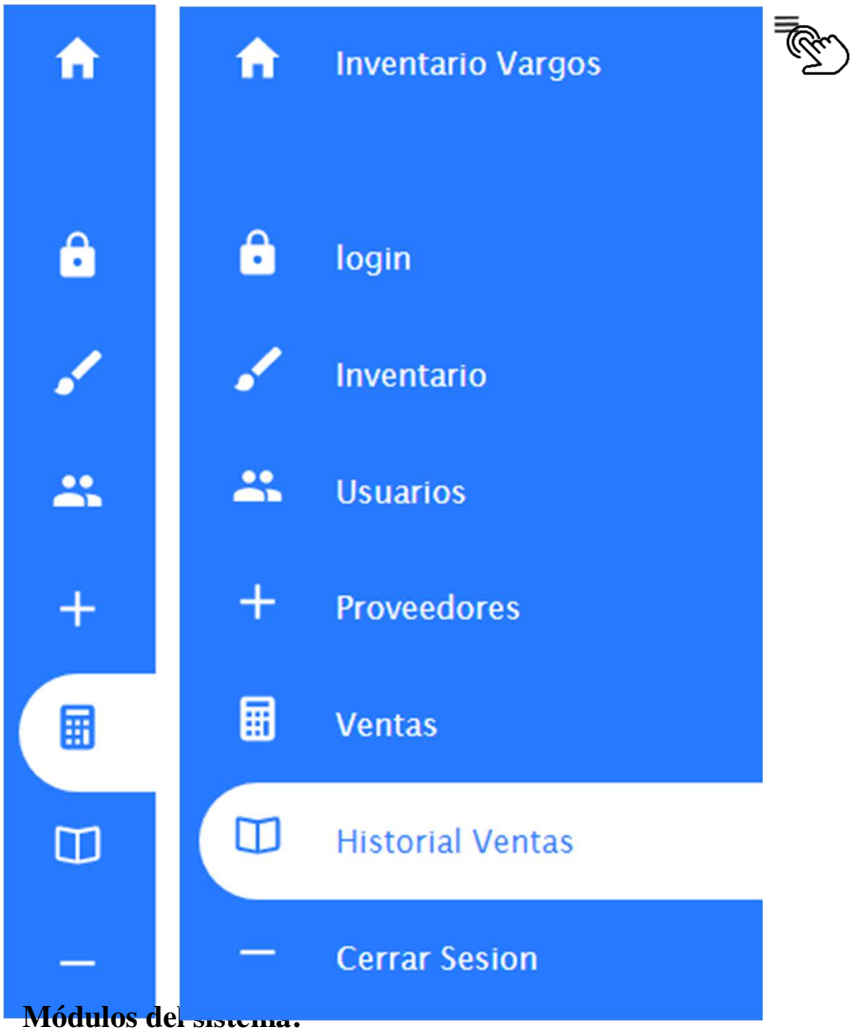
- Si los datos nombre de usuario y contraseña son incorrectos el sistema automáticamente mandara un mensaje de alerta.



2. Menú de navegación:

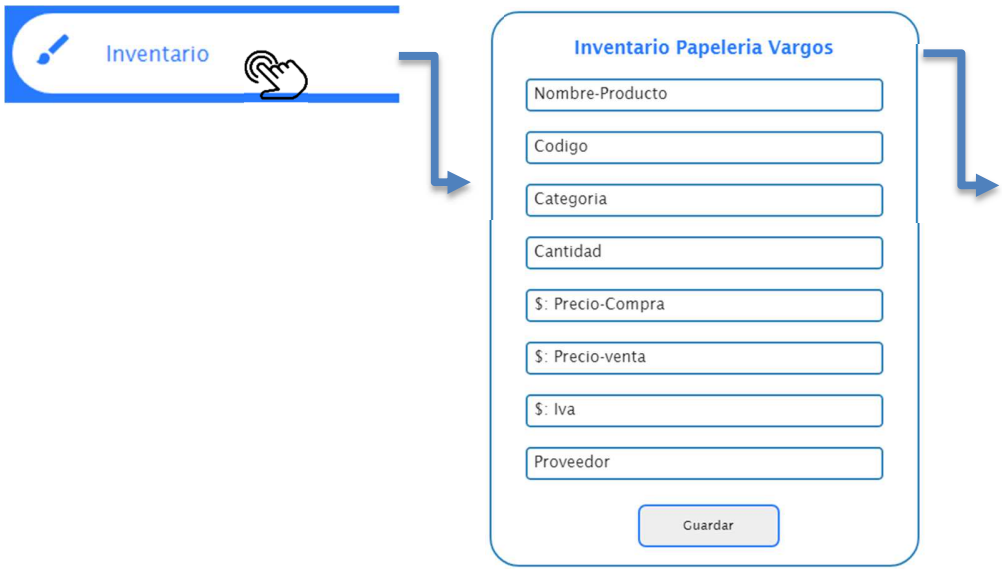
El menú de navegación es un menú desplegable el cual cuenta con 5 pestañas funcionales inventario, usuarios, proveedores, ventas, historial de ventas y cerrar sesión, al darle click

a cualquier pestaña lo re direccionara al módulo seleccionado.



3. Módulos del sistema:

Inventario: Al darle click al módulo inventario se mostrará un formulario con ocho campos correspondientes a los productos.

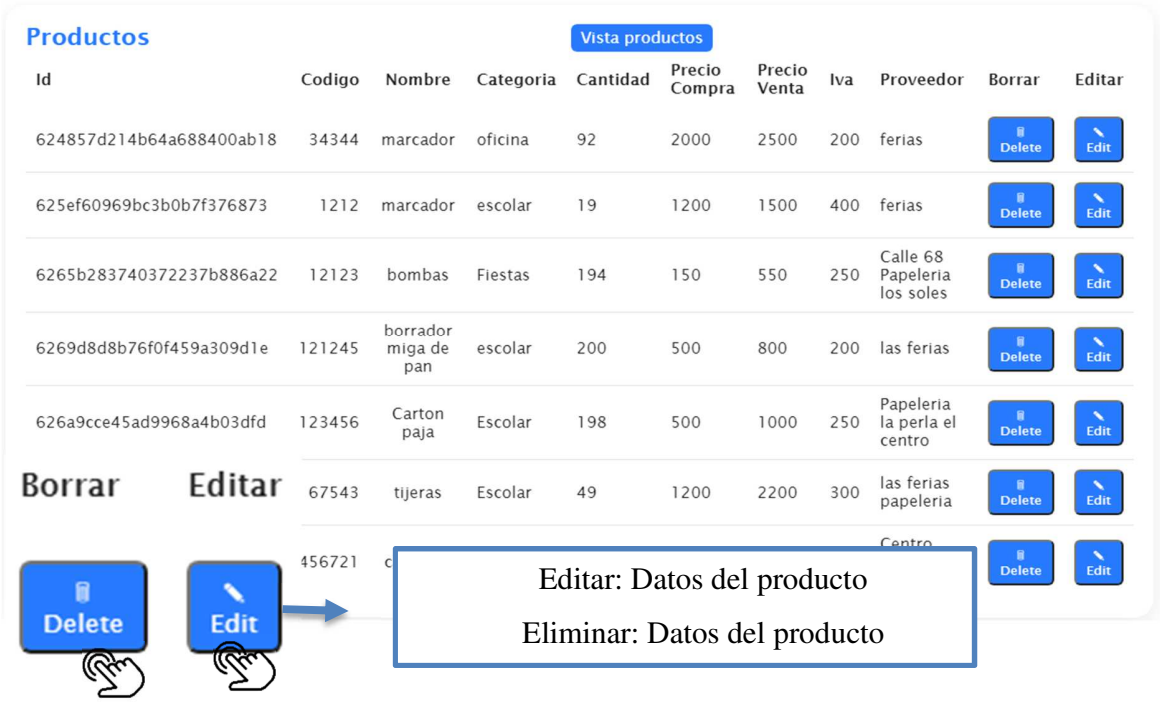


Registrar los datos de los productos.

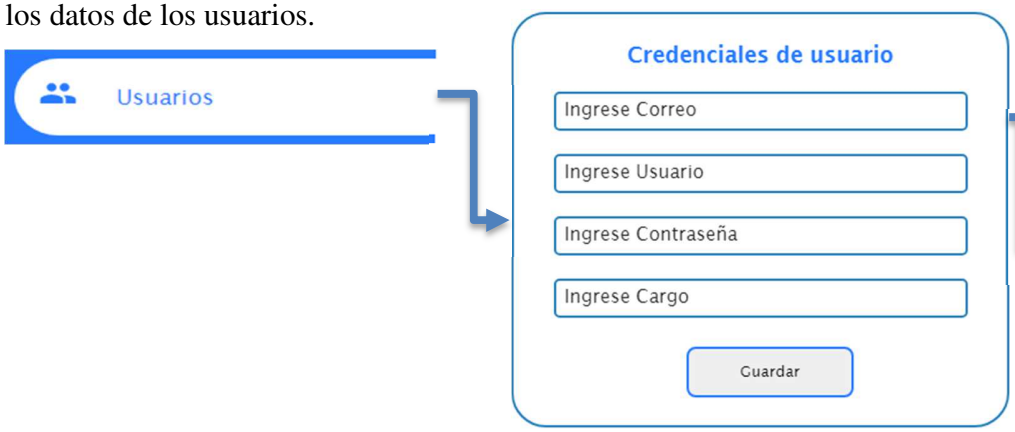
Al finalizar de registrar los datos del formulario darle click al botón guardar.

Tabla de los productos almacenados

Al guardar los datos del formulario se mostrará en la tabla el producto nuevo que fue almacenado.



Usuarios: Al dar click al módulo de usuarios se mostrará un formulario con cuatro campos de registro correspondientes a las credenciales de usuario y una tabla donde se almacenan los datos de los usuarios.

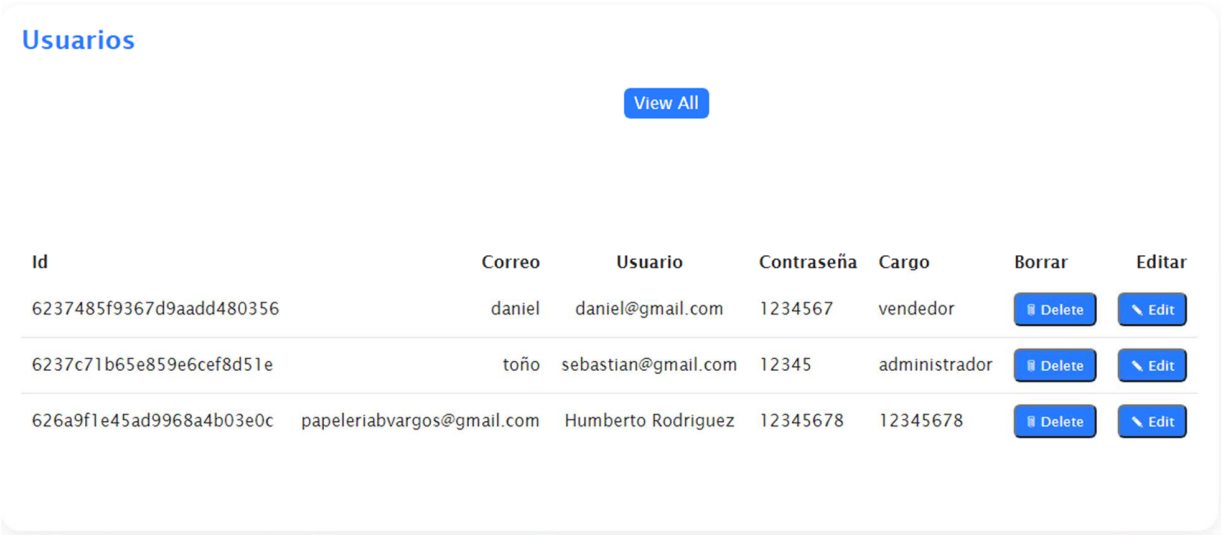


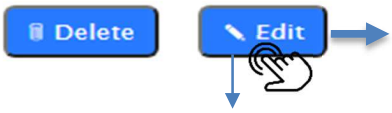
Registrar los datos de los usuarios.

Al finalizar de llenar los campos de los usuarios dar click en guardar.

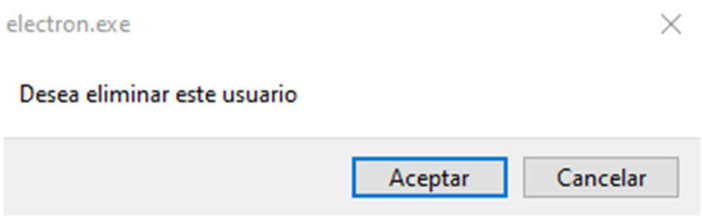
Tabla de los usuarios almacenados

Al guardar los datos se almacenará el usuario nuevo y se mostrará en la tabla.



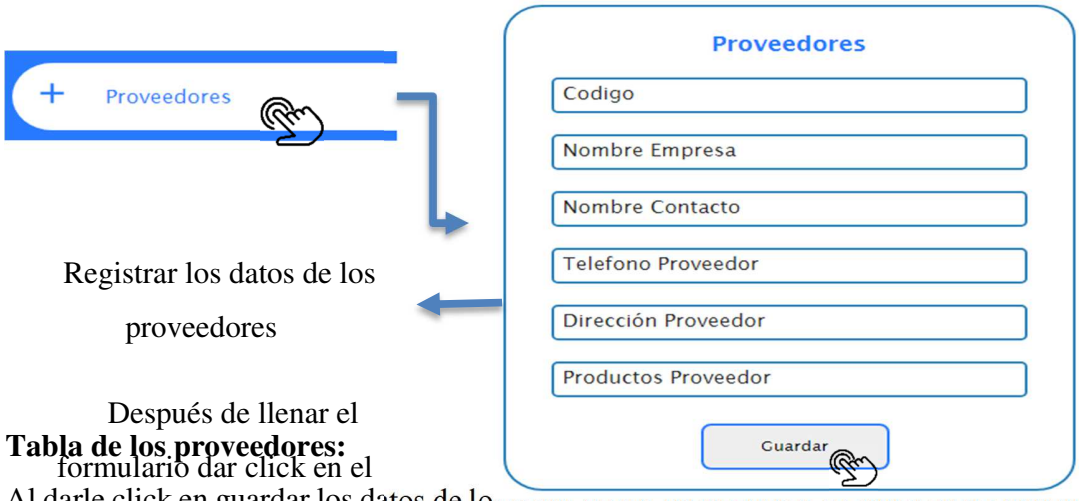


Al dar click en editar se podrá editar el campo que el usuario necesite.



Proveedores:

Al dar click al módulo de proveedores se mostrará un formulario con seis campos y una tabla donde se almacenan los datos de los proveedores.



Registrar los datos de los proveedores

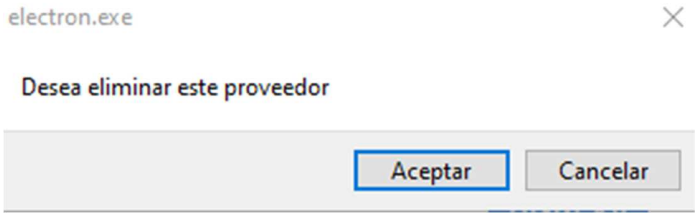
Después de llenar el formulario dar click en el botón guardar.

Tabla de los proveedores:

Al darle click en guardar los datos de los datos de los proveedores se almacenara uno nuevo y se mostrará en la tabla.

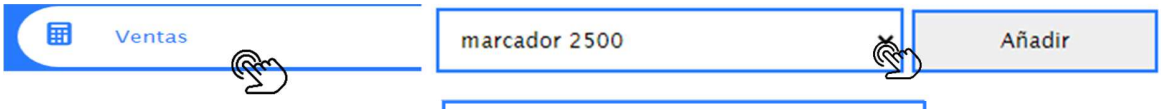
Al dar click en botón de editar se podrá editar el campo que el usuario necesite.

Al dar click en el botón eliminar el sistema enviara una alerta si quiere eliminar los datos de ese proveedor.

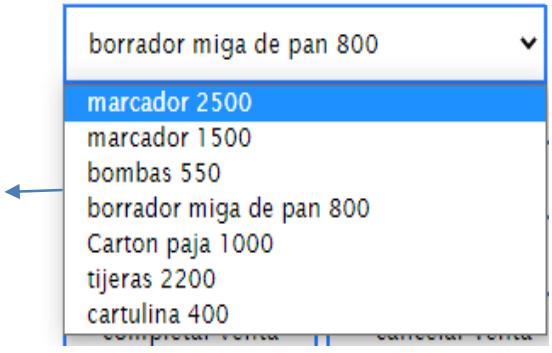


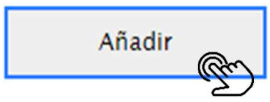
4. **Sistema de ventas**

Ventas: Al darle click al módulo de ventas se mostrará el sistema para gestionar la venta de los productos como se explica detalladamente a continuación:

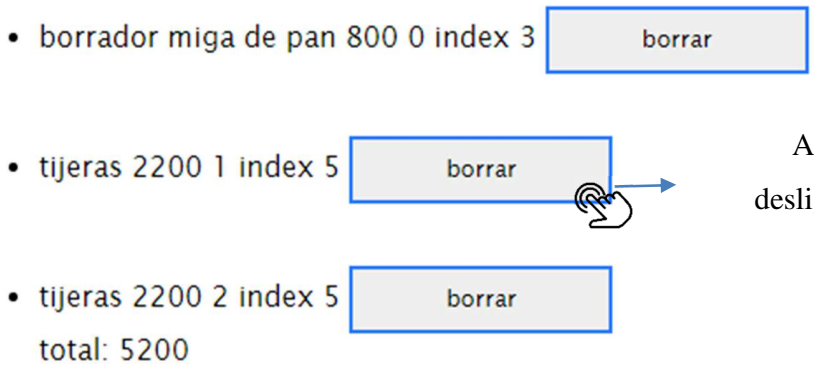


Al darle click en el botón desplegable se listarán los productos, (cada vez que se agrega un producto en el inventario se almacena para las ventas)





Al darle click en el botón añadir se añadirá un producto para la venta.

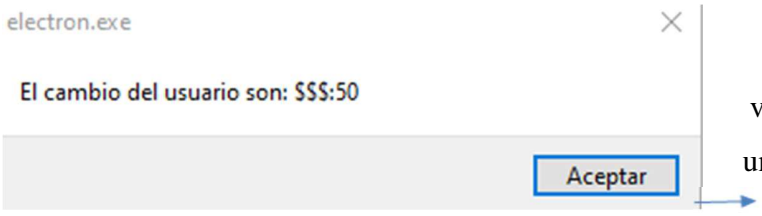


Al darle click en el botón borrar se desligará el producto que el cliente ya no dese llevar.

Después de seleccionar los productos se debe indicar el valor con el que el cliente paga $ 20.000

Dar click en completar venta

Dar click en cancelar venta si el cliente ya no desea llevar nada.



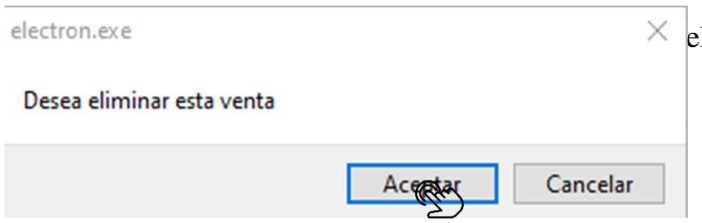
Al darle click a completar venta el programa le mostrara una alerta mostrando cuanto es el cambio del usuario.

Historial de ventas

Al darle click al módulo de historial de ventas se mostrarán todas las ventas almacenadas

en una tabla.

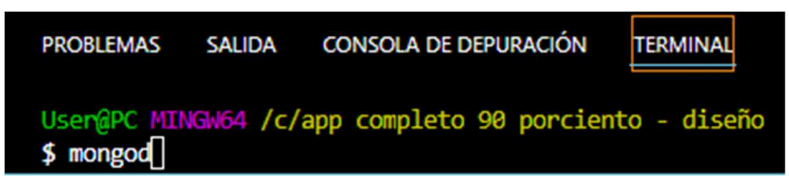
Al darle click a eliminar venta el programa mostrara una alerta si desea eliminar ese producto vendido.



Al darle click a aceptar se eliminará ese producto del historial de ventas.

Manual de procedimiento

1. **Procedimiento del desarrollo del software de inventario vargos:**
    - Descargar Node.js.
    - Descargar MongoDb para gestor de bases de datos.
    - Descargar electrón.
    - Ejecutar el framework de electrón en la terminar del editor de código el cual se utilizó visual studio code para instalar los paquetes con el siguiente comando. (npm install).
    - Ejecutar en otra terminal el comando (**mongod**) para encender la base de datos.



2. Código del desarrollo del software de escritorio.

- Conexión para la base de datos.

JS database.js

app > JS database.js > ...

```
1  const mongoose = require("mongoose");
2  const { MONGODB_URI } = require("./config");
3
4  mongoose
5    .connect(MONGODB_URI)
6    .then((db) => console.log("DB is connected"))
7    .catch((err) => console.log(err));
8
```

3. **Código index.html**

Menú de navegación del software:

```
<div id="menu" class="menu">
  <div class="container">
    <div class="navigation">
      <ul>
        <li>
          <a href="#">
            <span class="icon"><ion-icon name="home"></ion-icon></span>
            <span class="title" > Inventario Vargos</span>
          </a>
        </li>
        <li>
          <a href="#">
            <span class="icon"><ion-icon name="lock"></ion-icon></span>
            <span class="title" >login</span>
          </a>
        </li>
        <li>
          <a href="#">
            <span class="icon"><ion-icon name="brush"></ion-icon></span>
            <span class="title" onclick="mostrarVentana('inventario')">Inventario</span>
          </a>
        </li>
        <li>
          <a href="#">
            <span class="icon"><ion-icon name="people"></ion-icon></span>
            <span class="title" onclick="mostrarVentana('usuario')">Usuarios</span>
          </a>
        </li>
        <li>

          <a href="#">
```

```
            <li>
                <a href="#">
                    <span class="icon"><ion-icon name="calculator"></ion-icon></span>
                    <span class="title" onclick="mostrarVentana('vender')">Ventas</span>
                </a>
            </li>
            <li>
                <a href="#">
                    <span class="icon"><ion-icon name="book"></ion-icon></span>
                    <span class="title" onclick="mostrarVentana('lista-venta')">Historial Ventas</span>
                </a>
            </li>
            <li>
                <a href="#">
                    <span class="icon"><ion-icon name="remove"></ion-icon></span>
                    <span class="title" id="logout" onclick="logout('login')">Cerrar Sesion</span>
                </a>
            </li>


        </ul>
    </div>
</div>
</div>
```

Clase para botón desplegable:

```
<div class="main">
 <div class="topbar">
    <div class="toggle">
        <ion-icon name="menu"></ion-icon>
    </div>
```

HTML login de usuario:

```
<div id="login" class="login ventana">
 <form" class="form-body">
  <p class="text">Bienvenido</p>
 <img src="./img/users.png" alt="user-login">
 <input type="text" name="correo" placeholder="Ingrese nombre de usuario" id="correo-usuario">
 <input type="password" name="contraseña" placeholder="contraseña" id="pass-usuario">
 <button onclick="loguearse()">Loguearse</button>
 </form>
</div>
```

HTML Inventario de productos:

```
<div style= "display: none;" id="inventario" class="inventarios ventana">
  <form id="taskForm" class="card">
    <h3>Inventario Papeleria Vargos</h3>
    <input type="text" id="taskNombre" placeholder="Nombre-Producto" autofocus>
    <input type="text" id="taskCodigo" placeholder="Codigo" autofocus>
    <input type="text" id="taskCategoria" placeholder="Categoria" autofocus>
    <input type="text" id="taskCantidad" placeholder="Cantidad" autofocus>
    <input type="text" id="taskPreciocompra" placeholder="$: Precio-Compra" autofocus>
    <input type="text" id="taskPrecioventa" placeholder="$: Precio-venta" autofocus>
    <input type="text" id="taskIva" placeholder="$: Iva" autofocus>
    <input type="text" id="taskProveedor" placeholder="Proveedor" autofocus>


    <button type="submit" class="botonformulario btn-secondary">
      Guardar
    </button>
  </form>
```

Tabla inventario productos:

```
<!-- Inventarios -->
    <div class="details">
      <!-- order details list-->
        <div class="recentOrders">
            <div class="cardHeader">
                <h2>Productos</h2>
                <a href="#" class="btn">Vista productos</a>
            </div>
            <table>
                <thead>
                    <tr>
                        <td>Id</td>
                        <td>Codigo</td>
                        <td>Nombre</td>
                        <td>Categoria</td>
                        <td>Cantidad</td>
                        <td>Precio Compra</td>
                        <td>Precio Venta</td>
                        <td>Iva</td>
                        <td>Proveedor</td>
                        <td>Borrar</td>
                        <td>Editar</td>
                    </tr>
                </thead>
                <tbody id="taskList">

                </tbody>
            </table>
        </div>
      </div>
```

HTML usuarios:

```
<div style="display: none;" id="usuario" class="usuario ventana">
  <form id="usuarioForm" class="card">
    <h3>Credenciales de usuario</h3>
    <input type="text" id="usuarioName" placeholder="Ingrese Correo" autofocus>
    <input type= "text"id="usuarioDescription" placeholder="Ingrese Usuario" autofocus></input>
    <input type= "password"id="usuarioCorreo" placeholder="Ingrese Contraseña" autofocus></input>
    <input type= "text"id="usuarioCargo" placeholder="Ingrese Cargo" autofocus></input>
    <button type="submit" class="botonformulario btn-secondary">
      Guardar
    </button>
  </form>
```

Tabla usuarios:

```
<div class="details">
  <!-- order details list-->
  <div class="recentOrders">
    <div class="cardHeader">
      <h2>Usuarios</h2>
      <a href="#" class="btn">View All</a>
    </div>
    <table >
      <thead>
        <tr>
          <td>Id</td>
          <td>Correo</td>
          <td>Usuario</td>
          <td>Contraseña</td>
          <td>Cargo</td>
          <td>Borrar</td>
          <td>Editar</td>
        </tr>
      </thead>
      <tbody id="usuarioList">

      </tbody>
    </table>
  </div>
</div>

<!-- Apartado usuario -->
<ul id="usuarioList"></ul>
</div>
```

HTML Proveedores:

```
<div style=" display: none;" id="proveedores" class="proveedores ventana">

  <form id="proveedorForm" class="card">
    <h3>Proveedores</h3>
    <input type="text" id="proveedorCodigo" placeholder="Codigo" autofocus>
    <input type="text" id="proveedorName" placeholder="Nombre Empresa" autofocus>
    <input type="text" id="proveedorContacto" placeholder="Nombre Contacto" autofocus>
    <input type="text" id="proveedorTelefono" placeholder="Telefono Proveedor" autofocus>
    <input type="text" id="proveedorDireccion" placeholder="Dirección Proveedor" autofocus>
    <input type="text" id="proveedorProductos" placeholder="Productos Proveedor" autofocus>

    <button type="submit" class="botonformulario btn-secondary">
      Guardar
    </button>
  </form>
```

```
<!-- provvedor List -->
<div class="details">
  <!-- order details list-->
    <div class="recentOrders">
        <div class="cardHeader">
            <h2>Proveedores</h2>
            <a href="#" class="btn">View All</a>
        </div>
        <table>
            <thead>
                <tr>
                    <td>Id</td>
                    <td>Codigo</td>
                    <td>Nombre</td>
                    <td>Direccion</td>
                    <td>Telefono</td>
                    <td>Contacto</td>
                    <td>Productos</td>
                    <td>Borrar</td>
                    <td>Editar</td>
                </tr>
            </thead>
            <tbody id="proveedorList">

            </tbody>
        </table>
    </div>
  </div>
```

HTML módulo de ventas:

```
<!-- Apartado Ventas -->

<div style="display: none;" id="vender" class="vender ventana">

  <h3 class="ventasvargos">Sistema de ventas</h3>
  <select id="seleccionProductoLista">
  </select>
  <button class="ventaan" onclick="addProductoVenta()">Añadir</button>
  <input type="datetime-local" id="obtenerfecha">


  <p>paga con: <ion-icon name="card" class="cash"></ion-icon> <input type="number" name="" id="paga-con"></p>
  <button id="completar-venta">completar venta</button>
  <button id="cancelar-venta" onclick="cancelarVenta()">cancelar venta</button>

  <ul id="lista-productos-venta"></ul>
  <div id="total-suma"></div>
</div>

<div style=" display: none;" id="lista-venta" class="lista-venta ventana">
```

```
<div class="details">
  <!-- order details list-->
    <div class="recentOrders">
      <div class="cardHeader">
        <h2>Productos vendidos</h2>
        <a href="#" class="btn">Vista modulo ventas</a>
      </div>
      <table>
        <thead>
          <tr>
            <td>Id</td>
            <td>Nombre</td>
            <td>Precio</td>
            <td>Eliminar</td>
          </tr>
        </thead>
        <tbody id="ventaList">
        </tbody>
      </table>
    </div>

  </div>

  <!-- Venta List -->

  <ul id="ventaList"></ul >
 </div>
</div>
```

Scripts para utilizar los iconos en el programa:

```
<script src="ui/app.js"></script>
<script src="https://unpkg.com/ionicons@4.5.10-0/dist/ionicons.js"></script>
<script>
```

Funciones JavaScript menú desplegable:

```
//Menutoggle

let toggle = document.querySelector('.toggle');
let navigation = document.querySelector('.navigation');
let main = document.querySelector('.main');


toggle.onclick = function(){
    navigation.classList.toggle('active');
    main.classList.toggle('active');
}

// add hovered select
let list = document.querySelectorAll('.navigation li');
function activeLink(){
    list.forEach((item) =>
    item.classList.remove('hovered'));
    this.classList.add('hovered');

}

list.forEach((item) =>
item.addEventListener('mouseover',activeLink));
</script>
```

Configuración JavaScript:


```
app > JS config.js > ...
1   require("dotenv").config();
2
3   module.exports = {
4     MONGODB_URI: process.env.MONGODB_URI || "mongodb://localhost/electrondb",
5   };
6
```

Main JavaScript

```
const { BrowserWindow, ipcMain } = require("electron");
const Task = require("./models/Task");
const Usuario = require("./models/Usuario");
const Proveedor = require("./models/Proveedor");
const Venta = require("./models/Venta");


function createWindow() {
  const win = new BrowserWindow({
    width: 500,
    height: 700,
    webPreferences: {
      nodeIntegration: true,
      contextIsolation: false,
    },
    autoHideMenuBar: true,
  });

  win.loadFile("app/index.html");
}
```

IpcMain: Es el modulo el cual se utilizó para la comunicación desde el proceso principal a los procesos renderizados el cual se utiliza el proceso principal, por ende, maneja todos los mensajes síncronos y asíncronos.

```
ipcMain.on("new-task", async (e, arg) => {
  const newTask = new Task(arg);
  const taskSaved = await newTask.save();
  e.reply("new-task-created", JSON.stringify(taskSaved));
});

ipcMain.on("get-tasks", async (e, arg) => {
  const tasks = await Task.find();
  e.reply("get-tasks", JSON.stringify(tasks));
});

ipcMain.on("delete-task", async (e, args) => {
  const taskDeleted = await Task.findByIdAndDelete(args);
  e.reply("delete-task-success", JSON.stringify(taskDeleted));
});

ipcMain.on("update-task", async (e, args) => {
  console.log(args);
  const updatedTask = await Task.findByIdAndUpdate(
    args.idTaskToUpdate,
    { Codigo: args.Codigo, Nombre: args.Nombre, Categoria: args.Categoria, Cantidad: args.Cantidad,Preciocompra:
    { new: true }
  );
  e.reply("update-task-success", JSON.stringify(updatedTask));
});
```

```
ipcMain.on("new-usuario", async (e, arg) => {
  const newUsuario = new Usuario(arg);
  const usuarioSaved = await newUsuario.save();
  e.reply("new-usuario-created", JSON.stringify(usuarioSaved));
});

ipcMain.on("get-usuarios", async (e, arg) => {
  const usuarios = await Usuario.find();
  e.reply("get-usuarios", JSON.stringify(usuarios));
});

ipcMain.on("delete-usuario", async (e, args) => {
  const usuarioDeleted = await Usuario.findByIdAndDelete(args);
  e.reply("delete-usuario-success", JSON.stringify(usuarioDeleted));
});

ipcMain.on("update-usuario", async (e, args) => {
  console.log(args);
  const updatedUsuario = await Usuario.findByIdAndUpdate(
    args.idUsuarioToUpdate,
    { name: args.name, description: args.description,correo: args.correo,cargo: args.cargo},
    { new: true }
  );
  e.reply("update-usuario-success", JSON.stringify(updatedUsuario));
});
```

```
ipcMain.on("new-proveedor", async (e, arg) => {
  const newProveedor = new Proveedor(arg);
  const proveedorSaved = await newProveedor.save();
  e.reply("new-proveedor-created", JSON.stringify(proveedorSaved));
});

ipcMain.on("get-proveedores", async (e, arg) => {
  const proveedores = await Proveedor.find();
  e.reply("get-proveedores", JSON.stringify(proveedores));
});

ipcMain.on("delete-proveedor", async (e, args) => {
  const proveedorDeleted = await Proveedor.findByIdAndDelete(args);
  e.reply("delete-proveedor-success", JSON.stringify(proveedorDeleted));
});

ipcMain.on("update-proveedor", async (e, args) => {
  console.log(args);
  const updatedProveedor = await Proveedor.findByIdAndUpdate(
    args.idProveedorToUpdate,
    { name: args.name, telefono: args.telefono, direccion: args.direccion, contacto: args.contacto, productos: a
    { new: true }
  );
  e.reply("update-proveedor-success", JSON.stringify(updatedProveedor));
});
```

```
ipcMain.on("new-venta", async (e, arg) => {
  const newVenta = new Venta(arg);
  const ventaSaved = await newVenta.save();
  e.reply("new-venta-created", JSON.stringify(ventaSaved));
});

ipcMain.on("get-ventas", async (e, arg) => {
  const ventas = await Venta.find();
  e.reply("get-ventas", JSON.stringify(ventas));
});

ipcMain.on("delete-venta", async (e, args) => {
  const ventaDeleted = await Venta.findByIdAndDelete(args);
  e.reply("delete-venta-success", JSON.stringify(ventaDeleted));
});

ipcMain.on("update-venta", async (e, args) => {
  console.log(args);
  const updatedVenta = await Venta.findByIdAndUpdate(
    args.idVentaToUpdate,
    { name: args.name, precio: args.precio },
    { new: true }
  );
  e.reply("update-venta-success", JSON.stringify(updatedVenta));
});

module.exports = { createWindow };
```

App JavaScript:

Variables utilizadas en el desarrollo del software.

```
const { ipcRenderer } = require("electron");
//*.VARIABLES UTILIZADAS EN LA APP INVENTARIO......//
const taskForm = document.querySelector("#taskForm");
const taskCodigo = document.querySelector("#taskCodigo");
const taskList = document.querySelector("#taskList");
const taskNombre = document.querySelector("#taskNombre");
const taskCategoria = document.querySelector("#taskCategoria");
const taskCantidad = document.querySelector("#taskCantidad");
const taskPrecio = document.querySelector("#taskPreciocompra");
const taskPrecioventa = document.querySelector("#taskPrecioventa");
const taskIva = document.querySelector("#taskIva");
const taskProveedor = document.querySelector("#taskProveedor");
const codigoIdUsuario = document.querySelector("#codigoIdUsuario");
const formularioUsuario = document.querySelector("#formularioUsuario");
const usuarioForm = document.querySelector("#usuarioForm");
const usuarioName = document.querySelector("#usuarioName");
const usuarioDescription = document.querySelector("#usuarioDescription");
const usuarioList = document.querySelector("#usuarioList");
const usuarioCorreo = document.querySelector("#usuarioCorreo");
const usuarioCargo = document.querySelector("#usuarioCargo");
const proveedorForm = document.querySelector("#proveedorForm");
const proveedorName = document.querySelector("#proveedorName");
const proveedorTelefono = document.querySelector("#proveedorTelefono");
const proveedorDireccion = document.querySelector("#proveedorDireccion");
const proveedorList = document.querySelector("#proveedorList");
const proveedorContacto = document.querySelector("#proveedorContacto");
const proveedorProductos = document.querySelector("#proveedorProductos");
const proveedorCodigo = document.querySelector("#proveedorCodigo");

//*.VARIABLES VENTAS......//
const ventaForm = document.querySelector("#ventaForm");
const ventaName = document.querySelector("#ventaName");
const ventaPrecio = document.querySelector("#ventaPrecio");
const ventaList = document.querySelector("#ventaList");
```

Función eliminar y editar producto:

Funciones de los productos

```
function deleteTask(id) {
  const response = confirm("Desea eliminar este producto?");
  if (response) {
    ipcRenderer.send("Eliminar producto", id);
  }
  return;
}

function editTask(id) {
  updateStatus = true;
  idTaskToUpdate = id;
  const task = tasks.find((task) => task._id === id);
  taskCodigo.value = task.codigo;
  taskNombre.value = task.nombre;
  taskCategoria.value = task.categoria;
  taskCantidad.value = task.cantidad;
  taskPreciocompra.value = task.preciocompra;
  taskProveedor.value = task.proveedor;
  taskPrecioventa.value = task.precioventa;
  taskIva.value = task.iva;

}
```

```
function renderTasks(tasks) {
  taskList.innerHTML = "";
  tasks.map((t) => {
    taskList.innerHTML += `
          <tr class="card">
            <td>
            ${t._id}
            </td>
            <td>
            ${t.Codigo}
            </td>
            <td>
            ${t.Nombre}
            </td>
            <td>
            ${t.Categoria}
            </td>
            <td>
            ${t.Cantidad}
            </td>
            <td>
            ${t.Preciocompra}
            </td>
            <td>
            ${t.Precioventa}
            </td>
            <td>
            ${t.Iva}
            </td>
            <td>
            ${t.Proveedor}
            </td>
            <td>
            <button class="btn btn-danger" onclick="deleteTask('${t._id}')">
              Delete
            </button>
            </td>
            <td>
            <button class="btn btn-secondary" onclick="editTask('${t._id}')">
              Edit
            </button>
```

```
seleccionProductoLista.innerHTML = "";

for(let i = 0; i < tasks.length; i++){
  tasks[i].Precioventa
  seleccionProductoLista.innerHTML += `
  <option value="${tasks[i].Precioventa}" data-name="${tasks[i].Nombre}" data-index="${i}" data-cantidad="${ta
  ${tasks[i].Nombre}
  ${tasks[i].Precioventa}
  </option>
` ;
}
```

```
let tasks = [];

ipcRenderer.send("get-tasks");

taskForm.addEventListener("submit", async (e) => {
  e.preventDefault();

  const task = {
    Codigo: taskCodigo.value,
    Nombre: taskNombre.value,
    Categoria: taskCategoria.value,
    Cantidad: taskCantidad.value,
    Preciocompra: taskPreciocompra.value,
    Precioventa: taskPrecioventa.value,
    Iva: taskIva.value,
    Proveedor: taskProveedor.value,



  };

  if (!updateStatus) {
    ipcRenderer.send("new-task", task);
  } else {
    ipcRenderer.send("update-task", { ...task, idTaskToUpdate });
  }

  taskForm.reset();
});

ipcRenderer.on("new-task-created", (e, arg) => {
  console.log(arg);
  const taskSaved = JSON.parse(arg);
  tasks.push(taskSaved);
  console.log(tasks);
  renderTasks(tasks);
  alert("Task Created Successfully");
  taskName.focus();
});
```

```
ipcRenderer.on("get-tasks", (e, args) => {
  const receivedTasks = JSON.parse(args);
  tasks = receivedTasks;
  renderTasks(tasks);
});

ipcRenderer.on("delete-task-success", (e, args) => {
  const deletedTask = JSON.parse(args);
  const newTasks = tasks.filter((t) => {
    return t._id !== deletedTask._id;
  });
  tasks = newTasks;
  renderTasks(tasks);
});

ipcRenderer.on("update-task-success", (e, args) => {
  updateStatus = false;
  const updatedTask = JSON.parse(args);
  tasks = tasks.map((t, i) => {
    if (t._id === updatedTask._id) {
      t.Codigo = updatedTask.Codigo;
      t.Nombre = updatedTask.Nombre;
      t.Categoria = updatedTask.Categoria;
      t.Cantidad = updatedTask.Cantidad;
      t.Preciocompra = updatedTask.Preciocompra;
      t.Precioventa = updatedTask.Precioventa;
      t.Proveedor = updatedTask.Proveedor;
      t.Iva = updatedTask.Iva;


    }
    return t;
  });
  renderTasks(tasks);
});
```

Funciones de los usuarios

Función eliminar usuario

```
function deleteUsuario(id) {
  const response = confirm("Desea eliminar este usuario");
  if (response) {
    ipcRenderer.send("delete-usuario", id);
  }
  return;
}
```

Función editar usuario

```
function editUsuario(id) {
  updateStatus = true;
  idUsuarioToUpdate = id;
  const usuario = usuarios.find((usuario) => usuario._id === id);
  usuarioName.value = usuario.name;
  usuarioDescription.value = usuario.description;
  usuarioCorreo.value = usuario.correo;
  usuarioCargo.value = usuario.cargo;
}
```

Función rende rizar usuario

```
function renderUsuarios(usuarios) {
  usuarioList.innerHTML = "";
  usuarios.map((t) => {
    usuarioList.innerHTML += `

<tr class="card">
                <td>
                ${t._id}
                </td>
                <td>
                ${t.name}
                </td>
                <td>
                ${t.description}
                </td>
                <td>
                ${t.correo}
                </td>
                <td>
                ${t.cargo}
                </td>
                <td>
                <button class="btn btn-danger" onclick="deleteUsuario('${t._id}')">
                  Delete
                </button>
                </td>
                <td>
                <button class="btn btn-secondary" onclick="editUsuario('${t._id}')">
                  Edit
                </button>
                </td>
              </tr>
            `;
  });
}
```

```
let Usuarios = [];

ipcRenderer.send("get-usuarios");

usuarioForm.addEventListener("submit", async (e) => {
  e.preventDefault();

  const usuario = {
    name: usuarioName.value,
    description: usuarioDescription.value,
    correo: usuarioCorreo.value,
    cargo: usuarioCargo.value,
  };

  if (!updateStatus) {
    ipcRenderer.send("new-usuario", usuario);
  } else {
    ipcRenderer.send("update-usuario", { ...usuario, idUsuarioToUpdate });
  }

  usuarioForm.reset();
});

ipcRenderer.on("new-usuario-created", (e, arg) => {
  console.log(arg);
  const usuarioSaved = JSON.parse(arg);
  usuarios.push(usuarioSaved);
  console.log(usuarios);
  renderUsuarios(usuarios);
  alert("Usuario Created Successfully");
  usuarioName.focus();
});

ipcRenderer.on("get-usuarios", (e, args) => {
  const receivedUsuarios = JSON.parse(args);
  usuarios = receivedUsuarios;
  renderUsuarios(usuarios);
});
```

```
ipcRenderer.on("delete-usuario-success", (e, args) => {
  const deletedUsuario = JSON.parse(args);
  const newUsuarios = usuarios.filter((t) => {
    return t._id !== deletedUsuario._id;
  });
  usuarios = newUsuarios;
  renderUsuarios(usuarios);
});

ipcRenderer.on("update-usuario-success", (e, args) => {
  updateStatus = false;
  const updatedUsuario = JSON.parse(args);
  usuarios = usuarios.map((t, i) => {
    if (t._id === updatedUsuario._id) {
      t.name = updatedUsuario.name;
      t.description = updatedUsuario.description;
      t.correo = updatedUsuario.correo;
      t.cargo = updatedUsuario.cargo;
    }
    return t;
  });
  renderUsuarios(usuarios);
});
```

Funciones proveedores:

Función eliminar proveedor

```
function deleteProveedor(id) {
  const response = confirm("Desea eliminar este proveedor");
  if (response) {
    ipcRenderer.send("delete-proveedor", id);
  }
  return;
}
```

Función editar proveedor

```
function editProveedor(id) {
  updateStatus = true;
  idProveedorToUpdate = id;
  const proveedor = proveedores.find((proveedor) => proveedor._id === id);
  proveedorName.value = proveedor.name;
  proveedorTelefono.value = proveedor.telefono;
  proveedorDireccion.value = proveedor.direccion;
  proveedorContacto.value = proveedor.contacto;
  proveedorProductos.value = proveedor.productos;
  proveedorCodigo.value = proveedor.codigo;
}
```

```
function renderProveedores(proveedores) {
  proveedorList.innerHTML = "";
  proveedores.map((t) => {
    proveedorList.innerHTML += `
<tr class="card">
<td>
${t._id}
</td>
<td>
${t.codigo}
</td>
<td>
${t.name}
</td>
<td>
${t.direccion}
</td>
<td>
${t.telefono}
</td>
<td>
${t.contacto}
</td>
<td>
${t.productos}
</td>
<td>
<button class="btn btn-danger" onclick="deleteProveedor('${t._id}')">
  🗑 Delete
</button>
</td>
<td>
<button class="btn btn-secondary" onclick="editProveedor('${t._id}')">
  ✎ Edit
</button>
</td>

</tr>
`;
});
}
```

```
let proveedores = [];

ipcRenderer.send("get-proveedores");

proveedorForm.addEventListener("submit", async (e) => {
  e.preventDefault();

  const proveedor = {
    name: proveedorName.value,
    telefono: proveedorTelefono.value,
    direccion: proveedorDireccion.value,
    contacto: proveedorContacto.value,
    productos: proveedorProductos.value,
    codigo: proveedorCodigo.value,
  };

  if (!updateStatus) {
    ipcRenderer.send("new-proveedor", proveedor);
  } else {
    ipcRenderer.send("update-proveedor", { ...proveedor, idProveedorToUpdate });
  }

  proveedorForm.reset();
});

ipcRenderer.on("new-proveedor-created", (e, arg) => {
  console.log(arg);
  const proveedorSaved = JSON.parse(arg);
  proveedores.push(proveedorSaved);
  console.log(proveedores);
  renderProveedores(proveedores);
  alert("Proveedor Created Successfully");
  proveedorName.focus();
});

ipcRenderer.on("get-proveedores", (e, args) => {
  const receivedProveedores = JSON.parse(args);
  proveedores = receivedProveedores;
  renderProveedores(proveedores);
});
```

```
ipcRenderer.on("delete-proveedor-success", (e, args) => {
  const deletedProveedor = JSON.parse(args);
  const newProveedores = proveedores.filter((t) => {
    return t._id !== deletedProveedor._id;
  });
  proveedores = newProveedores;
  renderProveedores(proveedores);
});

ipcRenderer.on("update-proveedor-success", (e, args) => {
  updateStatus = false;
  const updatedProveedor = JSON.parse(args);
  proveedores = proveedores.map((t, i) => {
    if (t._id === updatedProveedor._id) {
      t.name = updatedProveedor.name;
      t.telefono = updatedProveedor.telefono;
      t.direccion = updatedProveedor.direccion;
      t.contacto = updatedProveedor.contacto;
      t.productos = updatedProveedor.productos;
      t.codigo = updatedProveedor.codigo;
    }
    return t;
  });
  renderProveedores(proveedores);
});
```

Funciones utilizadas en las ventas.

Función eliminar venta

```
function deleteVenta(id) {
  const response = confirm("Desea eliminar esta venta");
  if (response) {
    ipcRenderer.send("delete-venta", id);
  }
  return;
}
```

```
function renderVentas(ventas) {
  ventaList.innerHTML = "";
  ventas.map((t) => {
    ventaList.innerHTML += `
        <tr class="card">
        <td>
        ${t._id}
        </td>
        <td>
        ${t.name}
        </td>
        <td>
        ${t.precio}
        </td>
        <td>
          <button class="btn btn-danger" onclick="deleteVenta('${t._id}')">
            Delete
          </button>
          </td>
        </tr>
      `;
  });
}
```

```
let ventas = [];
let listaProductosCompra = [];
const seleccionProductoLista = document.querySelector("#seleccionProductoLista");


ipcRenderer.send("get-ventas");

const completarVenta = document.querySelector("#completar-venta");
const listaProductoVenta = document.querySelector("#lista-productos-venta");
const totalSuma = document.querySelector("#total-suma");
let sumaProductos = 0;
let n = 0;
const pagaCon = document.querySelector("#paga-con");

completarVenta.addEventListener("click", async (e) => {
  e.preventDefault();
  for (const value of listaProductosCompra) {
    const venta = {
      name: value.nombreProducto,
      precio: value.productoPrecio,
    };
    ipcRenderer.send("new-venta", venta);
    listaProductoVenta.innerHTML = "";
    let cantidadDeProductos = 0;
    for(let k = 0; k < listaProductosCompra.length; k++){
      if(value.indexProducto ==  listaProductosCompra[k].indexProducto){
        cantidadDeProductos++;
        console.log("cantidad de productos = " + cantidadDeProductos)
      }
    }


    restarProducto(value.indexProducto, value.cantidadProducto, cantidadDeProductos)
  }
```

```
 alert("El cambio del usuario son: $$$:" + (parseInt(pagaCon.value) - sumaProductos));

 totalSuma.innerHTML= "";
 pagaCon.value = "";
 sumaProductos = 0;
 listaProductosCompra =[];
 n = 0;
);
```

```
function addProductoVenta(){

  let productoSeleccionado = {
    nombreProducto: seleccionProductoLista.options[seleccionProductoLista.selectedIndex].dataset.name,
    indexProducto: seleccionProductoLista.options[seleccionProductoLista.selectedIndex].dataset.index,
    cantidadProducto: seleccionProductoLista.options[seleccionProductoLista.selectedIndex].dataset.cantidad,
    productoPrecio: seleccionProductoLista.value
  }

  listaProductosCompra.push(productoSeleccionado);
  listaProductoVenta.innerHTML += `
    <li>${ seleccionProductoLista.options[seleccionProductoLista.selectedIndex].dataset.name } ${ seleccionProdu
  `

  n ++;
  sumaProductos = sumaProductos + parseInt(seleccionProductoLista.value);
  totalSuma.innerHTML = `<p>total: ${ sumaProductos }</p>`;

}

function cancelarVenta(){
  listaProductosCompra = [];
  listaProductoVenta.innerHTML = "";
}

function borrarElementoArray(borrarElemento){
  sumaProductos = sumaProductos - parseInt(listaProductosCompra[borrarElemento].productoPrecio);
  totalSuma.innerHTML = `<p>total: ${ sumaProductos }</p>`;

  listaProductosCompra.splice(borrarElemento, 1);
  listaProductoVenta.innerHTML = "";
  //mostrar array
  n = listaProductosCompra.length;
  for (let step = 0; step < listaProductosCompra.length; step++) {
    listaProductoVenta.innerHTML += `
    <li>${ listaProductosCompra[step].nombreProducto } ${ listaProductosCompra[step].productoPrecio } ${step}<bu
    `
  }
}
```

```
ipcRenderer.on("new-venta-created", (e, arg) => {
  console.log(arg);
  const ventaSaved = JSON.parse(arg);
  ventas.push(ventaSaved);
  console.log(ventas);
  renderVentas(ventas);
});

ipcRenderer.on("get-ventas", (e, args) => {
  const receivedVentas = JSON.parse(args);
  ventas = receivedVentas;
  renderVentas(ventas);
});

ipcRenderer.on("delete-venta-success", (e, args) => {
  const deletedVenta = JSON.parse(args);
  const newVentas = ventas.filter((t) => {
    return t._id !== deletedVenta._id;
  });
  ventas = newVentas;
  renderVentas(ventas);
});



function restarProducto(idProductoResta, cantidadProducto, menosTantosProductos) {
  let idTaskToUpdate = tasks[idProductoResta]._id
  //console.log(idTaskToUpdate)

  const task = {
    Cantidad: cantidadProducto - menosTantosProductos,
  };
  console.log(idTaskToUpdate + " " + task.Cantidad + " " + menosTantosProductos)
  ipcRenderer.send("update-task", { ...task, idTaskToUpdate });
}
```

Funciona para loguearse el usuario

```
function loguearse() {
  let inputCorreo = document.getElementById("correo-usuario").value
  let inputPass = document.getElementById("pass-usuario").value
  let status;
  let indiceUsuario;

  for (let i = 0; i < usuarios.length; i++) {
    console.log(usuarios[i].name + " " + usuarios[i].correo)
    if((usuarios[i].name == inputCorreo) && (usuarios[i].correo == inputPass)){
      status = true;
      indiceUsuario = i;
    }
  }
  if(status == true){
    cargo = usuarios[indiceUsuario].cargo

    if(cargo == "vendedor"){
      mostrarVendedor()
    }
    if(cargo == "administrador"){
      mostrarAdministrador()
    }
  }else{
    alert("datos incorrectos")
  }


}
```