PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
FACULTAD DE INGENIERÍA
INGENIERÍA EN SISTEMAS DE INFORMACIÓN

DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN DEL STOCK DE PRODUCTOS APLICADO AL CASO DE ESTUDIO: MINI MARKET "PAOLA"

ORTEGA JUCA ARIEL ALEXANDER

DISERTACIÓN DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE: INGENIERO DE SISTEMAS DE INFORMACIÓN

Quito, junio 2022

Dedicatoria

Este trabajo de disertación está dedicado a:

A mi madre por apoyarme emocionalmente en los malos momentos y por darme ánimos para seguir adelante sin importar lo difícil de la situación.

A Dios por darme fuerzas y sabiduría para cumplir los objetivos que me he planteado y poder seguir mi camino sin mayores problemas.

A mi padre por siempre confiar en mí y apoyarme en la continuidad de mi educación sin dudarlo, tanto en lo económico como en lo anímico.

A mi hermano por siempre darme buenos consejos y por sus recomendaciones para poder ser un buen estudiante y profesional.

A mi familia por siempre ser tan unida y por su disposición de ayudar sin importar la situación o la hora.

A mis amigos por ser un buen apoyo moral y emocional cuando las cosas no salen de la mejor manera.

Agradecimiento

En este apartado quisiera agradecer a mis padres y a mi hermano, personas quienes me permitieron realizar mis estudios en la universidad y en la carrera que quería, por supuesto, también quiero agradecer por todo su apoyo económico y emocional a lo largo de todos estos años.

Asimismo, ofrezco mis profundos agradecimientos a mi familia, ya que ellos me ayudaron a encontrar oportunidades laborales con el fin de poner en práctica mis conocimientos y por ende tener un poco más de experiencia en futuros trabajos.

Por último, me gustaría agradecer a mis profesores, autoridades y empleados de la Universidad Católica del Ecuador por facilitarme las mejores oportunidades y recursos para fortalecer mi aprendizaje previo a una vida profesional.

Tabla de contenidos

Índice de tablas

Índice de gráficos

Resumen o Abstract

El siguiente trabajo de disertación tiene como finalidad la elaboración de una aplicación web que satisfaga la necesidad latente del mini market “Paola”, la cual es gestionar el stock de sus productos; por ende, en este documento se va a evidenciar la resolución de la problemática planteada empezando por una introducción en donde se va a conocer un poco más a detalle el negocio y la necesidad que se tiene, además, se va a plantear los objetivos los cuales se debe cumplir para dar por acabado el caso, asimismo, se va a redactar el alcance que dicha aplicación web tendrá.

Por consiguiente, se va a abordar el problema por medio de un marco teórico en donde se conozca un poco más sobre las herramientas, metodologías de desarrollo y arquitecturas que puedan servir para llevar a cabo este trabajo. Después, se escogerá la metodología que se va a emplear para el desarrollo de esta aplicación; por lo tanto, también se levantará los respectivos requerimientos funcionales y no funcionales para el caso.

Por último, se va a emplear la metodología escogida previamente para el respectivo desarrollo de la aplicación web de gestión de stock de productos para el mini market “Paola”, en donde se mostrará dicho desarrollo fase por fase con su respectiva explicación de lo que se hace y se pretende conseguir en cada una de estas mismas.

Capítulo 1: Introducción

1.1 Justificación

Mini Market Paola, es un negocio de víveres ubicado en la Av. Vicente Ramón Roca, tras la pandemia este negocio como muchos otros han sufrido muchos percances e incluso hasta crisis, ya que se ha perdido proveedores, clientes, productos y por ende dinero, dado que se tenía que hacer cuarentena y los estudiantes universitarios los cuales eran su principal fuente de dinero tuvieron que recibir clases en línea, de tal manera muchos negocios se han dado cuenta de la importancia y la ayuda que se puede ofrecer la tecnología a nivel de negocio para retomar las actividades e incluso optimizarlas, es por eso por lo que el proyecto propuesto de desarrollo de una aplicación web para la gestión del stock de productos para dicho Mini Market justifica el desarrollo de este mismo debido a que solventa una necesidad real latente.

Ahora bien, el proyecto pretende ayudar a llevar mejor la gestión del stock de sus productos, esto a través de la automatización de sus procesos para un manejo más organizando y óptimo de los productos que se adquieren y a la vez de los productos que se venden, así como sus proveedores, además, el proyecto promete incentivar el uso de la tecnología a nivel de pequeños negocios locales y sobre todo destacar al Mini Market Paola ante su competencia por su excelente desempeño impulsado por la aplicación web propuesta en este proyecto.

Abordando el tema de los inventarios, cabe destacar que son uno de los procesos más importantes dentro de un negocio, puesto que a partir de allí los administradores se pueden dar una idea de la demanda de los productos, así como un análisis para futuras compras. Por otro lado, contar con sistema automatizado de gestión de stock también representa una buena organización y agilidad dentro de en un negocio sea pequeño,

mediano o grande. Incluso, estos sistemas de gestión de stock proveen de información valiosa acerca del progreso o estancamiento actual, al igual que una cuenta exacta de los ingresos y egresos, productos sobrantes y faltantes, inclusive hasta dotando a los negocios a lo que respecta con la integridad y disponibilidad de la información.

Por otro lado, el contar con un inventario no solo mejora a un negocio con lo ya mencionado anteriormente, sino que también muestra a los altos cargos el patrimonio con el que cuenta de una forma ordenada y detallada, dicha manera de trabajar con los datos toma fuerza debido a que ayuda mucho en las actividades diarias del negocio, dado que es más fácil revisar y hacer controles de stock de los productos, lo cual influye para un análisis futuro de las ganancias previstas a través de la venta de estos productos (Tacuri & Rodriguez, 2020).

En síntesis, se sugiere que todo negocio deba plantearse la implementación de gestión de stocks en los negocios, dada la rentabilidad que esta conlleva en comparación a las ventajas con las está provee, ya como es una mayor competitividad a un precio bajo, mayor satisfacción con los clientes, versátil desarrollo en negocios de cualquier índole y por último el apoyo que implica en cuanto al manejo, planificación, estrategias y tomas de decisiones a futuro, por tales razones se puede considerar al inventario como uno de los activos más importantes dentro de un negocio, ya que es de menester para estos mismos mantener una cantidad de productos en stock adecuado; es decir, que no existan ni muchos tipos de productos en stock que no se vendan y que tampoco exista una carencia de productos específicos por que se acaben rápido, así se cumple con una oferta y demanda correcta con el fin de no perder ingresos por ventas (Bassantes, 2020).

Ahora bien, una vez aclarados estos puntos, es más fácil entender la relevancia de realizar la automatización de un sistema de gestión de stock de productos en el Mini

Market “Paola”, por el motivo que son bastantes los beneficios que conlleva desarrollar dicho sistema e incluso la necesidad de contar con uno sin importar el tipo de negocio; por lo tanto, la automatización de esta gestión de stock de productos es factible e indispensable desde cualquier punto de vista.

1.2 Planteamiento del problema

En vista que el Mini Market no cuenta con un sistema informático, ha provocado algunas dificultades dentro del negocio, debido a que los registros de las ventas, el inventario de los productos, la lista de los proveedores y sus respectivos procesos se realizan de manera manual y en papel; por ende, no existe tanta agilidad, lo que conlleva a perder un tiempo considerable, por ejemplo, la propietaria no sabe con exactitud la disponibilidad de los productos y sus proveedores hasta el momento de revisarlo físicamente en el inventario del Mini Market, generando incertidumbre e impidiendo una toma de decisión rápida, y cuando se trata de hacer un conteo del stock de todos los productos, la propietaria suele estar un par de horas efectuando esta actividad, además, a veces suele haber una tardanza en el cobro al cliente, ya que la propietaria no sabe el precio de ciertos productos. Otro problema que se ha hallado en el negocio por parte de la propietaria es el registro de las ventas debido a que lo plasma en un cuaderno, lo cual suele ser una herramienta propensa a cometer errores e incluso hasta perderse o dañarse, ya que algunas veces los registros de los productos no cuadran, ya sea por una mala operación matemática como una mala suma o que simplemente la propietaria se olvidó o se le perdió ese registro de venta. Con todo lo mencionado anteriormente, se deduce que existe dificultades en la gestión del inventario y registro de ventas, por lo cual se genera varias deficiencias; por lo tanto, se ha identificado una necesidad latente a solucionar en

el negocio. Todos los inconvenientes descritos en el párrafo anterior se podrán solventar con la aplicación web propuesta en el presente documento.

Por tal motivo se presenta la siguiente pregunta principal del proyecto:

¿La aplicación web propuesta ayudará a la administradora a gestionar mejor el stock de los productos del negocio?

Y las siguientes preguntas secundarias:

¿Qué metodología se puede usar para realizar el desarrollo de una aplicación web de manejo de stock de productos?

¿Cuáles son los requerimientos funcionales y no funcionales para la aplicación?

¿Una aplicación web ayudará a la administradora a gestionar el stock de los productos del negocio

1.3 Objetivos

1.3.1 Objetivo General

- Desarrollar una aplicación web para la gestión del stock de productos

1.3.2 Objetivos Específicos

- Identificar la metodología que se pueda usar para realizar el desarrollo de una aplicación web de manejo de stock de productos.
- Determinar los requerimientos funcionales y no funcionales para la aplicación.
- Desarrollar una aplicación web para gestionar el stock de los productos del negocio.

1.4 Metodología

En el presente trabajo, cuyo fin es desarrollar un aplicativo web para la gestión de productos de stock el mini market “Paola”, se basará en los fundamentos de la metodología de la investigación aplicada, dado el caso que su objetivo es aplicar los conocimientos adquiridos a lo largo del tiempo en un caso práctico, el cual ayude a resolver un problema de un actor social. Por supuesto, tal metodología investigativa es oportuna debido a que está hecha para lidiar con un propósito inmediato.

Una vez aclaradas las principales características de la metodología, cabe destacar que resulta perfecta para el caso de estudio, por el hecho que se cuenta con un actor social que en este caso va a ser la administradora del mini market, se cuenta con un problema a solventar, mismo que es la gestión de los productos de stock, además, la resolución de dicha problemática se efectuará a través de los fundamentos teóricos necesarios para la comprensión de la problemática, así como también para basarse en la selección de la mejor solución para la problemática ya mencionada. Finalmente, se podrá aplicar los conocimientos obtenidos durante la carrera.

Otra razón por la cual esta metodología es la ideal para este trabajo es el tiempo con el que se cuenta, mismo tiempo el cual es corto; por ende, se necesita ser práctico para la resolución del problema detectado, esto quiere decir que se debe ser lo más concreto posible, tomando en cuenta que dicho trabajo se trata de la resolución de una problemática social real, la cual debe tener de manera obligatoria un análisis previo de las posibles soluciones con las cuales se puede proceder a tratar la problemática (Nicaragua, 2018).

1.5 Alcance

Este proyecto tiene como alcance el desarrollo de una aplicación web para la gestión del stock de productos del Mini Market “Paola”, ya que el negocio aún realiza sus actividades a mano y papel, además, la administradora del negocio ha manifestado su deseo por automatizar algunos de sus procesos para mejorar su agilidad y eficacia, dicho esto, este proyecto tiene como alcance la automatización de las siguientes actividades, tomadas en base a las necesidades y recomendaciones planteadas por parte de la administradora del negocio.

En primer lugar, se desarrollará un módulo para la gestión de productos, esto quiere decir que la administradora será capaz de ingresar, eliminar, modificar y consultar consultar de manera general un producto; por ende, la administradora tendrá la facilidad de consultar el precio para el cliente y la cantidad en stock desde el sistema y ya no manualmente como lo hace en la actualidad, lo cual resulta ser un problema por la demora.

En segundo lugar, se desarrollará un módulo donde la administradora pueda ingresar, eliminar, modificar y consultar de manera general si así lo quiere, todos los proveedores con sus respectivos atributos.

En tercer lugar, se desarrollará un módulo donde la administradora sea capaz de gestionar las ventas, esto quiere decir que la administradora será capaz de ingresar, eliminar, modificar y consultar de manera general una venta al cliente.

En cuarto lugar, también se desarrollará un módulo en donde al igual que el módulo anterior, la administradora será capaz de gestionar las compras, por ende, podrá ingresar, eliminar, modificar y consultar consultar de manera general una compra al proveedor.

Por último, el sistema contará con un módulo donde la administradora pueda ver sus cuentas totales por mes hechas automáticamente o dicho de otra manera tendrá un módulo donde pueda observar sus ganancias en base a los ingresos y egresos totales por mes tomando en cuenta los productos que se hayan vendido y adquirido, lo cual ayuda mucho ya que la administradora mencionó que hacer las cuentas le resulta ser una actividad tediosa y demorosa, debido a que muchas veces se equivoca en algunas operaciones matemáticas, además, algunos registros de ventas y compras se suelen perder o no suelen coincidir; por lo tanto, se considera un completo problema y augurio.

Capítulo 2: Marco Teórico

2.1 Metodologías

A continuación, se procederá a profundizar fundamentos, lineamientos, fases y conceptos de algunas de las metodologías más utilizadas para el desarrollo de software, tales como son:

- Metodología SCRUM
- Metodología Tradicional
- Metodología WSDM

2.1.1 Metodología SCRUM

La metodología ágil de desarrollo de software SCRUM es un modelo el cual se le puede sacar mucho provecho al momento de desarrollar un proyecto debido a que brinda las siguientes ventajas como: tener una alta garantía que al final del proyecto el cliente saldrá satisfecho, versátil a la hora de efectuar los cambios que vayan surgiendo durante el desarrollo, gracias a los sprints el cliente siempre está al tanto del avance y funcionalidades del proyecto e incluso puede aportar con lo que se espera con respecto a los futuros avances, no exige mucha documentación como en las metodologías tradicionales, lo cual muchas veces resulta más tedioso, voluminoso y demoroso, se hace controles con reuniones más cortas y objetivas, los proyectos efectuados con esta metodología tienen una alta probabilidad éxito, dado que se centra más en conseguir los objetivos y la existencia de un mayor compromiso por parte de los involucrados (Guevara & Nicole, 2020).

Ahora bien, la metodología ágil de desarrollo de software SCRUM tiene sus orígenes alrededor de 1986 por Hirotaka Takeuchi y Ikujiro Nonaka, dicha metodología nace en base a una estrategia del deporte Rugby, la cual consiste en que cada miembro del equipo aporta por igual tanto defensivamente como ofensivamente para llegar a su objetivo, el cual es anotar un punto, es por eso que se adoptó el término de SCRUM para nombrar a esta metodología de desarrollo de software, dado que sus principios se fundamentan en sacar el producto sea como sea y en donde cada integrante aporta con lo que pueda con el fin de sacar un proyecto adelante de manera conjunta. Al pasar los años, la metodología fue cada vez consolidándose más y más, gracias a diversos autores que empezaron a aplicar los fundamentos y lineamientos de SCRUM para sus proyectos entre ellos están Jeff Sutherland y Ken Schwaber quienes también son llamados padres de esta metodología. En consecuencia, a toda esta revuelta y novedad, se empiezan a publicar libros los cuales redactan experiencias, consejos y resultados de aplicar la metodología ágil SCRUM en proyectos, motivos por los cuales empieza a tomar fuerza, reputación y contar con una documentación basta hasta convertirse en lo hoy en día es una metodología certificada y una de las más utilizadas a nivel empresarial (Hernández & Beltrán, 2020).

En cuanto, al ciclo de vida de esta metodología, comprende de cinco fases las cuales son: inicio, planificación y estimación, implementación, retrospectiva y lanzamiento.

En este sentido, en la fase de inicio se abarca todo lo respecto al análisis, necesidad y factibilidad del proyecto, se crean los backlogs o también conocidos como una lista de requerimientos, se aborda la formación del equipo de trabajo, el cual puede estar conformado entre cuatro a diez integrantes, donde debe existir las siguientes cuatro

entidades las cuales son: SCRUM máster, product owner, desarrolladores y cliente. Para una mejor compresión de estas entidades, se podría decir que el SCRUM Máster es el intermediario entre el cliente y el equipo de trabajo, el cual debe hacer seguir al píe de la letra los lineamientos de la metodología y solventar las inquietudes con respecto a esta, asimismo, debe solventar las inquietudes del equipo en cuanto al producto con el fin que el equipo sea más productivo y así obtener un producto con más calidad, por otro lado, el product owner es el responsable de entender a la perfección el giro del negocio y el producto que el cliente quiere para mejorar su emprendimiento, por lo que esta entidad también es la responsable de monitorear el desarrollo del producto y dar el visto bueno para continuar con dicho proceso o de lo contrario deberá corregir y sugerir los cambios respectivos. Por último, se tiene a las entidades de los desarrolladores y el cliente, que no es complicado de entender su rol, ya que el cliente es la persona a la cual se le va a entregar el producto y los desarrollados son los encargados de la ejecución de la parte técnica de producto, dicho de otro modo, son los que codifican, crean funcionalidades tanto en la parte de front-end como back-end, incluso hacen la conexión con la base de datos.

Continuando con las fases de la metodología SCRUM, se tiene la fase de planificación y estimación, fase en la cual crean y se estiman los sprint backlog, que es más que el conjunto de backlogs tomados de la fase anterior que se deben de desarrollar en un lapso de una a cuatro semanas. Después, se tiene la fase de implementación en donde se procederá a cumplir con los entregables y hacer las respectivas correcciones si son necesarias, esto se controla a través de dos tipos de reuniones, una conocida como daily stand-up, la cual consiste en hacer reuniones diarias cortas que no duren más de

quince minutos para ver el avance correspondiente y haciendo preguntas clave acerca de lo que se hizo el día anterior, el presente día y que se pretende hacer el siguiente día, también hay un espacio por si han existido dificultados o confusiones, la segunda reunión es el sprint, en donde se reúnen todas las entidades para la presentación de un entregable al cabo de entre una a cuatro semanas, posteriormente el cliente da sus observaciones las cuales deberán ser tomadas en cuenta como un añadido para el próximo sprint. Como cuarta fase se tiene la de retrospectiva, la cual no es más que una autoevaluación y una coevaluación como equipo con respecto al trabajo realizado, haciendo hincapié en las críticas constructivas y posibles soluciones para estas mismas. Por último, se tiene la fase de lanzamiento, aquí finalmente se entrega todo el producto ya desarrollado en base a lo acordado en los sprints anteriores, donde por supuesto se evidencie las correcciones propuestas por el cliente (Bautista, 2021).

En cuanto a casos favorables de empresas famosas e importantes que han utilizado SCRUM como metodología para desarrollar sus proyectos, se puede nombrar algunas tales como son: Amazon y Spotify. Por ejemplo, Amazon asegura que su enorme agilidad, flexibilidad y crecimiento empresarial es gracias a lo que SCRUM ofrece, dando aún más mérito a SCRUM, dado que Amazon ya contaba antes con metodologías agiles; sin embargo, no obtenían los mismos resultados. Amazon destaca de SCRUM la autonomía y empoderamiento que ofrece a sus equipos de trabajo, sin mencionar, la filosofía de contar con personal cuyas características sean ser siempre activos, participativos y no abstenerse al cambio.

Por otra parte, Spotify la cual es una empresa que suele tener cambios frecuentes, ya sea para eliminar, añadir o modificar funcionalidades, necesita de una agilidad enorme. Por tal motivo, SCRUM ha ofrecido una solución acorde a la problemática que tiene

dicha empresa, puesto que cada persona cuenta con habilidades particulares y complementarias a la de su grupo de compañeros de trabajo, lo cual ayuda con una mejor perspectiva para solventar la tarea que se quiera ejecutar, además, los grupos de trabajo no suelen ser más de diez personas, por lo que el grupo está obligado a ser participativo y comprometido con lo encomendado.

Cabe mencionar que estas empresas han optado por SCRUM por la capacidad que tiene de sacar productos al mercado en un corto tiempo, destacando a esta misma por la efectividad y productividad que ofrece, incluso hasta haciendo énfasis en lo multidisciplinarios que pueden llegar a ser sus trabajadores durante el desarrollo del proyecto (Sosa, 2020).

2.1.2 Metodología Tradicional

En cuanto a la metodología de desarrollo de software tradicional en cascada es un modelo el cual se le pude sacar mucho provecho al momento de desarrollar un proyecto debido a que brinda las siguientes virtudes como: una documentación basta, secuencialidad, organización, exigencia, claridad, detalle, ideal para proyectos pequeños o medianos, versátil a adoptar diferentes metodologías en cada una de sus etapas y segura al mitigar errores al final de cada etapa.

Abordando un poco más acerca de la historia de esta metodología, se puede decir que tiene sus orígenes alrededor de 1970 por Winston W. Royce para proyectos relacionados con la ingeniería civil, llegando a considerarse como una metodología organizada y rigurosa, en donde rige una secuencialidad de etapas que deben ser

completadas una por una para poder continuar con la siguiente, estas etapas son: análisis, diseño, desarrollo, implantación, pruebas y mantenimiento (Hadida & Troilo, 2020).

Es así como en la etapa de análisis se entiende el producto a realizar y se da el consentimiento del cliente, en la etapa de diseño se describe la manera en qué este va a ser realizado y como va a funcionar, en la etapa de desarrollo se ejecuta el plan para el respectivo desarrollo del producto acordado, la etapa de implantación viene a ser la capacitación al cliente e incorporación del producto en el negocio del cliente para ponerla en marcha correctamente, la etapa de pruebas es donde se examina si el producto tiene errores o vulnerabilidades que deben ser solventadas y por último se tiene la etapa de mantenimiento, la cual es la responsable de dar soporte, mejoras y cambios en el futuro, si es que así lo amerita el producto (Aguirre & Aguirre, 2020).

Una de las ventajas de trabajar con esta metodología es lo sencillo de implementarla, así como entender sus etapas secuenciales; además, asegura una gran cantidad de documentación en donde ampararse por si hay algún tipo de desacuerdo o conflicto con el cliente al momento de presentar el producto final. Adicionalmente, dicha metodología es la más certera y recomendable cuando se trata de proyectos pequeños o medianos, asimismo es sencillo constatar cuanto progreso se ha logrado en el proyecto por su secuencialidad, de igual forma es fácil evitar imprevistos que prolonguen el proyecto, ya sea por alguna disconformidad del cliente o algún nuevo requerimiento, esto es posible debido a que en la etapa de análisis se acuerda todo con respecto al producto final y los entregables que este mismo conlleve a detalle, incluyendo también el camino a seguir (Aguirre & Aguirre, 2020).

Dado que los negocios a menudo necesitan adaptarse rápido al cambio y por ende automatizar sus procesos para mantenerse en el mercado, es necesario meditar sobre que metodología es la adecuada para abordar la automatización de una área, módulo o proceso, ya como puede ser el caso de la automatización de un área contable, administrativa, transaccional, sistema de stock, etc. Para cada caso, se debe estudiar la metodología adecuada, su impacto, su ciclo de vida y su proyección a futuro; es decir, si va a ser un sistema en cambio constante o estático, escalable o fijo, entre otros factores, es de menester realizar un análisis de la situación previamente antes de abordar una metodología para un proyecto de manera negligente.

Por el motivo que el proyecto se trata de un aplicativo web para la gestión del stock de productos de un mini market, vale la pena analizar la metodología de software tradicional en cascada, ya que se trata de un proyecto pequeño pero necesario, el cual demanda requerimientos concisos en un lapso corto de tiempo, al igual que no es susceptible a cambios de último momento y posee una documentación basta como detallada.

Como se mencionó anteriormente, el desarrollo de un producto con esta metodología resulta ser intenso y estricto; sin embargo, no quiere decir que no se pueda mejorar esta situación e incluso hasta existen métodos que ayudan a agilizar esta metodología; no obstante, no se debe olvidar que los lineamientos de esta metodología exige un desarrollo por fases y en cada una de ellas se debe cumplir de carácter obligatorio las metas que se hayan establecido para cada una de estas fases respectivamente (González et al., 2019).

2.1.3 Metodología WSDM

A continuación, se hablará de la metodología de desarrollo web WSDM o también conocida por sus siglas en inglés Web Site Design Method, tal metodología apareció alrededor del año 1998 y fue creada específicamente para satisfacer las necesidades del desarrollo de aplicativos webs, dado que las metodologías tradicionales no lo hacían del todo bien.

Es por eso que toma fuerza la relevancia de usar WSDM en el sector de desarrollo web, ya que incluso brinda algunos otros aspectos que resultan beneficiosos para un desarrollo, tal y como es la facilidad de sacar productos en un corto lapso de tiempo, dado que se centra en complacer las necesidades expuestas por el cliente y por ello se utiliza técnicas que ayuden a alcanzar un mayor nivel de comprensión con respecto al problema, para así poder obtener de una manera más adecuada los requerimientos del cliente e interpretar de una forma más certera acerca de lo que el cliente espera del producto final.

Además, ofrece un alto detalle en cuanto a los tipos de usuario y sus roles, así como también sus limitaciones y rutas que pueden ejercer al utilizar la aplicación web, todo esto se lo refleja en los diagramas de navegación que se expone al cliente previamente y por supuesto también se lo presenta junto al modelo conceptual de entidad relación para evitar malentendidos o un desarrollo erróneo en el futuro (Menacho, 2021).

Seguidamente, se puede mencionar que esta metodología contempla cuatro fases en su ciclo de vida, las cuales son:

- Modelado de usuarios
- Diseño Conceptual
- Diseño de implementación

- Implementación

Para empezar, se va a hablar de la fase de modelo de usuarios, esta fase pretende identificar y clasificar todos los usuarios que utilizaran el aplicativo web, una vez ejecutada esta subfase se procede a describir el rol de cada usuario, esto quiere decir que se detalla lo que el usuario hace en la empresa y sus limitaciones; cabe destacar que para completar estas subfases, se puede efectuar ciertas preguntas de la razón de ser de la empresa y su manera de actuar, con el propósito de identificar y entender un poco más los usuarios existentes y sus roles.

La segunda fase es la del diseño conceptual, esta fase consiste en presentar un modelo conceptual de entidad relación, debido que aquí se puede evidenciar un poco más como los diferentes tipos de usuarios y objetos interactúan entre sí y también se contempla los tipos de datos que se van a recolectar, como fase subsiguiente se tiene la realización del diseño navegacional, en el cual se detalla la perspectiva de navegación por el aplicativo web de cada usuario, visualizando los accesos y las distintas actividades que pueden efectuar en el sistema, dependiendo el rol que tengan.

La tercera fase corresponde al diseño de implementación, en esta fase se crea los prototipos de interfaces para cada actividad posible en el sistema y dependiendo el tipo de usuario que sea, ya que cada usuario tiene una navegación diferente por el sistema, cabe mencionar que estos prototipos de interfaz tienen que tener de manera obligatoria un aspecto agradable para el cliente, y por supuesto se tiene que tener en cuenta las restricciones y funcionalidades que se hayan delimitado en las anteriores fases.

Para finalizar, se tiene la fase de implementación en donde una vez ya se haya escogido de manera objetiva el entorno de desarrollo y la arquitectura, se procede a codificar las funcionalidades acordadas y sus interfaces. Por consiguiente, una vez se

haya finalizado la codificación se procede a efectuar las pruebas correspondientes del sistema con el cliente para verificar que todo esté funcionando correctamente y constatar el cumplimiento de lo que se había acordado (Ríos et al., 2018).

2.2 Herramientas

Debido a que el proyecto se va a efectuar en un ambiente de un aplicativo web para la gestión del stock de productos del Mini Market “Paola”, se ha optado por poner a consideración las siguientes herramientas para el respectivo desarrollo en todo lo que respecta a bases de datos, modelamiento conceptual y navegacional, editores de código y frameworks tanto de front-end como de back-end, tales como son:

- Angular
- React
- MySQL
- PostgreSQL
- Visual Studio Code
- Notepad++
- Spring Boot
- Laravel
- Power Designer
- Creately

Para comprender un poco más acerca del funcionamiento, la importancia y las modalidades de estas herramientas es indispensable aclarar los conceptos de cada uno.

2.2.1 Angular

Para empezar, se puso en consideración el desarrollo del aplicativo web a partir del framework de Angular, debido a que tiene ventaja con respecto a otros frameworks dado que sus tiempos de respuesta son óptimos, tiene un alto rendimiento para proyectos de esta índole, y cuenta con la posibilidad de aplicar los conceptos que son mucha importancia en una aplicativo, tales como son el mantenimiento, la portabilidad y la usabilidad.

Ahora bien, lo que más aporta Angular al proyecto es la facilidad de reutilizar código, ya que cuenta con un estándar de componentes web; no obstante, la ventaja mencionada no es la única que ayuda a la hora de desarrollar, debido a que su principal lenguaje de programación es TypeScript, lo cual ayuda con respecto a todo lo que se haga en el código sea de una manera coherente y consistente, así como también su sintaxis. Por supuesto, el editor de código de Visual Studio Code ya cuenta con una forma fácil y rápida de instalar angular como una de sus extensiones (Espinoza, 2019).

2.2.2 React

React es una librería de código abierto escrita en JavaScript, la cual ayuda en cuanto a lo que es front-end, por la sencilla razón de contar con componentes que sean interactivos y que puedan ser reutilizados en cualquier momento. Esta librería fue desarrollada por Facebook y tiene sus inicios alrededor del 2013, es así como varias empresas populares como Instagram, Uber y Netflix han ido desarrollando sus interfaces con React, ya que esta librería tiene como objetivo desarrollar todo en una sola página lo cual resulta más cómodo para el usuario.

Una de las particularidades que ofrece React a la hora de trabajar es su desarrollo por módulos, por ello sus actualizaciones son más fáciles y rentables cuando las hay, igualmente ofrece uno de los rendimientos más altos en desarrollo, rentable para desarrollo de proyectos en corto tiempo y por supuesto, una de sus más grandes ventajas como se dijo anteriormente es su código abierto (Boduch, 2020).

2.2.3 MySQL

Por consiguiente, para el tema de la gestión de los datos en este proyecto se ha puesto en consideración la base de datos MySQL, dado que al ser SQL un lenguaje estandarizado, se ofrece una gran compatibilidad al momento de migrar datos, también existe bastante documentación y soporte con respecto a esta base de datos, asimismo también es de código abierto y extendido, dicha base de datos también maneja versiones gratuitas y por último se puede adaptar fácilmente a cualquier tipo de negocio pequeño o mediano dado que es una base de datos relacional, además controla de manera adecuada temas de seguridad como son los accesos y permisos (Sanchez, 2020).

2.2.4 PostgreSQL

PostgreSQL es un gestor de bases de datos relacional que se rige a los lineamientos de POO (programación orientada a objetos), cabe destacar que es de código abierto, dado que su mantenimiento y desarrollo es en mayor parte por desarrolladores independientes.

Entre una de sus grandes ventajas se debe tomar en cuenta la robustez que tiene en cuanto a documentación y soporte, sin mencionar que cuenta con una compatibilidad

amplia para muchos sistemas operativos. Además, es una de las bases de datos más seguras y confiables, ya sea por ataques externos o caídas de sistema. Por último, se puede destacar su interfaz gráfica amigable al usuario como es el pgAdmin (Barahona & Rebutti, 2019).

2.2.5 Visual Studio Code

En cuanto, al editor de código se ha tomado en cuenta Visual Studio Code, debido a que es una herramienta compatible con bastantes lenguajes de programación, asimismo como se trata de un desarrollo web, tiene más importancia dada la razón que trabaja con HTML, CSS, JavaScript y TypeScript; por ende, es sin duda una opción factible para el desarrollo del proyecto, puesto que es compatible con lenguajes y herramientas que se va a utilizar, asimismo su instalación es casi inmediata, así mismo ofrece una facilidad de agregar extensiones sin salir de Visual Studio Code e incluso cuenta con un gran conjunto de atajos al código, lo cual suele hacer que el desarrollador sea más eficaz y eficiente a la hora de desarrollar, además de que se lleva una gran experiencia de usuario. Adicionalmente Visual Studio Code cuenta con bastante soporte para los problemas durante el desarrollo, ya sea mediante foros, videos o documentación explicativa en diferentes páginas web debido a que gran parte de la comunidad de desarrolladores a nivel mundial lo usan (Tubay, 2019).

2.2.6 Notepad++

Notepad++ es un editor de texto, cuyo código fuente es libre y al igual que Visual Studio Code soporta varios lenguajes de programación como PHP, CSS, HTML, JavaScript y TypeScript; sin embargo, a pesar de ser de código libre y gratis, solo tiene soporte en Microsoft y su interfaz es precaria, aunque tenga la posibilidad de personalizarla (Escobar, 2019).

2.2.7 Spring Boot

Spring Boot es un framework diseñado especialmente para el desarrollo del back-end de una aplicación, cabe destacar que es de código abierto y trabaja con el lenguaje de programación de Java. Ahora bien, una de las principales causas por las que este marco de trabajo toma fuerza es por la eficiente capacidad de trabajar con arquitecturas para aplicaciones web, tales como son las arquitecturas SOAP y REST, es así como dicho framework ofrece mucha ayuda a la hora de desarrollar servicios, microservicios, establecer configuraciones, dependencias, crear APIs, reutilizar código, etc.

Por otro lado, Spring Boot también tiene una gran compatibilidad con diferentes IDEs, frameworks, herramientas y bases de datos, incluso existe una variedad de documentación y soporte para este marco de trabajo, ya sea en videos, páginas web o foros debido a la gran demanda que ha estado teniendo estos últimos años (Tito, 2020).

2.2.8 Laravel

Laravel es un marco de trabajo de código abierto creado en el 2011, el cual está especializado en el desarrollo de aplicaciones web, especialmente para el back-end con PHP, la finalidad de Laravel es desarrollar un código que sea simple y funcional. Trabaja mejor en arquitecturas MVC o también conocidas como arquitectura modelo, vista y controlador, en cuanto a su seguridad es bastante alta ya que cuenta con algoritmos de encriptación.

Siguiendo con la moda de las herramientas de desarrollo de la actualidad, permite una migración de bases de datos sencilla al igual que la modificación de código en cuanto programación, cabe mencionar que cuenta con mucha documentación y soporte actualmente, incluso hasta empresas grandes como Pfizer lo utilizan (Cíceri, 2019)

2.2.9 Power Designer

Power designer es una herramienta desarrollada por Sybase, la cual está especializada en el modelado empresarial; sin embargo, también contempla el modelado para lo que es bases de datos, ya como son los modelos conceptuales, físicos y lógicos. También cuenta con la posibilidad de elaborar casos de uso, diagramas secuenciales y diseños navegacionales, es compatible con varios sistemas operativos entre ellos Windows.

Una de las muchas ventajas de trabajar con power designer es el soporte que ofrece en cuanto a problemas o inquietudes que se susciten al utilizar esta herramienta y sus funcionalidades, ya como se mencionó anteriormente esta herramienta es

polifuncional y cuenta con técnicas de análisis y diseño bastante robustas y prácticas, aparte su interfaz gráfica es bastante intuitiva y amigable (Cerezo, 2019).

2.2.10 Creately

Creately es una herramienta la cual se dedica a la diagramación, contempla lo que son los diagramas UML, por lo cual se puede elaborar diagramas de flujo, secuenciales, navegacionales, clases, componentes, actividades, casos de uso, modelos conceptuales, físicos y lógicos, cabe mencionar que con esta herramienta se puede trabajar en línea y en una aplicación escritorio.

La herramienta es compatible con los sistemas operativos más conocidos como son Windows, Mac y Linux, por otro lado, creately también cuenta con una interfaz bastante simple pero a la vez eficaz, por lo cual es fácil familiarizarse con esta herramienta (Salazar, & Costaguta, s.f.).

2.3 Arquitecturas

2.3.1 Arquitectura REST

Las modalidades y lineamientos de la arquitectura REST o también llamada Transferencia de Estado Representacional están estrictamente relacionadas con el desarrollo web, por la razón que emplea el estándar HTTP para ejecución de sus métodos, usando diferentes formatos de mensajes como puede ser: JSON, XML, texto plano y entre otros, de igual manera REST también permite trabajar con cualquier lenguaje de programación ofreciendo un gran nivel de flexibilidad e incluso es capaz de reducir los tiempos de respuesta entre cliente-servidor por lo que mejora la experiencia de usuario, esto es posible dado la independencia entre cliente-servidor que ofrece la arquitectura debido a que generalmente solo se comunican mediante un JSON, cabe mencionar que para este tipo de arquitectura se debe separar el desarrollo del software en back-end y front-end con el fin de tener un código más ordenado y con mejores resultados.

Por otro lado, también se puede destacar las ventajas que existen al haber una independencia entre el cliente y el servidor, ya que por parte del cliente no le interesa la manera en que la API fue hecha y por parte del servidor no hay ningún cuestionamiento acerca de lo que se va a hacer con los datos obtenidos. Igualmente, resulta ser una arquitectura con un gran potencial en cuanto a la estabilidad, flexibilidad y fiabilidad, dado que se puede cambiar las herramientas con las que se trabaja para desarrollar el software, siempre y cuando las salidas de datos sean las mismas; es decir, que no se vean alteradas. Por consiguiente, se debe tomar en cuenta que aquí el front-end y el back-end no están mezclados del todo, lo cual permite que la carga en los servidores sea más baja en comparación a otras arquitectura porque es en el lado del cliente donde se envía el estado al servidor; por ende, no se tiene que atender todas las solicitudes en el servidor,

es por eso que se obtiene una mejor experiencia de usuario, dado que las respuestas a las peticiones son más rápidas y no existen mucho colapsos en el sistema (Cabrera & Navarro, 2020).

2.3.2 Arquitectura SOA

Los lineamientos de la arquitectura SOA o también llamada Arquitectura Orientada a Servicios, radica en la reutilización de sus componentes, de modo que el desarrollo de un software a través de esta modalidad garantiza que este mismo sea flexible y versátil por si en un futuro se necesite modificar, sobre todo aporta a las empresas por el hecho de crear servicios en menos tiempo y a un menor costo. Además, se puede hacer hincapié en el alto rendimiento que este desencadena, gracias a la manera en la cual los servicios interactúan con otros componentes y si no fuera poco, también interactúan con gran eficacia y eficiencia con componentes de terceros. Ahora bien, puesto que SOA integra todos sus componentes, facilita la posibilidad de que estos puedan trabajar en conjunto y que se comuniquen las veces que quieran de manera rápida, misma característica que es de menester para la creación de aplicaciones en diferentes sistemas, también cabe mencionar lo práctico que resulta ser esta manera de trabajar, dado que la depuración se hace por servicio; por ende, es rápido y sencillo, incluso si se necesita hacer una modificación o actualización se la puede hacer, ya que como cada servicio es independiente a otro, no se tiene errores colaterales.

Por último, se puede resaltar lo sencillo que resulta hacer un rediseño para esta arquitectura por la exigencia que esta misma tiene para mostrar con la mayor simplicidad sus procesos, lo cual es importante tomando en cuenta lo cambiantes que son los mercados hoy en día; no obstante, si algún servicio llega a fallar, la arquitectura SOA es

vulnerable a tener una falta de disponibilidad en todo su sistema hasta incluso tener una inconsistencia en su salida de datos, afectando directamente a la confiabilidad del sistema por la falta de calidad que este demuestra, de igual manera, si varios equipos intentan compartir un mismo recurso al mismo tiempo, existe una gran probabilidad que este mismo no esté disponible por la gran carga de procesamiento que se le exige (Cobo, & Alulema, 2021).

Capítulo 3: Caso de estudio: Desarrollar una aplicación para gestionar el stock de los productos del negocio

3.1 Determinación de la metodología que se pueda usar para realizar el desarrollo de una aplicación web de gestión de stock de productos

A continuación, se va a determinar la metodología más adecuada para el desarrollo de una aplicación web de gestión de stock de productos mediante el uso de dos tablas, la primera tabla tendrá parámetros que serán medidos de forma cualitativa y la segunda tabla tendrá parámetros que serán medidos de forma cuantitativa; por lo tanto, en la tabla cualitativa se compararan distintos parámetros para posteriormente hacer un análisis en base a la información obtenida en esta misma, por lo cual se tomará una decisión en base a que tan rentable resulta dicha metodología según el tiempo del proyecto, la necesidad real del mini market "Paola" y según el número de personas que van a desarrollar el producto. La segunda tabla contemplará parámetros los cuales se puedan cuantificar del 1 al 5, al final la metodología que más puntos sume y que más viable analíticamente sea, será la elegida para la ejecución del proyecto, entre las metodologías a comparar se tiene a la metodología SCRUM, la metodología tradicional y la metodología WSDM.

Ahora bien, como se puede evidenciar en la Tabla 1 se tiene algunos parámetros que se deben tomar en cuenta a la hora de desarrollar un proyecto con tal metodología y cuyas características pueden ser más convenientes dependiendo el tipo de proyecto.

Tabla 1

Comparativa cualitativa de metodologías de desarrollo de software

Metodología	SCRUM	Tradicional	WSDM
Técnica de modelado	Entidad-Relación	Entidad-Relación	Entidad-Relación
Representación gráfica	- Modelo conceptual - Diagramas UML, dependiendo los que se consideren necesarios	- Modelo conceptual - Diagrama de clases - Diagrama secuencial - Casos de uso	- Modelo conceptual - Diseño navegacional
Roles	- SCRUM Máster - Producto Owner - Desarrolladores - Cliente	No especifica, puede ser de una persona en adelante dependiendo el proyecto	No especifica, puede ser de una persona en adelante dependiendo el proyecto
Personalización	Alta	Media	Alta
Etapas	- Inicio - Planificación y estimación - Implementación - Retrospectiva - Lanzamiento	- Análisis - Diseño - Desarrollo - Implantación - Pruebas - Mantenimiento	- Modelado de usuarios - Diseño Conceptual - Diseño de implementación - Implementación

Nota. Autor: Ariel Ortega, 2022

Tal y como se observa en la Tabla 1 de la compararativa cualitativa de las metodologías para el desarrollo de software, se contempla que todas tienen la misma técnica de modelado, en cuanto a la representación gráfica se tiene que todas las metodologías usan el modelo conceptual; sin embargo, la SCRUM solo contempla los diagramas UML que considere necesarios, la tradicional contempla muchos diagramas a realizar y la WSDM solo considera necesario un diseño navegacional. Por otro lado, en los roles se estipula que para desarrollar la metodología SCRUM se necesita de al menos tres personas sin contar el cliente; no obstante, en la tradicional y en la WSDM no tienen roles y los responsables de asumir el proyecto pueden ser desde una persona en adelante dependiendo la magnitud de este desarrollo. En cuanto a la personalización del producto, se tiene que en la metodología SCRUM y WSDM son altas debido a su frecuente interacción con el cliente, pero en la tradicional es media por el motivo que hay mucha documentación y las etapas donde el cliente es protagonista del proyecto son pocas. En cuanto a las etapas, se tiene que en la metodología SCRUM existen cinco etapas, en la tradicional existen seis etapas y en la WSDM se cuenta con cuatro etapas.

Por último, se tiene la Tabla 2 en donde se muestran parámetros, los cuales determinan que tan apropiados y confiables son para el proyecto mediante valores del 1 al 5, siendo 1 el más bajo y 5 el más alto.

Tabla 2

Comparativa cuantitativa de metodologías de desarrollo de software

Metodología	SCRUM	Tradicional	WSDM
Fiabilidad	4	3	5
Usabilidad	4	3	4
Agilidad	5	2	5
Desarrollo Web	3	3	5
Curva de aprendizaje	4	5	4
Total	20	16	23

Nota. Autor: Ariel Ortega, 2022.

Como se muestra en la Tabla 1, la metodología que obtuvo la calificación más alta tomando como parámetros de desarrollo la fiabilidad, usabilidad, agilidad, desarrollo web y curva de aprendizaje es la WSDM con 23 puntos, mientras que la SCRUM y la tradicional obtuvieron 20 y 16 puntos respectivamente.

En conclusión, la metodología de desarrollo que se utilizará para el desarrollo de la aplicación web de gestión de stock de productos del mini market “Paula” en base a la comparativa de la Tabla 1 y la Tabla 2 es la WSDM, ya que es la que más se adapta con el proyecto, dada la razón que es una metodología exclusivamente para el desarrollo web, mientras que la tradicional y la SCRUM no lo son del todo, además con respecto a la documentación solo pide la necesaria mientras que la tradicional pide demasiada, lo cual no es oportuno para el poco tiempo que se tiene, además, los roles para la WSDM no se especifican y se puede tomar la responsabilidad una sola persona, mientras que en la SCRUM se necesitan al menos tres personas a parte del cliente, lo cual es inadecuado ya que en este proyecto solo intervendrá una persona en el desarrollo del proyecto, así mismo la personalización de la aplicación en WSDM es alta y contiene menos etapas que las demás metodologías, lo cual es bueno ya que como se dijo previamente no se cuenta con tanto tiempo. Para finalizar, se evidencio que la metodología WSDM es más beneficiosa para el proyecto con respecto a la fiabilidad, usabilidad y agilidad que se necesita, cabe recalcar que esta metodología es sencilla de comprenderla y desarrollarla.

3.2 Levantamiento de los requerimientos funcionales y no funcionales para la aplicación

Para el levantamiento de requerimientos funcionales y no funcionales de este proyecto, se va a basar en algunas pautas que propone el estándar de IEEE 830 que está orientada a la especificación de requisitos.

3.2.1 Requerimientos Funcionales

Para levantar los requerimientos funcionales se ha tenido reuniones con la administradora del mini market "Paola", a partir de estas mismas reuniones se ha comprendido la problemática y la necesidad real que existe para desarrollar la aplicación web; además, como evidencia de las reuniones que se han tenido, se ha generado una bitácora (anexo 1); asimismo, a partir de estas reuniones se ha conocido un poco más el negocio y su manera de organizarse, de esta forma se ha levantado los requerimientos funcionales y llegando a un acuerdo por ambas partes.

Ahora bien, como ya se mencionó anteriormente solo se va a utilizar algunas pautas del estándar IEEE 830 para la especificación de requisitos; por lo tanto, para levantar los requerimientos funcionales solo se va a tomar en cuenta las funcionalidades que va a tener el producto, los requisitos específicos y los casos de uso a nivel general y siguiente nivel, mas no los casos de uso a detalle.

3.2.1.1 Funcionalidades del producto

Las principales funcionalidades a desarrollar, según lo acordado con el cliente son las siguientes:

F0: Ingresar a la aplicación

F1: Gestión de productos

F1.1: Ingresar producto

F1.2: Eliminar producto

F1.3: Modificar producto

F1.4: Consultar producto

F2: Gestión de proveedores

F2.1: Ingresar proveedor

F2.2: Eliminar proveedor

F2.3: Modificar proveedor

F2.4: Consultar proveedor

F3: Gestión de ventas

F3.1: Ingresar venta

F3.2: Eliminar venta

F3.3: Modificar venta

F3.4: Consultar venta

F4: Gestión de compras

F4.1: Ingresar compra

F4.2: Eliminar compra

F4.3: Modificar compra

F4.4: Consultar compra

F5: Calcular ganancias

F5.1: Elegir fecha

F6: Salir de la aplicación

3.2.1.2 Requisitos específicos

Tabla 3

Requisito específico F0

Número de requisito	F0
Nombre del requisito	Ingresar a la aplicación
Fuente de requisito	Se pide el usuario y contraseña para poder acceder e interactuar con la aplicación web
Prioridad del requisito	Media

Nota. Autor: Ariel Ortega, 2022

Tabla 4

Requisito específico F1

Número de requisito	F1
Nombre del requisito	Gestión de productos
Fuente de requisito	Se le permite a la administradora ingresar, eliminar, modificar y consultar los productos que desee
Prioridad del requisito	Alta

Nota. Autor: Ariel Ortega, 2022

Tabla 5

Requisito específico F2

Número de requisito	F2
Nombre del requisito	Gestión de proveedores
Fuente de requisito	Se le permite a la administradora ingresar, eliminar, modificar y consultar los proveedores que desee
Prioridad del requisito	Alta

Nota. Autor: Ariel Ortega, 2022

Tabla 6

Requisito específico F3

Número de requisito	F3
Nombre del requisito	Gestión de ventas
Fuente de requisito	Se le permite a la administradora ingresar, eliminar, modificar y consultar las ventas que ha hecho
Prioridad del requisito	Alta

Nota. Autor: Ariel Ortega, 2022

Tabla 7

Requisito específico F4

Número de requisito	F4
Nombre del requisito	Gestión de compras
Fuente de requisito	Se le permite a la administradora ingresar, eliminar, modificar y consultar las compras que ha hecho
Prioridad del requisito	Alta

Nota. Autor: Ariel Ortega, 2022

Tabla 8

Requisito específico F5

Número de requisito	F5
Nombre del requisito	Ingresos por mes
Fuente de requisito	Se le permite a la administradora observar los ingresos totales por mes
Prioridad del requisito	Media

Nota. Autor: Ariel Ortega, 2022

Tabla 9

Requisito específico F6	
Número de requisito	F6
Nombre del requisito	Salir de la aplicación
Fuente de requisito	Permite a la administradora salir de la aplicación
Prioridad del requisito	Media

Nota. Autor: Ariel Ortega, 2022.

3.2.1.3 Casos de uso

A continuación, se va a realizar los casos de uso correspondientes para el desarrollo del aplicativo web de gestión de stock de productos del mini market “Paola”; por ende, se realizará el caso de uso a nivel general y los casos de uso al siguiente nivel; sin embargo, no se hará los casos de uso a detalle, dado que la metodología de desarrollo web WSDM ya contiene otros parámetros para efectuar un desarrollo eficaz.

Gráfico 1

Caso de uso general

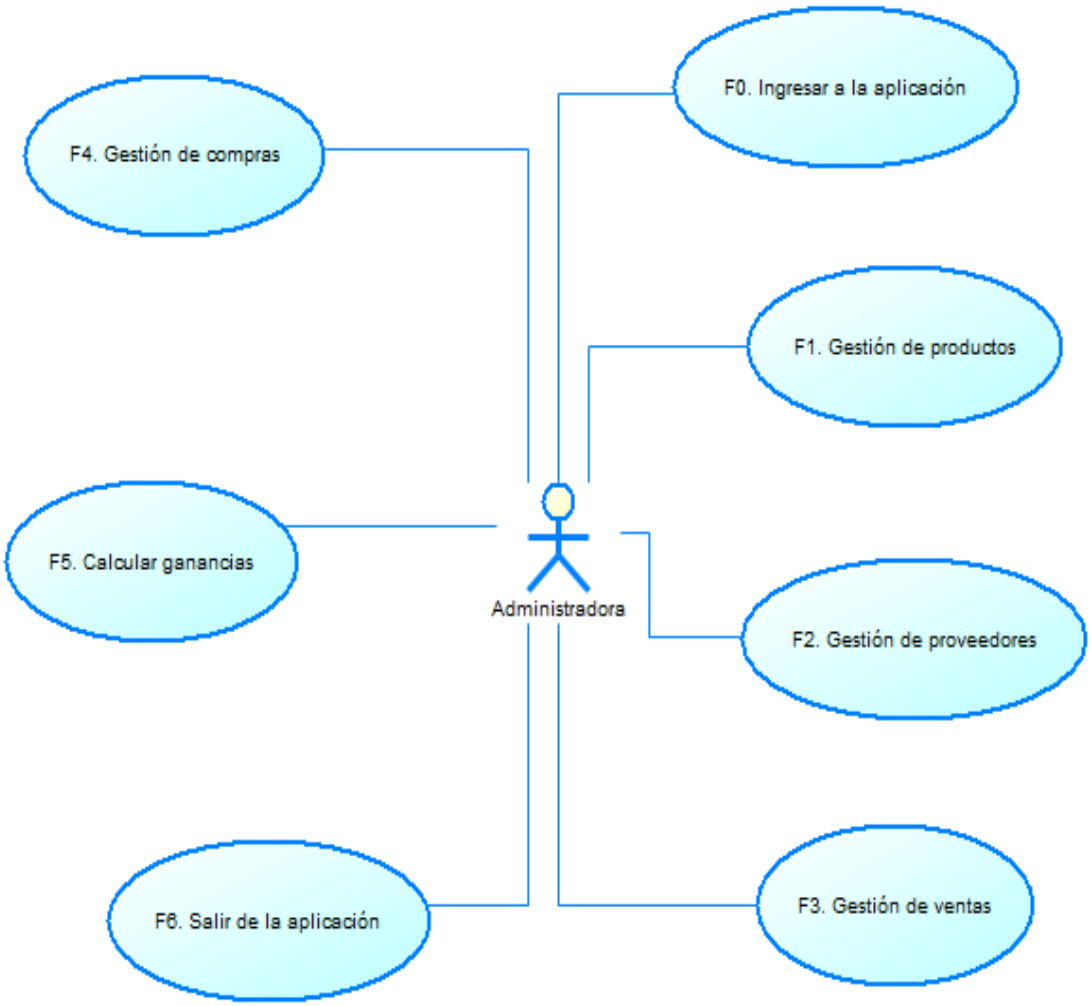


Gráfico 2

Caso de uso al siguiente nivel: F1. Gestión de productos

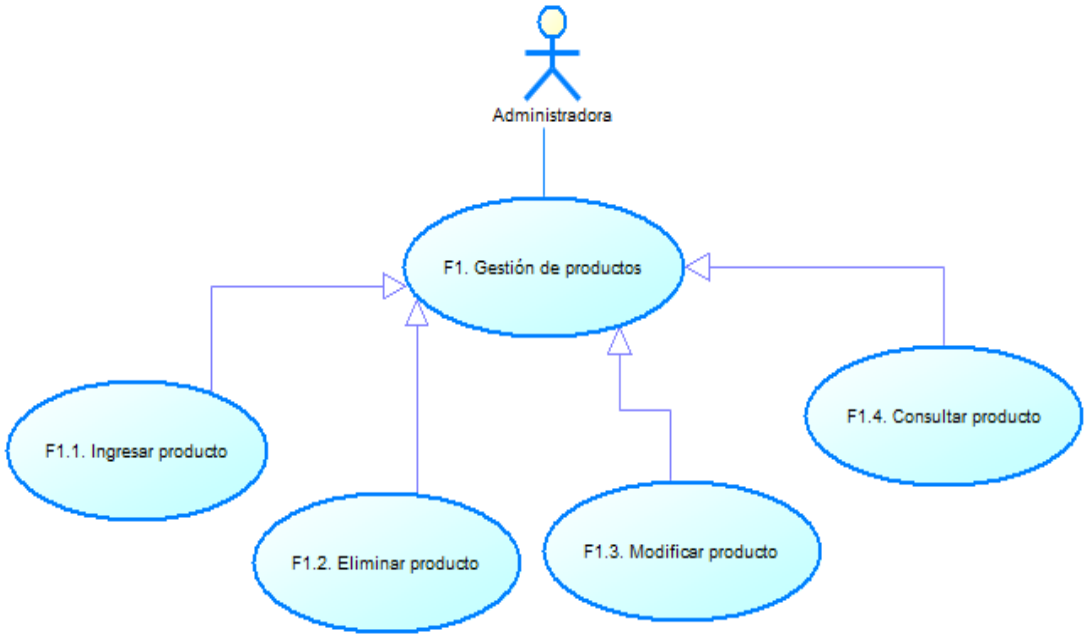


Gráfico 3

Caso de uso al siguiente nivel: F2. Gestión de proveedores

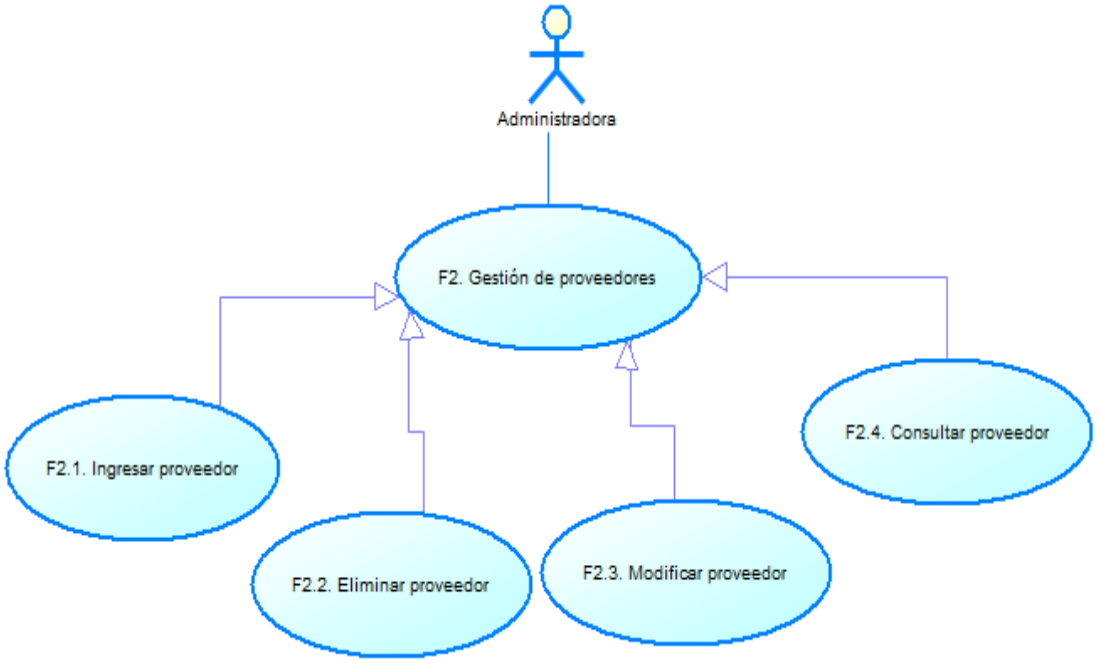


Gráfico 4

Caso de uso al siguiente nivel: F3. Gestión de ventas

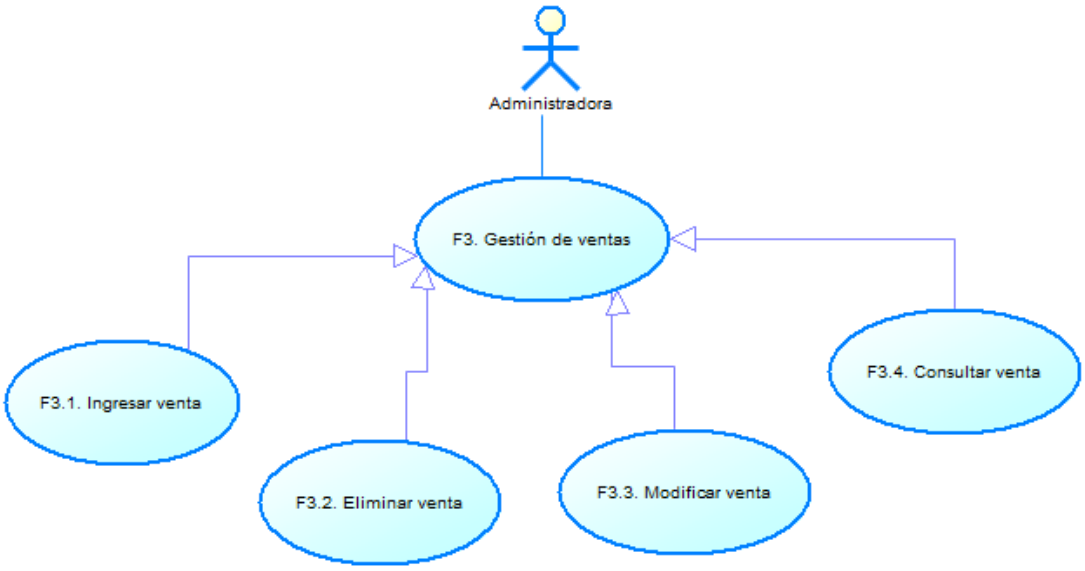


Gráfico 5

Caso de uso al siguiente nivel: F4. Gestión de compras

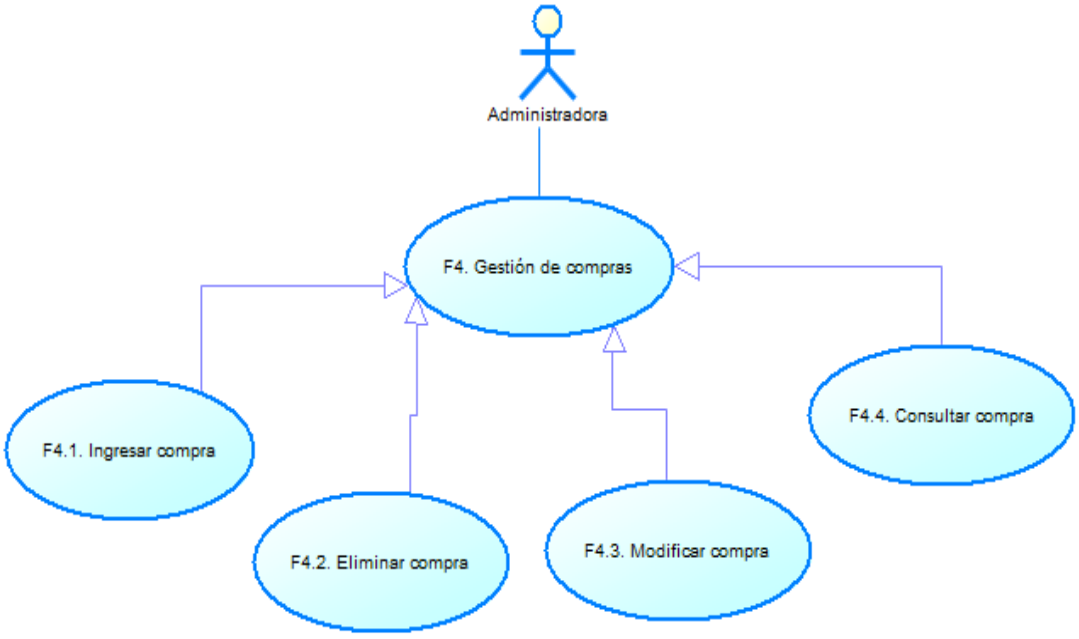
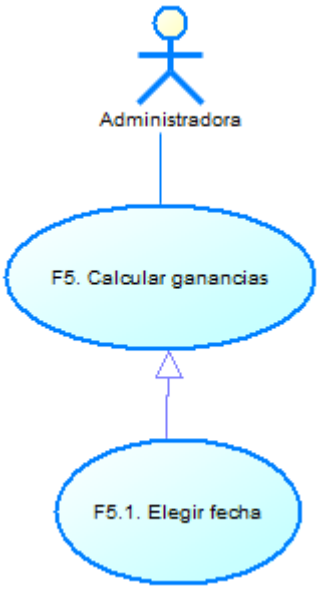


Gráfico 6

Caso de uso al siguiente nivel: F5. Calcular ganancias



3.2.2 Requerimientos No Funcionales

En el siguiente apartado se procederá a determinar los requerimientos no funcionales para el desarrollo de la aplicación web de gestión de stock de productos del mini market “Paola”; sin embargo, como ya se mencionó anteriormente los requerimientos no funcionales también se levantarán en base a ciertas pautas del estándar de IEEE 830 orientado a la especificación de requisitos; por ende, se tomará en cuenta todo lo que respecta con la interfaz de usuario, software, hardware, requisitos de rendimiento, seguridad, disponibilidad y mantenibilidad.

3.2.2.1 Interfaz de usuario

Con la administradora se ha llegado a un acuerdo, el cual consiste en que las interfaces deben ser lo más sencillo posible, con el fin que la administradora pueda navegar el sistema de forma intuitiva y eficaz, en cuanto a los colores se ha propuesto que el color principal sea el azul, además, se debe mostrar el logotipo del mini market en cada ventana. Por último, se requiere que cada funcionalidad tenga una ventana independiente.

3.2.2.2 Software

Para determinar las herramientas que se utilizaran en este proyecto, se va a proceder a evaluar estas mismas, tomando en cuenta que tienen que ser las que mejor se acoplen para la elaboración de la aplicación web para el caso de estudio planteado, esto se lo realizará con la ayuda de tablas comparativas y mediante parámetros los cuales se

puedan cuantificar del 1 al 5, al final la herramienta que más puntos sume y que más viable analíticamente sea, será la elegida para ayudar al desarrollo del proyecto.

Por tal motivo, en la siguiente tabla se va a comparar el mejor framework para el desarrollo del front-end de la aplicación web.

Tabla 10

Angular vs React

Front-End	Angular	React
Código Reutilizable	5	3
Independencia	5	3
Escalabilidad	4	4
Curva de Aprendizaje	4	2
Compatibilidad	4	3
Total	22	15

Nota. Autor: Ariel Ortega, 2022

Como se observa en la Tabla 10, el framework Angular obtuvo un total de 22 puntos mientras que el framework de React obtuvo 15 puntos; por lo tanto, el framework que se va a utilizar para el desarrollo del front-end de la aplicación web es Angular.

En la siguiente tabla se va a comparar MySQL y PostgreSQL para determinar la mejor base de datos relacional para aplicarla al caso de estudio.

Tabla 11

MySQL vs PostgreSQL

Base de Datos	MySQL	PostgreSQL
Compatibilidad	5	4
Interfaz	4	4
Seguridad	5	4
Flexibilidad	4	4
Curva de Aprendizaje	5	4
Licenciamiento	5	5
Total	28	25

Nota. Autor: Ariel Ortega, 2022

Como se evidencia en la Tabla 11, la base de datos MySQL obtuvo un total de 28 puntos mientras que la base de datos de PostgreSQL obtuvo 25 puntos; por lo tanto, la base de datos que se va a utilizar para el desarrollo de la aplicación web es MySQL, cabe mencionar que tanto MySQL como PostgreSQL obtuvieron un puntaje de 5 en el parámetro licenciamiento, ya que ambas tienen una versión gratuita y por ende no existe ningún problema al utilizar estas bases de datos para el desarrollo de la aplicación.

A continuación, se va a comparar el mejor editor de código para ayudar con el desarrollo de la aplicación web.

Tabla 12

Visual Studio Code vs Notepad++

Editor de Código	Visual Studio Code	Notepad++
Compatibilidad	5	2
Interfaz	4	2
Escalabilidad	5	2
Curva de Aprendizaje	4	3
Licenciamiento	5	5
Total	23	14

Nota. Autor: Ariel Ortega, 2022

Como se contempla en la Tabla 12, el editor de código Visual Studio Code obtuvo un total de 23 puntos mientras que Notepad++ obtuvo 14 puntos; por lo tanto, el editor de código que se va a utilizar para el desarrollo de la aplicación web es Visual Studio Code, cabe mencionar que tanto Visual Studio Code como Notepad++ obtuvieron un puntaje de 5 en el parámetro licenciamiento, ya que ambas son gratuitas y por ende no existe ningún problema al utilizarlas para el desarrollo de la aplicación.

Seguidamente, en esta tabla se va a comparar el mejor framework para el desarrollo del back-end de la aplicación web.

Tabla 13

Spring Boot vs Laravel

Back-End	Spring Boot	Laravel
Código Reutilizable	4	4
Compatibilidad	5	4
Escalabilidad	4	4
Curva de Aprendizaje	4	2
Total	17	14

Nota. Autor: Ariel Ortega, 2022

Como se observa en la Tabla 13, el framework Spring Boot obtuvo un total de 17 puntos mientras que el framework de Laravel obtuvo 14 puntos; por lo tanto, el framework que se va a utilizar para el desarrollo del back-end de la aplicación web es Spring Boot.

Finalmente, para la elaboración de los diagramas y del modelo conceptual del proyecto se va a comparar las herramientas de Power Designer y Creately.

Tabla 14

Power Designer vs Creately

Modelado y Diagramación	Power Designer	Creately
Interfaz	4	3
Compatibilidad	4	4
Escalabilidad	5	3
Curva de Aprendizaje	5	4
Licenciamiento	3	5
Total	21	19

Nota. Autor: Ariel Ortega, 2022

Como se evidencia en la Tabla 14, la herramienta Power Designer obtuvo un total de 21 puntos mientras que la herramienta de Creately obtuvo 19 puntos; por lo tanto, la herramienta para modelado y diagramación que se va a utilizar para el de la aplicación web es Power Designer, cabe mencionar que Power Designer obtuvo un puntaje de 3 en su licenciamiento debido a que no cuenta con una versión gratuita; sin embargo, cuenta con una versión de prueba, misma que provee las suficientes funcionalidades para poder generar los casos de uso y el modelo físico de la base de datos. Por otro lado, Creately si

obtuvo un puntaje de 5 en el parámetro licenciamiento, ya que esta herramienta si tiene una versión gratuita.

En resumen, se va a utilizar las siguientes herramientas para el desarrollo de la aplicación web de gestión de stock de productos del mini market “Paola”: Angular, MySQL, Visual Studio Code, Spring Boot y Power Designer

3.2.2.3 Arquitectura

A continuación, se va a elegir la arquitectura que más se acople al desarrollo del proyecto, las arquitecturas que se van a comparar en este caso son dos, las cuales son: la arquitectura REST (Transferencia de Estado Representacional) y la arquitectura SOA (Arquitectura orientada a servicios).

Para la selección de la arquitectura que más se acople al proyecto, se va a comparar estas mismas mediante una tabla la cual va a contener parámetros que serán evaluados de forma cuantitativa del 1 a 5, al final la arquitectura que más puntos será la elegida para el desarrollo del proyecto.

Tabla 15

Arquitectura REST vs Arquitectura SOA

Arquitecturas	Arquitectura REST	Arquitectura SOA
Rapidez	4	4
Compatibilidad	5	4
Escalabilidad	4	4
Fiabilidad	4	4
Curva de Aprendizaje	4	1
Total	21	17

Nota. Autor: Ariel Ortega, 2022

Tal y como se observa en la Tabla 15, la arquitectura que más se acopla al desarrollo de la aplicación web de gestión de stock de productos del mini market “Paola” es la arquitectura REST, ya que esta mismo obtuvo un total de 21 puntos mientras que la arquitectura SOA obtuvo un total de 17 puntos; por ende, se va a trabajar con la arquitectura REST en el desarrollo del proyecto.

3.2.2.4 Hardware

En cuanto al hardware, por la razón que la aplicación no es pesada ni exigente cuando se habla del consumo de recursos de un computador, se puede trabajar sin problemas con la computadora que ya cuenta el cliente; por lo tanto, no debe existir ningún problema para la computadora del cliente con la aplicación web de gestión de stock de productos y su respectivo procesamiento.

3.2.2.5 Requisitos de rendimiento

Dado que la aplicación web propuesta es solamente para la gestión de stock de productos de un mini market, misma aplicación la cual manejará un solo rol y para una persona, la cual es la administradora, no debe existir algún problema por trabajo simultaneo; además, la gestión de proveedores, productos, los registros de compras y ventas no deberían tardar en cuanto a procesamiento, asimismo, no debería tardar en calcular los ingresos totales por mes.

3.2.2.6 Seguridad

Ahora bien, en cuanto al tema de seguridad se contará con un usuario y contraseña para poder acceder e interactuar con la aplicación web, además, también se hara uso de la herramienta IPBan, la cual sirve para evitar el ingreso de personas no autorizadas o bloquear direcciones IPs sospechosas que pretendan hacer un mal uso de la información.

3.2.2.7 Disponibilidad

La aplicación web estará disponible el 100% del tiempo; sin embargo, para la ejecución de las principales actividades, la administradora solo lo necesita en el horario de 8:00 a.m. a 8:00 p.m. todos los días.

3.2.2.8 Mantenibilidad

Al tratarse de una aplicación web de gestión de stock de productos de un mini market, la mantenibilidad no es alta; sin embargo, solo hay que estar pendiente por existe algún fallo en la base de datos o en alguna operación en cuanto a lógica.

3.3 Desarrollo de una aplicación para gestionar el stock de los productos del negocio con la metodología WSDM

3.3.1 Modelado de usuarios

Para empezar, se va a plantear una serie de preguntas al cliente acerca del mini market y su manera de operar, con el fin de recolectar información que sea útil para elaborar adecuadamente la aplicación web. Por consiguiente, se procoederá a clasificar el grupo de usuarios existentes y a describir estos mismos, cabe mencionar que todas las preguntas van a ser contestadas por el cliente en las reuniones acordadas en la bitácora (Anexo 1).

¿A qué se dedica el mini market "Paola"?

El mini market "Paola" se dedica a la comercialización de todo tipo de víveres como, por ejemplo: frituras, frutas, verduras, chicles, leches, galletas, colas, aguas, jugos, helados, cervezas, etc.

¿Qué se pretende solventar con la aplicación web?

Con la aplicación web, el mini market tiene previsto solventar algunos problemas como el registro de compras y ventas que se efectúen, así como también llevar una cuenta de las ganancias por mes. Por último, se deberá contemplar la gestión de los productos y sus respectivos proveedores.

¿Cuántas personas atienden en el negocio y cuáles son sus roles?

Dado que es un negocio familiar, el mini market "Paola" es atendido generalmente por la administradora y su hijo; sin embargo, a veces también lo ayuda a atender su hija y su esposo; por lo tanto, cuatro personas atienden este mismo y todas tienen el permiso de realizar las mismas actividades cuando están a cargo; por ende, todos podrían tener el rol de administradora sin ningún conflicto.

¿Cuántas personas van a utilizar la aplicación web?

Como ya se mencionó anteriormente se trata de un negocio familiar de un total de cuatro personas y la administradora ha sabido manifestar que todas estas personas deben poder acceder a la aplicación web y utilizar las funcionalidades que requieran sin ninguna restricción; es decir, todos los miembros de la familia vendrían a asumir el rol de administradora con todos los permisos que este posee.

3.3.1.1 Clasificación de usuarios

En esta etapa se va a identificar a los tipos de usuarios que van a hacer uso de la aplicación web; no obstante, para este caso solo va a haber un tipo de usuario, el cual es el de administradora, ya que como se aclaró en las preguntas anteriores respondidas por el cliente, es un negocio familiar en la cual ayudan cuatro personas y se pretende que todas puedan utilizar la aplicación web por igual sin ninguna restricción.

3.3.1.2 Descripción de los grupos de usuarios

En la siguiente tabla, se va a describir a los tipos de usuarios identificados en la anterior etapa que en este caso solo es la administradora.

Tabla 16

Descripción de los grupos de usuarios

Tipo de usuario	Información
Administradora	Para comprender un poco más las funcionalidades a la que puede acceder este tipo de usuario, cabe mencionar que se puede observar de forma ilustrativa en los gráficos 1, 2, 3, 4, 5 y 6, los cuales son los casos de uso general y al siguiente nivel de este usuario. Por consiguiente, se va a indicar las actividades que puede realizar la administradora. • Productos: Puede ingresar, eliminar, modificar y consultar los productos en la aplicación web. • Proveedores: Puede ingresar, eliminar, modificar y consultar los proveedores en la aplicación web. • Ventas: Puede ingresar, eliminar, modificar y consultar las ventas efectuadas en la aplicación web. • Compras: Puede ingresar, eliminar, modificar y consultar las compras efectuadas en la aplicación web. • Cálculo de ganancias: Puede consultar las ganancias obtenidas por mes en una fecha determinada. Dado que se trata de un usuario tipo administradora, puede acceder a todas las funcionalidades de la aplicación web sin restricción.

Nota. Autor: Ariel Ortega, 2022

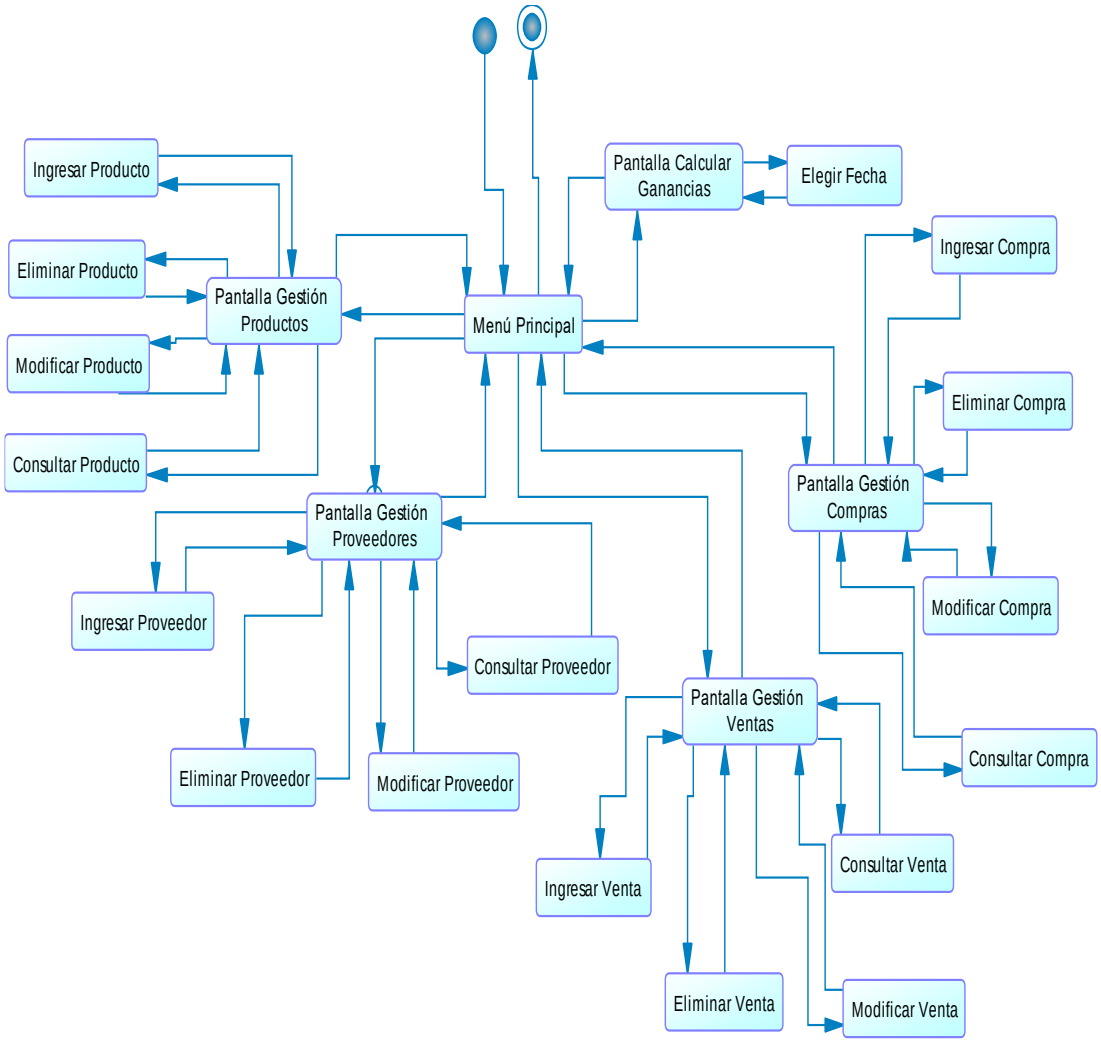
3.3.2 Diseño Conceptual

En esta fase se va a realizar el diseño de cómo estará organizada la información en la aplicación, tomando en cuenta que pueden existir varias vías de navegación dependiendo el usuario; sin embargo, para el caso del mini market “Paola” solo existe un tipo de usuario el cual es el de la administradora; por ende, en esta fase se podrá ver la perspectiva de navegación del tipo de usuario de administradora por medio del modelo navegacional, además, también se va a realizar el modelo físico de la base de datos para la respectiva aplicación web de gestión de stock de productos.

3.3.2.1 Diseño del modelo navegacional

Gráfico 7

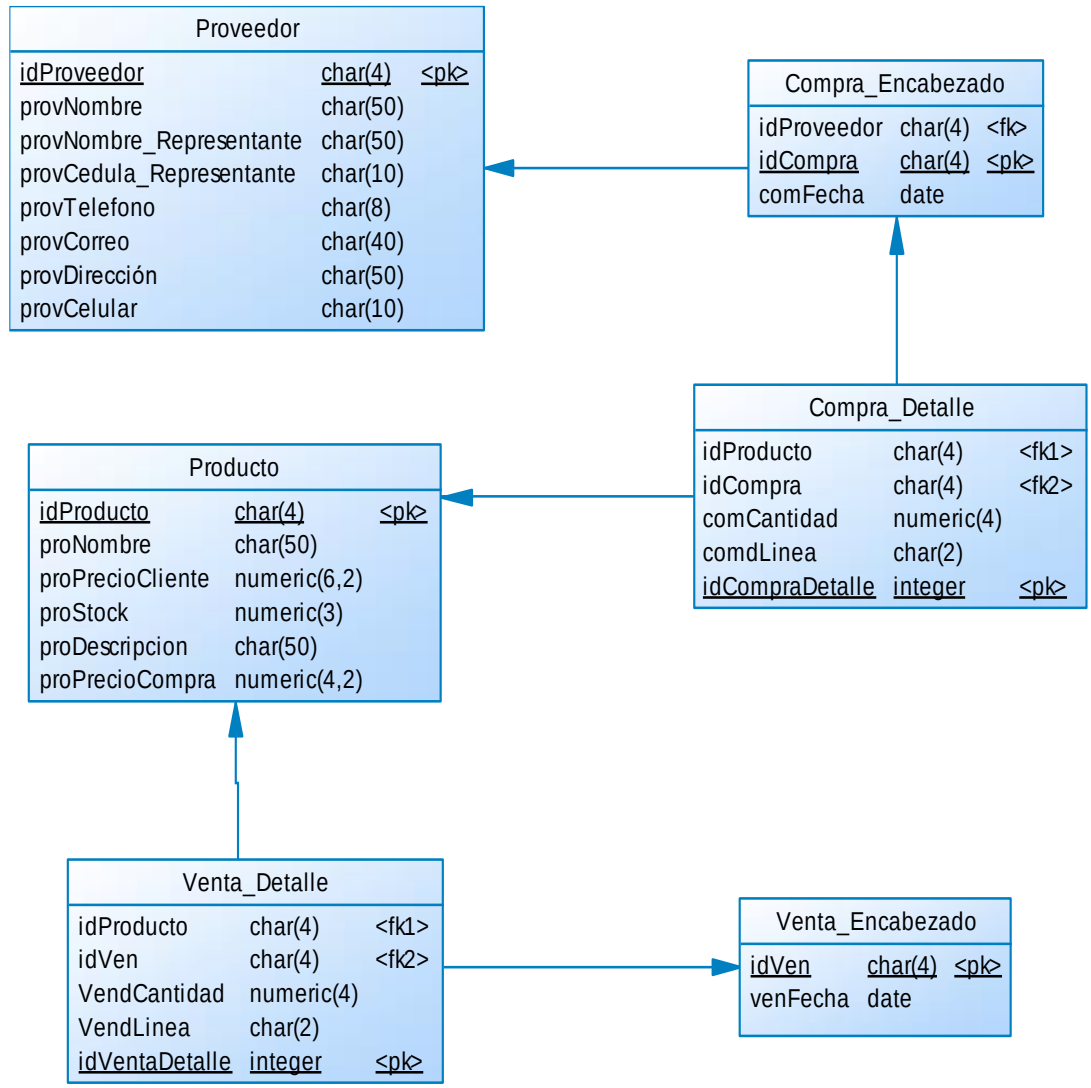
Modelo Navegacional



3.3.2.2 Diseño del modelo físico de la base de datos

Gráfico 8

Modelo Físico de la Base de Datos



3.3.3 Diseño de implementación

En esta fase se va a crear los prototipos de las interfaces para cada actividad en la aplicación web, dependiendo el tipo de usuario que sea; sin embargo, para este caso solo se cuenta con un tipo de usuario el cual es el de la administradora; por ende, se va a mostrar todas las interfaces con las respectivas actividades que este tipo de usuario pueda hacer, eso sí tomando en cuenta las restricciones y funcionalidades que ha sabido delimitar la administradora en el levantamiento de requisitos, como por ejemplo, se debe evidenciar que las interfaces sean lo más simples e intuitivas posibles con el fin que la aplicación no sea confusa, además, se debe constar que se pueda solventar todas las funcionalidades planteadas en ventanas independientes. Por último, se debe observar el azul como color predominante de la aplicación, asimismo, el logotipo del mini market "Paola" siempre se debe poder visualizar en todo momento que se utilice la aplicación.

A continuación, se va a mostrar las respectivas interfaces que solventaran con la necesidad existente del mini market "Paola".

Gráfico 9

Iniciar Sesión

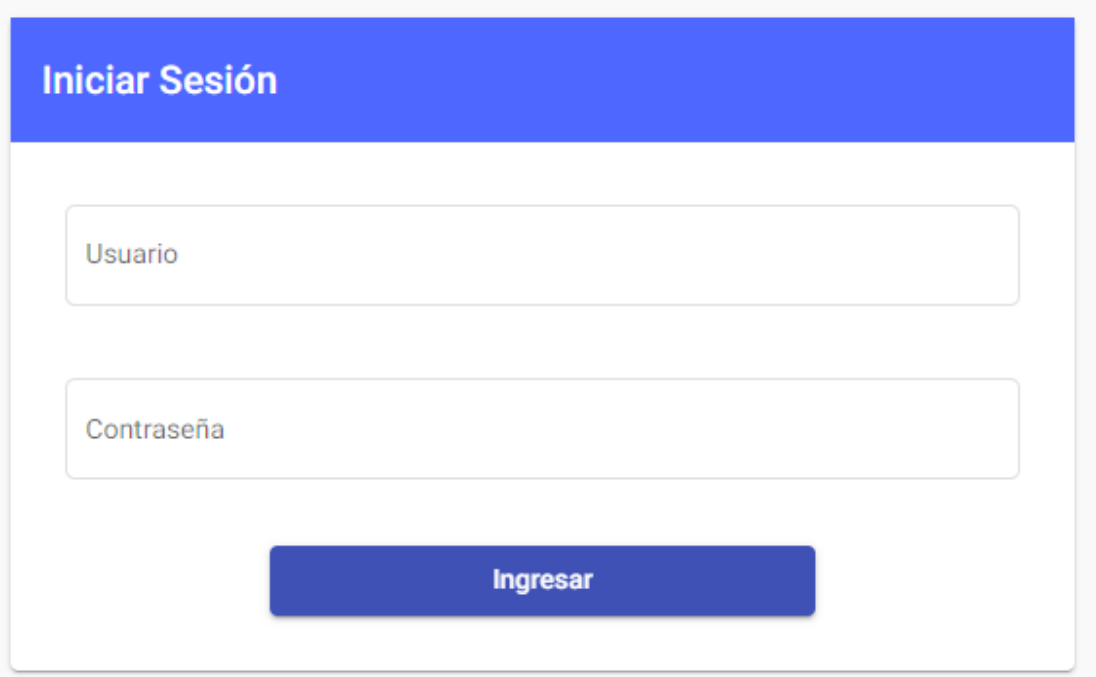


Gráfico 10

Menú Principal

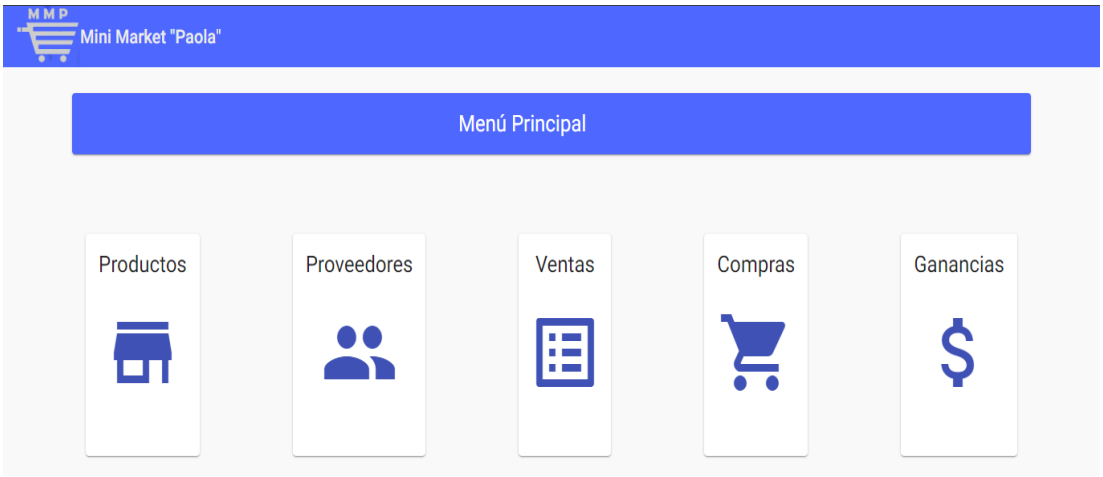


Gráfico 11

Gestión de Productos

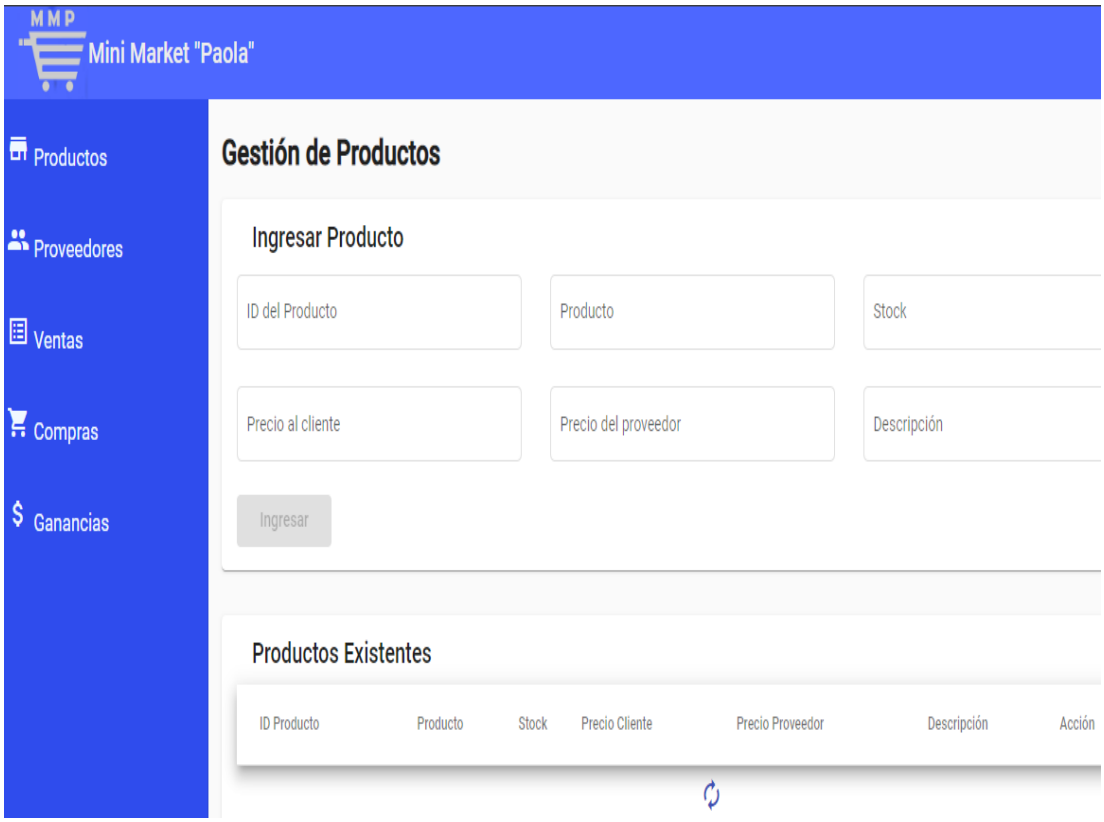


Gráfico 12

Gestión de Proveedores

Gráfico 13

Gestión de Ventas

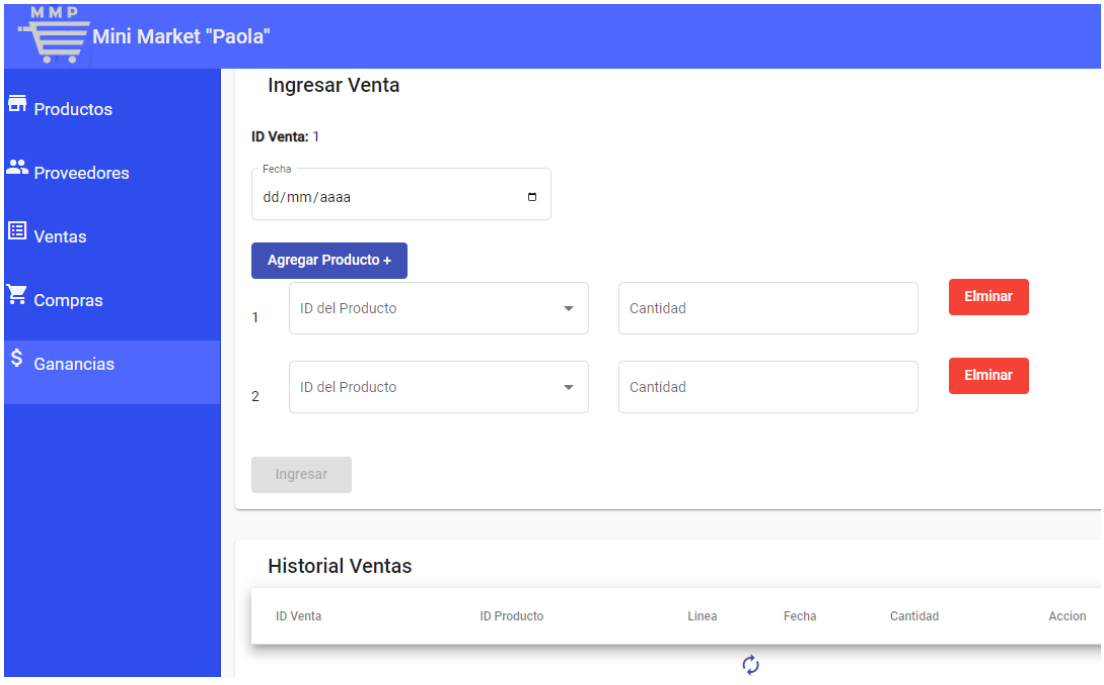


Gráfico 14

Gestión de Compras

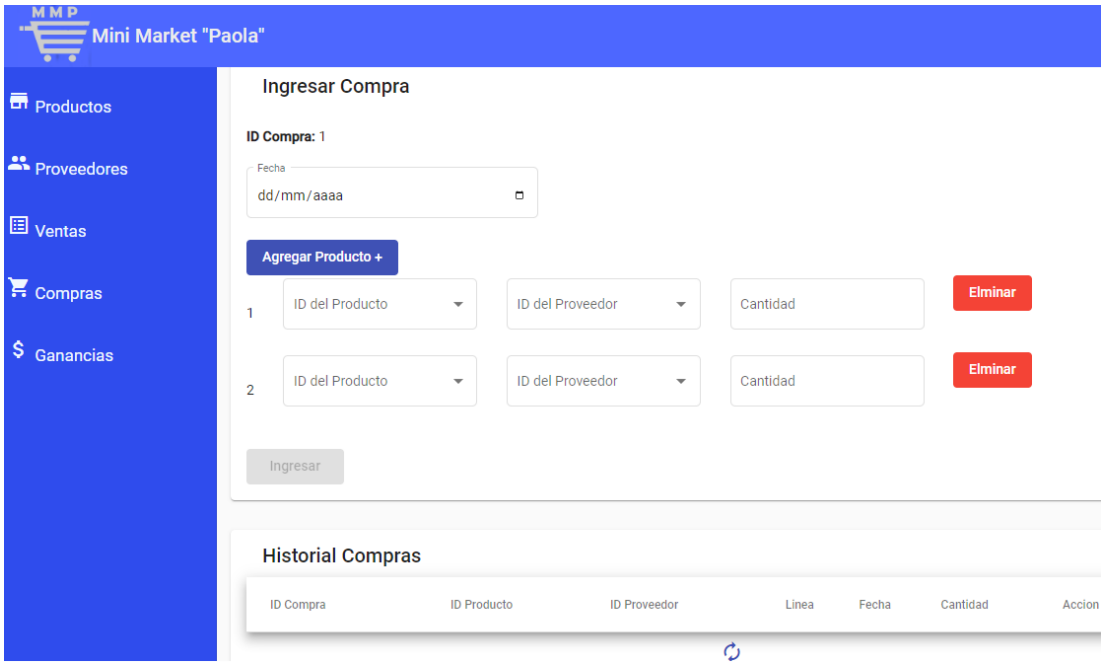
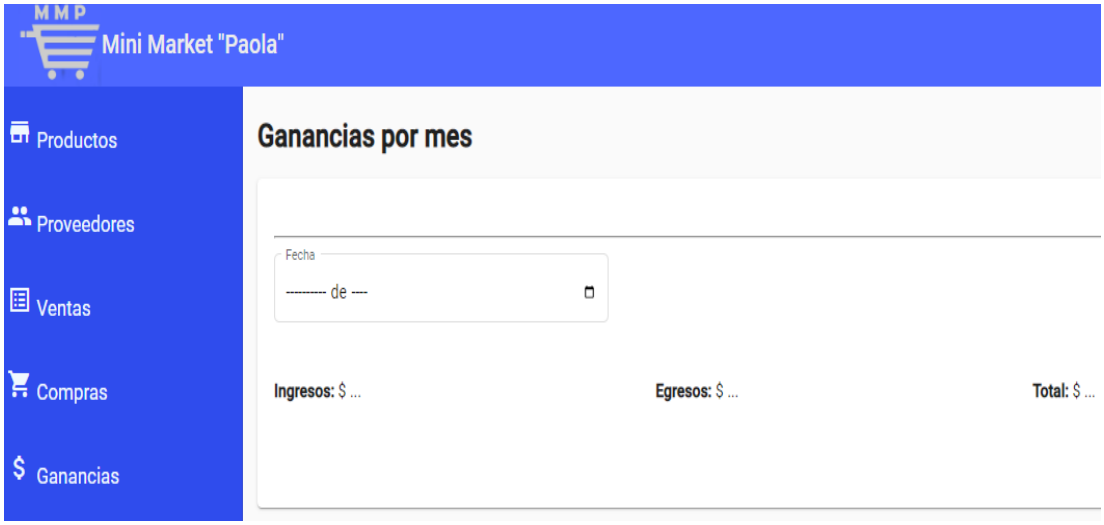


Gráfico 15

Ganancias



3.3.4 Implementación

En esta fase finalmente se va a codificar lo acordado con el cliente en base al levantamiento de requerimientos realizado, además, se tomará en cuenta que las interfaces deben ser lo más idénticas posibles a las mostradas en la etapa anterior, dado que la administradora del mini market “Paola” ya las aprobó, tanto la interfaz como las funcionalidades que procura cumplir.

Por otro lado, cabe mencionar que, según lo estipulado con la administradora, la implementación de la aplicación web se pretende hacer dentro de unos meses; por ende, en este documento solo se evidenciará el desarrollo de esta misma con sus respectivas pruebas de software.

Las pruebas que se van a efectuar a la aplicación web van a ser únicamente unitarias.

3.3.4.1 Pruebas Unitarias

En este apartado se va a efectuar las pruebas unitarias a la aplicación web desarrollada, en la cual se va a evidenciar el funcionamiento correcto de las funcionalidades de cada módulo que se acordó con la administradora del mini market “Paola” previamente, en estas pruebas no se va a en cuenta los módulos o procesos que podrían verse afectados por la dependencia o relación que se tenga con la funcionalidad que se ponga a prueba, ya que se busca solo comprobar el funcionamiento de manera individual y no en conjunto.

A continuación, se va a mostrar las pruebas unitarias respectivas que se le realizaron a la aplicación web.

Gráfico 16

Prueba Unitaria: Iniciar Sesión

Gráfico 17

Prueba Unitaria: Iniciar Sesión Fallida

Gráfico 18

Prueba Unitaria: Iniciar Sesión Exitosa

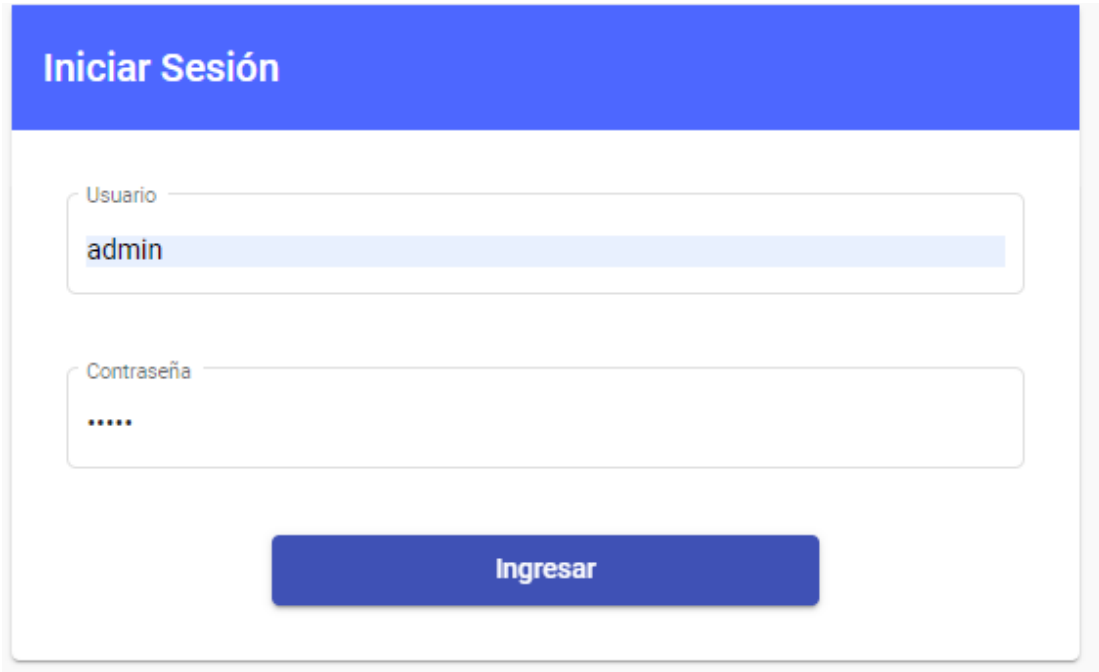
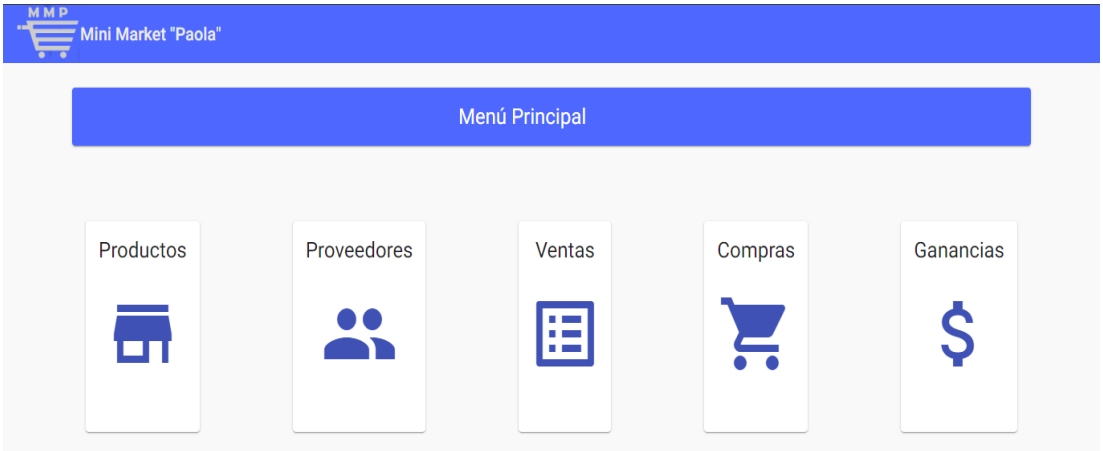


Gráfico 19

Prueba Unitaria: Iniciar Sesión Exitosa – Menú Principal



Como se evidencia en el Gráfico 16, se tiene la respectiva ventana para iniciar sesión; sin embargo, si no se ingresa el usuario o contraseña correcta el sistema no permitirá al usuario ingresar a la aplicación web, tal y como se muestra en la Gráfico 17, por otro lado, si se ingresa el usuario y contraseña correcta así como se observa en el Gráfico 18, el sistema permitirá al usuario ingresar a la aplicación web, para ser más certeros le permite ingresar al menú principal como se puede apreciar en el Gráfico 19.

Ahora bien, en los Gráficos 20, 21, 22, 23, 24 y 25 se va a mostrar como el usuario se puede dirigir directamente a la ventana de la funcionalidad que necesite desde el menú principal.

Gráfico 20

Prueba Unitaria: Menú Principal

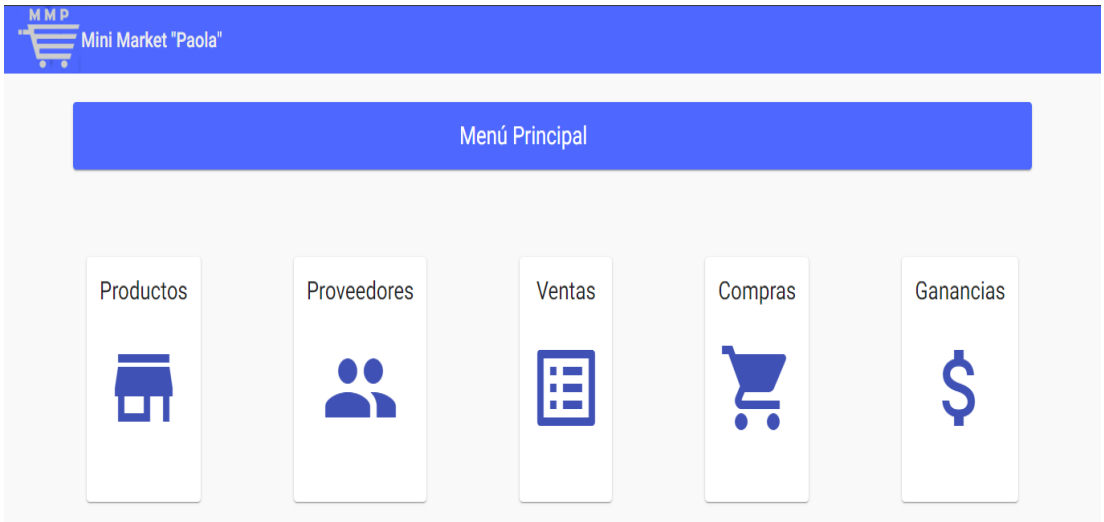


Gráfico 21

Prueba Unitaria: Menú Principal - Productos

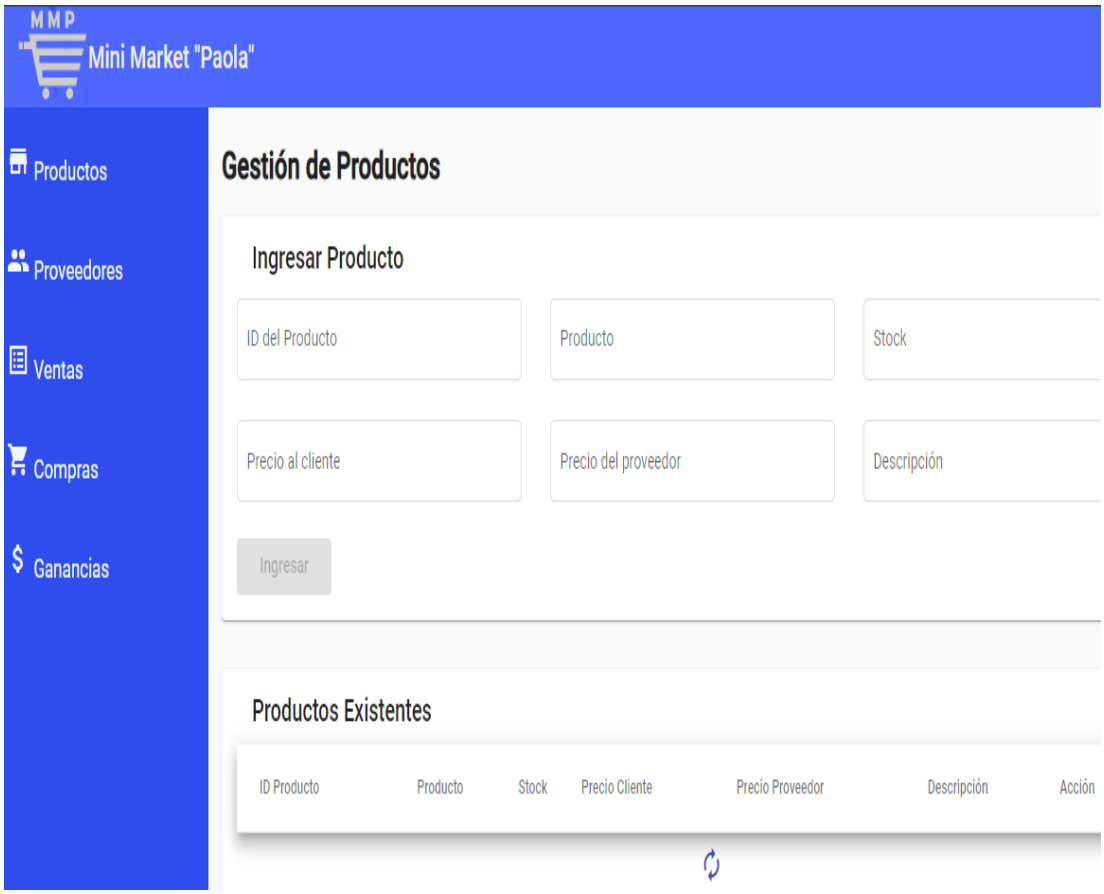


Gráfico 22

Prueba Unitaria: Menú Principal - Proveedores

Gráfico 23

Prueba Unitaria: Menú Principal - Ventas

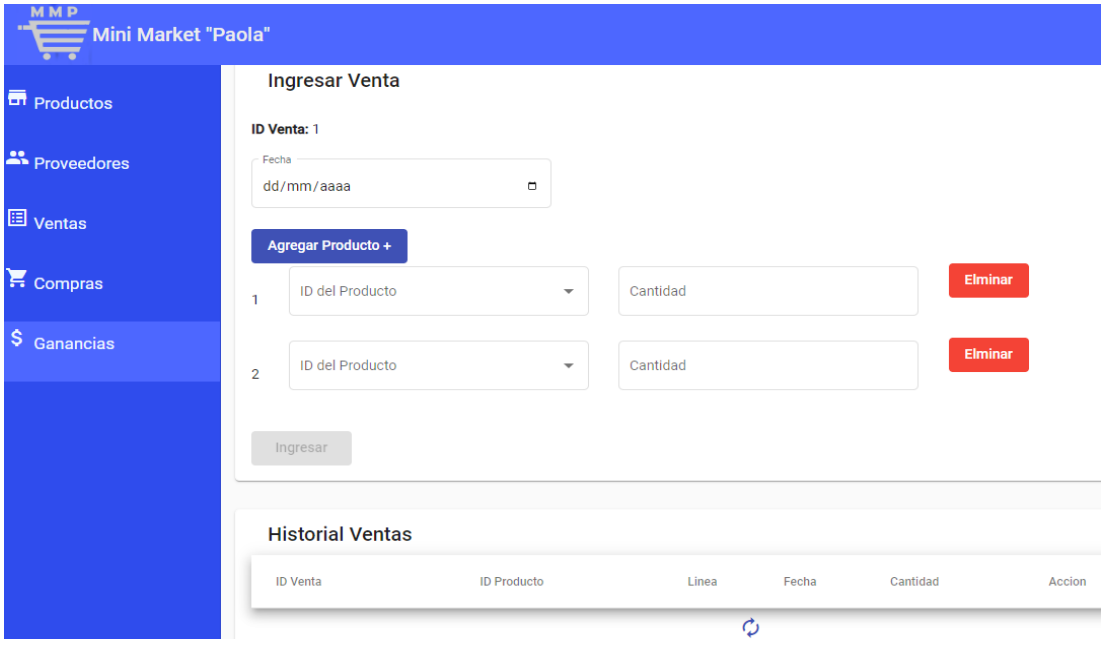


Gráfico 24

Prueba Unitaria: Menú Principal - Compras

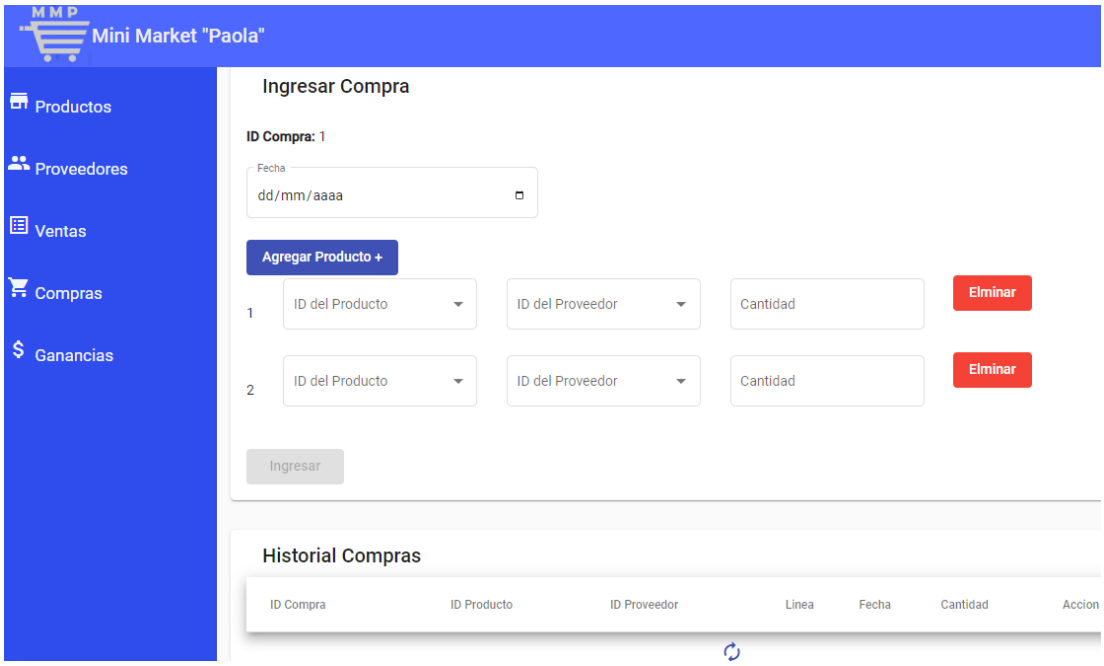
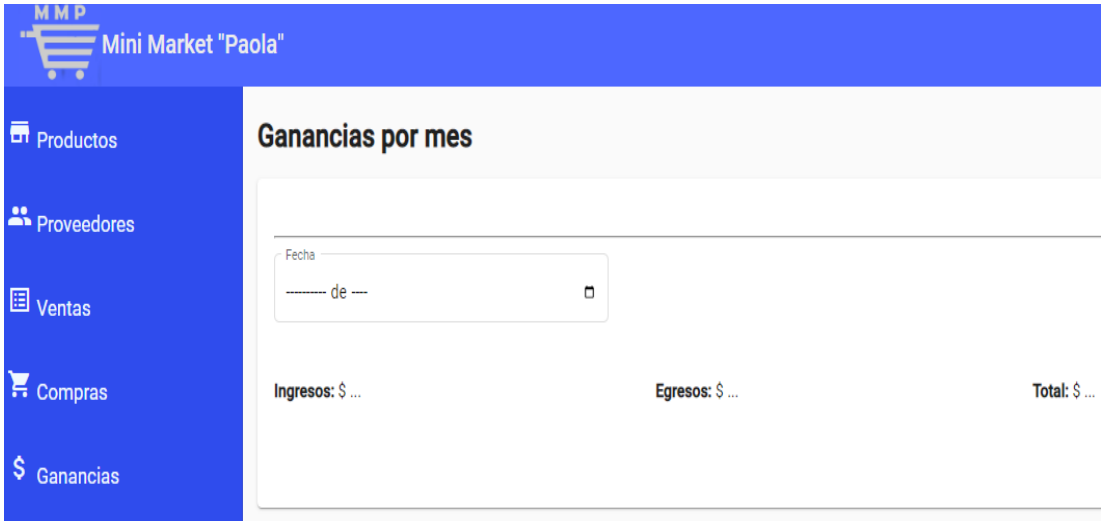


Gráfico 25

Prueba Unitaria: Menú Principal - Ganancias



A continuación, se va a mostrar el ingreso de un producto a través de un formulario, cabe destacar que la aplicación web dejará ingresar un producto solo sí cada campo del formulario se llena correctamente, es por eso por lo que en el Gráfico 26 no se habilita el botón de ingresar, mientras que en el Gráfico 27 ya se habilita, ya que todos los campos del formulario se encuentran llenados correctamente; por ende, al usuario se le permite ingresar el producto solicitado.

Gráfico 26

Prueba Unitaria: Ingresar Producto - Antes

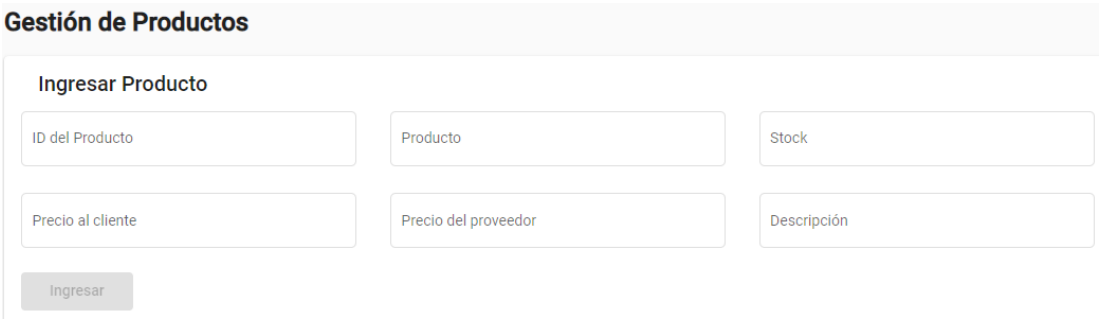
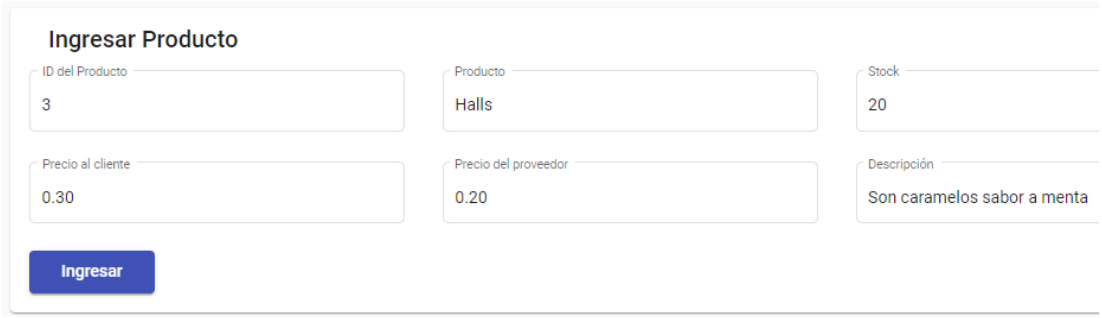


Gráfico 27

Prueba Unitaria: Ingresar Producto - Después



En la siguiente prueba unitaria, se va a mostrar como la aplicación web es capaz de modificar un producto que ya haya sido registrado previamente, es por eso por lo que se va a modificar el producto “Halls”, producto que fue ingresado en la prueba anterior , el cual se ver que ya consta en la tabla de productos tal y como se observa en el Gráfico 28, además se puede evidenciar un icono de un lápiz azul el cual permite abrir un cuadro en donde se puede modificar dicho producto como se muestra en el Gráfico 29, aquí se va a cambiar su descripción en donde dice “Son caramelos sabor a menta” por “Son caramelos sabor a menta pequeños” , una vez modificado el campo se procede a aplicar el cambio a través del botón “Guardar”, finalmente en el Gráfico 30 se puede ver como el producto “Halls” ya se encuentra modificado tal y como se lo requería.

Gráfico 28

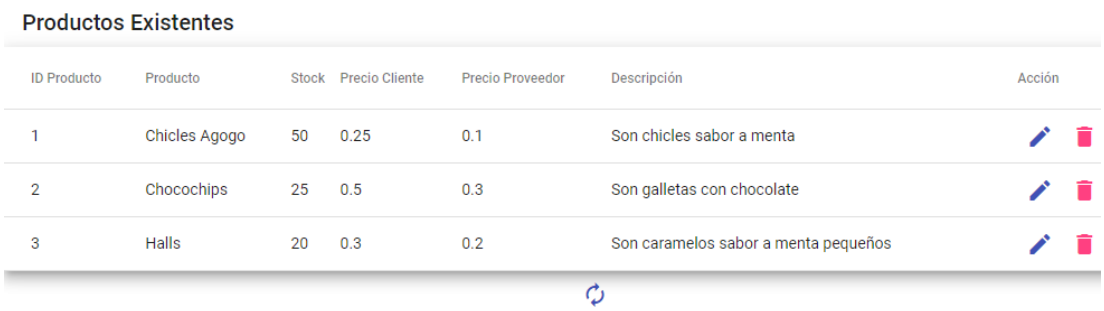
Prueba Unitaria: Modificar Producto – Cuadro

Gráfico 29

Prueba Unitaria: Modificar Producto – Guardar

Gráfico 30

Prueba Unitaria: Modificar Producto – Exitoso



En esta prueba unitaria, se va a evidenciar como la aplicación web es capaz de eliminar un producto que ya haya sido registrado previamente, es por eso por lo que se va a eliminar el producto “Halls”, producto el cual se puede ver que ya consta en la tabla de productos tal y como se observa en el Gráfico 31, además se puede evidenciar un icono de un basurero rosa el cual permite abrir un cuadro en donde se puede eliminar dicho producto como se muestra en el Gráfico 32, aquí se va a observar todos los datos de dicho producto que se va a eliminar; por lo tanto, para eliminar el producto se debe presionar el botón “Eliminar”, finalmente en el Gráfico 32 se puede ver como el producto “Halls” ya no se encuentra en la tabla.

Gráfico 31

Prueba Unitaria: Eliminar Producto – Cuadro

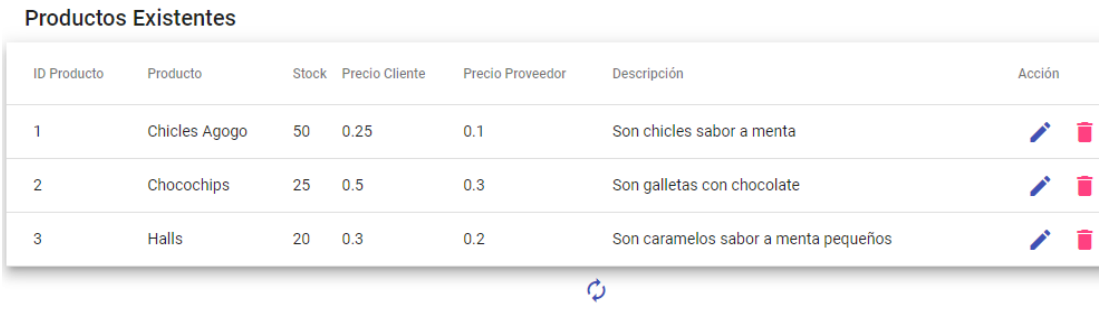


Gráfico 32

Prueba Unitaria: Eliminar Producto – Eliminar

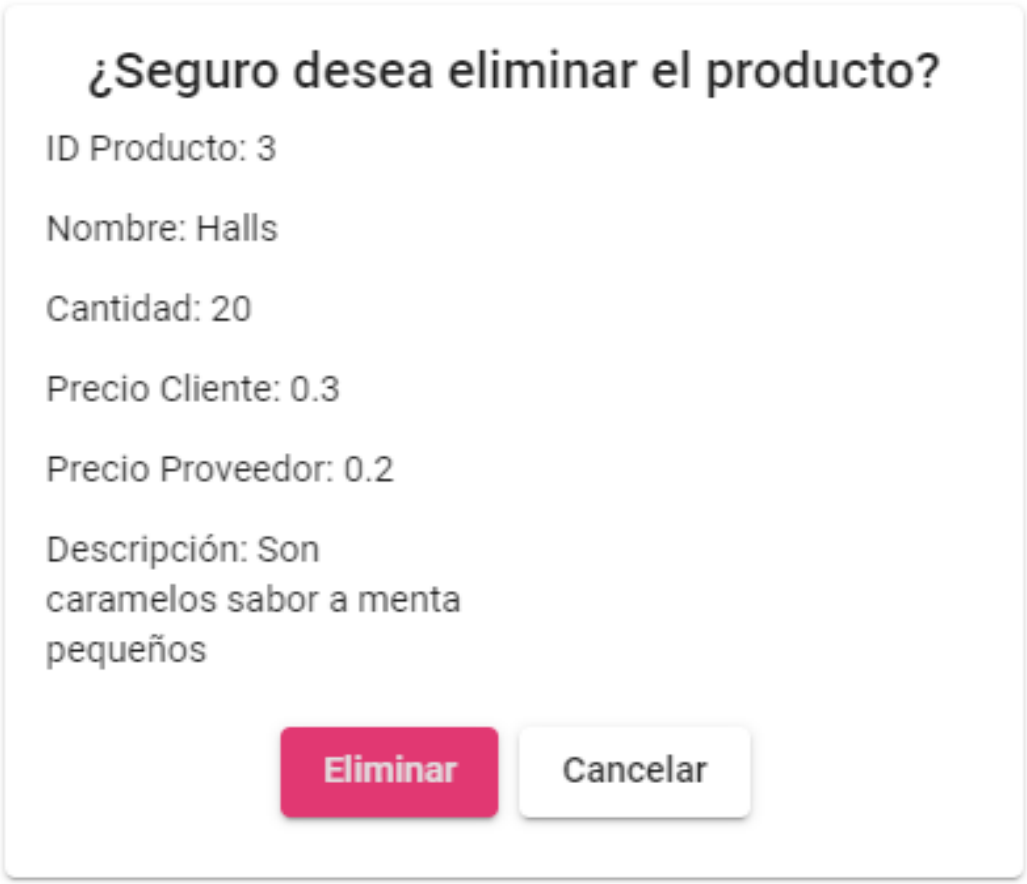


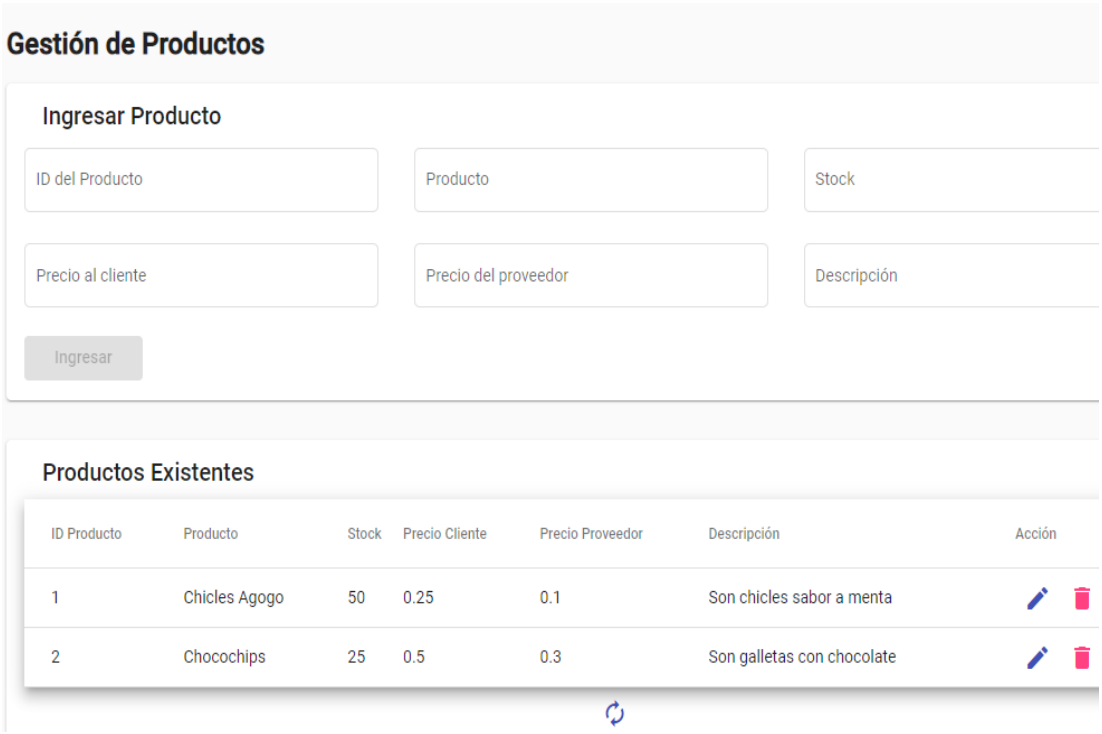
Gráfico 33

Prueba Unitaria: Eliminar Producto – Exitoso

Como última prueba unitaria para la funcionalidad de producto, se puede evidenciar en el Gráfico 34 que apenas se abre la ventana de producto, también se carga una tabla con toda la información de todos los productos registrados hasta el momento, permitiendo así al usuario consultar de manera general todos los productos que posee.

Gráfico 34

Prueba Unitaria: Consultar Producto



Posteriormente, se va a mostrar el ingreso de un proveedor a través de un formulario, cabe destacar que la aplicación web dejará ingresar un proveedor solo sí cada campo del formulario se llena correctamente, es por eso por lo que en el Gráfico 35 no se habilita el botón de ingresar, mientras que en el Gráfico 36 ya se habilita, ya que todos los campos del formulario se encuentran llenados correctamente; por ende, al usuario se le permite ingresar el proveedor solicitado.

Gráfico 35

Prueba Unitaria: Ingresar Proveedor - Antes

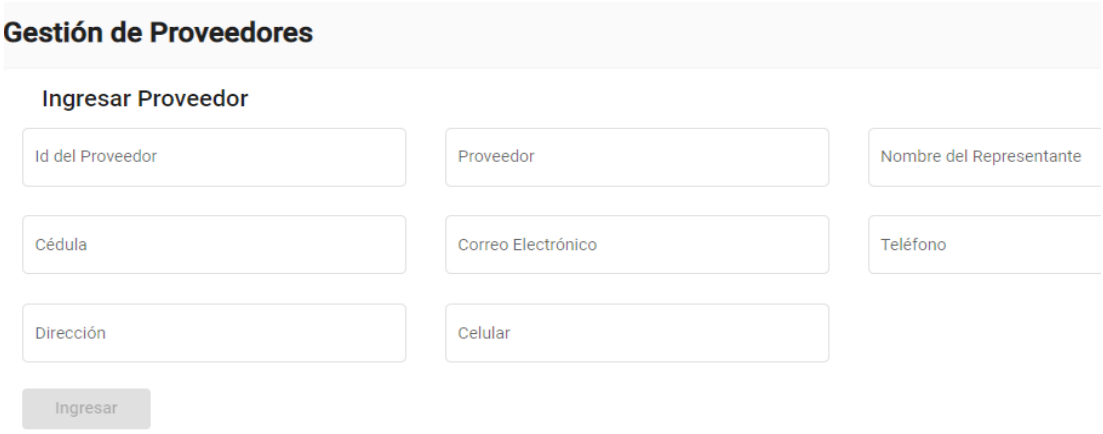
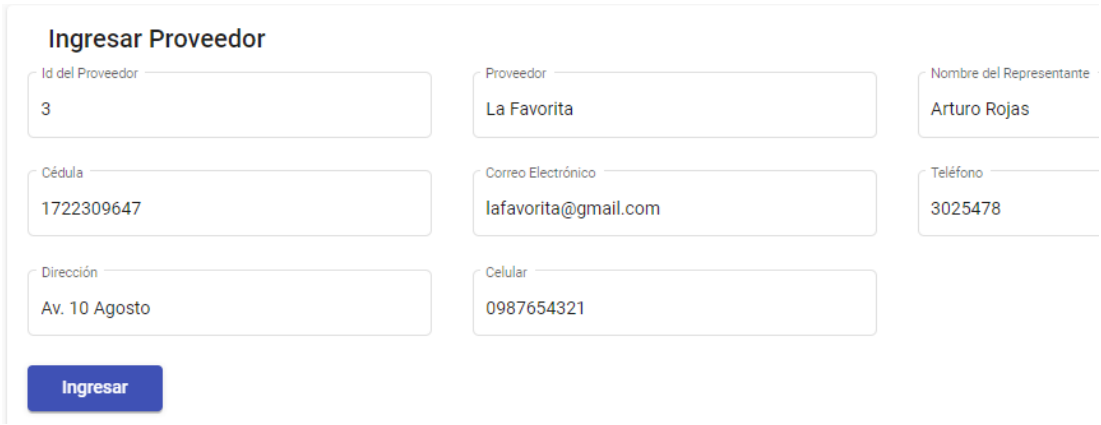


Gráfico 36

Prueba Unitaria: Ingresar Proveedor - Después



En la siguiente prueba unitaria, se va a mostrar como la aplicación web es capaz de modificar un proveedor que ya haya sido registrado previamente, es por eso por lo que se va a modificar el proveedor "La Favorita", proveedor que fue ingresado en la prueba anterior, el cual se puede ver que ya consta en la tabla de proveedores tal y como se observa en el Gráfico 37, además se puede evidenciar un icono de un lápiz azul el cual permite abrir un cuadro en donde se puede modificar dicho proveedor como se muestra en el Gráfico 38, aquí se va a cambiar su dirección en donde dice "Av. 10 de Agosto" por "Av. Shyris" , una vez modificado el campo se procede a aplicar el cambio a través del botón "Guardar", finalmente en el Gráfico 39 se puede ver como el proveedor "La Favorita" ya se encuentra modificado tal y como se lo requería.

Gráfico 37

Prueba Unitaria: Modificar Proveedor – Cuadro

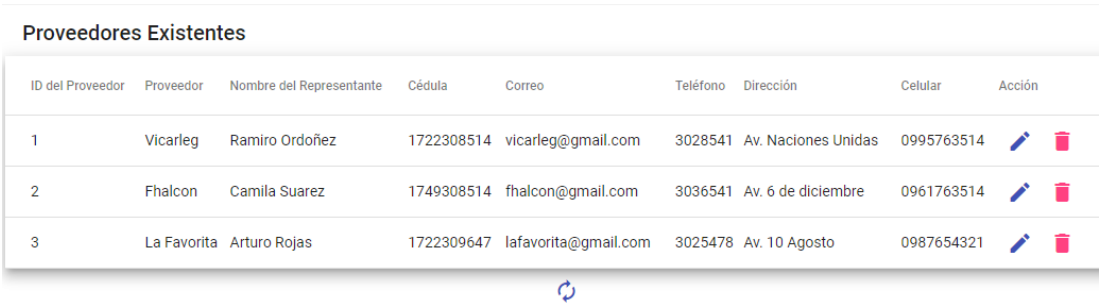


Gráfico 38

Prueba Unitaria: Modificar Proveedor – Guardar

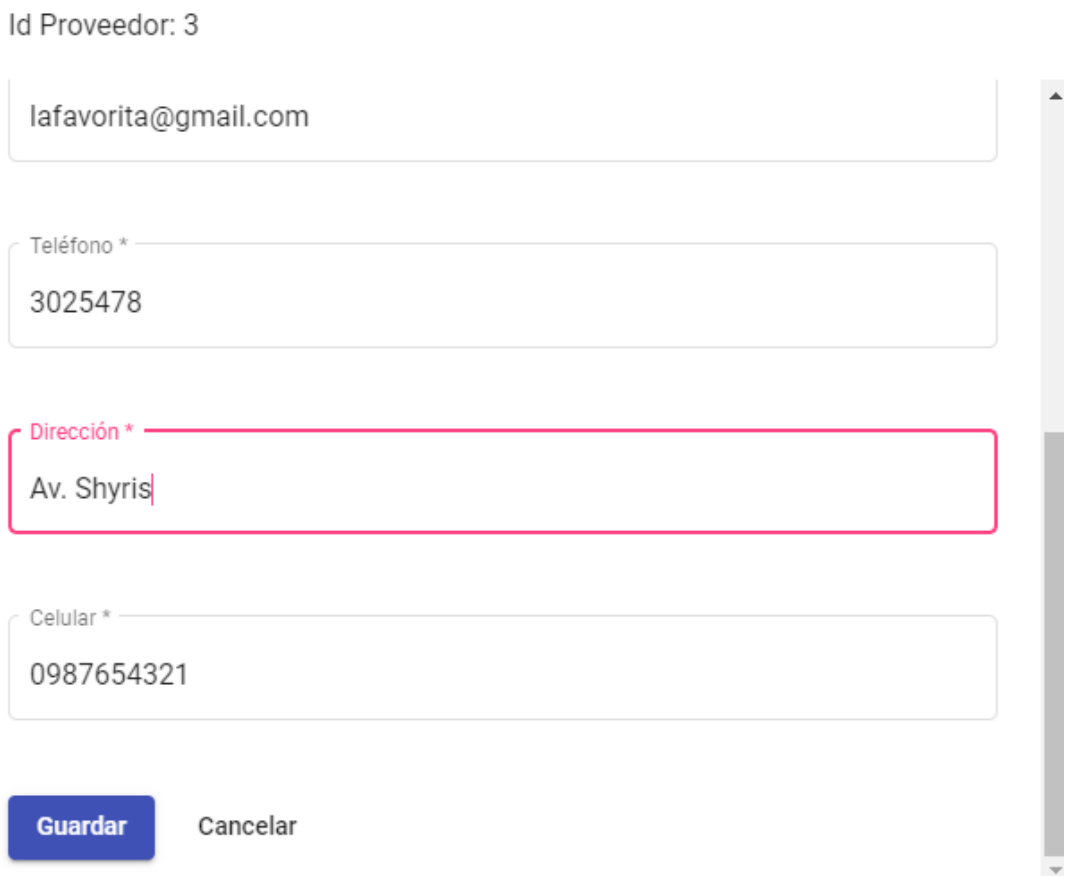
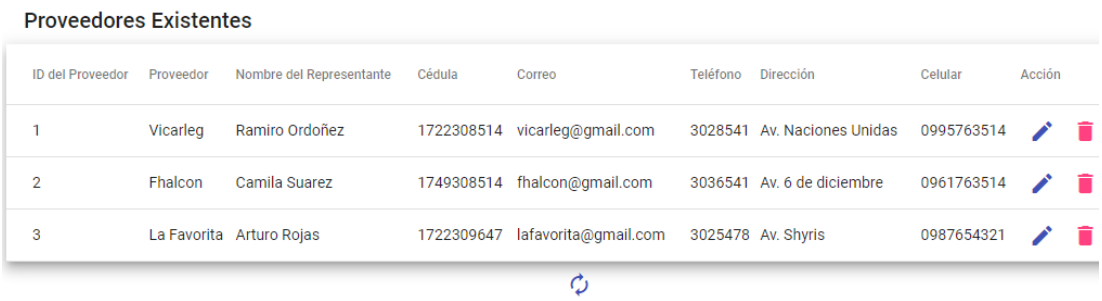


Gráfico 39

Prueba Unitaria: Modificar Proveedor – Exitoso



En esta prueba unitaria, se va a evidenciar como la aplicación web es capaz de eliminar un proveedor que ya haya sido registrado previamente, es por eso por lo que se va a eliminar el proveedor “La Favorita”, proveedor el cual se puede ver que ya consta en la tabla de proveedores tal y como se observa en el Gráfico 40, además se puede evidenciar un icono de un basurero rosa el cual permite abrir un cuadro en donde se puede eliminar dicho proveedor como se muestra en el Gráfico 41, aquí se va a observar todos los datos de dicho proveedor que se va a eliminar; por lo tanto, para eliminar el proveedor se debe presionar el botón “Eliminar”, finalmente en el Gráfico 42 se puede ver como el proveedor “La Favorita” ya no se encuentra en la tabla.

Gráfico 40

Prueba Unitaria: Eliminar Proveedor – Cuadro

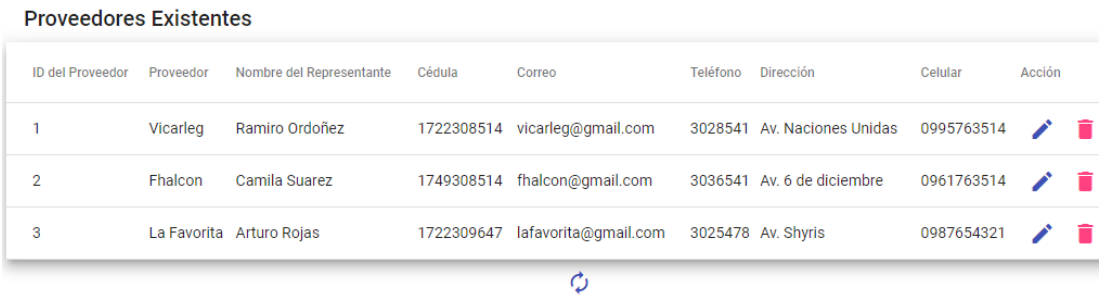
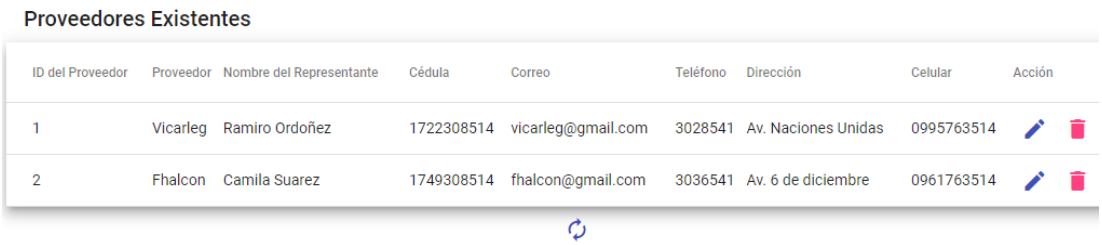


Gráfico 41

Prueba Unitaria: Eliminar Proveedor – Eliminar

Gráfico 42

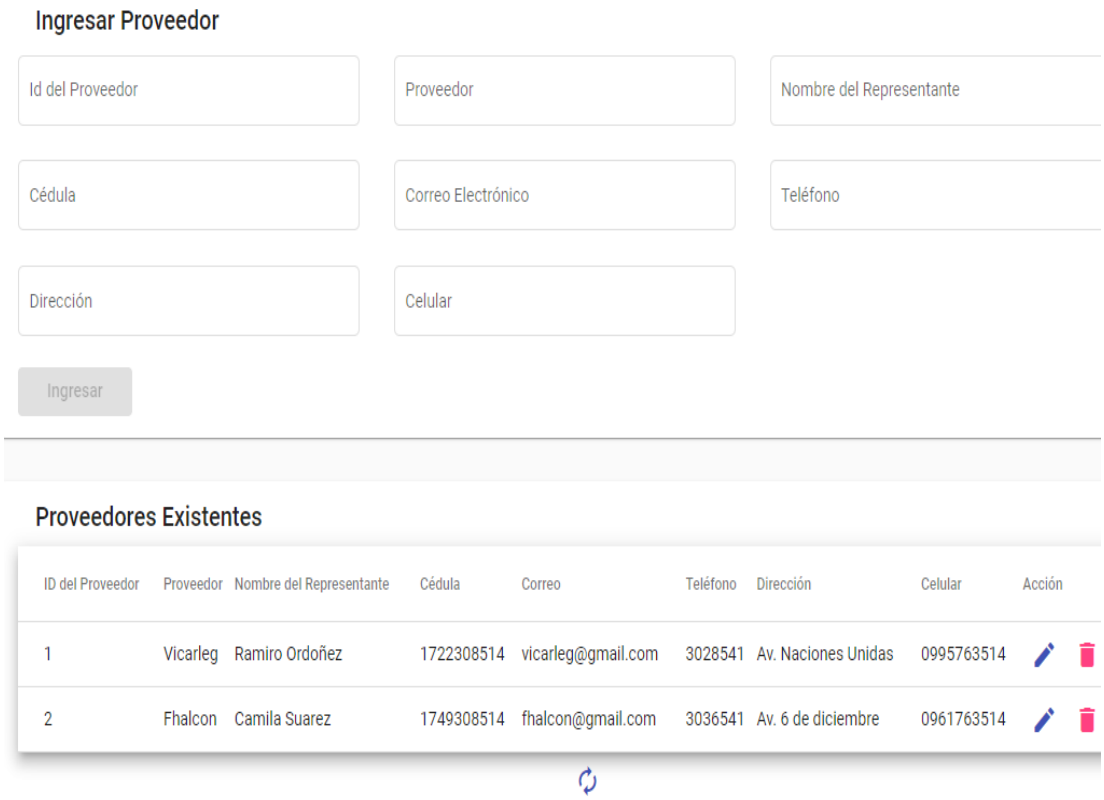
Prueba Unitaria: Eliminar Proveedor – Exitoso



Como última prueba unitaria para la funcionalidad de proveedor, se puede evidenciar en el Gráfico 43 que apenas se abre la ventana de proveedor, también se carga una tabla con toda la información de todos los proveedores registrados hasta el momento, permitiendo así al usuario consultar de manera general todos los proveedores que tiene.

Gráfico 43

Prueba Unitaria: Consultar Proveedor



A continuación, se va a mostrar el ingreso de una venta a través de un formulario, este mismo también cuenta con un botón de “Agregar Producto +” el cual permite ingresar más de un producto en una venta permaneciendo en la misma ventana, cabe destacar que la aplicación web dejará ingresar una venta solo sí cada campo del formulario se llena correctamente, es por eso por lo que en el Gráfico 44 no se habilita el botón de ingresar, mientras que en el Gráfico 45 ya se habilita, ya que todos los campos del formulario se encuentran llenados correctamente; por ende, al usuario se le permite ingresar la venta, por último, vale la pena mencionar que el “Id Venta” se actualiza solo y por lo tanto el “Id Venta” correspondiente para esta prueba unitaria es el número cuatro ya que ya existen otras ventas que han sido ingresadas previamente.

Gráfico 44

Prueba Unitaria: Ingresar Venta - Antes

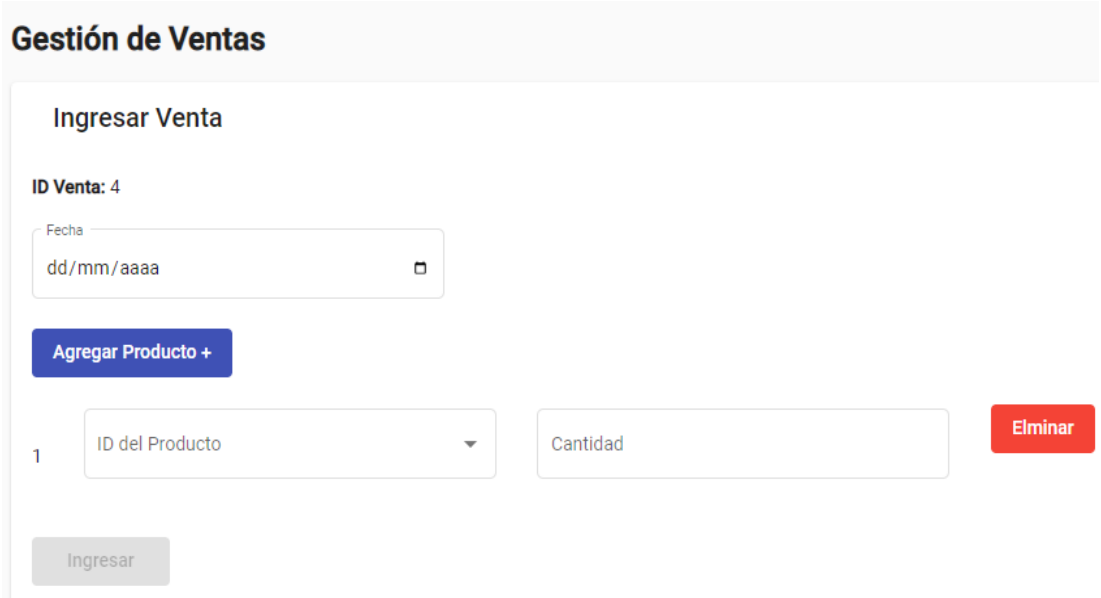
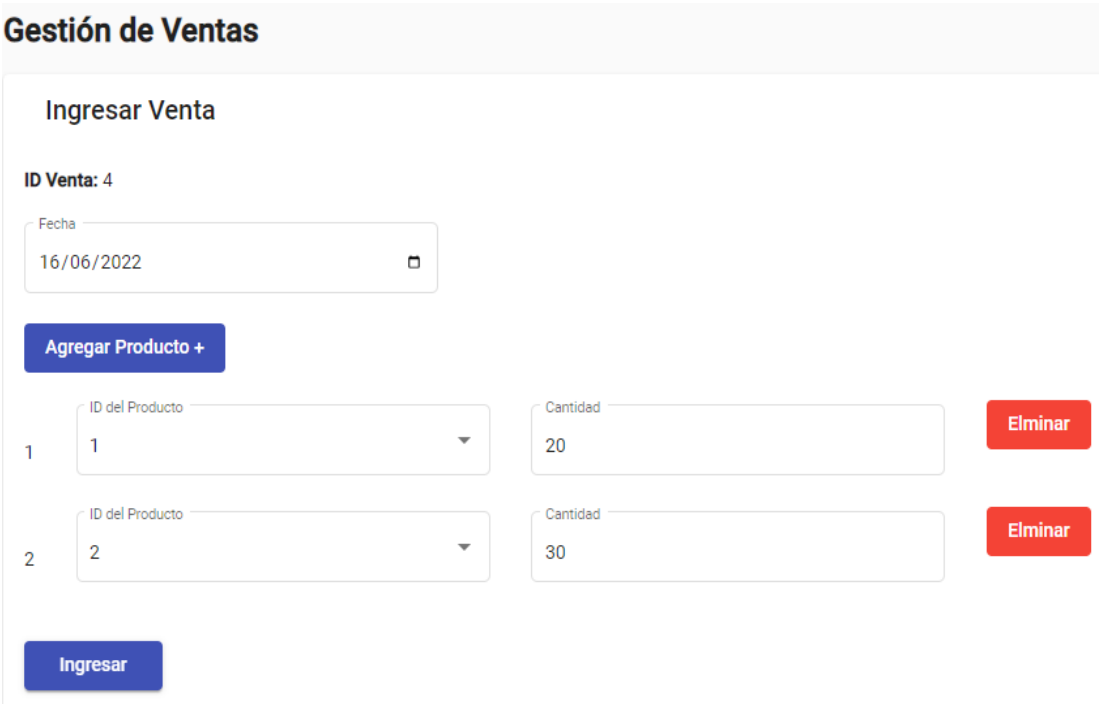


Gráfico 45

Prueba Unitaria: Ingresar Venta - Después



En la siguiente prueba unitaria, se va a mostrar como la aplicación web es capaz de modificar una venta que ya haya sido registrada previamente, es por eso por lo que se va a modificar la venta con Id cuatro, venta que fue ingresada en la prueba anterior, la cual se puede ver que ya consta en la tabla de ventas tal y como se observa en el Gráfico 46, además se puede evidenciar un icono de un lápiz azul el cual permite abrir un cuadro en donde se puede modificar dicha venta como se muestra en el Gráfico 47, aquí se va a cambiar la cantidad del producto uno en donde dice “20” por “40” , una vez modificado el campo se procede a aplicar el cambio a través del botón “Guardar”, finalmente en el Gráfico 48 se puede ver como la venta del producto uno con Id cuatro ya se encuentra modificada tal y como se lo requería.

Gráfico 46

Prueba Unitaria: Modificar Venta – Cuadro

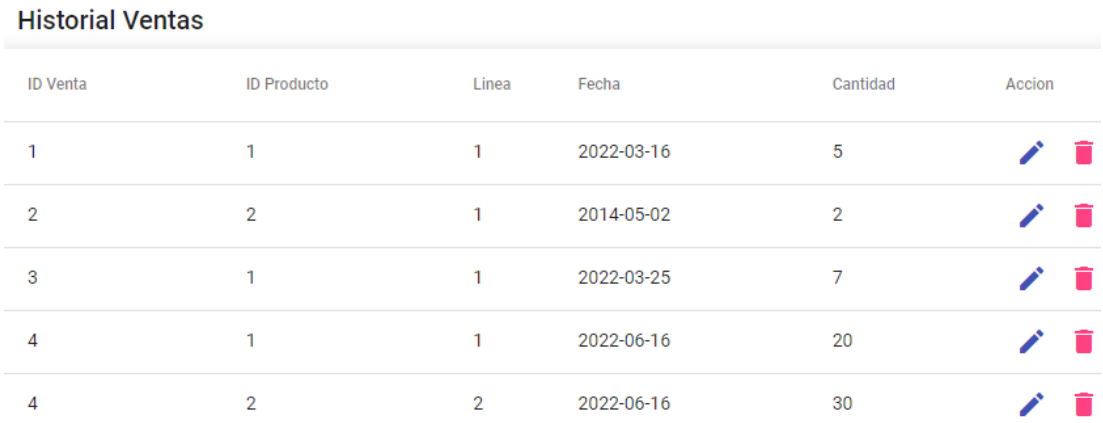


Gráfico 47

Prueba Unitaria: Modificar Venta – Guardar

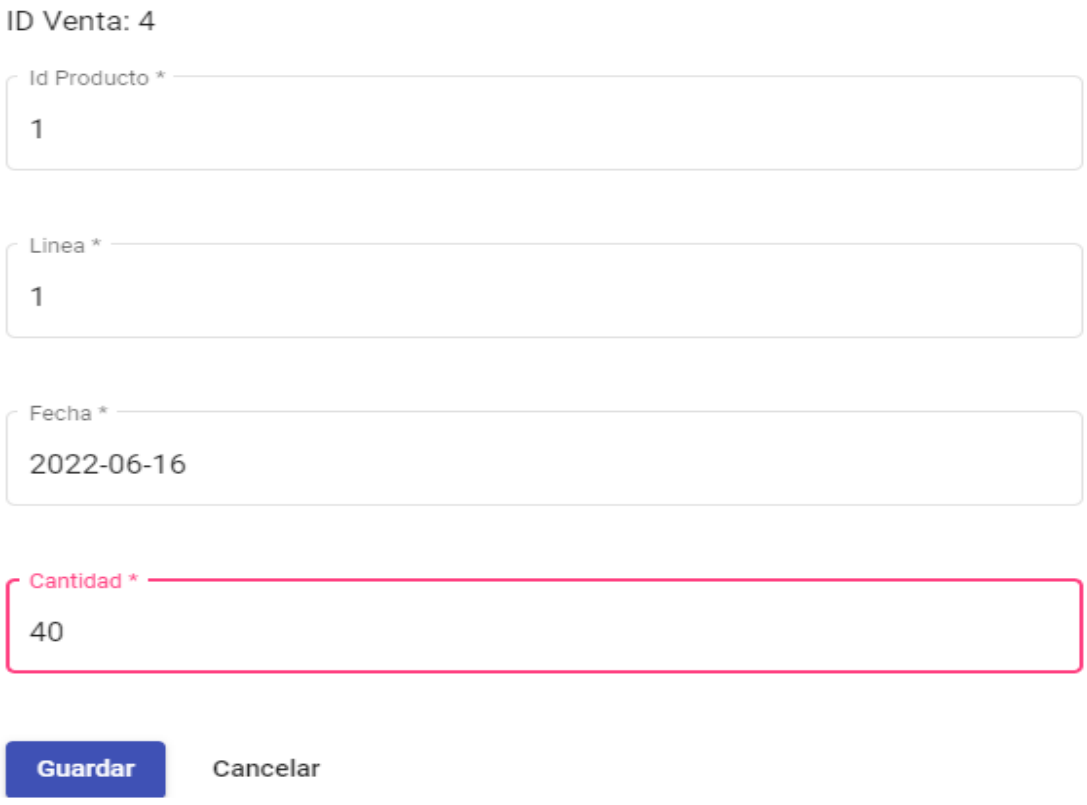
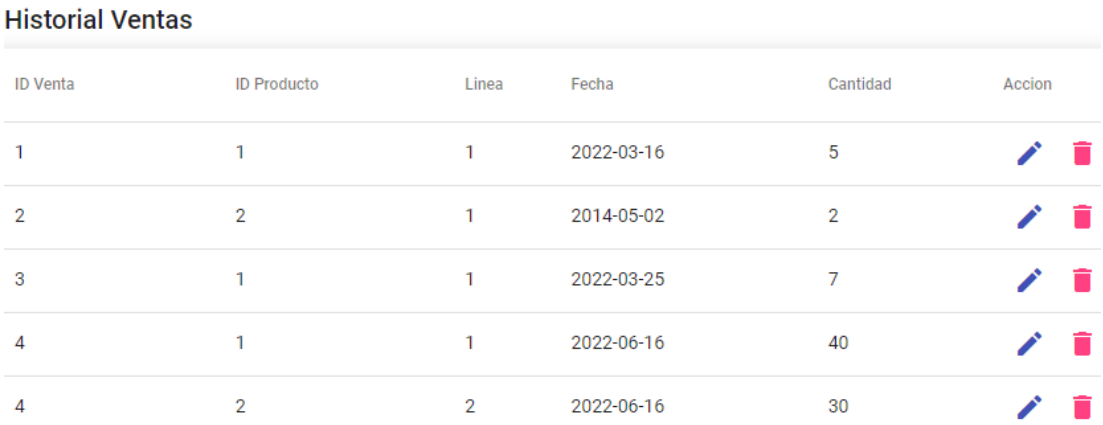


Gráfico 48

Prueba Unitaria: Modificar Venta – Exitoso



En esta prueba unitaria, se va a evidenciar como la aplicación web es capaz de eliminar una venta que ya haya sido registrada previamente, es por eso por lo que se va a eliminar el producto dos de la venta cuatro, venta la cual se puede ver que ya consta en la tabla de ventas tal y como se observa en el Gráfico 49, además se puede evidenciar un icono de un basurero rosa el cual permite abrir un cuadro en donde se puede eliminar dicha venta como se muestra en el Gráfico 50, aquí se va a observar todos los datos de dicha venta que se va a eliminar; por lo tanto, para eliminar la venta se debe presionar el botón “Eliminar”, finalmente en el Gráfico 51 se puede ver como el producto dos de la venta cuatro ya no se encuentra en la tabla.

Gráfico 49

Prueba Unitaria: Eliminar Venta – Cuadro

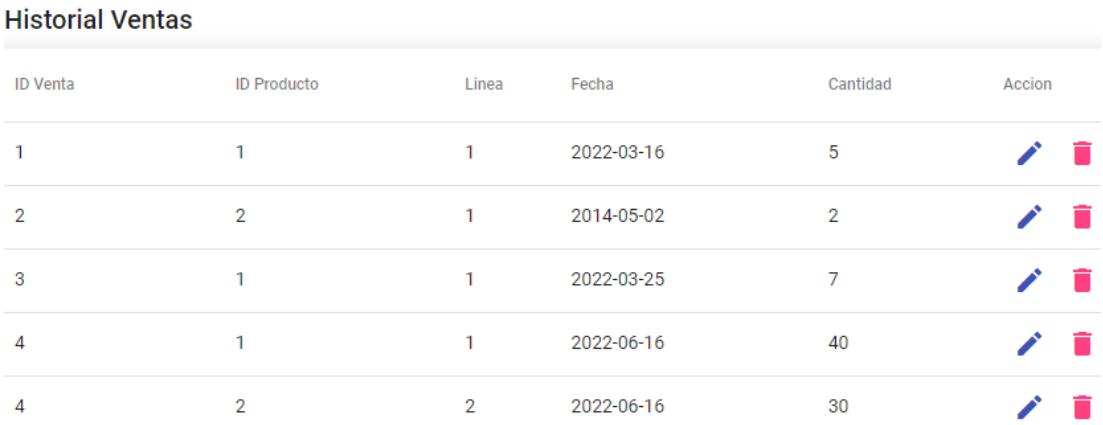


Gráfico 50

Prueba Unitaria: Eliminar Venta – Eliminar

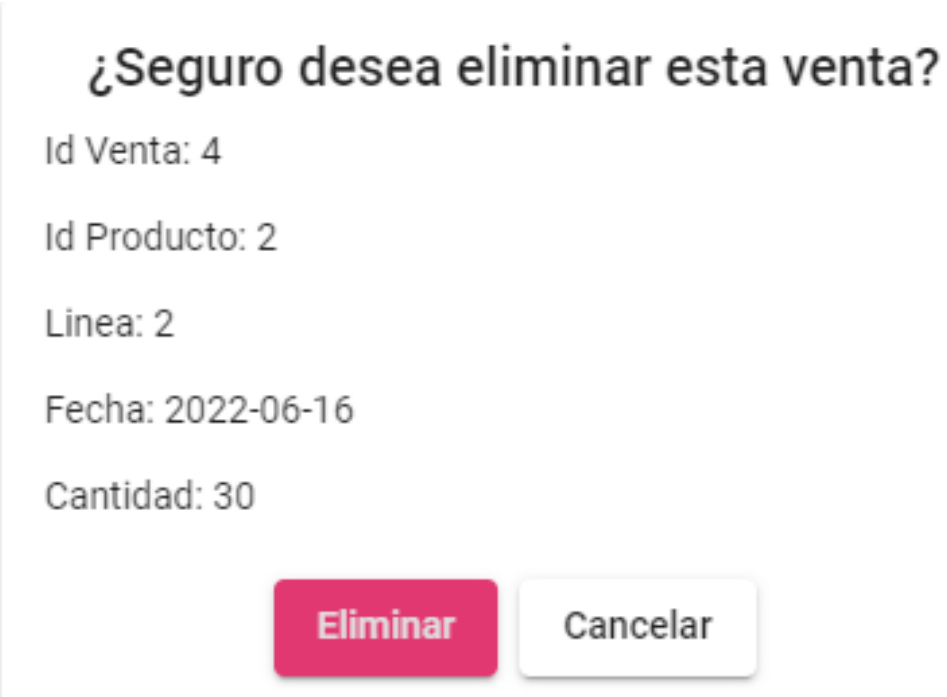
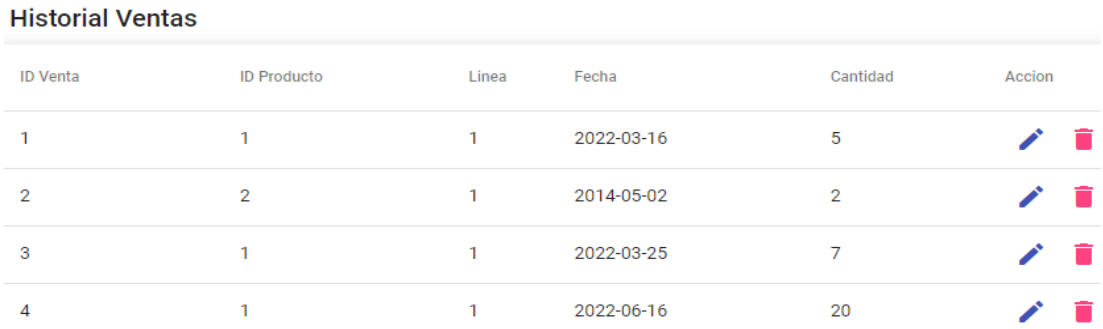


Gráfico 51

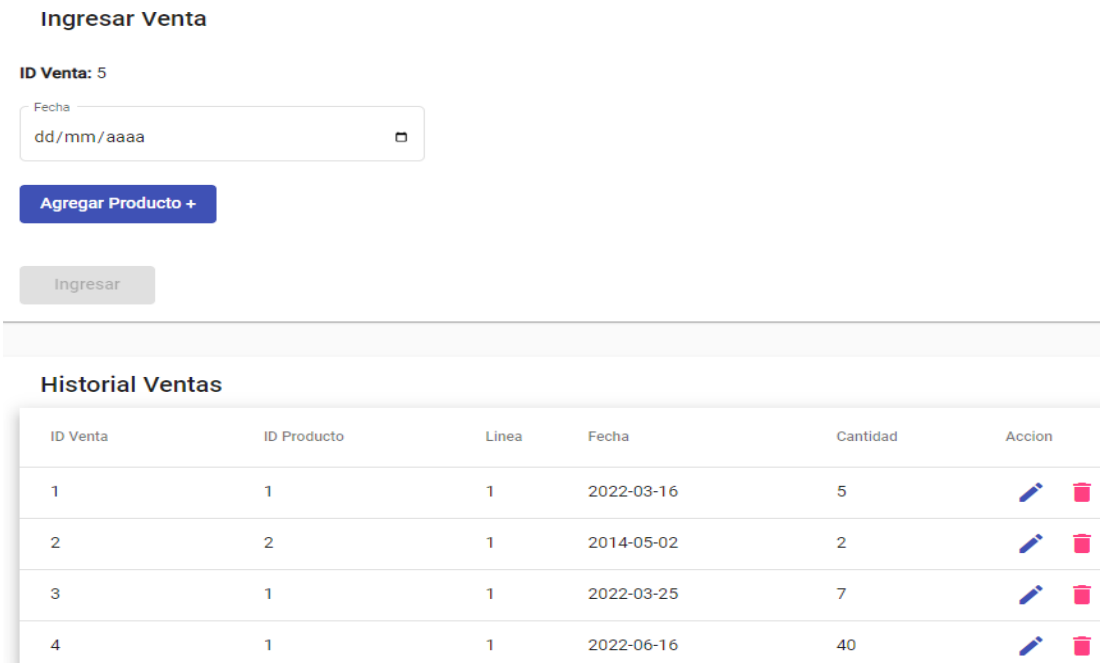
Prueba Unitaria: Eliminar Venta – Exitoso



Como última prueba unitaria para la funcionalidad de venta, se puede evidenciar en el Gráfico 52 que apenas se abre la ventana de venta, también se carga una tabla con toda la información de todas las ventas registradas hasta el momento, permitiendo así al usuario consultar de manera general todas las ventas que ha hecho.

Gráfico 52

Prueba Unitaria: Consultar Venta



A continuación, se va a mostrar el ingreso de una compra a través de un formulario, este mismo también cuenta con un botón de "Agregar Producto +" el cual permite ingresar más de un producto en una compra permaneciendo en la misma ventana, cabe destacar que la aplicación web dejará ingresar una compra solo sí cada campo del formulario se llena correctamente, es por eso por lo que en el Gráfico 53 no se habilita el botón de ingresar, mientras que en el Gráfico 54 ya se habilita, ya que todos los campos del formulario se encuentran llenados correctamente; por ende, al usuario se le permite ingresar la compra, por último, vale la pena mencionar que el "Id Compra" se actualiza solo y por lo tanto el "Id Compra" correspondiente para esta prueba unitaria es el número cuatro ya que ya existen otras compras que han sido ingresadas previamente.

Gráfico 53

Prueba Unitaria: Ingresar Compra - Antes

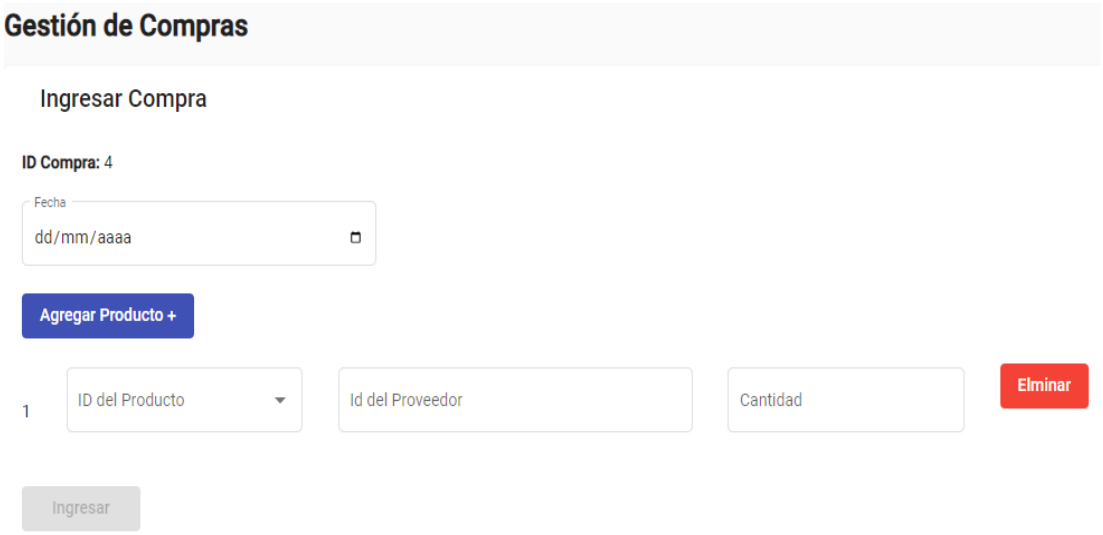
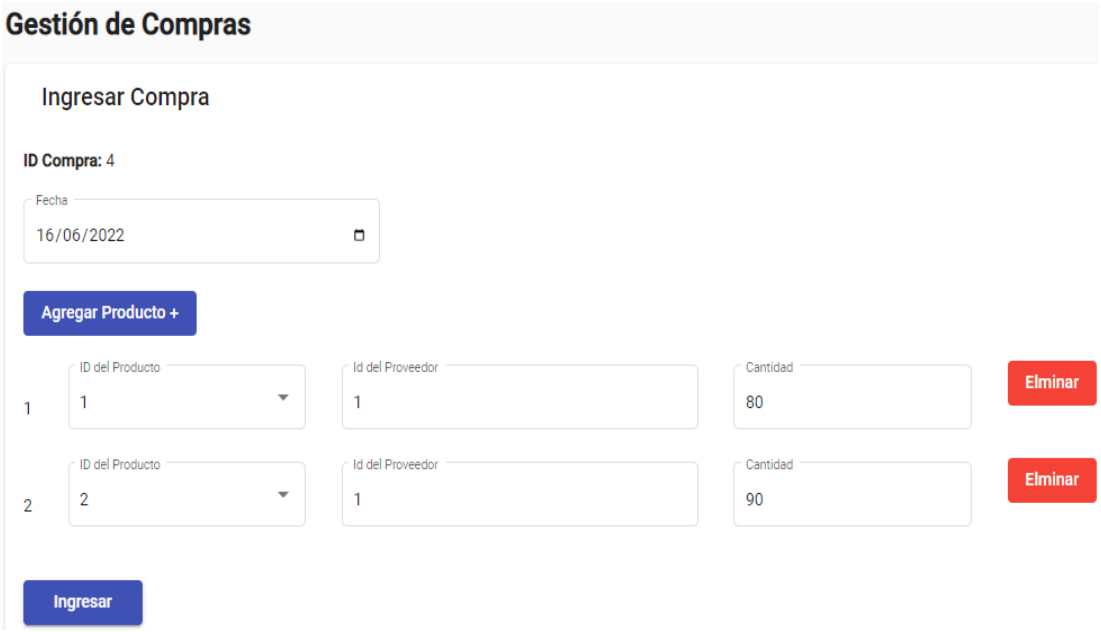


Gráfico 54

Prueba Unitaria: Ingresar Compra - Después



En la siguiente prueba unitaria, se va a mostrar como la aplicación web es capaz de modificar una compra que ya haya sido registrada previamente, es por eso por lo que se va a modificar la compra con Id cuatro, compra que fue ingresada en la prueba anterior, la cual se puede ver que ya consta en la tabla de compra tal y como se observa en el Gráfico 55, además se puede evidenciar un icono de un lápiz azul el cual permite abrir un cuadro en donde se puede modificar dicha compra como se muestra en el Gráfico 56, aquí se va a cambiar la cantidad del producto uno en donde dice “80” por “100” , una vez modificado el campo se procede a aplicar el cambio a través del botón “Guardar”, finalmente en el Gráfico 57 se puede ver como la compra del producto uno con Id cuatro ya se encuentra modificada tal y como se lo requería.

Gráfico 55

Prueba Unitaria: Modificar Compra – Cuadro

Gráfico 56

Prueba Unitaria: Modificar Compra – Guardar

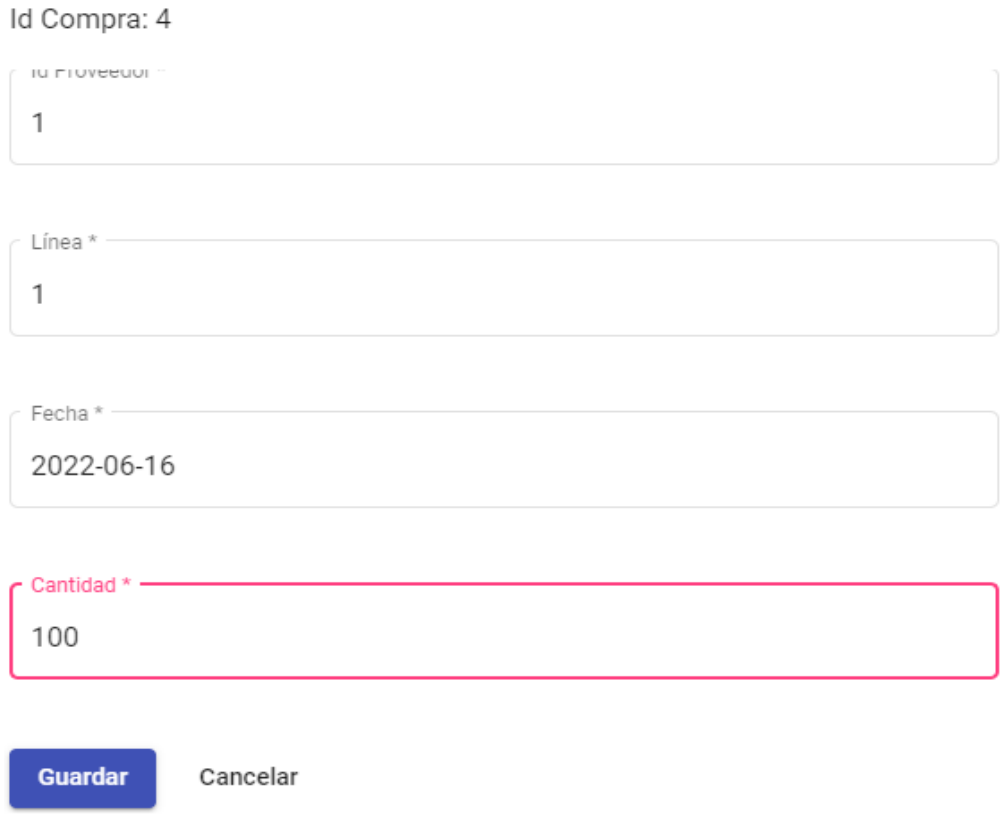
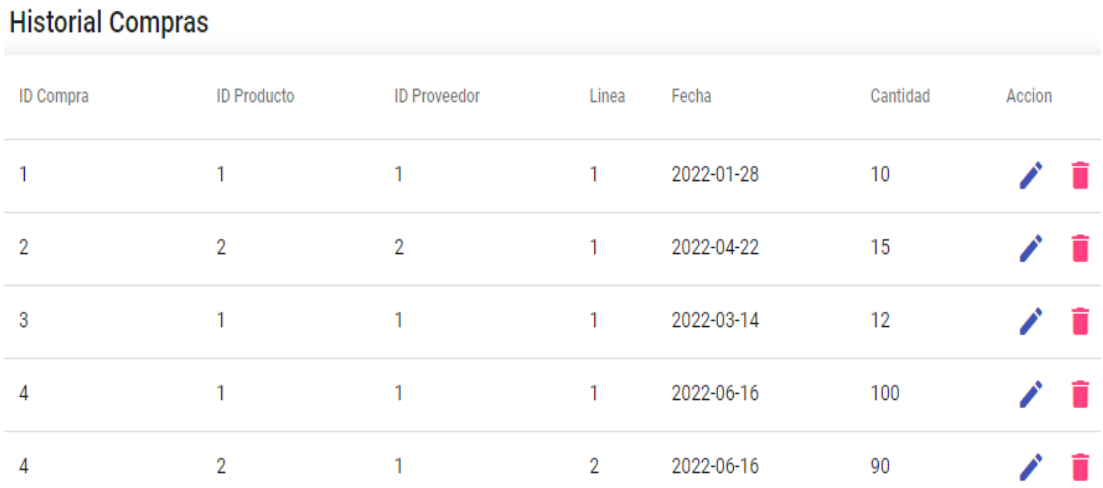


Gráfico 57

Prueba Unitaria: Modificar Compra – Exitoso



En esta prueba unitaria, se va a evidenciar como la aplicación web es capaz de eliminar una compra que ya haya sido registrada previamente, es por eso por lo que se va a eliminar el producto dos de la compra cuatro, compra la cual se puede ver que ya consta en la tabla de compras tal y como se observa en el Gráfico 58, además se puede evidenciar un icono de un basurero rosa el cual permite abrir un cuadro en donde se puede eliminar dicha compra como se muestra en el Gráfico 59, aquí se va a observar todos los datos de dicha compra que se va a eliminar; por lo tanto, para eliminar la compra se debe presionar el botón “Eliminar”, finalmente en el Gráfico 60 se puede ver como el producto dos de la compra cuatro ya no se encuentra en la tabla.

Gráfico 58

Prueba Unitaria: Eliminar Compra – Cuadro

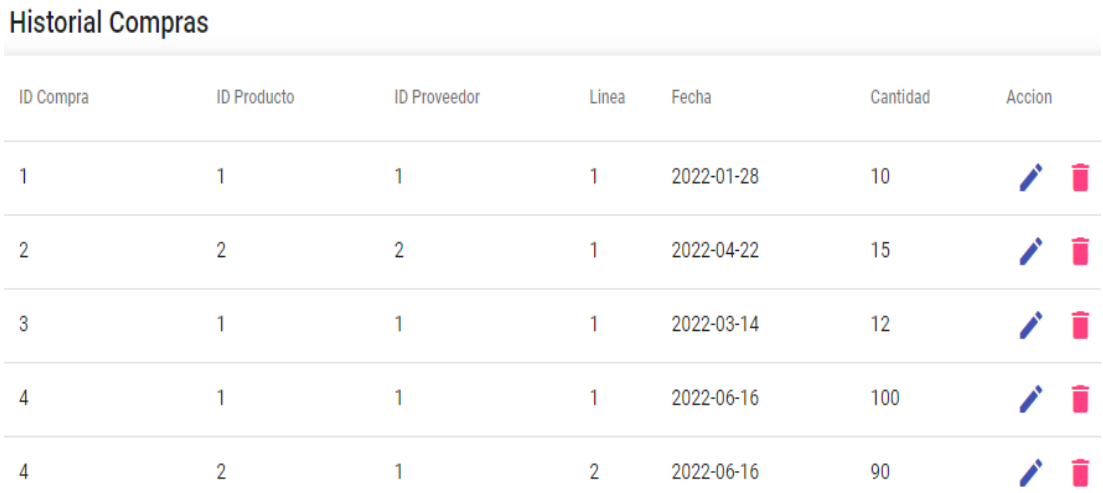


Gráfico 59

Prueba Unitaria: Eliminar Compra – Eliminar

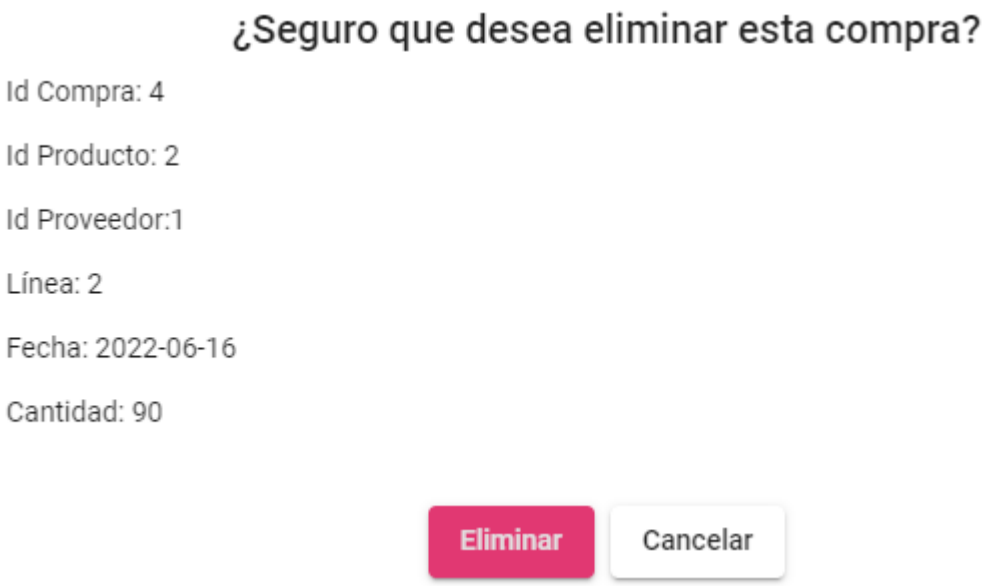


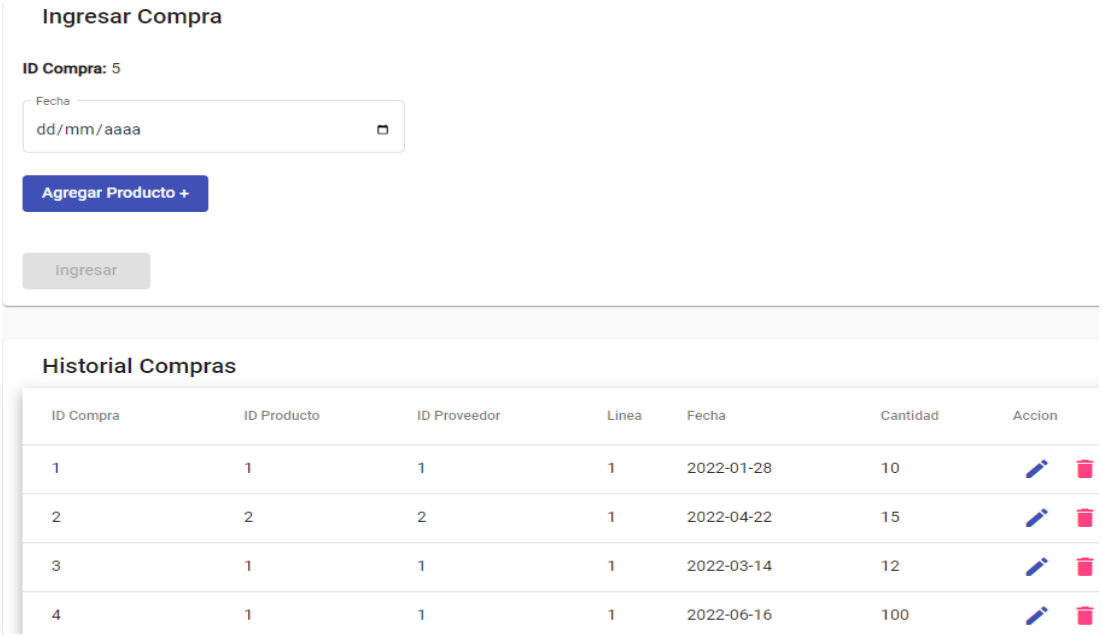
Gráfico 60

Prueba Unitaria: Eliminar Compra – Exitoso

Como última prueba unitaria para la funcionalidad de compra, se puede evidenciar en el Gráfico 61 que apenas se abre la ventana de compra, también se carga una tabla con toda la información de todas las compras registradas hasta el momento, permitiendo así al usuario consultar de manera general todas las compras que ha hecho.

Gráfico 61

Prueba Unitaria: Consultar Compra



Como última prueba unitaria, se tiene la ventana de ganancias la cual cuenta con un campo para que el usuario escoja un mes y un año con el fin de calcular sus ingresos, egresos y total en dicha fecha, tal y como se muestra en el Gráfico 62; por ende, una vez el usuario escoge el mes y el año que en este caso se ha escogido el mes de marzo del año 2022, la aplicación web procede a mostrar automáticamente sus ingresos, egresos y total que son $3.75, $3.00 y $0.75 respectivamente en dicho mes y año, tal y como se evidencia en el Gráfico 63.

Gráfico 62

Prueba Unitaria: Ganancias Antes

Gráfico 63

Prueba Unitaria: Ganancias Después

Capítulo 4: Conclusiones y Recomendaciones

4.1 Conclusiones

En este trabajo de disertación se identificó la metodología que se puede usar para realizar el desarrollo de una aplicación web de gestión de stock de productos para el mini market “Paola”, la cual es la metodología WSDM, dado que su objetivo es sacar el producto en un corto tiempo ya que se enfoca más en el desarrollo de este misma que en la documentación.

En conclusión, la metodología WSDM es la más rentable para el desarrollo de una aplicación web de gestión de stock de productos para el mini market “Paola” en comparación a la metodología SCRUM y la tradicional, debido a que resulta ser más ágil que la tradicional y no se necesita de un determinado número de roles como en la SCRUM.

Como se ha podido observar en este trabajo de disertación, se determinó los requerimientos funcionales y no funcionales para la aplicación mediante el uso de la propia metodología WSDM y el estándar IEEE 830, ya que así se pudo detallar y delimitar un poco más acerca de lo que se esperaba del producto final.

En síntesis, se desarrolló una aplicación web para gestionar el stock de los productos del mini market “Paola”, la cual se la dividió en front-end y back-end porque de esta manera se garantiza un mejor control de errores y lógica, ya que se tiene un código más ordenado.

Finalmente, se desarrolló una aplicación web para gestionar el stock de los productos del mini market “Paola” con una arquitectura REST porque de esta forma se generan APIs que mejoran el rendimiento de esta misma por sus servicios reutilizables.

4.2 Recomendaciones

Una vez concluido el presente trabajo, se recomienda investigar más sobre la metodología que vaya más acorde al caso de estudio, ya que algunas veces se puede necesitar una metodología que provea más documentación que agilidad o viceversa.

En base a los resultados obtenidos, se propone comparar varias metodologías por el hecho que cada una de estas ofrece distintas ventajas que pueden hacer que un proyecto sea efectuado de una manera más acorde a lo solicitado.

Se aconseja indagar un poco más acerca de los distintos estándares y técnicas existentes para levantar requerimientos, dado que no todas las metodologías de desarrollo contemplan una fase adecuada para el levantamiento de estos mismos.

Se sugiere también que se tome en cuenta dividir el desarrollo de una aplicación web de esta índole en front-end y back-end, con el fin de facilitar la resolución de problemas y errores que surjan durante el desarrollo de esta misma.

Antes de finalizar, se recomienda que se investigue un poco más sobre las arquitecturas que propongan un desarrollo mediante APIs, debido a que estas ayudan bastante en el rendimiento de la aplicación y la velocidad de comunicación.

Bibliografía

Aguirre Barrera, J., & Aguirre Barrera, S. (2021). Metodologías para el desarrollo de Proyectos.

Bassantes Quingatuña, K. L. (2020). Gestión de stocks y su relación en la rentabilidad de la empresa Autorepuestos Universal Importaciones (Bachelor's thesis, Pontificia Universidad Católica del Ecuador).

Barahona Alcívar, S. B., & Rebutti Izquierdo, J. A. (2019). Plataforma tecnológica para contribuir a la planeación urbana de la ciudad de Guayaquil dirigido a la transportación, enfocado a la administración de base de datos Postgresql garantizando la disponibilidad 24/7 de la información, desarrollando esquemas de respaldos automáticos, monitoreo, depuración, seguridad y sincronización (Bachelor's thesis).

Bautista Téllez, D. F. Metodología SCRUM en complemento con otras herramientas ágiles para el desarrollo de proyectos y su utilidad en el área de construcción

Boduch, A., & Derks, R. (2020). React and React Native: A complete hands-on guide to modern web and mobile development with React. js. Packt Publishing Ltd.

Cerezo Goya, R. A. (2019). Desarrollo de un sistema web para la gestión de espacios físicos del Centro Deportivo Metropolitano Iñaquito (Bachelor's thesis, Quito, 2019.).

Cíceri, M. (2019). Introducción a Laravel: Aplicaciones robustas ya gran escala. RedUsers.

Cobo Jaramillo, N. M., & Alulema Flores, D. O. Diseño de una arquitectura para el control telemático de sistemas ciber físicos basada en microservicios y con tolerancia a fallos.

Diego Navarro, R., & Cabrera Lozada, R. D. (2020). Aplicación basada en arquitectura de microservicios.

Escobar Hernández, H. (2019). Sistema de información que controla y administra el inventario y las ventas de una pequeña empresa comercial.

Espinoza Pereda, J. C. (2019). Análisis de Framework Javascript y MVC con serializador de datos y su incidencia en rendimiento en los últimos 4 años.

González González, F., & Calero Castañeda, S. L. (2019). Comparación de las metodologías cascada y ágil para el aumento de la productividad en el desarrollo de software (Doctoral dissertation, Universidad Santiago de Cali).

Hadida, S., & Troilo, F. (2020). La agilidad en las organizaciones: trabajo comparativo entre metodologías ágiles y de cascada en un contexto de ambigüedad y transformación digital (No. 756). Serie Documentos de Trabajo.

Hernández-Salazar, Edwin. Beltrán, Carlos Alberto. SCRUM. Un enfoque práctico de metodología ágil para la ingeniería de software. Revista Tecnol.Investig.Academia TIA. ISSN: 23448288, 8 (2), pp. 61-73. Bogotá-Colombia

Menacho Huisa, D. L. (2021). Sistema web para el proceso de ventas en la Botica "Pharma Medical".

Nicaragua, E. (2018). Metodología de la investigación e investigación aplicada para Ciencias Económicas y Administrativas. Revista de La Universidad Autónoma, 1-89.

Nicole, E., & Guevara, G. (2020). Análisis comparativo del ciclo de vida en el Método de desarrollo de software Híbrido EssUp versus RUP y SCRUM: Una revisión sistemática de la literatura.

Ríos, J. R. M., Ordóñez, M. P. Z., Segarra, M. J. C., & Zerda, F. G. G. (2018). Comparación de metodologías en aplicaciones web. 3C Tecnología: glosas de innovación aplicadas a la pyme, 7(1), 1-19.

Salazar, N. I., & Costaguta, R. Evaluación de algunas herramientas utilizadas por estudiantes universitarios en actividades grupales.

Sánchez Jordán, A. K. (2020). Estudio para la implementación de un sistema informático para el registro de cobro de pensiones de la Unidad Educativa Babahuyus de la ciudad de Babahoyo (Bachelor's thesis, Babahoyo, UTB-FAFI 2020).

Sosa Aguilar, J. J. Trabajo de investigación del aumento de la calidad de servicios y productos a través de aplicación de metodologías ágiles.

Tacuri Ordoñez, J. C., & Rodríguez Alcívar, C. J. (2020). Desarrollo de una aplicación web para la actualización de inventario en el negocio de "Plasti-Quil" en la ciudad de Milagro (Bachelor's thesis).

Tito Chulde, P. A. (2020). Desarrollo de un sistema de gestión y control de procesos para la microempresa bazar y papelería San Antonio utilizando Spring Boot (Bachelor's thesis).

Tubay Aveiga, B. S. (2020). DISEÑO DE UNA PÁGINA WEB INFORMATIVA PARA LA PUBLICIDAD DE LA EMPRESA MAKTUB DE LA CIUDAD

DE GUAYAQUIL 2019 (Bachelor's thesis, Instituto Superior Tecnológico Bolivariano de Tecnología.).

Anexos

Anexo 1

Bitácora en donde constan las reuniones y los temas mantenidos con el encargado del Mini Market "Paola" para el desarrollo de la aplicación web.

Bitácora de las reuniones mantenidas con el responsable del Mini Market "Paola"

Fecha programada	Actividad	Descripción	Responsable	Firma
26-04-2022	Levantamiento de requerimientos funcionales	Se trató todo lo que respecta con el levantamiento de requerimientos funcionales	Alex Angulo	
28-04-2022	Levantamiento de requerimientos no funcionales	Se trató todo lo que respecta con el levantamiento de requerimientos no funcionales.	Alex Angulo	
03-05-2022	Corrección de levantamiento de requerimientos	Se corrigió los malentendidos en el levantamiento de requerimientos (funcionales y no fun.)	Alex Angulo	
05-05-2022	Clasificasión y descripción de los grupos de usuarios	Se habló a detalle acerca de los procesos y las personas que trabajan aquí (roles y restricciones).	Alex Angulo	
11-05-2022	Diseño de la aplicación web	Se mostró un boceto de como se vería la interfaz de la aplicación web, tanto como y donde funcionaría cada cosa.	Alex Angulo	
13-05-2022	Corrección del diseño de la aplicación web	Se corrigió los bocetos de las interfaces tal y como lo prefirió el responsable, con el fin que sea lo más intuitiva posible.	Alex Angulo	

Anexo 2

El siguiente código fuente contempla a todo lo que respecta con el desarrollo del front-end de la aplicación de gestión de stock de productos del mini market “Paola”.

```
<ng-container [formGroup]="form">
      <h1 style="font-weight: bold">Gestión de Compras</h1>
   <mat-card  >
      <mat-card-header>
         <mat-card-title>
            Ingresar Compra
         </mat-card-title>
      </mat-card-header>
      <br>
      <mat-card-content style="font-weight: bold">
        ID Compra:
       <mat-label style="font-weight: normal" input type="text" disabled="disabled">
5 </mat-label>
      </mat-card-content>

      <mat-card-content fxLayout="column wrap" >

            <div fxLayout="row wrap" fxLayoutGap="3%">
```

```
                <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-md="90">
                <mat-label>Fecha</mat-label>
                <input type="date" formControlName="COMFECHA" matInput >

                <mat-error>
                 Campo requerido.
                </mat-error>

              </mat-form-field>



          </div>

       <div class="example-button-row">
         <button mat-flat-button color="primary" (click)="agregarCompra()">Agregar Producto + </button>
        </div>
       <br>

                          <div class="form-group row" [formGroupName]="i" formArrayName="compra" *ngFor="let compra of compra.controls; let i = index" fxLayout="row wrap" fxLayoutGap="3%">
```

<mat-label fxLayoutAlign="center center" style="font-weight: normal" input type="text" disabled="disabled"> 1 </mat-label>

<mat-form-field appearance="outline" fxFlex.gt-md="20" fxFlex.lt-md="90">

<mat-label>ID del Producto </mat-label>

<mat-select formControlName="idProducto" (selectionChange)="IdChange(this.form.value.idProducto)" matInput>

<ng-container *ngFor="let productos of productoId">

<mat-option [value]="productos.idProducto">{{productos.idProducto}}</mat-option>

</ng-container>

</mat-select>

<mat-error>

Campo Requerido.

</mat-error>

</mat-form-field>

<mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-md="90">

<mat-label>Id del Proveedor</mat-label>

```
                <input  formControlName="idProveedor" maxlength="4" matInput >

                <mat-error>
                   Campo requerido.
                </mat-error>

              </mat-form-field>

                <mat-form-field appearance="outline" fxFlex.gt-md="20" fxFlex.lt-md="90">
                <mat-label>Cantidad</mat-label>
                 <input type="number" formControlName="COMDCANTIDAD" matInput >
                                                                    <mat-error *ngIf="form.controls['COMDCANTIDAD'].hasError('required')">
                   Campo Requerido.
                </mat-error>
              </mat-form-field>

            <div >
```

```
                    <button class="btn btn-danger" color="warn" mat-flat-button (click)="removerCompra(i)">Elminar</button>
                </div>
            </div>



        <br>


        <div>
                <button [disabled]="!form.valid" (click)="ingresar()" mat-raised-button color='primary' fxFlex.gt-md="10" fxFlex.lt-md="90">
                Ingresar
            </button>
        </div>


    </mat-card-content>
</mat-card>


<mat-card style="margin-top: 3%;margin-block-end: 10%;">
    <mat-card-header>
        <mat-card-title >
            Historial Compras
```

```
      </mat-card-title>

   </mat-card-header>
   <mat-card-content fxLayout="column">

      <div>
            <table mat-table [dataSource]="transactions" class="mat-elevation-z8" fxFlex.gt-md="100%">                    <!-- Item Column -->

         <ng-container matColumnDef="IdCompra">
            <th mat-header-cell *matHeaderCellDef> ID Compra</th>
             <td mat-cell *matCellDef="let transaction"> {{transaction.idCompra}} </td>
            <td mat-footer-cell *matFooterCellDef> Total </td>
         </ng-container>

         <ng-container matColumnDef="ID del Producto">
            <th mat-header-cell *matHeaderCellDef> ID Producto</th>
                              <td mat-cell *matCellDef="let transaction"> {{transaction.compraDetalleModelo[0].idProducto }} </td>

         </ng-container>
```

```
<ng-container matColumnDef="IdProv">
   <th mat-header-cell *matHeaderCellDef> ID Proveedor</th>
    <td mat-cell *matCellDef="let transaction"> {{transaction.idProveedor }} </td>

</ng-container>

<ng-container matColumnDef="ComLinea">
   <th mat-header-cell *matHeaderCellDef> Linea </th>
                    <td mat-cell *matCellDef="let transaction"> {{transaction.compraDetalleModelo[0].comdLinea}} </td>
   <td mat-footer-cell *matFooterCellDef> Total </td>
</ng-container>

<ng-container matColumnDef="ComFecha">
   <th mat-header-cell *matHeaderCellDef> Fecha </th>
    <td mat-cell *matCellDef="let transaction"> {{transaction.comFecha}} </td>

</ng-container>
<ng-container matColumnDef="ComCantidad">
   <th mat-header-cell *matHeaderCellDef> Cantidad </th>
                    <td mat-cell *matCellDef="let transaction" > {{transaction.compraDetalleModelo[0].comCantidad}} </td>
```

```
    </ng-container>

    <ng-container matColumnDef="accion">
      <th mat-header-cell  *matHeaderCellDef >Accion</th>
      <td mat-cell *matCellDef="let row2;let i=index;">
        <button mat-icon-button color='primary' >
          <mat-icon>edit</mat-icon> </button>
        <button mat-icon-button color='accent' >
          <mat-icon >delete</mat-icon>
        </button>
      </td>

    </ng-container>
    <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
    <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
  </table>

</div>
  <div  align =" center"fxFlex.gt-md="10" fxFlex.lt-md="90">
    <button (click)="cargarDatos()" mat-icon-button color='primary' >
      <mat-icon>
        autorenew
```

```
                </mat-icon>
              </button>
            </div>
          </mat-card-content>

    </mat-card>


</ng-container>
.label{
    font-size: large;
}
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { ComprasComponent } from './compras.component';

describe('ComprasComponent', () => {
  let component: ComprasComponent;
  let fixture: ComponentFixture<ComprasComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ ComprasComponent ]
```

```
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(ComprasComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});

import { Component, OnInit } from '@angular/core';

import { FormGroup, FormControl, Validators, FormBuilder, FormArray } from '@angular/forms';
import { Subscription } from 'rxjs';
import { ServiciosService } from 'src/app/servicios.service';
interface Transaction {
  idCompra: string,
```

```
  idProducto: string,
  idProveedor:String,
  comFecha: Date,
  comCantidad: number,
  comdLinea: string,
  compraDetalleModelo:String[]



}

@Component({
  selector: 'app-compras',
  templateUrl: './compras.component.html',
  styleUrls: ['./compras.component.scss']
})
export class ComprasComponent implements OnInit {

        displayedColumns: string[] = ['IdCompra', 'ID del Producto','IdProv','ComLinea','ComFecha','ComCantidad','accion'];
  transactions: Transaction[] = [];
  index: number=0;
  constructor(private servicio: ServiciosService, private formBuilder:FormBuilder) { }
```

```
 cedulasClientes : any[] =[]
 productoId : any[] =[]
total:number=0;
 ngOnInit(): void {
   this.servicio.findClients().subscribe((res) => {
     this.cedulasClientes = res;
     console.log('d',res)
   })
   this.servicio.findProducto().subscribe((res) => {
     this.productoId= res;
     console.log('f',res)
   })
   this.cargarDatos()

 }
 form = new FormGroup({
   IDVENTA: new FormControl('', [Validators.required]),
   COMFECHA: new FormControl('', [Validators.required]),
   IDPRODUCTO: new FormControl('', [Validators.required]),
   COMDCANTIDAD: new FormControl('', [Validators.required]),
   VENLINEA: new FormControl('', [Validators.required]),
```

```
  IDPROV: new FormControl('', [Validators.required]),



 })
 cargarDatos(){
  this.servicio.findCompras().subscribe((res) => {
   this.transactions = res;
   console.log("compras", this.transactions)


  })



 }
 CantidadChange(id:number){
            //this.total = (this.productoId[this.form.value.IDPRODUCTO-
1].PRECIOPRODUCTO)*id
  for(let i =0;i<this.productoId.length;i++){
   if(this.productoId[i].IDPRODUCTO==this.form.value.IDPRODUCTO){
                                                                this.total=
this.productoId[i].PRECIOPRODUCTO*this.form.value.COMDCANTIDAD
   }
  }
  console.log("total",this.total)
```

```
  }
  IdChange(id:number){
                                /*                this.total=(this.productoId[id-
1].PRECIOPRODUCTO*this.form.value.COMDCANTIDAD)
    this.form.value.TOTALVENTA=this.total*/
    for(let i =0;i<this.productoId.length;i++){
      if(this.productoId[i].IDPRODUCTO==this.form.value.IDPRODUCTO){
                                                                    this.total=
this.productoId[i].PRECIOPRODUCTO*this.form.value.COMDCANTIDAD
      }
    }
  }
  ingresar(){
    this.form.value.TOTALVENTA=this.total

    this.servicio.insertCompras(this.form.value).subscribe(
      response=>{
        console.log(response);
      },
      error=>{
        console.log(error);
      });
      console.log("datos ",this.form.value)
```

```
}
editar(){


}


get compra(){
  return this.registerForm.get('compra') as FormArray;
}

registerForm = this.formBuilder.group({

  compra: this.formBuilder.array([])
});

agregarCompra(){
  const telefonoFormGroup  = this.formBuilder.group({
    IDPRODUCTO:'',
    IDPROV:'',
    COMDCANTIDAD:''
  });
  this.compra.push(telefonoFormGroup);
}
```

```
 removerCompra(indice: number) {
   this.compra.removeAt(indice);
 }
}
<div >
   <mat-drawer-container style="height: 90vh;"  autosize>
      <mat-drawer mode="side" opened class="drawer">
         <ul class="ul" >
            <li style="    padding-bottom: 30px;
            ">
                        <a  href='formulario/producto'  style="color:  aliceblue;text-decoration:none"><mat-icon >store</mat-icon>   Productos</a>
         </li >
         <li style="    padding-bottom: 30px;
            ">
                     <a  href='formulario/ingresar-cliente'  style="color:  aliceblue;text-decoration:none"><mat-icon >people</mat-icon>   Proveedores</a>

            </li>
            <li style="    padding-bottom: 30px;
            ">
                        <a  href='formulario/ventas'  style="color:  aliceblue;text-decoration:none"><mat-icon>list_alt</mat-icon>   Ventas</a>
            </li>
```

```
            <li style="    padding-bottom: 30px;
            ">
                        <a href='formulario/compras' style="color: aliceblue;text-decoration:none"> <mat-icon>shopping_cart</mat-icon>  Compras</a>

            </li>
            <li style="    padding-bottom: 30px;
            ">
                        <a href='formulario/ganancias' style="color: aliceblue;text-decoration:none"> <mat-icon>attach_money</mat-icon>  Ganancias</a>
            </li>
          </ul>
       </mat-drawer>
       <mat-drawer-content class="content">
          <router-outlet></router-outlet>
       </mat-drawer-content>
     </mat-drawer-container>


</div>
.productos{
   color:white;
   margin-top: 1.5em;
```

```
    background-color: rgb(182, 182, 182);
    height: 2em;
    text-align:center;
}
.content{
    padding-top: 1.5em;

}
.drawer{
    width: 17%;
    background-color: #2F4CED ;
}
li:hover{
    background-color: #4D67FF;
    color: white;
    padding:5px
}
li{
    padding:5px;
    text-align:left ;
}
.ul{
    list-style:none;
```

```
    padding-left: 0;
    font-size:17px;

}
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { FormsGeneralComponent } from './forms-general.component';
describe('FormsGeneralComponent', () => {
  let component: FormsGeneralComponent;
  let fixture: ComponentFixture<FormsGeneralComponent>;
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ FormsGeneralComponent ]
    })
    .compileComponents();
  });
  beforeEach(() => {
    fixture = TestBed.createComponent(FormsGeneralComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
```

```
  });
});
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-forms-general',
  templateUrl: './forms-general.component.html',
  styleUrls: ['./forms-general.component.scss']
})
export class FormsGeneralComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
  }
}
<ng-container [formGroup]="form">
     <h1 style="font-weight: bold;">Gestión de Proveedores</h1>
    <mat-card >
       <mat-card-header>
          <mat-card-title>
             Ingresar Proveedor
          </mat-card-title>
       </mat-card-header>
       <mat-card-content fxLayout="column wrap" >
          <div fxLayout="row wrap" fxLayoutGap="3%">
```

```
                <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-
md="90">
                <mat-label>Id del Proveedor</mat-label>
                <input  formControlName="idProveedor" maxlength="4" matInput >

                <mat-error>
                   Campo requerido.
                </mat-error>
              </mat-form-field>

                <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-
md="90">
                <mat-label>Proveedor</mat-label>
                 <input formControlName="provNombre" matInput maxlength="50"
matInput>
                <mat-error>
                   Campo Requerido.
                </mat-error>
              </mat-form-field>
                <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-
md="90">
                <mat-label>Nombre del Representante</mat-label>
                                <input  formControlName="provNombreRepresentante"
matInput  maxlength="50" matInput>
```

```
            <mat-error>
              Campo Requerido.
            </mat-error>
          </mat-form-field>
            <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-md="90">
            <mat-label>Cédula</mat-label>
            <input type="number" formControlName="provCedulaRepresentante" matInput maxlength="10" matInput>
            <mat-error>
              Campo Requerido.
            </mat-error>
          </mat-form-field>
            <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-md="90">
            <mat-label>Correo Electrónico</mat-label>
            <input formControlName="provCorreo" matInput maxlength="40" matInput >
            <mat-error *ngIf="form.controls['provCorreo'].hasError('email')">
              Email Inválido.
            </mat-error>
            <mat-error *ngIf="form.controls['provCorreo'].hasError('required')">
              Campo Requerido.
            </mat-error>
```

```
        </mat-form-field>

            <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-md="90">
            <mat-label>Teléfono</mat-label>
                    <input type="number" formControlName="provTelefono" matInput maxlength="10" matInput>
            <mat-error>
              Campo Requerido.
            </mat-error>
          </mat-form-field>
            <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-md="90">
            <mat-label>Dirección</mat-label>
             <input formControlName="provDireccion" matInput maxlength="50" matInput >
            <mat-error>
              Campo Requerido.
            </mat-error>
          </mat-form-field>
            <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-md="90">
            <mat-label>Celular</mat-label>
            <input formControlName="provCelular" matInput maxlength="10" >
```

```
                <mat-error>
                    Campo Requerido.
                </mat-error>
            </mat-form-field>
        </div>
        <div>
            <button [disabled]="!form.valid" (click)="createCliente()" mat-raised-button color='primary' fxFlex.gt-md="10" fxFlex.lt-md="90">
                Ingresar
            </button>
        </div>
    </mat-card-content>
</mat-card>
<mat-card style="margin-top: 3%;margin-block-end: 10%;">
    <mat-card-header>
        <mat-card-title >
            Proveedores Existentes
        </mat-card-title>

    </mat-card-header>
    <mat-card-content fxLayout="column">

        <div>
```

```
<table mat-table [dataSource]="transactions" class="mat-elevation-z8" fxFlex.gt-md="100%">
<!-- Item Column -->
<ng-container matColumnDef="IdProv">
<th mat-header-cell *matHeaderCellDef> ID del Proveedor </th>
<td mat-cell *matCellDef="let transaction"> {{transaction.idProveedor}} </td>
<td mat-footer-cell *matFooterCellDef> Total </td>
</ng-container>
<ng-container matColumnDef="ProvNombre">
<th mat-header-cell *matHeaderCellDef> Proveedor </th>
<td mat-cell *matCellDef="let transaction"> {{transaction.provNombre}} </td>
<td mat-footer-cell *matFooterCellDef> Total </td>
</ng-container>

<!-- Cost Column -->
<ng-container matColumnDef="ProvNomRepre">
<th mat-header-cell *matHeaderCellDef> Nombre del Representante </th>
<td mat-cell *matCellDef="let transaction"> {{transaction.provNombreRepresentante }} </td>

</ng-container>
```

```
<ng-container matColumnDef="ProvCedula">
  <th mat-header-cell *matHeaderCellDef> Cédula  </th>
                    <td mat-cell *matCellDef="let transaction">
{{transaction.provCedulaRepresentante}} </td>

</ng-container>
<ng-container matColumnDef="ProvCorreo">
  <th mat-header-cell *matHeaderCellDef> Correo </th>
                    <td mat-cell *matCellDef="let transaction">
{{transaction.provCorreo}} </td>

</ng-container>

<ng-container matColumnDef="ProvTelefono">
  <th mat-header-cell *matHeaderCellDef> Teléfono </th>
 <td mat-cell *matCellDef="let transaction"> {{transaction.provTelefono
}} </td>

</ng-container>

<ng-container matColumnDef="ProvDireccion">
  <th mat-header-cell *matHeaderCellDef> Dirección </th>
                    <td mat-cell *matCellDef="let transaction">
{{transaction.provDireccion }} </td>
```

```
</ng-container>

<ng-container matColumnDef="ProvCelular">
  <th mat-header-cell *matHeaderCellDef> Celular </th>
  <td mat-cell *matCellDef="let transaction"> {{transaction.provCelular }} </td>

</ng-container>

<ng-container matColumnDef="accion">
  <th mat-header-cell  *matHeaderCellDef >Acción</th>
  <td mat-cell *matCellDef="let row2;let i=index;">
    <button mat-icon-button color='primary' (click)="editar(i, row2.idProveedor, row2.provNombre, row2.provNombreRepresentante,row2.provCedulaRepresentante,row2.provCorreo, row2.provTelefono,row2.provDireccion,row2.provCelular )">
      <mat-icon>edit</mat-icon> </button>
    <button mat-icon-button color='accent' (click)="deleteItem(i, row2.idProveedor, row2.provNombre, row2.provNombreRepresentante,row2.provCedulaRepresentante,row2.provCorreo,row2.provTelefono,row2.provDireccion,row2.provCelular)">
      <mat-icon >delete</mat-icon>
    </button>
  </td>
```

```
            </ng-container>
            <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
            <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
          </table>

        </div>
        <div  align =" center"fxFlex.gt-md="10" fxFlex.lt-md="90">
          <button (click)="cargarDatos()" mat-icon-button color='primary' >
            <mat-icon>
              autorenew
            </mat-icon>
          </button>
        </div>
      </mat-card-content>

    </mat-card>
</ng-container>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { IngresarClienteComponent } from './ingresar-cliente.component';
```

```
describe('IngresarClienteComponent', () => {
  let component: IngresarClienteComponent;
  let fixture: ComponentFixture<IngresarClienteComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ IngresarClienteComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(IngresarClienteComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});

import { ChangeDetectorRef } from '@angular/core';
```

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms';
import { MatDialog } from '@angular/material/dialog';
import { ServiciosService } from 'src/app/servicios.service';
import { EditarDialogComponent } from '../editar-dialog/editar-dialog.component';
import { EliminarDialogComponent } from '../eliminar-dialog/eliminar-dialog.component';


interface Transaction {
  idProveedor: string,
  provNombre: string;
  provNombreRepresentante: string;
  provCedulaRepresentante: string;
  provCorreo: string;
  provTelefono: string;
  provDireccion: string;
  provCelular:string;

}
interface Datos{

}
```

```
@Component({
  selector: 'app-ingresar-cliente',
  templateUrl: './ingresar-cliente.component.html',
  styleUrls: ['./ingresar-cliente.component.scss']
})
export class IngresarClienteComponent implements OnInit {

  cliente: any[] = []
        displayedColumns: string[] = ['IdProv','ProvNombre', 'ProvNomRepre','ProvCedula','ProvCorreo','ProvTelefono','ProvDireccion','ProvCelular','accion'];
  transactions: Transaction[] = [];
  index: number=0;

    constructor(private servicio: ServiciosService,private changeDetectorRefs: ChangeDetectorRef,public dialog: MatDialog) { }
  form = new FormGroup({
    idProveedor: new FormControl('', [Validators.required]),
    provNombre: new FormControl('', [Validators.required]),
    provNombreRepresentante: new FormControl('', [Validators.required]),
    provCedulaRepresentante: new FormControl('', [Validators.required]),
    provCorreo: new FormControl('', [Validators.required,Validators.email]),
    provTelefono: new FormControl('', [Validators.required]),
    provDireccion: new FormControl('', [Validators.required]),
```

```
  provCelular: new FormControl('', [Validators.required]),



})




ngOnInit(): void {
  this.cargarDatos()

}

cargarDatos(){
  this.servicio.findClients().subscribe((res) => {
    this.transactions = res;
    console.log("cliente", this.transactions)

  })

}
createCliente():void{
  this.servicio.create(this.form.value).subscribe(
    response=>{
```

```
      console.log(response);


    },
    error=>{
      console.log(error);
    });
    console.log("datos ",this.form.value)


  }
```

editar(i: number, idProveedor: string, provNombre: string, provNombreRepresentante: string, provCedulaRepresentante: string,provCorreo: string,provTelefono: string,provDireccion: string, provCelular: string) {

```
  this.form.value.idProveedor = idProveedor;
  // index row is used just for debugging proposes and can be removed
  this.index = i;
  console.log(this.index);
  const dialogRef = this.dialog.open(EditarDialogComponent, {
```

data: {idProveedor: idProveedor, provNombre: provNombre, provNombreRepresentante: provNombreRepresentante, provCedulaRepresentante: provCedulaRepresentante,provCorreo: provCorreo,provTelefono: provTelefono ,provDireccion: provDireccion, provCelular: provCelular }

```
  });


  }
```

```
deleteItem(i: number, idProveedor: string, provNombre: string, provNombreRepresentante: string, provCedulaRepresentante: string,provCorreo: string,provTelefono: string,provDireccion: string , provCelular: string) {
  this.index = i;
  this.form.value.idProveedor = idProveedor;
  const dialogRef = this.dialog.open(EliminarDialogComponent, {
        data: {idProveedor: idProveedor, provNombre: provNombre, provNombreRepresentante: provNombreRepresentante, provCedulaRepresentante: provCedulaRepresentante,provCorreo: provCorreo,provTelefono: provTelefono,provDireccion: provDireccion , provCelular: provCelular}
  });
 }
}
<div fxLayout="row" fxLayoutAlign="space-around start" class="login-div">
 <mat-card class="login-card" fxFlex.gt-md="40" fxFlex="100">
    <mat-toolbar class="toolbar"> Iniciar Sesión</mat-toolbar>
    <mat-card-content fxLayout="column" style="margin-top: 0em;" class="login-card-content" fxLayoutGap="2.4%">
```

```
<ng-container [formGroup]="form" >

  <mat-form-field appearance="outline" >
    <mat-label>Usuario</mat-label>
    <input matInput   formControlName="usuario" >
    <mat-error>Campo Requerido</mat-error>
  </mat-form-field>


  <mat-form-field appearance="outline">
    <mat-label>Contraseña</mat-label>
    <input matInput type="password" formControlName="clave"  >
    <mat-error >Campo Requerido</mat-error>
  </mat-form-field>

 <div fxLayoutAlign="center">

        <button mat-raised-button color="primary" style="width: 20em;"
type="submit" (click)="login()">
    Ingresar
  </button>
```

```
        </div>



      </ng-container>
    </mat-card-content>
  </mat-card>
  </div>



.login-div {
    background-color: #f9f9f9;
    min-height: calc(100vh - 64px);
}
.login-card {
    padding: 0em;
    margin-top: 2em;
    justify-content: center;
    text-align: center;
}
.login-card-content {
    padding: 2em;
    margin-top: 0em;
    justify-content: center;
```

```
    text-align: center;
}
.login-card-header {
    justify-content: center;
}
.login-card h1 {
    font-weight: 500;
    font-size: 135%;
}
.toolbar {

    background-color: #4D67FF ; //dark blue: #002341; black: #1e1e1e; blue: #0275d8
    color: #ffffff; //#fff

}

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { LoginComponent } from './login.component';

describe('LoginComponent', () => {
  let component: LoginComponent;
  let fixture: ComponentFixture<LoginComponent>;
```

```
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ LoginComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(LoginComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});

import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
```

```
import {MatSnackBar, MatSnackBarHorizontalPosition} from '@angular/material/snack-bar';
import { openTimedSnackBar } from 'src/app/utils';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss']
})
export class LoginComponent implements OnInit {

  user: string="admin";
  password: string="admin";
  horizontalPosition: MatSnackBarHorizontalPosition = 'center';
  constructor(private router: Router,private snackBar: MatSnackBar) {}
  ngOnInit(): void {
  }
  form=new FormGroup({
    usuario: new FormControl('',[Validators.required]),
    clave: new FormControl('',[Validators.required])

  })
```

```
 login() {

   console.log(this.form.value.usuario);
   console.log(this.form.value.clave);

   if(this.form.value.usuario==this.user && this.form.value.clave==this.password)
     {
       this.router.navigateByUrl('menu')
     }else{
       openTimedSnackBar(this.snackBar, 'Usuario Invalido!', 'mat-warn')



     }
 }
 onLogin(){

 }

}

<ng-container >
```

```
<div fxLayoutGap=5% fxLayout="column" fxLayoutAlign="space-around center" class="menu-div">
    <mat-card class="form-title" >
      <h1 style="padding-left: 40%;">Menú Principal</h1>
    </mat-card>
    <div fxLayoutAlign="row" fxLayoutGap=10% class="content">
      <mat-card class="card-menu" >
        <h1 align="center" >
          Productos
        </h1>

          <br>
          <br>

          <div align="center" >
                <button color="primary" mat-icon-button class="icon-button" (click)="productos()">
                  <mat-icon >store</mat-icon>
                </button>
          </div>
```

```
</mat-card>
<mat-card class="card-menu" >
   <h1 align="center" >
      Proveedores
   </h1>

      <br>
      <br>

      <div align="center" >
                  <button color="primary" mat-icon-button class="icon-button"(click)="proveedores()" >
            <mat-icon >people</mat-icon>
         </button>
      </div>

</mat-card>
<mat-card class="card-menu" >
   <h1 align="center" >
      Ventas
```

```
        </h1>

            <br>
            <br>

            <div align="center" >
                        <button color="primary" mat-icon-button class="icon-button"(click)="ventas()" >
                    <mat-icon >list_alt</mat-icon>
                </button>
            </div>


        </mat-card>
        <mat-card class="card-menu" >
            <h1 align="center" >
                Compras
            </h1>

            <br>
            <br>

            <div align="center" >
```

```
                    <button color="primary" mat-icon-button class="icon-button"(click)="compras()" >
                        <mat-icon >shopping_cart</mat-icon>
                    </button>
                </div>


        </mat-card>
        <mat-card class="card-menu" >
            <h1 align="center" >
                Ganancias
            </h1>

                <br>
                <br>

                <div align="center" >
                    <button color="primary" mat-icon-button class="icon-button"(click)="ganancias()" >
                        <mat-icon >attach_money</mat-icon>
                    </button>
                </div>
```

```
            </mat-card>
        </div>
    </div>
</ng-container>
.menu-div {
    background-color: #f9f9f9;
    height: calc(90vh);



}
.content{
    height: 100vh;
}
.form-title {
    background-color: #4D67FF;
}
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { MenuPrincipalComponent } from './menu-principal.component';

describe('MenuPrincipalComponent', () => {
  let component: MenuPrincipalComponent;
```

```
  let fixture: ComponentFixture<MenuPrincipalComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ MenuPrincipalComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(MenuPrincipalComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});

import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
@Component({
```

```
  selector: 'app-menu-principal',
  templateUrl: './menu-principal.component.html',
  styleUrls: ['./menu-principal.component.scss']
})
export class MenuPrincipalComponent implements OnInit {

  constructor(private router: Router) { }

  ngOnInit(): void {
  }
productos(){
  this.router.navigateByUrl('formulario/producto')

}
proveedores(){
  this.router.navigateByUrl('formulario/ingresar-cliente')

}
ventas(){
  this.router.navigateByUrl('formulario/ventas')

}
compras(){
```

```
  this.router.navigateByUrl('formulario/compras')


}
ganancias(){
  this.router.navigateByUrl('formulario/ganancias')


}
}


<ng-container [formGroup]="form">
     <h1 style="font-weight: bold;">Gestión de Productos</h1>
    <mat-card >


        <mat-card-header>
            <mat-card-title>
                Ingresar Producto
            </mat-card-title>
        </mat-card-header>
        <mat-card-content fxLayout="column wrap" >
            <div fxLayout="row wrap" fxLayoutGap="3%">


                    <mat-form-field  appearance="outline"  fxFlex.gt-md="30"  fxFlex.lt-
md="90">
```

```
<mat-label>ID del Producto</mat-label>
<input  formControlName="idProducto" maxlength="10" matInput >

<mat-error>
  Campo requerido.
</mat-error>

</mat-form-field>

<mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-md="90">
<mat-label>Producto</mat-label>
<input formControlName="proNombre" matInput maxlength="35">
<mat-error>
  Campo Requerido.
</mat-error>
</mat-form-field>

<mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-md="90">
<mat-label>Stock</mat-label>
<input type="number" formControlName="proStock" matInput maxlength="10">
```

```
            <mat-error>
              Campo Requerido.
            </mat-error>
          </mat-form-field>

            <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-md="90">
            <mat-label>Precio al cliente</mat-label>
                  <input type="number" formControlName="proPrecioCliente" matInput maxlength="5" >
            <mat-error>
              Campo Requerido.
            </mat-error>
          </mat-form-field>

            <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-md="90">
            <mat-label>Precio del proveedor</mat-label>
                  <input type="number"formControlName="proPrecioCompra" matInput maxlength="5" >
            <mat-error>
              Campo Requerido.
            </mat-error>
          </mat-form-field>
```

```
                <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-
md="90">
                <mat-label>Descripción</mat-label>
                 <input formControlName="proDescripcion" matInput maxlength="50"
>
                <mat-error>
                   Campo Requerido.
                </mat-error>
              </mat-form-field>



          </div>
          <div>
             <button [disabled]="!form.valid" (click)="ingresarProducto()" mat-raised-
button color='primary' fxFlex.gt-md="10" fxFlex.lt-md="90">
                Ingresar
              </button>
          </div>


       </mat-card-content>
    </mat-card>
```

```
<mat-card style="margin-top: 3%;margin-block-end: 10%;">
    <mat-card-header>
      <mat-card-title >
          Productos Existentes

      </mat-card-title>

    </mat-card-header>
    <mat-card-content fxLayout="column">

      <div>
            <table mat-table [dataSource]="transactions" class="mat-elevation-z8" fxFlex.gt-md="100%">                    <!-- Item Column -->
          <ng-container matColumnDef="ID del Producto">
            <th mat-header-cell *matHeaderCellDef> ID Producto </th>
           <td mat-cell *matCellDef="let transaction"> {{transaction.idProducto}} </td>
            <td mat-footer-cell *matFooterCellDef> Total </td>

          </ng-container>
          <ng-container matColumnDef="Producto">
            <th mat-header-cell *matHeaderCellDef> Producto </th>
                                <td mat-cell *matCellDef="let transaction"> {{transaction.proNombre}} </td>
```

```
            <td mat-footer-cell *matFooterCellDef> Total </td>
          </ng-container>

          <!-- Cost Column -->
          <ng-container matColumnDef="Cantidad">
            <th mat-header-cell *matHeaderCellDef> Stock </th>
            <td mat-cell *matCellDef="let transaction"> {{transaction.proStock }}
</td>

          </ng-container>

          <ng-container matColumnDef="Precio">
            <th mat-header-cell *matHeaderCellDef> Precio Cliente </th>
                                <td mat-cell *matCellDef="let transaction">
{{transaction.proPrecioCliente}} </td>

          </ng-container>


          <ng-container matColumnDef="PrecioProv">
            <th mat-header-cell *matHeaderCellDef> Precio Proveedor </th>
                                <td mat-cell *matCellDef="let transaction">
{{transaction.proPrecioCompra}} </td>
```

```
</ng-container>

<ng-container matColumnDef="ProductoDescripcion">
  <th mat-header-cell *matHeaderCellDef> Descripción </th>
                    <td mat-cell *matCellDef="let transaction">
{{transaction.proDescripcion}} </td>

</ng-container>

<ng-container matColumnDef="accion">
  <th mat-header-cell  *matHeaderCellDef >Acción</th>
  <td mat-cell *matCellDef="let row2;let i=index;">
                    <button mat-icon-button color='primary'
(click)="editar(row2.idProducto,row2.proNombre,row2.proStock,row2.proPrecioCli
ente,row2.proPrecioCompra,row2.proDescripcion)">
      <mat-icon>edit</mat-icon> </button>
                    <button mat-icon-button color='accent'
(click)="deleteItem(row2.idProducto,row2.proNombre,row2.proStock,row2.proPreci
oCliente,row2.proPrecioCompra, row2.proDescripcion)">
      <mat-icon >delete</mat-icon>
    </button>
  </td>

</ng-container>
```

```
                <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
                <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
              </table>

            </div>
            <div  align =" center"fxFlex.gt-md="10" fxFlex.lt-md="90">
              <button (click)="cargarDatos()" mat-icon-button color='primary' >
                <mat-icon>
                  autorenew
                </mat-icon>
              </button>
            </div>
          </mat-card-content>

      </mat-card>

</ng-container>
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { ProductoComponent } from './producto.component';

describe('ProductoComponent', () => {
  let component: ProductoComponent;
```

```
  let fixture: ComponentFixture<ProductoComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ ProductoComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(ProductoComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});

import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms';
import { ServiciosService } from 'src/app/servicios.service';
```

```
import { MatDialog } from '@angular/material/dialog';
import {EditarDialogProductosComponent} from '../dialogs/editar-dialog-productos/editar-dialog-productos.component';
import {EliminarDialogProductosComponent} from '../dialogs/eliminar-dialog-productos/eliminar-dialog-productos.component'
interface Transaction {
  idProducto: string,
  proNombre: string;
  proStock: number;
  proPrecioCliente: number;
  proPrecioCompra: number;
  proDescripcion: String;

}

@Component({
  selector: 'app-producto',
  templateUrl: './producto.component.html',
  styleUrls: ['./producto.component.scss']
})
export class ProductoComponent implements OnInit {

    displayedColumns: string[] = ['ID del Producto','Producto', 'Cantidad','Precio','PrecioProv', 'ProductoDescripcion','accion'];
```

```
transactions: Transaction[] = [];
index: number=0;

constructor(private servicio:ServiciosService,private dialog:MatDialog) { }

ngOnInit(): void {
  this.cargarDatos()

}
form = new FormGroup({
  idProducto: new FormControl('', [Validators.required]),
  proNombre: new FormControl('', [Validators.required]),
  proStock: new FormControl('', [Validators.required]),
  proPrecioCliente: new FormControl('', [Validators.required]),
  proPrecioCompra: new FormControl('', [Validators.required]),
  proDescripcion: new FormControl('', [Validators.required]),

})

cargarDatos(){
  this.servicio.findProducto().subscribe((res) => {
    this.transactions = res;
    console.log("producto", this.transactions)
```

```
  })


}
ingresarProducto():void {
  this.servicio.insertProducto(this.form.value).subscribe(
    response=>{
      console.log(response);


    },
    error=>{
      console.log(error);
    });
  }
                editar(idProducto:number, proNombre:number,proStock:
number,proPrecioCliente:number, proPrecioCompra:number , proDescripcion:String
) {
    const dialogRef = this.dialog.open(EditarDialogProductosComponent, {
       data: {idProducto: idProducto, proNombre: proNombre, proStock: proStock,
proPrecioCliente: proPrecioCliente, proPrecioCompra:
proPrecioCompra, proDescripcion:proDescripcion }
    });
  }
```

```
deleteItem(idProducto:number, proNombre:number,proStock: number,proPrecioCliente:number, proPrecioCompra: number,proDescripcion:String) {
    const dialogRef = this.dialog.open(EliminarDialogProductosComponent, {
      data: {idProducto: idProducto, proNombre: proNombre, proStock: proStock, proPrecioCliente: proPrecioCliente, proPrecioCompra: proPrecioCompra, proDescripcion:proDescripcion  }
    });
  }

}
```

```
<ng-container [formGroup]="form">
    <h1 style="font-weight: bold">Gestión de Ventas</h1>
  <mat-card  >
    <mat-card-header>
      <mat-card-title>
        Ingresar Venta
      </mat-card-title>
    </mat-card-header>
    <br>
    <mat-card-content style="font-weight: bold">
      ID Venta:
```

```
        <mat-label style="font-weight: normal" input type="text" disabled="disabled">
5 </mat-label>
      </mat-card-content>

      <mat-card-content fxLayout="column wrap" >

        <div fxLayout="row wrap" fxLayoutGap="3%">

              <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-
md="90">
            <mat-label>Fecha</mat-label>
            <input type="date" formControlName="VENFECHA" matInput >

            <mat-error>
              Campo requerido.
            </mat-error>

          </mat-form-field>



        </div>
```

```
<div  class="example-button-row">
  <button  mat-flat-button color="primary" (click)="agregarVenta()">Agregar Producto + </button>
 </div>
 <br>

                    <div class="form-group row" [formGroupName]="i" formArrayName="venta" *ngFor="let venta of venta.controls; let i = index" fxLayout="row wrap" fxLayoutGap="3%">

  <mat-label fxLayoutAlign="center center" style="font-weight: normal" input type="text" disabled="disabled"> 1 </mat-label>

        <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-md="90">

    <mat-label>ID del Producto </mat-label>

                        <mat-select formControlName="idProducto" (selectionChange)="IdChange(this.form.value.idProducto)">

      <ng-container *ngFor="let productos of productoId">

                                <mat-option [value]="productos.idProducto">{{productos.idProducto}}</mat-option>
```

```
            </ng-container>
          </mat-select>
          <mat-error>
            Campo Requerido.
          </mat-error>
        </mat-form-field>

          <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-md="90">
          <mat-label>Cantidad</mat-label>
           <input type="number" formControlName="VENDCANTIDAD" matInput >
                                                                    <mat-error *ngIf="form.controls['VENDCANTIDAD'].hasError('required')">
            Campo Requerido.
          </mat-error>
        </mat-form-field>

         <div>
                <button class="btn btn-danger" color="warn" mat-flat-button (click)="removerVenta(i)">Elminar</button>
          </div>
        </div>
```

```
        <br>


        <div>
            <button [disabled]="!form.valid" (click)="ingresar()"  mat-raised-button color='primary' fxFlex.gt-md="10" fxFlex.lt-md="90">
              Ingresar
            </button>
        </div>


    </mat-card-content>
  </mat-card>


  <mat-card style="margin-top: 3%;margin-block-end: 10%;">
    <mat-card-header>
      <mat-card-title >
        Historial Ventas


      </mat-card-title>


    </mat-card-header>
    <mat-card-content fxLayout="column">
```

```
<div>
    <table mat-table [dataSource]="transactions" class="mat-elevation-z8"
fxFlex.gt-md="100%">                    <!-- Item Column -->

    <ng-container matColumnDef="IdVenta">
      <th mat-header-cell *matHeaderCellDef> ID Venta</th>
        <td mat-cell *matCellDef="let transaction"> {{transaction.idVen}}
</td>
      <td mat-footer-cell *matFooterCellDef> Total </td>
    </ng-container>

    <ng-container matColumnDef="ID del Producto">
      <th mat-header-cell *matHeaderCellDef> ID Producto</th>
                    <td mat-cell *matCellDef="let transaction">
{{transaction.ventaDetalleModelo[0].idProducto }} </td>

    </ng-container>

    <ng-container matColumnDef="VenLinea">
      <th mat-header-cell *matHeaderCellDef> Linea </th>
```

<td mat-cell *matCellDef="let transaction">
{{transaction.ventaDetalleModelo[0].vendLinea}} </td>

<td mat-footer-cell *matFooterCellDef> Total </td>

</ng-container>

<ng-container matColumnDef="VenFecha">

<th mat-header-cell *matHeaderCellDef> Fecha </th>

<td mat-cell *matCellDef="let transaction"> {{transaction.venFecha}}
</td>

</ng-container>

<ng-container matColumnDef="VenCantidad">

<th mat-header-cell *matHeaderCellDef> Cantidad </th>

<td mat-cell *matCellDef="let transaction">
{{transaction.ventaDetalleModelo[0].vendCantidad}} </td>

</ng-container>

<ng-container matColumnDef="accion">

<th mat-header-cell *matHeaderCellDef >Accion</th>

<td mat-cell *matCellDef="let row2;let i=index;">

<button mat-icon-button color='primary' >

<mat-icon>edit</mat-icon> </button>

<button mat-icon-button color='accent' >

```
                        <mat-icon >delete</mat-icon>
                    </button>
                </td>

            </ng-container>
            <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
            <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
        </table>

    </div>
    <div  align =" center"fxFlex.gt-md="10" fxFlex.lt-md="90">
        <button (click)="cargarDatos()" mat-icon-button color='primary' >
            <mat-icon>
                autorenew
            </mat-icon>
        </button>
    </div>
  </mat-card-content>

</mat-card>


</ng-container>
```

```
.label{
   font-size: large;
}
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { VentasComponent } from './ventas.component';

describe('VentasComponent', () => {
  let component: VentasComponent;
  let fixture: ComponentFixture<VentasComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ VentasComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(VentasComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });
```

```
  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, Validators, FormBuilder, FormArray } from '@angular/forms';
import { ServiciosService } from 'src/app/servicios.service';
interface Transaction {
  idVen: string,
  idProducto: string,
  venFecha: Date,
  vendCantidad: number,
  vendLinea: string,
  ventaDetalleModelo:String[]

}
@Component({
  selector: 'app-ventas',
  templateUrl: './ventas.component.html',
  styleUrls: ['./ventas.component.scss']
})
```

```
export class VentasComponent implements OnInit {
        displayedColumns: string[] = ['IdVenta', 'ID del Producto','VenLinea','VenFecha','VenCantidad','accion'];
  transactions: Transaction[] = [];
  index: number=0;
  constructor(private servicio: ServiciosService, private formBuilder:FormBuilder) { }
  cedulasClientes : any[] =[]
  productoId : any[] =[]
 total:number=0;
  ngOnInit(): void {
    this.servicio.findClients().subscribe((res) => {
      this.cedulasClientes = res;
      console.log('d',res)
    })
    this.servicio.findProducto().subscribe((res) => {
      this.productoId= res;
      console.log('f',res)
    })
    this.cargarDatos()

  }
  form = new FormGroup({
    IDVEN: new FormControl('', [Validators.required]),
```

```
  VENFECHA: new FormControl('', [Validators.required]),
  IDPRODUCTO: new FormControl('', [Validators.required]),
  VENDCANTIDAD: new FormControl('', [Validators.required]),
  VENLINEA: new FormControl('', [Validators.required]),



 })
 cargarDatos(){
  this.servicio.findVentas().subscribe((res) => {
   this.transactions = res;
   console.log("ventas", this.transactions)

  })



 }
 CantidadChange(id:number){
            //this.total = (this.productoId[this.form.value.IDPRODUCTO-
1].PRECIOPRODUCTO)*id
  for(let i =0;i<this.productoId.length;i++){
   if(this.productoId[i].IDPRODUCTO==this.form.value.IDPRODUCTO){
                                                                this.total=
this.productoId[i].PRECIOPRODUCTO*this.form.value.VENDCANTIDAD
```

```
    }
   }
   console.log("total",this.total)
  }
  IdChange(id:number){
                                /*                this.total=(this.productoId[id-
1].PRECIOPRODUCTO*this.form.value.VENDCANTIDAD)
   this.form.value.TOTALVENTA=this.total*/
   for(let i =0;i<this.productoId.length;i++){
    if(this.productoId[i].IDPRODUCTO==this.form.value.IDPRODUCTO){
                                                                this.total=
this.productoId[i].PRECIOPRODUCTO*this.form.value.VENDCANTIDAD
    }
   }
  }
  ingresar(){
   this.form.value.TOTALVENTA=this.total

   this.servicio.insertVentas(this.form.value).subscribe(
    response=>{
     console.log(response);

    },
```

```
    error=>{
      console.log(error);
    });
    console.log("datos ",this.form.value)
}
editar(){

}

get venta(){
  return this.registerForm.get('venta') as FormArray;
}

registerForm = this.formBuilder.group({

  venta: this.formBuilder.array([])
});

agregarVenta(){
  const telefonoFormGroup  = this.formBuilder.group({
    IDPRODUCTO:",
    VENDCANTIDAD:"
  });
```

```
    this.venta.push(telefonoFormGroup);
  }
  removerVenta(indice: number) {
    this.venta.removeAt(indice);
  }

}

import { Component, NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AppComponent } from './app.component';
import { LoginComponent } from './componentes/login/login.component';
import { FormsGeneralComponent } from './componentes/forms-general/forms-general.component';
import { VentasComponent } from './componentes/ventas/ventas.component';
import { ProductoComponent } from './componentes/producto/producto.component';
import { ComprasComponent } from './componentes/compras/compras.component';
import { GananciasComponent } from './componentes/ganancias/ganancias.component';
import { MenuPrincipalComponent } from './componentes/menu-principal/menu-principal.component';

import { IngresarClienteComponent } from './componentes/ingresar-cliente/ingresar-cliente.component';
```

```
const routes: Routes = [
  { path: "", component: AppComponent },
  { path: "login", component: LoginComponent },
  { path: "menu", component: MenuPrincipalComponent},
  {
    path: 'formulario',
    component: FormsGeneralComponent,
    children:[
      { path: "ingresar-cliente", component: IngresarClienteComponent },
      {path: "ventas", component:VentasComponent },
      {path: "producto",component:ProductoComponent},
      {path: "compras",component:ComprasComponent},
      {path: "ganancias",component:GananciasComponent}
    ]
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

```
<div >

  <mat-toolbar color="primary" class="toolbar" >
    <mat-toolbar-row class="row">

       <img class="logo"src="assets/img/mini-market.jpg" >

     <span>Mini Market "Paola"</span>
   </mat-toolbar-row>
  </mat-toolbar>
    <router-outlet></router-outlet>
</div>
.toolbar {
    position: sticky;
    position: -webkit-sticky;
    top: 0%;
    left: 0%;
    z-index: 1000;
    background-color: #4D67FF ; //dark blue: #002341; black: #1e1e1e; blue: #0275d8
    color: #ececec; //#fff
    width:100%;
    box-shadow: #0c0c0c ;
```

```
}
.logo{

        max-height:100%;
        min-height:0%;
        width:6%;/*valor inicial*/

}

import { TestBed } from '@angular/core/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [
        RouterTestingModule
      ],
      declarations: [
        AppComponent
      ],
    }).compileComponents();
```

```
  });

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have as title 'paginaWebAppi'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('paginaWebAppi');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement as HTMLElement;
                                                expect(compiled.querySelector('.content
span')?.textContent).toContain('paginaWebAppi app is running!');
  });
});
```

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'PaginaWebAppi';
}

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule, ReactiveFormsModule} from '@angular/forms';
import {MatMenuModule} from '@angular/material/menu';
import { AppRoutingModule } from './app-routing.module';
import {MatSidenavModule} from '@angular/material/sidenav';

import { AppComponent } from './app.component';
import { MatCardModule } from '@angular/material/card';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatGridListModule } from '@angular/material/grid-list';
import { MatButtonModule } from '@angular/material/button';
```

```
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatInputModule } from '@angular/material/input';
import { MatRadioModule } from '@angular/material/radio';
import { MatSelectModule } from '@angular/material/select';
import  {MatTableModule} from '@angular/material/table' ;
import { MatListModule } from '@angular/material/list';
import { MatSnackBarModule} from '@angular/material/snack-bar'
import { MatDatepickerModule } from '@angular/material/datepicker'
import { MatNativeDateModule } from '@angular/material/core';
import { MatCheckboxModule } from '@angular/material/checkbox';
import { MatIconModule } from '@angular/material/icon';
import { MatSliderModule } from '@angular/material/slider';
import { MatExpansionModule } from '@angular/material/expansion';
import { MatDialogModule } from '@angular/material/dialog';
import { MatTooltipModule } from '@angular/material/tooltip';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { FlexLayoutModule } from '@angular/flex-layout';
import { CommonModule } from '@angular/common';
import { FormsGeneralComponent } from './componentes/forms-general/forms-general.component';
import { LoginComponent } from './componentes/login/login.component';
import { HttpClientModule, HTTP_INTERCEPTORS } from "@angular/common/http";
```

```
import { IngresarClienteComponent } from './componentes/ingresar-cliente/ingresar-cliente.component';

import { EditarDialogComponent } from './componentes/editar-dialog/editar-dialog.component';

import { EliminarDialogComponent } from './componentes/eliminar-dialog/eliminar-dialog.component';

import { VentasComponent } from './componentes/ventas/ventas.component';

import { ProductoComponent } from './componentes/producto/producto.component';

import { ComprasComponent } from './componentes/compras/compras.component';

import { GananciasComponent } from './componentes/ganancias/ganancias.component';

import { MenuPrincipalComponent } from './componentes/menu-principal/menu-principal.component';

import { EditarDialogVentasComponent } from './componentes/dialogs/editar-dialog-ventas/editar-dialog-ventas.component';

import { EditarDialogProductosComponent } from './componentes/dialogs/editar-dialog-productos/editar-dialog-productos.component';

import { EliminarDialogProductosComponent } from './componentes/dialogs/eliminar-dialog-productos/eliminar-dialog-productos.component';


@NgModule({
  declarations: [
    AppComponent,
    FormsGeneralComponent,
```

```
    LoginComponent,
    IngresarClienteComponent,
    EditarDialogComponent,
    EliminarDialogComponent,
    VentasComponent,
    ProductoComponent,
    ComprasComponent,
    GananciasComponent,
                MenuPrincipalComponent,  EditarDialogVentasComponent,
EditarDialogProductosComponent,  EliminarDialogProductosComponent,
ComprasComponent, GananciasComponent,

  ],
  imports: [
    FlexLayoutModule,
    HttpClientModule,
    MatMenuModule,
    ReactiveFormsModule,
    CommonModule,
    MatSidenavModule,
    FormsModule,
    BrowserModule,
    AppRoutingModule,
```

```
    MatToolbarModule,
    MatGridListModule,
    MatButtonModule,
    MatRadioModule,
    MatCardModule,
    MatFormFieldModule,
    MatInputModule,
    MatSelectModule,
    MatTableModule,
    MatListModule,
    MatSnackBarModule,
    MatCheckboxModule,
    MatDatepickerModule,
    MatNativeDateModule,
    MatIconModule,
    MatSliderModule,
    MatExpansionModule,
    MatDialogModule,
    MatTooltipModule,
    BrowserAnimationsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
```

```
})
export class AppModule { }

import { TestBed } from '@angular/core/testing';

import { ServiciosService } from './servicios.service';

describe('ServiciosService', () => {
  let service: ServiciosService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(ServiciosService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});

import { environment } from './../environments/environment';
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
```

```
import { Observable } from 'rxjs';


const httpOptions = {
  headers: new HttpHeaders({
    'Content-Type': 'application/json',
  })
}
@Injectable({
  providedIn: 'root'
})
export class ServiciosService {
  private findClientesUrl: string = environment.apiProveedor
  private findVentasUrl: string = environment.apiVenta
  private findProductoURl: string = environment.apiProducto
  private findComprasUrl: string = environment.apiCompra
  private findGananciasUrl: string = "papeleria/api/ganancias/"
  constructor(private http:HttpClient) { }


  public findClients(): Observable<any[]> {
    const url = this.findClientesUrl
    return this.http.get<any[]>(url, httpOptions)
  }
```

```
public create(datos: any):  Observable<any>{
  return this.http.post(this.findClientesUrl, datos);
}

public delete(id: any):  Observable<any>{
  return this.http.delete(this.findClientesUrl+id);
}
public findVentas(): Observable<any[]> {
  const url = this.findVentasUrl
  return this.http.get<any[]>(url, httpOptions)
}


public insertVentas(datos:any): Observable<any>{
  return this.http.post(this.findVentasUrl,datos)
}
public insertCompras(datos:any): Observable<any>{
  return this.http.post(this.findComprasUrl,datos)
}
public findProducto(): Observable<any[]> {
  const url = this.findProductoURl
  return this.http.get<any[]>(url, httpOptions)
```

```
}
public insertProducto(data:any):Observable<any>{
  const url = this.findProductoURl
  return this.http.post(url,data)
}
public put(datos:any,id:any){
  const url = `${this.findClientesUrl}${id}`
  return this.http.put(url,datos);

}
public modificarProductos(datos:any,id:any){
  const url = `${this.findProductoURl}${id}`
  return this.http.put(url,datos);

}
public eliminrProductos(id:any){
  return this.http.delete(this.findProductoURl+id)
}
public findCompras(): Observable<any[]> {
  const url = this.findComprasUrl
  return this.http.get<any[]>(url, httpOptions)
}
```

```
  public findGanancias(): Observable<any[]> {
    const url = this.findGananciasUrl
    return this.http.get<any[]>(url, httpOptions)
  }
}

export const environment = {
  production: false,
  apiProducto: "/api/producto/",
  apiProveedor: "/api/proveedor/",
  apiVenta:"/api/ventae/venta?venta",
  apiCompra:"/api/comprae/compra?compra"
};


<ng-container [formGroup]="form">
        <h1 style="font-weight: bold">Ganancias por mes</h1>
    <mat-card  >
        <br>
        <hr>
<div>
```

```
                <mat-form-field appearance="outline" fxFlex.gt-md="30" fxFlex.lt-
md="90">
                  <mat-label>Fecha</mat-label>
                  <input type="month" formControlName="FECHA"  matInput  >

                  <mat-error>
                   Campo requerido.
                  </mat-error>

                </mat-form-field>
</div>
                <mat-card-content fxLayout="column wrap" >
                  <br>

                  <div fxLayout="column wrap" fxLayoutGap="3%">
                <mat-card-content style="font-weight: bold">
                  Ingresos:
                          <mat-label style="font-weight: normal" input type="text"
disabled="disabled">
                  <td mat-cell *matCellDef="let transaction"> {{transaction.Ingresos }}
</td>
                 </mat-label>
                </mat-card-content>
```

```
<mat-card-content style="font-weight: bold">
  Egresos:
        <mat-label style="font-weight: normal" input type="text" disabled="disabled">
  <td mat-cell *matCellDef="let transaction"> {{transaction.Egresos }} </td>
</mat-label>
</mat-card-content>

<mat-card-content style="font-weight: bold">
  Total:
        <mat-label style="font-weight: normal" input type="text" disabled="disabled">
  <td mat-cell *matCellDef="let transaction"> {{transaction.Total }} </td>
</mat-label>
</mat-card-content>


    </div>
  </mat-card-content>
</mat-card>
</ng-container>
.label{
```

```
    font-size: large;
}
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { GananciasComponent } from './ganancias.component';
describe('GananciasComponent', () => {
  let component: GananciasComponent;
  let fixture: ComponentFixture<GananciasComponent>;
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ GananciasComponent ]
    })
    .compileComponents();
  });
  beforeEach(() => {
    fixture = TestBed.createComponent(GananciasComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });
  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
import { Component, OnInit } from '@angular/core';
```

```
import { FormGroup, FormControl, Validators } from '@angular/forms';
import { ServiciosService } from 'src/app/servicios.service';
interface Transaction {
  COMFECHA: Date,
}

@Component({
  selector: 'app-ganancias',
  templateUrl: './ganancias.component.html',
  styleUrls: ['./ganancias.component.scss']
})
export class GananciasComponent implements OnInit {

  transactions: Transaction[] = [];
  index: number=0;
  constructor(private servicio: ServiciosService) { }
  cedulasClientes : any[] =[]
  productoId : any[] =[]
 total:number=0;
  ngOnInit(): void {
    this.servicio.findGanancias().subscribe((res) => {
      this.cedulasClientes = res;
      console.log('d',res)
```

```
  })
  this.servicio.findGanancias().subscribe((res) => {
    this.productoId= res;
    console.log('f',res)
  })
  this.cargarDatos()

}
form = new FormGroup({
  FECHA: new FormControl('', [Validators.required]),

})
cargarDatos(){
  this.servicio.findCompras().subscribe((res) => {
    this.transactions = res;
    console.log("compras", this.transactions)
  })
}
}
```

Anexo 3

El siguiente código fuente contempla a todo lo que respecta con el desarrollo del back-end de la aplicación de gestión de stock de productos del mini market “Paola”.

```
package com.example.demo.controlador;
import com.example.demo.modelo.CompradModelo;
import com.example.demo.servicios.CompradServicio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.ArrayList;
import java.util.Optional;
@RestController
@RequestMapping("/api/comprad")
public class CompradControlador {
    @Autowired
    CompradServicio compradServicio;
    @GetMapping()
    public ArrayList<CompradModelo> getComprad(){
        return compradServicio.getComprad();
    }
    @PostMapping()
    public CompradModelo postComprad(@RequestBody CompradModelo comprad){
        return this.compradServicio.postComprad(comprad);
```

```
    }
    @GetMapping(path="/{id}")
    public Optional<CompradModelo> getById(@PathVariable("id")Integer id){
        return this.compradServicio.getById(id);
    }
    @DeleteMapping(path="/{id}")
    public String deleteById(@PathVariable("id")Integer id){
        boolean ok=this.compradServicio.deleteComprad(id);
        if(ok){
            return"Se eliminó la compra con id: "+id;
        }else
            return "No se pudo eliminar la compra con id: "+id;
    }
    @PutMapping(path="/{id}")
      public String putComprad(@PathVariable("id") Integer id, @RequestBody
CompradModelo comprad){
        boolean ok=this.compradServicio.putComprad(id, comprad);
        if (ok){
            return "Se actualizó el detalle de la compra con id: "+id;
        }
        else
            return "No se pudo actualizar el detalle de la compra con id: "+id;
    }
```

```
}
package com.example.demo.controlador;
import com.example.demo.modelo.CompraeModelo;
import com.example.demo.servicios.CompraeServicio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.ArrayList;
import java.util.Optional;
@RestController
@RequestMapping("/api/comprae")
public class CompraeControlador {
    @Autowired
    CompraeServicio compraeServicio;
    @GetMapping()
    public ArrayList<CompraeModelo> getComprae(){
        return compraeServicio.getComprae();
    }
    @PostMapping()
    public CompraeModelo postComprae(@RequestBody CompraeModelo comprae){
        return this.compraeServicio.postComprae(comprae);
    }
    @GetMapping(path="/{id}")
    public Optional<CompraeModelo> getById(@PathVariable("id")String id){
```

```
        return this.compraeServicio.getById(id);
    }
    @DeleteMapping(path="/{id}")
    public String deleteById(@PathVariable("id")String id){
        boolean ok=this.compraeServicio.deleteComprae(id);
        if(ok){
            return"Se eliminó la compra con id: "+id;
        }else
            return "No se pudo eliminar la compra con id: "+id;
    }
    @PutMapping(path="/{id}")
        public String putComprae(@PathVariable("id") String id, @RequestBody
CompraeModelo comprae){
        boolean ok=this.compraeServicio.putComprae(id, comprae);
        if (ok){
            return "Se actualizó el encabezado de la compra con id: "+id;
        }
        else
            return "No se pudo actualizar el encabezado de la compra con id: "+id;
    }
    @GetMapping("/compra")
                                    public                ArrayList<CompraeModelo>
getCompraJoin(@RequestParam("compra")String f){
```

```
        return compraeServicio.getCompraJoin();
    }
    @GetMapping("/egreso")
     public ArrayList<CompraeModelo> getEgresos(@RequestParam("egreso")String
f1){
        return compraeServicio.getEgresos();
    }
}
package com.example.demo.controlador;
import com.example.demo.modelo.ProductoModelo;
import com.example.demo.servicios.ProductoServicio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.ArrayList;
import java.util.Optional;
@RestController
@RequestMapping("/api/producto")
public class ProductoControlador {
    @Autowired
    ProductoServicio productoServicio;
    @GetMapping()
    public ArrayList<ProductoModelo> getProducto(){
        return productoServicio.getProducto();
```

```
    }
    @PutMapping(path="/{id}")
        public String putProducto(@PathVariable("id") String id, @RequestBody
ProductoModelo producto){
        boolean ok=this.productoServicio.putProducto(id, producto);
        if (ok){
            return "Se actualizó el producto con id: "+id;
        }
        else
            return "No se pudo actualizar el producto con id: "+id;
    }
    @GetMapping(path="/{id}")
    public Optional<ProductoModelo> getById(@PathVariable("id")String id){
        return this.productoServicio.getById(id);
    }
    @DeleteMapping(path="/{id}")
    public String deleteById(@PathVariable("id")String id){
        boolean ok=this.productoServicio.deleteProducto(id);
        if(ok){
            return"Se eliminó el producto con id: "+id;
        }else
            return "No se pudo eliminar el producto con id: "+id;
    }
```

```
    @PostMapping()
    public ProductoModelo postProducto(@RequestBody ProductoModelo producto){
        return this.productoServicio.postProducto(producto);
    }
}
package com.example.demo.controlador;
import com.example.demo.modelo.ProveedorModelo;
import com.example.demo.servicios.ProveedorServicio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.ArrayList;
import java.util.Optional;
@RestController
@RequestMapping("/api/proveedor")
public class ProveedorControlador {
    @Autowired
    ProveedorServicio proveedorServicio;
    @GetMapping()
    public ArrayList<ProveedorModelo> getProveedor(){
        return proveedorServicio.getProveedor();
    }
    @PostMapping()
```

```
    public ProveedorModelo postProveedor(@RequestBody ProveedorModelo proveedor){
        return this.proveedorServicio.postProveedor(proveedor);
    }
    @GetMapping(path="/{id}")
    public Optional<ProveedorModelo> getById(@PathVariable("id")String id){
        return this.proveedorServicio.getById(id);
    }
    @DeleteMapping(path="/{id}")
    public String deleteById(@PathVariable("id")String id){
        boolean ok=this.proveedorServicio.deleteProveedor(id);
        if(ok){
            return"Se eliminó el proveedor con id: "+id;
        }else
            return "No se pudo eliminar el proveedor con id: "+id;
    }
    @PutMapping(path="/{id}")
    public String putProveedor(@PathVariable("id") String id, @RequestBody ProveedorModelo proveedor){
        boolean ok=this.proveedorServicio.putProveedor(id, proveedor);
        if (ok){
            return "Se actualizó el proveedor con id: "+id;
        }
```

```
        else
            return "No se pudo actualizar el proveedor con id: "+id;
    }
}
package com.example.demo.controlador;
import com.example.demo.modelo.VentadModelo;
import com.example.demo.servicios.VentadServicio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.ArrayList;
import java.util.Optional;
@RestController
@RequestMapping("/api/ventad")
public class VentadControlador {
    @Autowired
    VentadServicio ventadServicio;
    @GetMapping()
    public ArrayList<VentadModelo> getVentad(){
        return ventadServicio.getVentad();
    }
    @PostMapping()
    public VentadModelo postVentad(@RequestBody VentadModelo ventad){
        return this.ventadServicio.postVentad(ventad);
```

```
    }
    @GetMapping(path="/{id}")
    public Optional<VentadModelo> getById(@PathVariable("id")Integer id){
        return this.ventadServicio.getById(id);
    }
    @DeleteMapping(path="/{id}")
    public String deleteById(@PathVariable("id")Integer id){
        boolean ok=this.ventadServicio.deleteVentad(id);
        if(ok){
            return"Se eliminó la venta con id: "+id;
        }else
            return "No se pudo eliminar la venta con id: "+id;
    }
    @PutMapping(path="/{id}")
        public String putVentad(@PathVariable("id") Integer id, @RequestBody
VentadModelo ventad){
        boolean ok=this.ventadServicio.putVentad(id, ventad);
        if (ok){
            return "Se actualizó el detalle de venta con id: "+id;
        }
        else
            return "No se pudo actualizar el detalle de venta con id: "+id;
    }
```

```
}
package com.example.demo.controlador;
import com.example.demo.modelo.VentaeModelo;
import com.example.demo.servicios.VentaeServicio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.ArrayList;
import java.util.Optional;
@RestController
@RequestMapping("/api/ventae")
public class VentaeControlador {
    @Autowired
    VentaeServicio ventaeServicio;
    @GetMapping()
    public ArrayList<VentaeModelo> getVentae(){
        return ventaeServicio.getVentae();
    }
    @PostMapping()
    public VentaeModelo postVentae(@RequestBody VentaeModelo ventae){
      return this.ventaeServicio.postVentae(ventae);
    }
    @GetMapping(path="/{id}")
    public Optional<VentaeModelo> getById(@PathVariable("id")String id){
```

```
        return this.ventaeServicio.getById(id);
    }
    @DeleteMapping(path="/{id}")
    public String deleteById(@PathVariable("id")String id){
        boolean ok=this.ventaeServicio.deleteVentae(id);
        if(ok){
            return"Se eliminó la venta con id: "+id;
        }else
            return "No se pudo eliminar la venta con id: "+id;
    }
    @PutMapping(path="/{id}")
        public String putVentae(@PathVariable("id") String id, @RequestBody
VentaeModelo ventae){
        boolean ok=this.ventaeServicio.putVentae(id, ventae);
        if (ok){
            return "Se actualizó el encabezado de la venta con id: "+id;
        }
        else
            return "No se pudo actualizar el encabezado de la venta con id: "+id;
    }
    @GetMapping("/venta")
     public ArrayList<VentaeModelo> getVentaJoin(@RequestParam("venta")String
f){
```

```
        return ventaeServicio.getVentaJoin();
    }
}
package com.example.demo.modelo;
import javax.persistence.*;
@Entity
@Table(name="compra_detalle")
public class CompradModelo {
    @Column(name="idproducto")
    private String idProducto;
    @Id
    @Column(name="idcompradetalle")
    private Integer idCompraDetalle;
    @Column(name="idcompra")
    private String idCompra;
    @Column(name="comcantidad")
    private Integer comCantidad;
    @Column(name="comdlinea")
    private String comdLinea;
    public String getIdProducto() {
        return idProducto;
    }
    public void setIdProducto(String idProducto) {
```

```
    this.idProducto = idProducto;
}
public Integer getIdCompraDetalle() {
    return idCompraDetalle;
}
public void setIdCompraDetalle(Integer idCompraDetalle) {
    this.idCompraDetalle = idCompraDetalle;
}
public String getIdCompra() {
    return idCompra;
}
public void setIdCompra(String idCompra) {
    this.idCompra = idCompra;
}
public Integer getComCantidad() {
    return comCantidad;
}
public void setComCantidad(Integer comCantidad) {
    this.comCantidad = comCantidad;
}
public String getComdLinea() {
    return comdLinea;
}
```

```
    public void setComdLinea(String comdLinea) {
        this.comdLinea = comdLinea;
    }
}
package com.example.demo.modelo;
import javax.persistence.*;
import java.util.Date;
import java.util.Set;
@Entity
@Table(name="compra_encabezado")
public class CompraeModelo {
    @Id
    @Column(name="idcompra")
    private String idCompra;
    @Column(name="idproveedor")
    private String idProveedor;
    @Column(name="comfecha")
    @Temporal(TemporalType.DATE)
    private Date comFecha;
        @OneToMany(targetEntity = CompradModelo.class, mappedBy =
"idCompraDetalle", orphanRemoval = false, fetch = FetchType.LAZY)
    private Set<CompradModelo> compraDetalleModelo;
    public Set<CompradModelo> getCompraDetalleModelo() {
```

```
        return this.compraDetalleModelo;
    }
    public void setVentaDetalleModelo(Set<CompradModelo> compraDetalleModelo)
{
        this.compraDetalleModelo = compraDetalleModelo;
    }
    public String getIdCompra() {
        return idCompra;
    }
    public void setIdCompra(String idCompra) {
        this.idCompra = idCompra;
    }
    public String getIdProveedor() {
        return idProveedor;
    }
    public void setIdProveedor(String idProveedor) {
        this.idProveedor = idProveedor;
    }
    public Date getComFecha() {
        return comFecha;
    }
    public void setComFecha(Date comFecha) {
        this.comFecha = comFecha;
```

```
    }
}
package com.example.demo.modelo;
import javax.persistence.*;
@Entity
@Table(name="producto")
public class ProductoModelo {
    @Id
    @Column(name="idproducto")
    private String idProducto;
    @Column(name="pronombre")
    private String proNombre;
    @Column(name="propreciocliente")
    private Float proPrecioCliente;
    @Column(name="prostock")
    private Integer proStock;
    @Column(name="prodescripcion")
    private String proDescripcion;
    @Column(name="propreciocompra")
    private Float proPrecioCompra;
        @OneToMany(targetEntity = CompradModelo.class, mappedBy =
"idCompraDetalle", orphanRemoval = false, fetch = FetchType.LAZY)
    private Set<CompradModelo> compradModelo;
```

```
public Set<CompradModelo> getCompradModelo() {
    return compradModelo;
}
public void setCompradModelo(Set<CompradModelo> compradModelo) {
    this.compradModelo = compradModelo;
}
public String getIdProducto() {
    return idProducto;
}
public void setIdProducto(String idProducto) {
    this.idProducto = idProducto;
}
public String getProNombre() {
    return proNombre;
}
public void setProNombre(String proNombre) {
    this.proNombre = proNombre;
}
public Float getProPrecioCliente() {
    return proPrecioCliente;
}
public void setProPrecioCliente(Float proPrecioCliente) {
    this.proPrecioCliente = proPrecioCliente;
```

```
    }
    public Integer getProStock() {
        return proStock;
    }
    public void setProStock(Integer proStock) {
        this.proStock = proStock;
    }
    public String getProDescripcion() {
        return proDescripcion;
    }
    public void setProDescripcion(String proDescripcion) {
        this.proDescripcion = proDescripcion;
    }
    public Float getProPrecioCompra() {
        return proPrecioCompra;
    }
    public void setProPrecioCompra(Float proPrecioCompra) {
        this.proPrecioCompra = proPrecioCompra;
    }
}
package com.example.demo.modelo;
import javax.persistence.Column;
import javax.persistence.Entity;
```

```
import javax.persistence.Id;
import javax.persistence.Table;
@Entity
@Table(name="proveedor")
public class ProveedorModelo {
    @Id
    @Column(name="idproveedor")
    private String idProveedor;
    @Column(name="provnombre")
    private String provNombre;
    @Column(name="provnombrerepresentante")
    private String provNombreRepresentante;
    @Column(name="provcedularepresentante")
    private String provCedulaRepresentante;
    @Column(name="provtelefono")
    private String provTelefono;
    @Column(name="provcorreo")
    private String provCorreo;
    @Column(name="provdireccion")
    private String provDireccion;
    @Column(name="provcelular")
    private String provCelular;
    public String getIdProveedor() {
```

```
    return idProveedor;
}
public void setIdProveedor(String idProveedor) {
    this.idProveedor = idProveedor;
}
public String getProvNombre() {
    return provNombre;
}
public void setProvNombre(String provNombre) {
    this.provNombre = provNombre;
}
public String getProvNombreRepresentante() {
    return provNombreRepresentante;
}
public void setProvNombreRepresentante(String provNombreRepresentante) {
    this.provNombreRepresentante = provNombreRepresentante;
}
public String getProvCedulaRepresentante() {
    return provCedulaRepresentante;
}
public void setProvCedulaRepresentante(String provCedulaRepresentante) {
    this.provCedulaRepresentante = provCedulaRepresentante;
}
```

```
public String getProvTelefono() {
    return provTelefono;
}
public void setProvTelefono(String provTelefono) {
    this.provTelefono = provTelefono;
}
public String getProvCorreo() {
    return provCorreo;
}
public void setProvCorreo(String provCorreo) {
    this.provCorreo = provCorreo;
}
public String getProvDireccion() {
    return provDireccion;
}
public void setProvDireccion(String provDireccion) {
    this.provDireccion = provDireccion;
}
public String getProvCelular() {
    return provCelular;
}
public void setProvCelular(String provCelular) {
    this.provCelular = provCelular;
```

```
    }
}
package com.example.demo.modelo;
import javax.persistence.*;
@Entity
@Table(name="venta_detalle")
public class VentadModelo {
    @Id
    @Column(name="idventadetalle")
    private Integer idVentaDetalle;
    @Column(name="idproducto")
    private String idProducto;
    @Column(name="idven" )
    private String idVen;
    @Column(name="vendcantidad")
    private Integer vendCantidad;
    @Column(name="vendlinea")
    private String vendLinea;
    public Integer getIdVentaDetalle() {
        return idVentaDetalle;
    }
    public void setIdVentaDetalle(Integer idVentaDetalle) {
        this.idVentaDetalle = idVentaDetalle;
```

```
}
public String getIdProducto() {
    return idProducto;
}
public void setIdProducto(String idProducto) {
    this.idProducto = idProducto;
}
public String getIdVen() {
    return idVen;
}
public void setIdVen(String idVen) {
    this.idVen = idVen;
}
public Integer getVendCantidad() {
    return vendCantidad;
}
public void setVendCantidad(Integer vendCantidad) {
    this.vendCantidad = vendCantidad;
}
public String getVendLinea() {
    return vendLinea;
}
public void setVendLinea(String vendLinea) {
```

```
        this.vendLinea = vendLinea;
    }
}
package com.example.demo.modelo;
import javax.persistence.*;
import java.util.Date;
import java.util.Set;
@Entity
@Table(name="venta_encabezado")
public class VentaeModelo {
    @Id
    @Column(name="idven")
    private String idVen;
    @Column(name="venfecha")
    @Temporal(TemporalType.DATE)
    private Date venFecha;
    @OneToMany(targetEntity = VentadModelo.class, mappedBy = "idVentaDetalle",
orphanRemoval = false, fetch = FetchType.LAZY)
    private Set<VentadModelo> ventaDetalleModelo;
    public Set<VentadModelo> getVentaDetalleModelo() {
        return ventaDetalleModelo;
    }
    public void setVentaDetalleModelo(Set<VentadModelo> ventaDetalleModelo) {
```

```
        this.ventaDetalleModelo = ventaDetalleModelo;
    }
    public String getIdVen() {
        return idVen;
    }
    public void setIdVen(String idVen) {
        this.idVen = idVen;
    }
    public Date getVenFecha() {
        return venFecha;
    }
    public void setVenFecha(Date venFecha) {
        this.venFecha = venFecha;
    }
}
package com.example.demo.repositorio;
import com.example.demo.modelo.CompradModelo;
import org.springframework.data.repository.CrudRepository;
import java.util.ArrayList;
public interface CompradRepositorio extends CrudRepository<CompradModelo, Integer> {
    public abstract ArrayList<CompradModelo> findAll();
}
```

```
package com.example.demo.repositorio;

import com.example.demo.modelo.CompraeModelo;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.CrudRepository;

import java.util.ArrayList;

public interface CompraeRepositorio extends CrudRepository<CompraeModelo, String> {

    @Query(value="select v.idCompra, v.idProducto, b.idProveedor, v.comdLinea, b.comFecha, v.comCantidad from compra_detalle v inner join compra_encabezado b where b.idCompra= v.idCompra order by v.idCompra", nativeQuery=true )

   ArrayList<CompraeModelo> getJoin();

    @Query(value=" select sum(v.vendCantidad*p.proPrecioCliente) as ingresos, sum(m.comCantidad*p.proPrecioCompra) as egresos from venta_detalle v inner join venta_encabezado b inner join producto p inner join compra_encabezado n inner join compra_detalle m where year(b.venFecha) = year('?') and month(b.venFecha) = month('?') and b.idVen = v.idVen and p.idProducto=v.idProducto and year(n.comFecha) = year('?') and month(n.comFecha) = month('?') and n.idCompra = m.idCompra and p.idProducto=m.idProducto order by p.idProducto", nativeQuery=true )

   ArrayList<CompraeModelo> getEgresos();

}

package com.example.demo.repositorio;

import com.example.demo.modelo.ProductoModelo;

import org.springframework.data.repository.CrudRepository;

import java.util.ArrayList;
```

```
public interface ProductoRepositorio  extends CrudRepository<ProductoModelo, String> {
   public abstract ArrayList<ProductoModelo> findAll();
}
package com.example.demo.repositorio;
import com.example.demo.modelo.ProveedorModelo;
import org.springframework.data.repository.CrudRepository;
import java.util.ArrayList;
public interface ProveedorRepositorio  extends CrudRepository<ProveedorModelo, String> {
   public abstract ArrayList<ProveedorModelo> findAll();
}
package com.example.demo.repositorio;
import com.example.demo.modelo.VentadModelo;
import org.springframework.data.repository.CrudRepository;
import java.util.ArrayList;
public interface VentadRepositorio extends CrudRepository<VentadModelo, Integer>
{
   public abstract ArrayList<VentadModelo> findAll();
}
package com.example.demo.repositorio;
import com.example.demo.modelo.VentaeModelo;
import java.util.ArrayList;
import org.springframework.data.jpa.repository.Query;
```

```
import org.springframework.data.repository.CrudRepository;

public interface VentaeRepositorio extends CrudRepository<VentaeModelo, String>
{

        @Query(value="select v.idVen, v.idProducto, v.vendLinea, b.venFecha, v.vendCantidad from venta_detalle v inner join venta_encabezado b where b.idVen= v.idVen order by v.idVen", nativeQuery=true )

   ArrayList<VentaeModelo> getJoin();

}


package com.example.demo.servicios;

import com.example.demo.modelo.CompradModelo;

import com.example.demo.repositorio.CompradRepositorio;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.Optional;

@Service

public class CompradServicio {

   @Autowired

   CompradRepositorio compradRepositorio;

   public ArrayList<CompradModelo> getComprad(){

      return (ArrayList<CompradModelo> ) compradRepositorio.findAll();

   }

   public CompradModelo postComprad(CompradModelo comprad){
```

```
        return compradRepositorio.save(comprad);
    }
    public Optional<CompradModelo> getById(Integer id){
        return compradRepositorio.findById(id);
    }
    public Boolean deleteComprad(Integer id){
        try{
            compradRepositorio.deleteById(id);
            return true;
        }catch(Exception ex){
            return false;
        }
    }
    public Boolean putComprad(Integer  id, CompradModelo comprad){
        try{
            compradRepositorio.save(comprad);
            return true;
        }catch(Exception ex){
            return false;
        }
    }
}
package com.example.demo.servicios;
```

```
import com.example.demo.modelo.CompraeModelo;
import com.example.demo.repositorio.CompraeRepositorio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.ArrayList;
import java.util.Optional;
@Service
public class CompraeServicio {
    @Autowired
    CompraeRepositorio compraeRepositorio;
    public ArrayList<CompraeModelo> getComprae(){
        return (ArrayList<CompraeModelo> ) compraeRepositorio.findAll();
    }
    public CompraeModelo postComprae(CompraeModelo comprae){
        return compraeRepositorio.save(comprae);
    }
    public Optional<CompraeModelo> getById(String id){
        return compraeRepositorio.findById(id);
    }
    public Boolean deleteComprae(String id){
        try{
            compraeRepositorio.deleteById(id);
            return true;
```

```
        }catch(Exception ex){
            return false;
        }
    }
    public Boolean putComprae(String  id, CompraeModelo comprae){
        try{
            compraeRepositorio.save(comprae);
            return true;
        }catch(Exception ex){
            return false;
        }
    }
    public ArrayList<CompraeModelo> getCompraJoin(){
        return (ArrayList<CompraeModelo> ) compraeRepositorio.getJoin();
    }
    public ArrayList<CompraeModelo> getEgresos(){
        return (ArrayList<CompraeModelo> ) compraeRepositorio.getEgresos();
    }
}
package com.example.demo.servicios;

import com.example.demo.modelo.ProductoModelo;
import com.example.demo.repositorio.ProductoRepositorio;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.ArrayList;
import java.util.Optional;
@Service
public class ProductoServicio {
    @Autowired
    ProductoRepositorio productoRepositorio;
    public ArrayList<ProductoModelo> getProducto(){
        return (ArrayList<ProductoModelo> ) productoRepositorio.findAll();
    }
    public ProductoModelo postProducto(ProductoModelo producto){
        return productoRepositorio.save(producto);
    }
    public Optional<ProductoModelo> getById(String id){
        return productoRepositorio.findById(id);
    }
    public Boolean deleteProducto(String id){
        try{
            productoRepositorio.deleteById(id);
            return true;
        }catch(Exception ex){
            return false;
```

```
        }
    }
    public Boolean putProducto(String  id, ProductoModelo producto){
        try{
            productoRepositorio.save(producto);
            return true;
        }catch(Exception ex){
            return false;
        }
    }
}
package com.example.demo.servicios;
import com.example.demo.modelo.ProveedorModelo;
import com.example.demo.repositorio.ProveedorRepositorio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.ArrayList;
import java.util.Optional;

@Service
public class ProveedorServicio {
    @Autowired
    ProveedorRepositorio proveedorRepositorio;
```

```
public ArrayList<ProveedorModelo> getProveedor(){
   return (ArrayList<ProveedorModelo> ) proveedorRepositorio.findAll();
}
public ProveedorModelo postProveedor(ProveedorModelo proveedor){
   return proveedorRepositorio.save(proveedor);
}
public Optional<ProveedorModelo> getById(String id){
   return proveedorRepositorio.findById(id);
}
public Boolean deleteProveedor(String id){
   try{
      proveedorRepositorio.deleteById(id);
      return true;
   }catch(Exception ex){
      return false;
   }
}
public Boolean putProveedor(String  id, ProveedorModelo proveedor){
   try{
      proveedorRepositorio.save(proveedor);
      return true;
   }catch(Exception ex){
      return false;
```

```
        }
    }
}
package com.example.demo.servicios;
import com.example.demo.modelo.VentadModelo;
import com.example.demo.repositorio.VentadRepositorio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.ArrayList;
import java.util.Optional;
@Service
public class VentadServicio {
    @Autowired
    VentadRepositorio ventadRepositorio;
    public ArrayList<VentadModelo> getVentad(){
        return (ArrayList<VentadModelo> ) ventadRepositorio.findAll();
    }
    public VentadModelo postVentad(VentadModelo ventad){
        return ventadRepositorio.save(ventad);
    }
    public Optional<VentadModelo> getById(Integer id){
        return ventadRepositorio.findById(id);
    }
```

```
    public Boolean deleteVentad(Integer id){
        try{
            ventadRepositorio.deleteById(id);
            return true;
        }catch(Exception ex){
            return false;
        }
    }
    public Boolean putVentad(Integer  id, VentadModelo ventad){
        try{
            ventadRepositorio.save(ventad);
            return true;
        }catch(Exception ex){
            return false;
        }
    }
}
package com.example.demo.servicios;
import com.example.demo.modelo.VentaeModelo;
import com.example.demo.repositorio.VentaeRepositorio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.ArrayList;
```

```
import java.util.Optional;
@Service
public class VentaeServicio {
    @Autowired
    VentaeRepositorio ventaeRepositorio;
    public ArrayList<VentaeModelo> getVentae(){
        return (ArrayList<VentaeModelo> ) ventaeRepositorio.findAll();
    }
    public VentaeModelo postVentae(VentaeModelo ventae){
        return ventaeRepositorio.save(ventae);
    }
    public Optional<VentaeModelo> getById(String id){
        return ventaeRepositorio.findById(id);
    }
    public Boolean deleteVentae(String id){
        try{
            ventaeRepositorio.deleteById(id);
            return true;
        }catch(Exception ex){
            return false;
        }
    }
    public Boolean putVentae(String  id, VentaeModelo ventae){
```

```
        try{
            ventaeRepositorio.save(ventae);
            return true;
        }catch(Exception ex){
            return false;
        }
    }
    public ArrayList<VentaeModelo> getVentaJoin(){
        return (ArrayList<VentaeModelo> ) ventaeRepositorio.getJoin();
    }
}
spring.datasource.url=jdbc:mysql://localhost:3306/minimarket
spring.datasource.username=root
spring.datasource.password=xxxxxxxxxx
spring.jpa.hibenate.ddl-auto=none
```