ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO DE SISTEMA DE GESTIÓN DE INVENTARIOS PARA UNA PYME TIPO LAVADORA Y LUBRICADORA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

ESTEBAN LEANDRO MACHADO OBANDO

esteban.machado@epn.edu.ec

JOSEPH ESTUARDO BRAVO MACAS

joseph.bravo@epn.edu.ec

DIRECTOR: RAÚL DAVID MEJÍA NAVARRETE, M.Sc.

david.mejia@epn.edu.ec

Quito, Agosto 2022

AVAL

Certifico que el presente trabajo fue desarrollado por Esteban Leandro Machado Obando y Joseph Estuardo Bravo Macas, bajo mi supervisión.

Raúl David Mejía Navarrete, M.Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Nosotros, Esteban Leandro Machado Obando y Joseph Estuardo Bravo Macas, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejamos constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

Esteban Leandro Machado Obando

Joseph Estuardo Bravo Macas

DEDICATORIA

Dedico con orgullo este Trabajo de Titulación a mi madre Gladys Obando por ser el pilar principal en mi formación profesional, quien día a día estuvo junto a mí depositando su confianza. Las palabras no alcanzan para describir su ayuda, sus consejos y por ser ejemplo de lucha y perseverancia durante toda mi vida.

Esteban Leandro Machado Obando

DEDICATORIA

Este Trabajo de Titulación se lo dedico a mi madre Merci Macas por ser un ejemplo de superación y perseverancia, a mi hermano Bryan Bravo por ser mi apoyo y fuente de motivación en esta etapa de mi vida, a Lusdary Reyes y a Cristy Asanza por impulsarme a ser mejor cada día y mostrarme la luz al final del camino.

Joseph Estuardo Bravo Macas

AGRADECIMIENTO

A Dios por sus bendiciones y por permitirme cumplir con esta meta en mi vida.

A mi madre, Gladys Obando, por todo el apoyo incondicional entregado durante toda mi vida.

A mi abuelita, Marina Obando, quien siempre cuido y estuvo pendiente de mi durante toda su vida.

A los grandes amigos y compañeros que durante toda la carrera fueron un apoyo fundamental: Karolina Goyes, Jairo Viracocha, Alan Crespo, Sebastián Mullo, Ariana Rivas, Joseph Bravo, Bryan Santander, Matías Romero. Gracias por los momentos de estudio, risas, paseos, giras, y por hacer del paso por la universidad una experiencia inolvidable.

A David Mejía, M.Sc. por compartir la paciencia, el apoyo, el tiempo y sus conocimientos durante la dirección del presente Trabajo de Titulación.

A los docentes de la Escuela Politécnica Nacional por contribuir en mi formación profesional.

Esteban Leandro Machado Obando

AGRADECIMIENTO

A mí, por estar siempre firme, por saber esforzarme, por hacer las cosas bien y pensar que no hay mejor logro que aquel que se obtiene con trabajo honesto.

A mi madre, por apoyarme, por estar pendiente de mi cada día de mi vida, por todo el cariño que me dio fuerza para seguir tras cada caída.

A mi hermano por apoyarme y mostrarme que todo lo podemos alcanzar con perseverancia, que estamos hechos para superarnos y que el camino nunca termina siempre hay algo más por hacer, aprender o ganar.

A mi hermana por impulsarme a ser mejor, a ser un ejemplo.

A Cristy por toda la paciencia y estar ahí en los mejores y peores momentos.

A los grandes amigos y compañeros que durante toda la carrera fueron un apoyo fundamental: Isaac Sanchez, Carlos Montalvo, Robinson Iza, Andres Ortega, Esteban Machado, Bryan Santander, Matías Romero. Gracias por enseñarme a trabajar en equipo, a compartir conocimiento, gracias por el tiempo de calidad y hacer de la experiencia universitaria algo inolvidable.

A todos los docentes de la Escuela Politécnica Nacional por su guía y sobre todo a David Mejía, M.Sc. por compartir la paciencia, el apoyo, el tiempo y sus conocimientos durante la dirección del presente Trabajo de Titulación.

Joseph Estuardo Bravo Macas

ÍNDICE DE CONTENIDO

RESUMEN

En el presente Trabajo de Titulación se desarrolló un prototipo de sistema de gestión de inventario para una PYME lavadora y lubricadora.

El prototipo permite administrar el inventario de productos, realizar búsquedas de productos, administrar clientes, proveedores y usuarios del sistema, así también, permite generar comprobantes de compra y venta, generar reportes de compras y ventas, y visualizar notificaciones.

Este documento está organizado en 4 capítulos.

El primer capítulo contiene una breve definición de la arquitectura de 4 niveles, además detalla las tecnologías y los *frameworks* utilizados, y finalmente presenta una descripción acerca de la metodología Kanban aplicada al desarrollo de software.

En el segundo capítulo se definen los requerimientos del prototipo a través de entrevistas y encuestas. A partir de estas, se generan las historias de usuario y casos de uso. A continuación, se esquematizan los diagramas de bases de datos, clases, actividad, y secuencia. Finalmente se resume la implementación de las funcionalidades principales del prototipo.

En el tercer capítulo se presentan los resultados de las pruebas de funcionamiento, las pruebas de aceptación y los cambios realizados al prototipo.

El cuarto capitulo exponen las conclusiones y recomendaciones generadas durante el desarrollo de este Trabajo de Titulación.

Por último, en los anexos se presenta la encuesta realizada para el levantamiento de los requerimientos, los diagramas de secuencia, el código del prototipo, el manual de instalación y el manual de usuario.

PALABRAS CLAVE: prototipo, inventario, administración, Kanban.

ABSTRACT

This Degree Project presents a prototype of an inventory management system was developed for a washing and lubricating cars company.

The prototype allows to manage the inventory of products, perform product searches, manage customers, suppliers, and system users, generate purchase and sales receipts, generate purchase and sales reports, and display notifications.

This document is composed of 4 chapters.

The first chapter contains a brief definition of the 4-layer architecture, it also details the technologies and frameworks used, and finally a description of the Kanban methodology applied to software development is presented.

In the second chapter the requirements of the prototype are defined through interviews and surveys. From these, user stories and use cases are generated. Next, the database, class, activity, and sequence diagrams are schematized. Finally, the implementation of the main functionalities of the prototype are summarized.

The third chapter focuses on the results of the functional tests, the acceptance tests and the changes made to the prototype.

The fourth chapter presents the conclusions and recommendations acquired during the development of this Degree Project.

Finally, the annexes present the requirements survey, the sequence diagrams, code, and the user's manual.

KEYWORDS: prototype, inventory, management, Kanban.

3. INTRODUCCIÓN

El presente Trabajo de Titulación se origina ante la falta de un sistema de inventario en la PYME “Lavadora y lubricadora El Chino”. Esta PYME carece de un software para el control de inventario, que permita gestionar dicho inventario, así como administrar datos de los proveedores autorizados y la disponibilidad de los diferentes productos que se ofrecen a los clientes para el cuidado y mantenimiento de sus vehículos.

Actualmente toda esta información es gestionada a mano de manera tradicional, aunque el ingreso y salida de productos es registrada con fecha, es complicado tener una vista general de los productos disponibles.

Este trabajo inicia describiendo el proceso de levamiento de la información, diseño, implementación y pruebas ejecutadas. El prototipo fue desarrollado haciendo uso de Windows *Forms*, Windows *Communication Foundation*, *Data Access Object* y SQL (*Structured Query Language*) *Server* para la base de datos.

El prototipo corresponde a un sistema distribuido, el cual está conformado por una aplicación de escritorio y un servicio web. El prototipo cuenta con una interfaz de inicio de sesión capaz de distinguir entre los dos tipos de usuarios (administrador y empleado). El usuario empleado, puede realizar las siguientes tareas: buscar productos, generar comprobantes de venta, administrar clientes, generar reportes de venta y visualizar las notificaciones. El usuario administrador, además de poder ejecutar las tareas del empleado, tendrá la capacidad de realizar la administración de productos, usuarios y proveedores, generar comprobantes de compra y generar reportes de compra.

1.1 OBJETIVOS

El objetivo general del presente Trabajo de Titulación es desarrollar un prototipo de sistema gestión de inventario para una PYME tipo lavadora y lubricadora.

Los objetivos específicos del presente Trabajo de Titulación son:

- Analizar las herramientas adecuadas para el desarrollo del presente trabajo de titulación.
- Diseñar con base a los requerimientos de la PYME los componentes necesarios para el prototipo.

- Implementar el prototipo de acuerdo con el diseño.
- Evaluar los resultados de las pruebas de funcionalidad.

1.2 ALCANCE

En este trabajo se plantea desarrollar un prototipo de sistema distribuido que permita llevar un control de inventario, y administrar datos de proveedores y productos. Las herramientas que se usarán para el desarrollo del prototipo son: Windows *Forms* (.Net *Framework*), Windows *Communication Foundation* (WCF), SQL Server y GIT.

Se aplicará una encuesta a 5 empleados de la PYME con el fin de determinar los requerimientos funcionales y no funcionales del prototipo, mismos que se detallarán como historias de usuario.

El prototipo contará con los siguientes módulos:

- Inicio de sesión: Permitirá iniciar sesión mediante el uso de un usuario/correo y una contraseña.
- Gestión de comprobantes: Permitirá registrar la salida de mercancía por medio de un comprobante de compra imprimible, que contendrá información como: tipo, cantidad y precio de los productos despachados.
- Gestión de productos: Permitirá realizar el CRUD (crear, visualizar, actualizar y eliminar) mediante el uso del código de barras de cada uno de los productos, además en el caso de la visualización se dispondrá de mecanismos de búsqueda con base en criterios como: nombre, código, marca entre otros que se determinen durante el desarrollo del prototipo.
- Gestión de empleados: Permitirá realizar el CRUD (crear, visualizar, actualizar y eliminar) de empleados.
- Gestión de proveedores: Permitirá realizar el CRUD (crear, visualizar, actualizar y eliminar) de proveedores.
- Reportes: Permitirá la visualización de los ingresos y egresos por día y por mes.

El usuario podrá iniciar sesión mediante el uso de un usuario/correo y una contraseña, estas credenciales determinarán el rol del usuario dentro del prototipo. Se implementarán dos tipos de roles: el rol de empleado permitirá realizar una búsqueda de productos

disponibles; además podrá registrar la salida de mercancía por medio de un comprobante de compra imprimible y realizar la administración de clientes, y el rol de administrador permitirá, además de las funciones del empleado, realizar la gestión de empleados, proveedores y productos; así como durante el ingreso de mercadería de proveedores se hará uso del código de barras de los productos; y la visualización de los ingresos y egresos por día y por mes.

Además, el prototipo contará con un panel de notificaciones en el cual se indicará la escasez de un producto. Para poder indicar la escasez de un producto se definirá un límite configurable que dependerá de cada uno de los productos.

Para el desarrollo del presente trabajo de titulación se utilizará la metodología Kanban.

En el presente trabajo se utilizará una arquitectura de 4 niveles como se puede apreciar en la Figura 3.1.

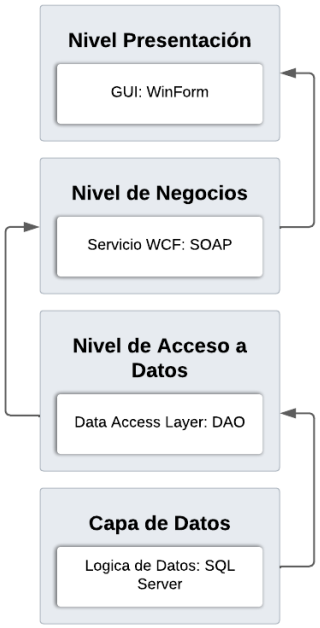


Figura 3.1 Diagrama de la arquitectura.

Para evaluar los resultados del prototipo se realizarán pruebas de funcionalidad durante las diferentes fases de desarrollo, adicionalmente se realizarán pruebas de validación al mismo con la ayuda de empleados de la PYME y finalmente se realizará una encuesta para evaluar si el prototipo cumple con los requerimientos levantados.

Finalmente se aclara que en este Trabajo de Titulación se generará un producto final demostrable.

1.3 MARCO TEÓRICO

Esta sección describe los conceptos en los que se sustenta el presente Trabajo de Titulación. En primer lugar, se presenta la arquitectura utilizada y un resumen de las tecnologías, herramientas y *frameworks*[1] utilizados. Posteriormente, se detalla la metodología Kanban y el tipo de pruebas aplicadas al prototipo.

3.1.1 ARQUITECTURA DE CUATRO NIVELES

La arquitectura de cuatro niveles consiste en dividir la aplicación en niveles que se distribuyen entre el cliente y el servidor, como se puede apreciar en la Figura 3.2. Esto con el fin de que cada nivel cumpla con un determinado rol permitiendo solventar problemas como escalabilidad, flexibilidad, seguridad, tolerancia a fallos, etc. [1].

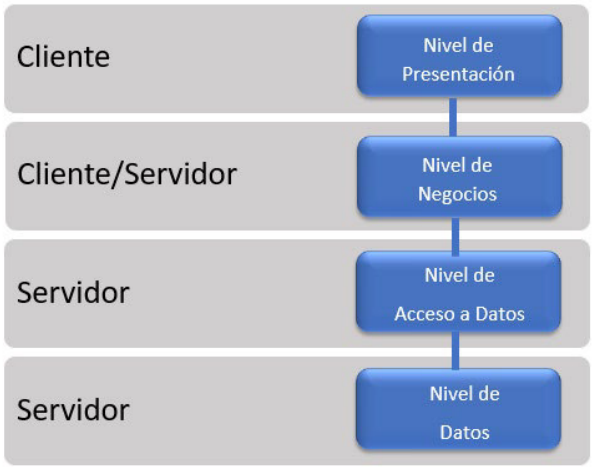


Figura 3.2 Arquitectura de 4 niveles

En esta arquitectura, cada uno de los niveles se coloca de forma horizontal en diferentes equipos en una misma capa[2], de esta forma se restringe la comunicación a capas

[1] *Framework*: Es un conjunto de herramientas y módulos que pueden ser utilizados para varios proyectos.

[2] Capa: Según el modelo TCP/IP es un conjunto de protocolos que permiten la interacción con las capas que se encuentran por encima y por debajo de ellas.

subyacentes. Los niveles que componen esta arquitectura son: nivel de presentación, nivel de negocios, nivel de acceso a datos y nivel de datos.

Nivel de Presentación: Se encarga de interpretar las acciones del usuario y presentar los conceptos de negocio mediante el uso de interfaces de usuario (UI). Las responsabilidades de este nivel son la explotación de los procesos de negocio, así como bridar información acerca del proceso en ejecución y la configuración de las reglas de validación de dicha interfaz [2]. En este nivel se emplea usualmente un navegador web, y se usan tecnologías como HTML[3], CSS[4], JavaScript[5], mediante una aplicación web o mediante una aplicación de escritorio; incluso hoy en día se pueden implementar aplicaciones para dispositivos móviles.

Nivel de Negocios: Se encarga de la lógica del negocio, es decir, es responsable de encapsular la lógica de negocio, brindar información sobre los procesos de negocio y de implementar las reglas del negocio. En este nivel se implementa la funcionalidad principal del sistema, y se tiene en cuenta todas las consideraciones que debe tomar el software antes, durante y después de realizar una transacción, por ello se lo considera esencial en la arquitectura. El contar con una diferenciación clara entre las reglas del negocio y los detalles de implementación e infraestructura es la principal fortaleza del nivel de negocios, esta diferenciación permite aislar la lógica de negocio de la aplicación mientras se realizan mantenimientos o cambios en otros niveles [2].

Nivel de Acceso a Datos: Se encarga de proporcionar acceso tanto a los datos internos del sistema (por ejemplo, la base de datos), como a los datos expuestos fuera del sistema, tales como servicios[6] web de sistemas externos. Este nivel se compone mediante clases o componentes que contienen la lógica para acceder a las fuentes de datos. Los componentes de acceso a datos realizan directamente la comunicación con la base de datos y las operaciones de persistencia como: consultar, insertar, editar y eliminar registros, para lo cual se suele emplear sentencias escritas en SQL[7].

---

[3] HTML (*HyperText Markup Language*): es el lenguaje de marcado estándar para el diseño páginas web.

[4] CSS (*Cascading Style Sheets*): Es un lenguaje utilizado para diseñar y estructurar páginas web.

[5] JavaScript: Es un lenguaje de programación web interpretado de alto nivel.

[6] Servicios: Los servicios consisten generalmente en un conjunto de operaciones que pueden ser utilizadas por un cliente.

[7] SQL (*Structured Query Language*): Es un lenguaje de programación diseñado para administrar y recuperar información en bases de datos.

Nivel de Datos: Se encarga de almacenar de forma persistente la información de la aplicación.

3.1.2 BASES DE DATOS

Una base de datos es una colección estructurada de información, normalmente contralada por un sistema de gestión de base de datos (DBMS – *Database Managment System*), típicamente los datos son organizados en filas y columnas que a su vez componen una serie de tablas, haciendo posible que el procesamiento y la consulta de los datos sean eficientes. Posteriormente es posible acceder, administrar, modificar, actualizar, controlar y organizar estos fácilmente.

Existen diferentes gestores de bases de datos entre ellos SQL Server, los gestores requieren de un lenguaje de consultas estructurado como SQL.

SQL Server es una plataforma de Microsoft que brinda un sistema de base de datos relacionales[8] el cual posee una variedad de características y herramientas que ayudan al desarrollo y administración de bases de datos [3]. El principal servicio de SQL Server es el motor de base de datos; el motor es responsable de asegurar, procesar y optimizar el acceso a los datos relacionales.

3.1.3 DATA ACCESS OBJECT

El patrón de acceso DAO (*Data Access Object*) trata de desvincular el acceso a los datos de su almacenamiento subyacente, esto permite elegir diferentes fuentes de datos, de ser necesario, sin cambiar la lógica de negocios. El patrón de acceso DAO se visualiza en la Figura 3.3 y está compuesto por cuatro componentes [4]:

- **Objeto de negocios:** representa la lógica del negocio, se encarga de saber qué y cómo modificar el contenido, mas no cómo almacenarlo.
- **Objeto de acceso a datos:** oculta la fuente de datos, es decir, impide que el nivel de negocios se comunique directamente con la base de datos, para ello, es necesario pasar por el objeto de acceso a datos

[8] Base de datos relacionales: Es una colección de elementos organizados en tablas descritas formalmente donde cada fila contiene una instancia única y representa una relación entre un conjunto de valores.

- **Fuente de datos:** es donde realmente se encuentran los datos.
- **Objeto de transferencia:** es un objeto intermedio utilizado para transportar los datos desde el objeto de acceso a datos al objeto de negocios y viceversa. Para que un cambio en el objeto de transferencia sea permanente primero debe confirmarse en el objeto de acceso a datos.

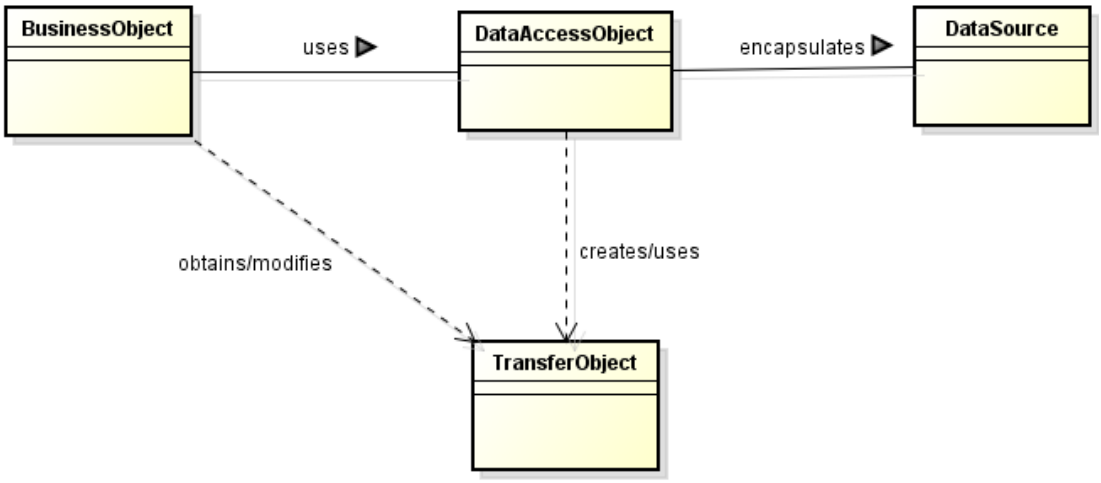


Figura 3.3 Elementos del patrón DAO

3.1.4 SERVICIO WCF

WCF (Windows *Communication Foundation*) es un conjunto de herramientas que definen las interacciones de los servicios, el *marshalling*[9] y la gestión de varios protocolos para el desarrollo y despliegue de servicios en Windows. WCF proporciona una forma homogénea de desarrollar aplicaciones distribuidas mediante un modelo de programación orientado a los servicios [5], [6].

3.1.4.1 Modelo de Programación WCF

Entender el modelo de programación WCF implica tener en cuenta algunos conceptos necesarios para configurar y consumir un servicio WCF.En la Figura 3.4 se presentan los componentes de WCF.

[9] *Marshaling*: Consiste en tomar una colección de datos y ensamblarlos en forma adecuada para transmitirlos en un mensaje.

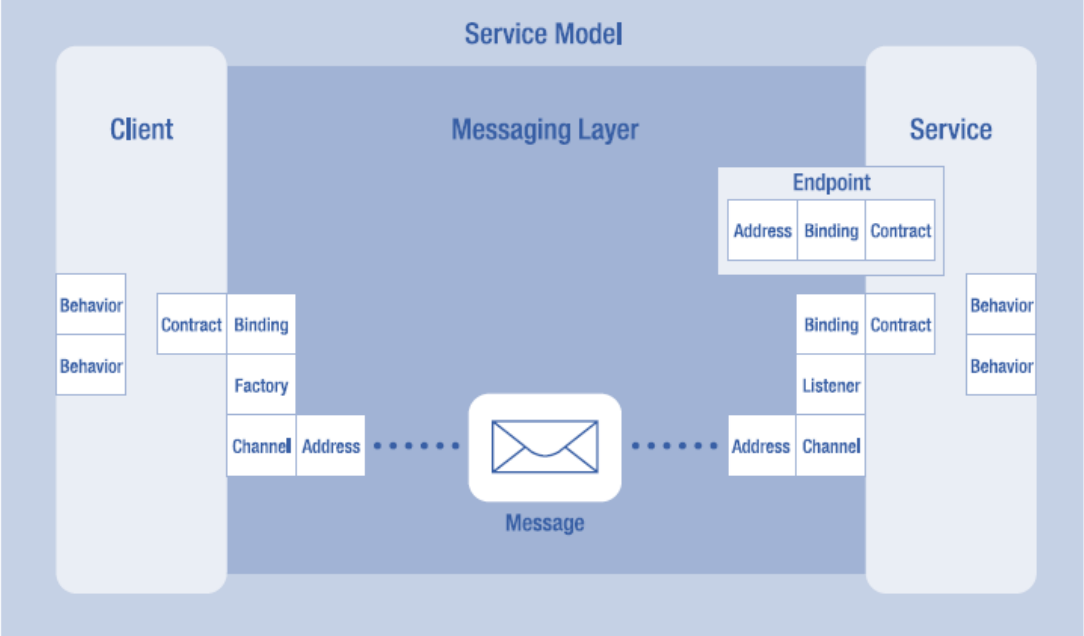


Figura 3.4 Arquitectura de WCF

3.1.4.1.1 Addresses

La dirección es una parte fundamental para poder hacer uso de un servicio, cada servicio está asociado a una dirección única la cual está compuesta por dos elementos principales: la ubicación del servicio y el protocolo o esquema de transporte utilizado para comunicarse con el servicio. El formato de una dirección es el siguiente:

```
scheme://<machinename>[:port]/path
```

Donde:

- `Scheme`: Es cualquier tipo de protocolo de transporte soportado.
- `Machinename`: Es el nombre del servidor.
- `Port`: Es el puerto de escucha del servicio.
- `Path`: es la ubicación del servicio y permite la diferenciación de servicios en un mismo equipo.

3.1.4.1.2 Bindings

Binding especifica como el cliente se comunica con el servicio. Su rol es controlar:

- El transporte: Se pueden utilizar diferentes protocolos de transporte, como HTTP[10], MSMQ[11], IPC [12] y TCP[13], siendo el primero el más utilizado.
- Los canales: Pueden manejar mensajes síncronos, asíncronos, de una vía y bidireccionales.
- La codificación: Se admiten varios esquemas de codificación tales como: XML[14], para permitir la interoperabilidad; codificación binaria, para optimizar el rendimiento; o, MTOM[15], para cargas útiles grandes.
- Los protocolos *Web Service* (WS): Soporta una variedad de protocolos WS, entre ellos *WS-Security*[16], *WS-Federation*[17], *WS-Reliability*[18], etc.

WCF define 5 *bindings* de uso frecuente: *Basic binding*, *TCP binding*, *IPC binding*, *WS binding* y *MSMQ binding*. Siendo el primero el más utilizado, el cual se implementa mediante la clase `BasicHttpsBinding` y permite exponer un servicio WCF como un servicio web ASMX[19].

[10] HTTP (*Hypertext Transfer Protocol*): Protocolo de comunicación que permite las transferencias de información a través de archivos en la Word Wide Web.

[11] MSMQ (*Microsoft Message Queuing*): Es una implementación de colas de mensajes desarrollada por Microsoft.

[12] IPC (*Inter Porcess Communication*): Es un mecanismo que nos permite intercambiar datos entre múltiples subprocesos en uno o más programas o procesos.

[13] TCP (*Transmission Control Protocol*): Protocolo que define cómo establecer y mantener una conversación a través de la red.

[14] XML (*Extensible Markup Language*): Es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos.

[15] MTOM (*Message Transmission Optimization Mechanism*): Es un mecanismo que permite un envío eficiente de datos binarios entre servicios Web.

[16] WS- Security (*Web Service Security*): Es un protocolo para suministrar seguridad a servicios web.

[17] WS-Federation (*Web Service Federation*): Es un *framework* que proporciona un lenguaje general y un mecanismo seguro para conectar a los usuarios y los recursos.

[18] WS-Federation (*Web Service Reliability*): Es un protocolo basado en SOAP para intercambiar mensajes SOAP.

[19] ASMX (*Active Server Methods*): Ofrece la capacidad de crear servicios web que envían mensajes mediante SOAP.

3.1.4.1.3 Contracts

Un contrato es una forma estándar para describir lo que hace un servicio, independientemente de la plataforma. Todos los servicios en WCF requieren de un contrato. Se definen cuatro tipos de contratos:

- *Service contracts*: Es una declaración de comportamiento expuesto.
- *Data contracts*: Define los tipos de datos que pasan hacia y desde el servicio.
- *Faul contracts*: Define que errores son generados por el servicio y su manejo.
- *Message contracts*: Permite al servicio interactuar directamente con los mensajes.

3.1.4.1.4 Endpoints

WCF formaliza la relación entre un *address*, que define dónde se encuentra el servicio; un *binding*, que define cómo se comunica con el servicio; un *contract* que define lo que hace el servicio como *endpoint*. Este conjunto de elementos es conocido como el ABC del servicio.

3.1.4.2 Contratos en WCF

***Service Contract*:** Especifica las operaciones o métodos que se exponen al mundo exterior. También se lo conoce como la interfaz del servicio o el comportamiento (*behavior*) expuesto del servicio. Describe lo que se puede esperar del servicio y sus requisitos para su uso [6]. Para crear un contrato es necesario implementar una interfaz o una clase en .NET, en la misma, se incluye el atributo `ServiceContract` para indicar que es un contrato de servicio y el atributo `OperationContract` para indicar los métodos que serán expuestos por el servicio.

Data Contract: Es un acuerdo formal que especifica los datos que se intercambiarán entre el servicio y el cliente. El `DataContract` determina qué parámetro y tipo de retorno se serializará o deserializará para ser transmitido o recibido entre las dos partes. El `DataContract` se publica en los metadatos del servicio, facilitando a los clientes la conversión desde una representación de datos neutral e independiente de la tecnología hacia sus representaciones nativas. Este proceso de conversión se denomina serialización [5]. En WCF la serialización es el proceso de convertir un objeto en una representación XML, serializando solo los campos públicos y los valores de propiedad de un objeto. El

proceso de reconstruir el mismo objeto partiendo del XML se denomina deserialización. Una vez se acepta o devuelve una operación de servicio con cualquier tipo de parámetro, WCF utiliza `DataContractSerializer` para serializar y deserializar ese parámetro [5].

3.1.4.3 Servicio WCF con SOAP

SOAP (*Simple Object Access Protocol*) es un protocolo de comunicación ligero basado en XML, que se utiliza para intercambiar información tipificada y estructurada entre cliente y servidor. SOAP trabaja independientemente de la plataforma, es decir, permite invocar métodos sin conocer detalles específicos del software que se ejecuta en las máquinas remotas [6].

SOAP establece un esquema para representar el contenido de los mensajes de solicitud y respuesta, además de un esquema para la comunicación de documentos mediante el uso de XML. En la comunicación se pueden utilizar diferentes protocolos de transporte, como HTTP, SMTP [20], TCP o UDP[21] , siendo el primero el más utilizado.

Las API[22] de SOAP han sido implementadas en varios lenguajes de programación, tales como Java [23], JavaScript, Python[24] , .NET[25], etc., lo que permite que el programador no tenga que preocuparse de los detalles de su implementación.

3.1.5 INTERFACES DE USUARIO

La interfaz de usuario (UI) es el punto en el cual interactúan aplicaciones y usuarios. Una interfaz gráfica de usuario (GUI) permite a los usuarios comunicarse con los dispositivos electrónicos mediante una representación visual, la cual es una combinación de iconos

[20] SMTP (*Simple Mail Transfer Protocol*): Protocolo de red utilizado para el intercambio de mensajes de correo electrónico.

[21] UDP (*User Datagram Protocol*): Protocolo que permite el envío de datagramas a través de la red sin que se haya establecido una conexión.

[22] API (*Application Programming Interface*): Interfaz que permite la comunicación entre dos sistemas o plataformas diferentes mediante un conjunto de protocolos y definiciones.

[23] Java: Es un lenguaje de programación general orientado a objetos y basado en clases.

[24] Python: Es un lenguaje de programación multiparadigma y multipropósito.

[25] .NET: Es un *framework* de Microsoft independiente de la plataforma de hardware que permite un rápido desarrollo de aplicaciones.

gráficos como menús, cursores, pestañas, ventanas, barras de desplazamiento, entre otros.

El objetivo es crear una interfaz intuitiva para que el usuario se comunique con la aplicación, recibiendo una respuesta visual de forma inmediata al realizar una tarea [7].

3.1.5.1 Windows Forms para .NET

Windows Forms es una tecnología para el desarrollo de aplicaciones inteligentes para .NET Framework, esta tecnología permite simplificar la ejecución de tareas habituales de una aplicación, como la lectura y escritura mediante la administración de un conjunto de bibliotecas.

En cuanto a productividad, Windows Forms ofrece funcionalidades como la de arrastrar y soltar controles visuales, fragmentos de código reutilizable, acceso al hardware y al sistema de archivos local. Además de brindar facilidades al momento de desplegar y actualizar la aplicación.

Windows Forms ofrece una gran variedad de herramientas que facilitan el desarrollo de aplicaciones de escritorio, entre las cuales se encuentran:

- Formularios: Es una superficie en la cual el usuario puede visualizar la información de la aplicación, normalmente se añaden controles a los formularios para que el usuario pueda interactuar mediante la ayuda del mouse o del teclado.
- Controles: Es un elemento reutilizable que encapsula funcionalidades de la interfaz de usuario permitiendo el ingreso de datos o la visualización de estos. Windows Forms proporciona varios controles listos para usar y también brinda los elementos necesarios para la creación de controles personalizados.
- Eventos: Un evento es una acción que se produce cuando un usuario interactúa con un control, esta acción se puede responder o manejar mediante código del programa o del sistema.

3.1.6 SISTEMA DE CONTROL DE VERSIONES

El proceso de almacenar y recuperar cambios durante el desarrollo de un proyecto es denominado control de versiones. Un sistema de control de versiones (SCV) permiten a un

equipo de desarrolladores administrar las versiones previas de un proyecto conjuntamente con los archivos que componen el mismo con el fin de corregir errores o analizar funciones [8].

Entre las principales funciones de un SCV se destacan las siguientes:

- Guardar un historial de cambios realizados al proyecto.
- Administrar las versiones de un proyecto
- Visualizar las diferencias entre versiones.
- Regresar a una versión anterior deshaciendo los cambios desde la versión antigua hasta la actual.
- Permitir el desarrollo en paralelo de un mismo proyecto, mediante la creación de ramas de trabajo.

El esquema de un sistema de control de versiones se puede apreciar en la Figura 3.5.

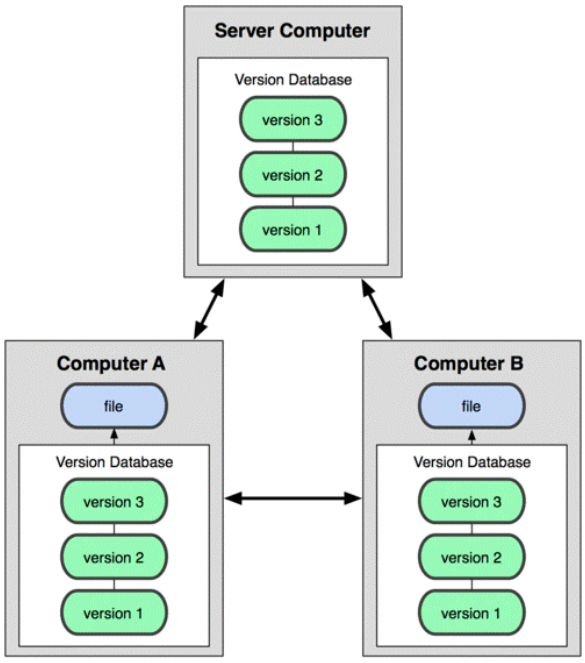


Figura 3.5 Esquema de un sistema de control de versiones

3.1.6.1 Git

Git es uno de los más populares sistemas de control de versiones, se caracteriza por ser rápido, eficiente, gratuito y de código abierto. Permite la gestión de todo tipo de proyectos desde simples hasta complejos [9].

Git está constituido por una estructura de estados como se muestra en la Figura 3.6, la cual está dividida en:

- Directorio de trabajo (*Working Directory*): Es una carpeta local que contiene los archivos de la versión actualizada del proyecto, sobre los cuales se van a realizar los cambios.
- Área de preparación (*Staging Area*): Es un archivo contenido en el repositorio local que incluye la información de los cambios que se efectuarán en la siguiente confirmación.
- Repositorio Local (*Local Repository*): Es el directorio de Git donde se almacenan los metadatos y la base de datos de objetos del proyecto. Es lo que se copia cuando se clona un repositorio por lo tanto es la parte más importante de Git.
- Repositorio Remoto (*Remote Repository*): Es una copia del proyecto alojada en la nube, que contiene los últimos cambios confirmados. Esta copia permite acceder al proyecto desde cualquier parte del mundo.

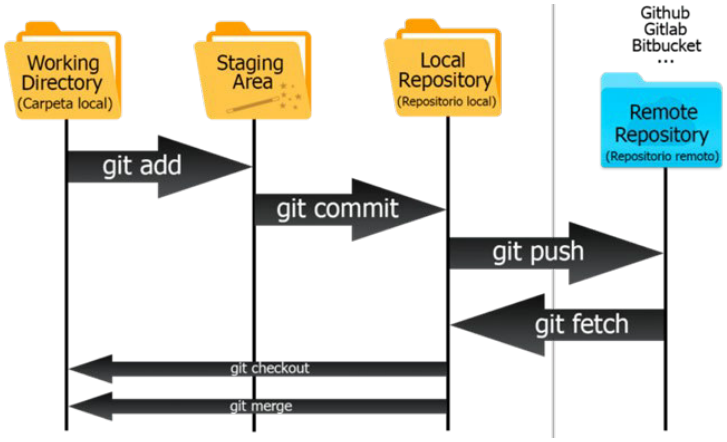


Figura 3.6 Funcionamiento de GIT

Actualmente la gran mayoría de los sistemas de control de versiones disponen de algún mecanismo para soportar el uso de múltiples ramas, el trabajar con ramas implica tomar la rama principal de desarrollo o llamada en muchos casos *master* y a partir de ella continuar trabajando y generando cambios sin que estos afecten a la rama principal.

Las ramas de Git son muy útiles cuando: se trabaja en equipos de desarrollo, se necesita añadir nuevas funciones o simplemente solucionar un error. Conforme avanza el proyecto,

estas ramas deberán ser incorporadas a la rama principal mediante un proceso conocido como *merge*, el cual puede ser de dos tipos [9].

Fast Forward Merge: Como se puede apreciar en la Figura 3.7, si entre la rama destino y la rama actual ha ocurrido un proceso lineal, la fusión consiste en desplazar el puntero a la rama destino (ver la Figura 3.8). Esto es considerado un avance rápido (*Fast Forward*).

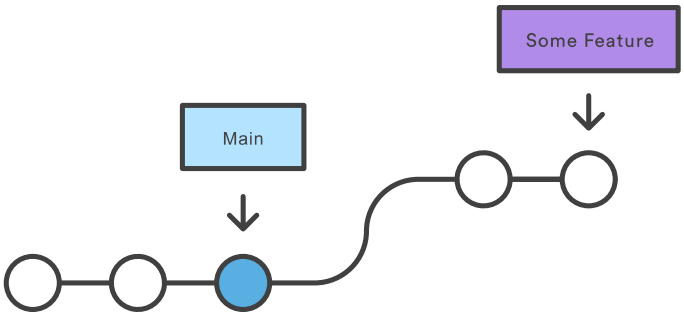


Figura 3.7 Antes de realizar el *Fast Forward Merge*

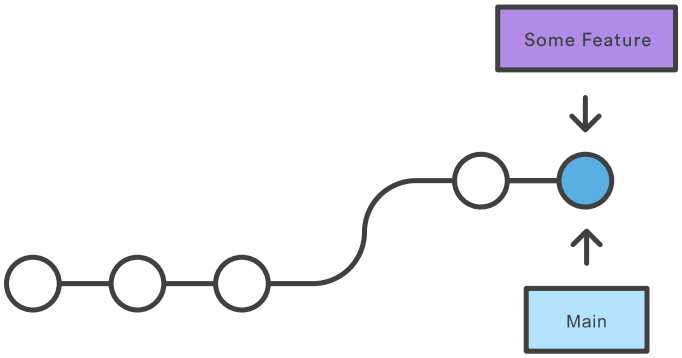


Figura 3.8 Después de realizar el *Fast Forward Merge*

3-Way Merge: Si no ha ocurrido un proceso lineal, Git procede a realizar una fusión de 3 vías la cual consiste en tomar como referencia tres eventos: el último *commit* de la rama actual (*main tip*), el último *commit* de la rama destino (*feature tip*) y un *commit* común a las dos ramas (*common base*), como se muestra en Figura 3.9. A partir de los 3 eventos se genera la confirmación de fusión que permite incorporar los cambios de las dos ramas

dando como resultado el *commit* de fusión (*new merge commit*) como se visualiza en la Figura 3.10.

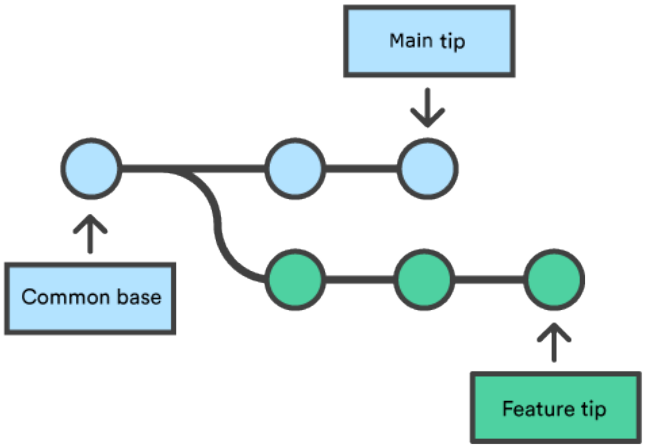


Figura 3.9 Antes de realizar el *3-Way Merge*

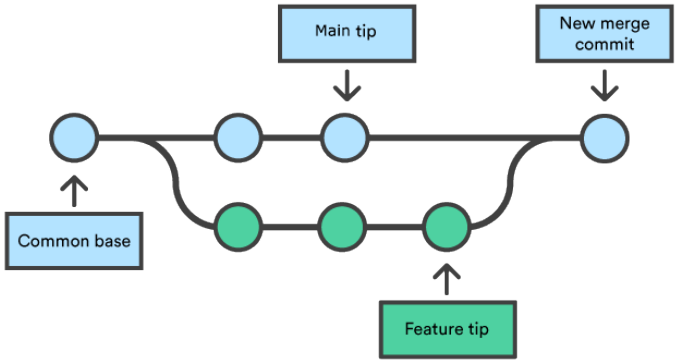


Figura 3.10 Después de realizar el *3-Way Merge*

Si se han realizado cambios en la misma parte, de un mismo archivo, de las dos ramas inmersas en la fusión, Git no conseguirá ejecutar la fusión de manera automática. En este caso informará que ha ocurrido un conflicto y mostrará los archivos que deben resolverse manualmente.

Para utilizar Git se puede hacer uso de varios tipos de herramientas, la línea de comandos es una de las más utilizadas debido a que permite hacer uso de todas las funciones del sistema y solucionar todo tipo de conflictos o contratiempos. En la Tabla 3.1 se muestran los comandos básicos de Git.

Tabla 3.1. Comandos básicos de Git

Comando	Descripción
`git init`	Toma un directorio y lo convierte en un nuevo repositorio Git permitiendo controlar sus versiones.
`git add`	Añade contenido al área de preparación o *staging area* para la siguiente confirmación
`git commit`	Toma todos los contenidos del área de preparación y se procede a registrar una nueva instantánea en la base. También el puntero de la rama avanza en la rama actual.
`git status`	Permite visualizar los estados de los archivos en el directorio de trabajo, muestra información acerca de archivos modificados con o sin seguimiento y con o sin confirmación.
`git log`	Permite visualizar la historia de un proyecto desde la última instantánea hacia atrás. Por defecto muestra la historia de la rama actual
`git branch`	Es un comando que permite gestionar las ramas. Mediante este comando se puede crear, eliminar y cambiar el nombre a las ramas de un proyecto.
`git merge`	Se utiliza para la fusión de una o más ramas dentro de la rama actual de trabajo.
`git pull`	Mediante este comando Git descargará desde el repositorio remoto la última versión confirmada e intentará combinarlo en la rama actual.
`git push`	Se utiliza para subir las modificaciones realizadas en el directorio y sus archivos al directorio remoto.
`git reberse`	Es un comando que permite capturar todos los cambios realizados y confirmados en una rama para replicarlo sobre otra.

3.1.6.2 GitHub

GitHub es un servicio para el alojamiento de repositorios Git, este repositorio remoto permite la colaboración de varios desarrolladores a través de Git. Además, también es un sistema de desarrollo que posibilita realizar la planificación y seguimiento del desarrollo del código [10].

GitHub ofrece algunos recursos adicionales que pueden ayudar al desarrollo de un proyecto, entre los más importantes están: *Wikis*[26], *GitHub Pages*[27] y *Releases*[28].

3.1.7 KANBAN

Kanban es una metodología de desarrollo ágil que tiene como principal objetivo determinar las tareas por hacer y cambiar su prioridad en base al avance del proyecto, en esta metodología la cadena de trabajo está disponible y visible para todos, lo que hace posible detectar donde se producen atascos en caso de ser necesario [11].

El enfoque principal de Kanban es el desarrollo incremental basado en la división del trabajo en partes, estas partes se escriben en una tarjeta y se colocan en la columna inicial de un tablero, este puede tener tantas columnas como el equipo considera necesarias; en la Figura 3.11 se puede apreciar un ejemplo de un tablero Kanban estándar.

Esta visualización tiene como objetivo que cada uno de los miembros del equipo tenga claro el trabajo que debe realizar y el avance de este a lo largo del tablero, siempre teniendo en cuenta la prioridad de cada actividad [12].

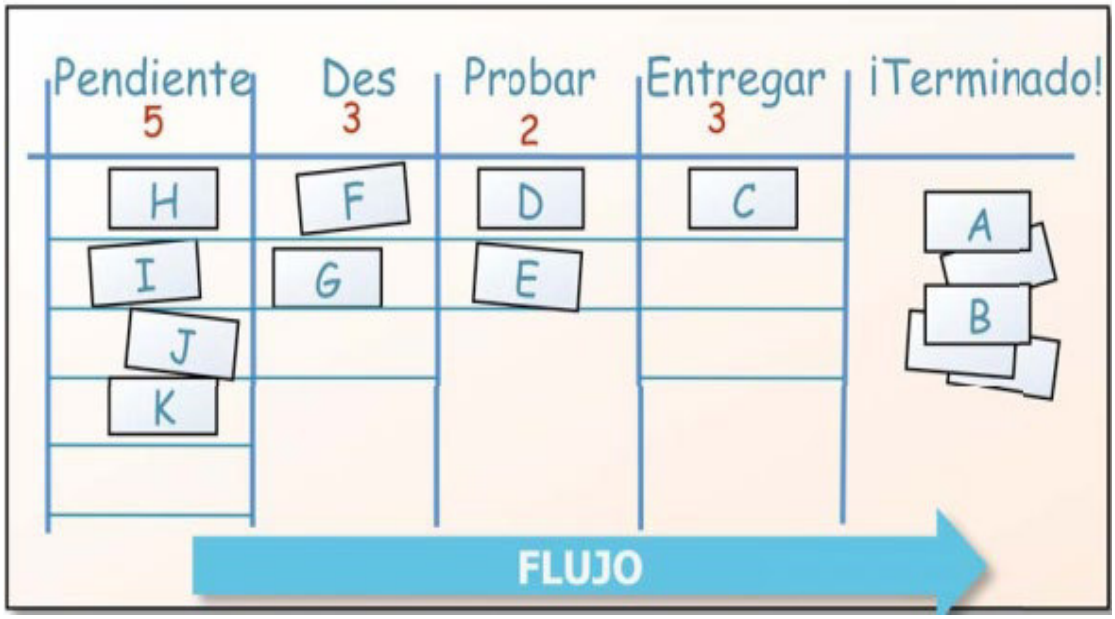


Figura 3.11 Plantilla de Tablero Kanban

[26] *Wikis*: Proporciona información sobre el proyecto: que hace, como funciona y quien puede contribuir.

[27] GitHub *Pages*: Es un sitio Web alojado en GitHub.

[28] *Releases*: Son versiones finales del proyecto que pueden ser descargadas.

En Kanban no se establecen roles, no obstante, de ser necesario estos podrían implementarse, siempre y cuando estos roles añadan valor al proyecto y no entren en conflicto con otros elementos del proceso [13].

Una tarjeta Kanban representa una tarea, e incluye información relevante acerca de esta y su estado en el tablero desde que se inicia hasta que se consideren finalizadas. Estas tarjetas establecen un canal de comunicación entre los miembros del equipo.

Los tableros físicos tienden a ser complicados cuando el flujo de trabajo es complejo, debido a que los miembros del equipo tendrán que acudir frecuentemente al tablero para tenerlo actualizado. Existen soluciones de software que permiten una mejor organización y visualización del trabajo, la conexión remota es una de las características más importantes que contiene el software ya que permiten administrar equipos de trabajo remotos [14].

3.1.7.1 Trello

Trello es una aplicación colaborativa la cual se encuentra disponible en todas las plataformas, como se puede apreciar en la Figura 3.12, Trello permite la organización de tareas en un tablero distribuido por columnas, cada columna está compuesta por tareas o instancias que son representadas por tarjetas. Las tarjetas son la representación básica de cada una de las tareas [15].

Los elementos principales de Trello son:

- Tablero: En este se crean tantas columnas como los desarrolladores del proyecto consideren necesario, cada columna representa una categoría fija que contiene tarjetas, estas se pueden mover de una columna a otra.
- Columnas: Son útiles tanto para ordenar las tareas como para lograr identificar el trabajo y los objetivos de las diferentes áreas del proyecto.
- Tarjetas: Las tarjetas cuentan con el contenido concreto de todos los apartados del proyecto. Los elementos principales son: descripción, comentarios, etiquetas y miembros. Cada una de estas tarjetas actúa de manera autónoma.
- Comentarios: Cada miembro del proyecto puede brindar una retroalimentación con respecto a cada tarea. Además, se puede cargar enlaces o documentos respecto al tema.

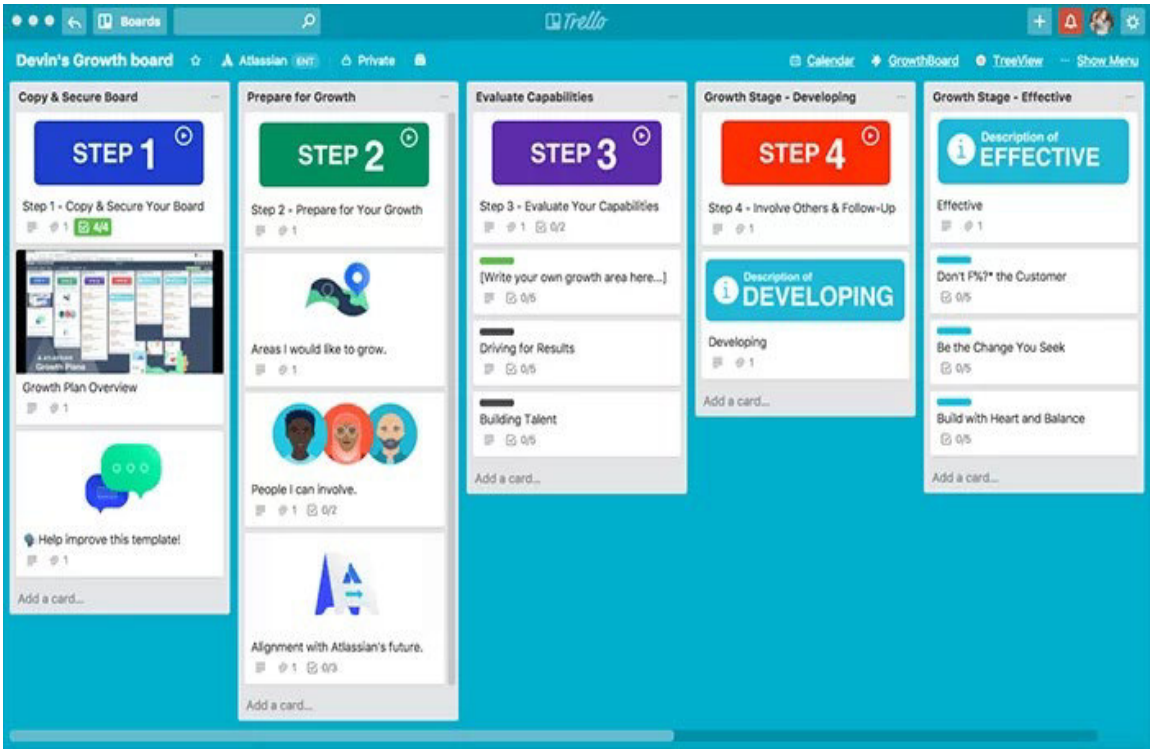


Figura 3.12 Plataforma de Trello

3.1.8 PRUEBAS

De acuerdo a [16], “las pruebas de software son un proceso, o una serie de procesos, diseñados para asegurar que el código informático hace lo que debe hacer y, a la inversa, que no haga nada que no se desee”. Dependiendo del modelo de prueba, está puede ser implementada en cualquier punto del proceso de desarrollo.

Pruebas de Funcionamiento: Las pruebas de funcionamiento son un proceso mediante el cual se pretende determinar las diferencias entre las historias de usuario[29] y el funcionamiento del sistema. Para la realización de estas pruebas, es necesario la creación de casos de prueba a partir de un análisis previo.

Pruebas de Aceptación: Las pruebas de aceptación son responsabilidad del cliente o el usuario final del sistema. En estas, el usuario determina si el sistema cumple con sus requerimientos iniciales y sus necesidades actuales, permitiendo evaluar su satisfacción con el producto entregado.

[29] Historias de usuario (HU): Las historias de usuario son una representación de un requisito descrito en el lenguaje común del usuario.

3.1.9 RELACIÓN CON TRABAJOS ANTERIORES DEL ÁREA

Una vez revisada la biblioteca de la Facultad de Ingeniería Eléctrica y Electrónica se encontraron dos trabajos que guardan relación con el presente trabajo de titulación. La principal diferencia es la tecnología que se empleó en la interfaz del cliente; El trabajo de titulación “DESARROLLO DE UN SISTEMA DISTRIBUIDO BASADO EN SOA Y MVC PARA EL DESPLIEGUE DE INFORMACIÓN MÉDICA DE PASIENTES” utilizó una tecnología web MVC.NET enfocada en el despliegue de información médica de pacientes, mientras que, el trabajo de titulación “IMPLEMENTACIÓN DE UNA APLICACIÓN WEB MOVIL SOBRE UNA ARQUITECTURA ORIENTADA A SERVICIOS SOA” desarrollo diferentes clientes en lenguajes como Java y .NET con el fin de probar la arquitectura orientada a servicios (SOA). El presente trabajo de titulación utiliza una aplicación de escritorio desarrollada en C# mediante Windows Forms para la interfaz del cliente y como servicio web se utiliza WCF con SOAP. El presente trabajo de titulación está enfocado en un sistema de control de inventario para la PYME “Lavadora y lubricadora El Chino”.

4. METODOLOGÍA

Con la finalidad de establecer una visión general del prototipo se realizó una entrevista con el Gerente General de la “Lavadora y Lubricadora el Chino”, en la cual se determinó el alcance y los requerimientos iniciales del prototipo. Adicionalmente, se realizó una encuesta a un total de cinco empleados con el objetivo de recabar información para generar las historias de usuario y los casos de uso.

A partir de las historias de usuario se definió un listado de tareas, mismas que se incluyeron en el *backlog*[30].y en un tablero Kanban, el mismo que cuenta con un grupo de columnas (pendiente, en proceso y finalizado) mismas que representan las fases de desarrollo del prototipo.

Como parte del diseño del prototipo se desarrollaron los siguientes diagramas: casos de uso, arquitectura, esquema relacional de la base de datos, clases, actividades y secuencia.

Como parte de la implementación se procedió a la creación de un repositorio en GitHub con el fin de contar con un sistema de control de versiones y adecuar un ambiente de desarrollo, el cual incluye la instalación del IDE[31] (*Integrated Development Environment*) Visual Studio 2019 con el *framework* .NET, así como las extensiones GitHub Extension, Microsoft Reporting Service Projects y los paquetes Nuget[32] como Microsoft.SqlServer.Types, Microsoft.ReportingService.ReportViewer.Control.Winforms y el administrador de bases de datos MS SQL Managment Studio.

Posteriormente se realizó la codificación de la base de datos de acuerdo con lo planteado en el esquema relacional, del mismo modo, se realizó la codificación del nivel de acceso a datos, el nivel de negocios y finalmente se realizaron las interfaces de usuario.

Por último, se realizaron pruebas para verificar el correcto funcionamiento de cada uno de los niveles con sus componentes y de cada una de las funcionalidades del prototipo ejecutando pruebas de las funcionalidades de creación, búsqueda, edición y eliminación de productos y usuarios; generación, búsqueda y anulación de comprobantes; generación

[30] *Backlog*: Es una lista priorizada de funcionalidades que debe contener un producto.

[31] IDE (*Integrated Development Environment*): Es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software

[32] Paquete Nuget: Es un archivo en formato ZIP con la extensión .nupkg que contiene código compilado (archivos DLL).

y exportación de reportes y generación de notificaciones automáticas. Posteriormente se realizaron pruebas de validación con cinco empleados de la PYME y a través de una encuesta se evaluó si el prototipo cumple con los requerimientos levantados.

4.1 LEVANTAMIENTO DE INFORMACIÓN

4.1.1 ENCUESTAS

Para el desarrollo del prototipo se realizó un total de 5 encuestas, cada una con 11 preguntas al personal de la PYME “Lavadora y Lubricadora el Chino”. El objetivo de la encuesta fue la obtención de información detallada para la generación de las historias de usuario y los casos de uso.

El formato de la encuesta realizada se encuentra en el ANEXO A.

En esta sección se describirán los resultados obtenidos. La pregunta 1 busca identificar los principales productos ofertados por el negocio con el fin de realizar un enfoque especial a los mismos.

En la Figura 4.1 se puede observar que, aceites y filtros son los principales productos vendidos, adicionalmente se aprecia la existencia de otros productos los cuales serán tomados de manera genérica.

Figura 4.1 Resultados de la pregunta 1

La pregunta 2 busca definir las características generales de un producto con el fin de modelar la entidad `producto`. Con base en los resultados presentados en la Figura 4.2 la

entidad contará con los siguientes atributos: marca, descripción, código de barras, precio de venta, precio de compra, margen de venta, cantidad y cantidad mínima; dejando de lado los atributos: número de lote y fecha de fabricación.

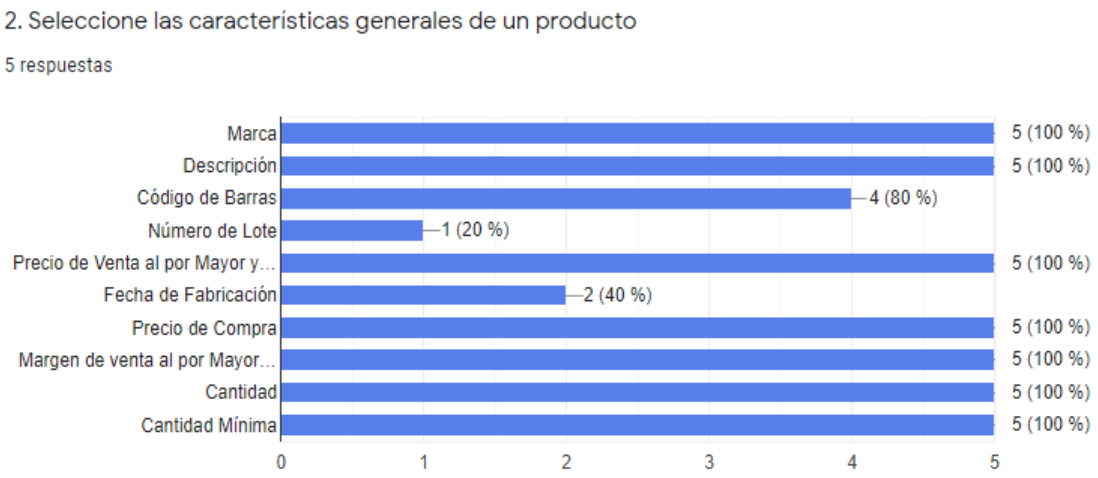


Figura 4.2 Resultados de la pregunta 2

Las preguntas 3 y 4 tuvieron como objetivo permitir obtener las características específicas del producto aceite y del producto filtro, como se observa en la Figura 4.3 y en la Figura 4.4, respectivamente.

En el caso del producto aceite se puede se pueden destacar los siguientes atributos: presentación, SAE, API, tipo de combustible y tipo de aceite. En el caso del producto Filtro los atributos son: tipo de filtro, rosca y código de filtro. Estas características serán añadidas a las características generales obtenidas para un producto.

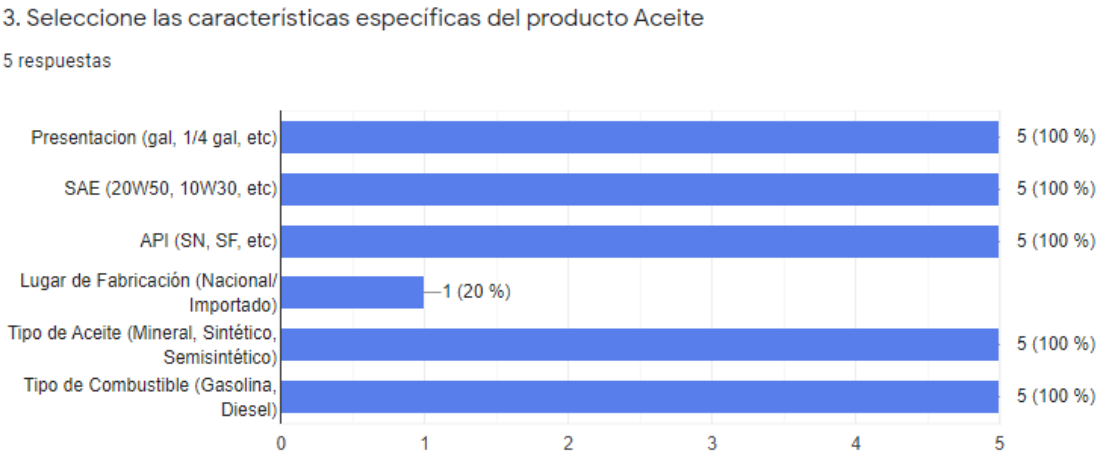


Figura 4.3 Resultados de la pregunta 3

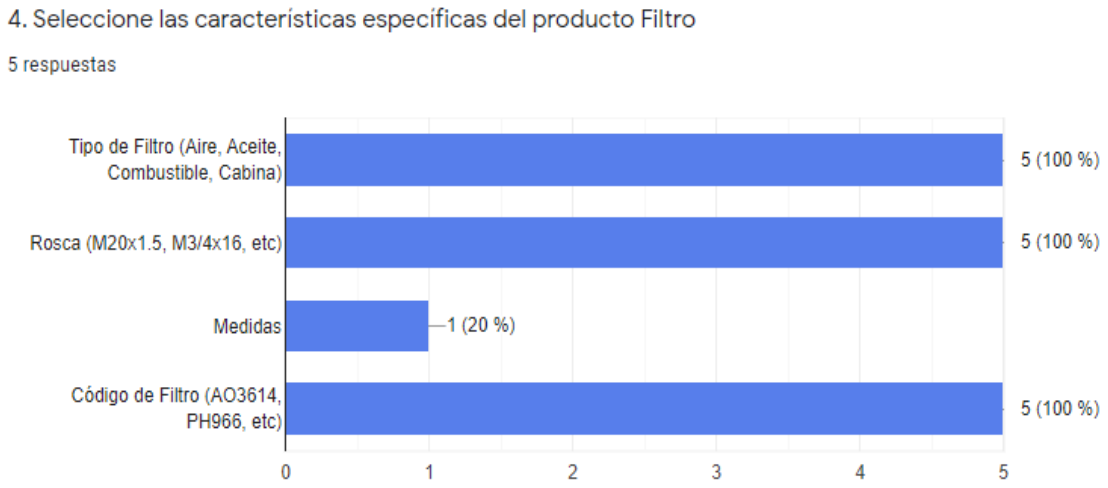


Figura 4.4 Resultados de la pregunta 4

El objetivo de la pregunta 5 es obtener los parámetros necesarios de un vehículo para identificarlo y relacionarlo con los productos. Con base a las respuestas obtenidas la entidad `vehículo` constará con los siguientes atributos: marca, modelo, año y motor, como se observa en la Figura 4.5.

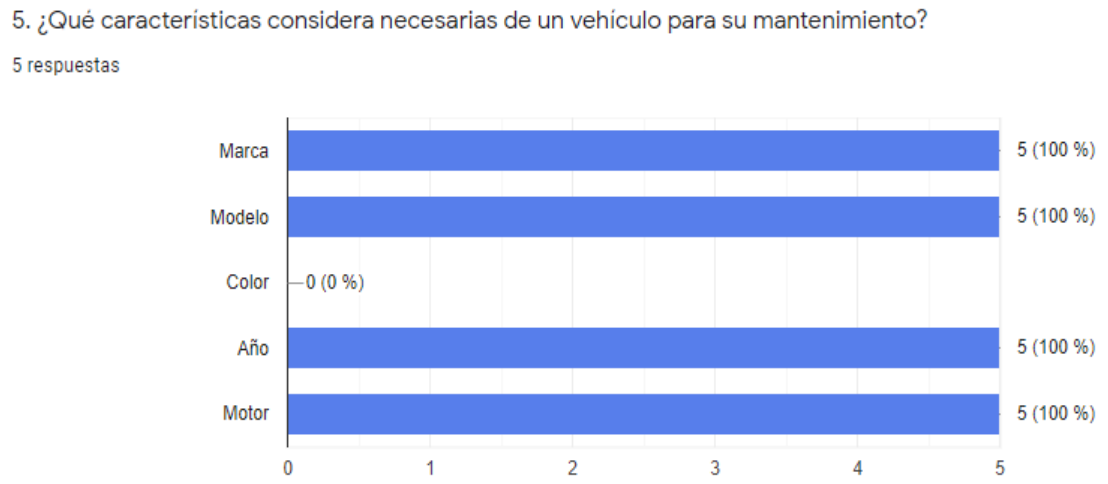


Figura 4.5 Resultados de la pregunta 5

La pregunta 6 ha sido planteada con la finalidad de determinar los parámetros que sean relevantes tanto para la lavadora como para el cliente/proveedor, los resultados se presentan en la Figura 4.6. En este caso la entidad `comprobante` contara con los

siguientes atributos: datos del cliente/proveedor, lista de productos, fecha, subtotal, IVA, total y descuento.

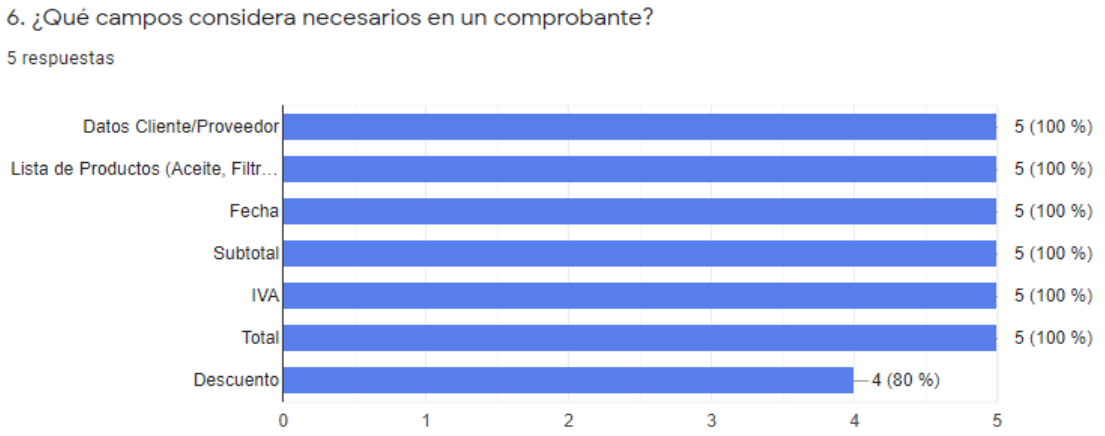


Figura 4.6 Resultados de la pregunta 6

Las preguntas 7 y 8 respectivamente, se realizaron con el fin de obtener los capos necesarios de un cliente/proveedor, en el caso de la entidad `cliente` la información más destacada es: nombre, apellido, cédula, dirección, teléfono y correo; como se puede apreciar en la Figura 4.7 la nacionalidad no es una característica relevante de un cliente. Así mismo, para la entidad `proveedor` se descartó el campo régimen como una característica deseable. Los datos con mayor aceptación han sido: nombre, apellido, RUC, empresa, teléfono, correo y dirección, como se visualiza en la Figura 4.8. Esta información también se utiliza al momento de generar los comprobantes de compra/venta y se visualiza en el documento imprimible.

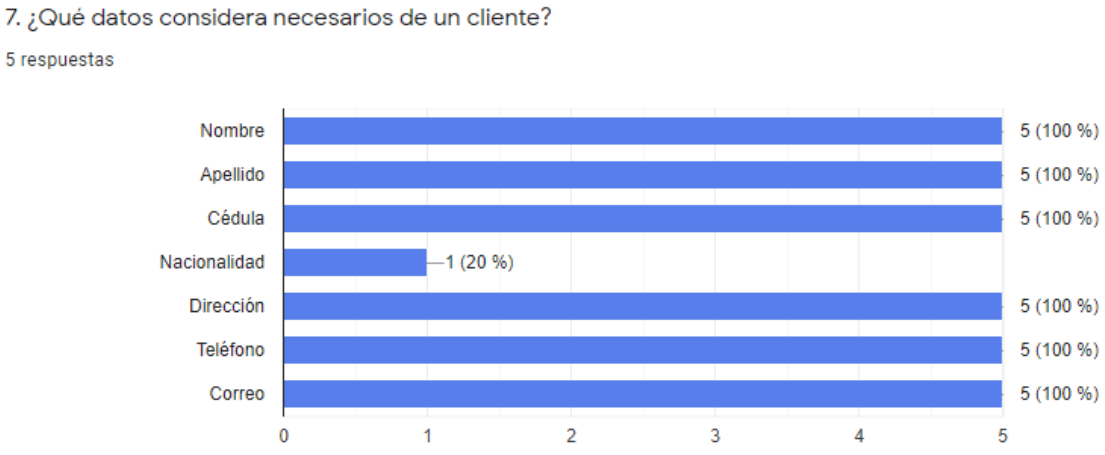


Figura 4.7 Resultados de la pregunta 7

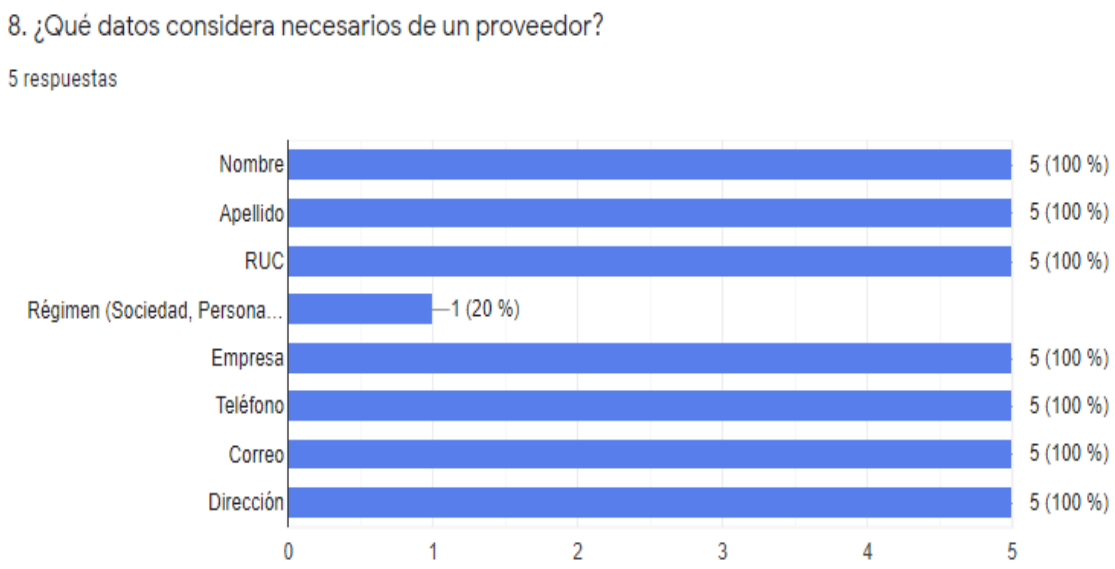


Figura 4.8 Resultados de la pregunta 8

La pregunta 9 está destinada a recabar información acerca de la importancia de anular un comprobante de compra/venta. Como se destalla en la Figura 4.9 la mayoría de los empleados considera necesaria esta característica en el prototipo.

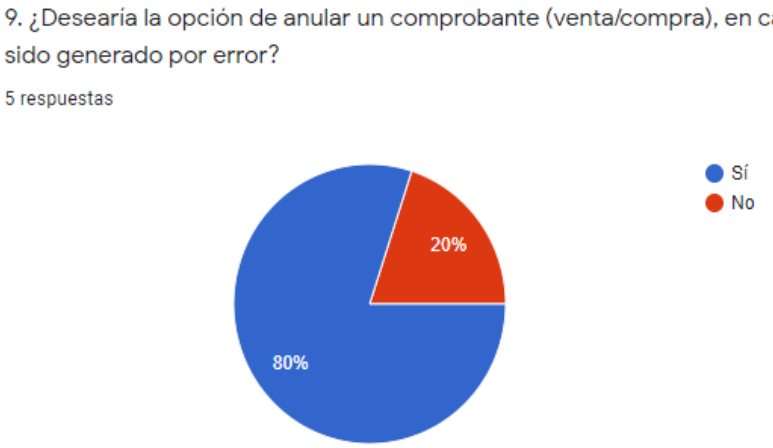


Figura 4.9 Resultados de la pregunta 9

La pregunta 10 tiene como finalidad conocer los intervalos de tiempo en los cuales se debe generar un reporte, según los resultados obtenidos en la Figura 4.10 los rangos idóneos son: diario, semanal y mensual, sin embargo, se ha considerado la posibilidad de que se seleccionen rangos personalizados.

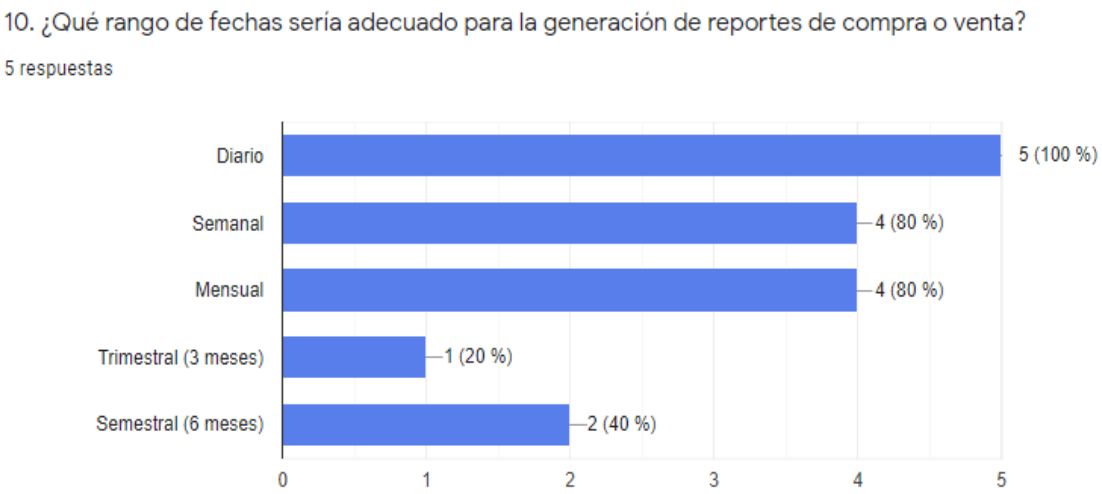


Figura 4.10 Resultados de la pregunta 10

La pregunta 11 refleja la necesidad de contar con un sistema de notificaciones ante la escasez de algún producto, como se puede ver en la Figura 4.11 la totalidad de los empleados indicaron que es una funcionalidad útil.

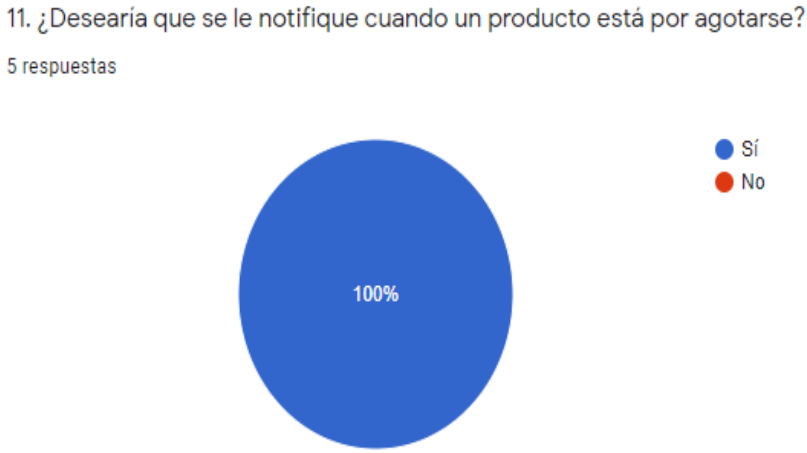


Figura 4.11 Resultados de la pregunta 11

4.1.2 HISTORIAS DE USUARIO

Tomando en cuenta las entrevistas y las encuestas realizadas al personal de la lavadora, se procedió a generar las historias de usuario correspondientes, las cuales se presentan en la Tabla 4.1 y en la Tabla 4.2.

Tabla 4.1 Historias de usuario parte 1

ID	Título	Descripción
HU_01	Autenticación de usuarios	Como usuario administrador y empleado, es necesario poder iniciar sesión con credenciales para ingresar al sistema y determinar sus funciones.
HU_02	Búsqueda de productos	Como usuario administrador y empleado, se necesita realizar una búsqueda los productos que se encuentran dentro del inventario para determinar su disponibilidad.
HU_03	Búsqueda de productos por vehículo	Como administrador y empleado, se necesita conocer que productos están disponibles para un vehículo en particular.
HU_04	Administración de productos	Como usuario administrador, es necesario el poder administrar (crear, editar, eliminar) los productos del inventario.
HU_05	Generación de comprobantes de venta	Como administrador y empleado, al realizar una venta es necesario generar un comprobante de venta que incluya los datos de los productos vendidos, así como los precios y los datos del cliente.
HU_06	Generación de comprobantes de compra	Como usuario administrador, al realizar una compra es necesario generar un comprobante de compra que incluya los datos de los productos adquiridos, así como los precios y los datos del proveedor.
HU_07	Administración de clientes	Como administrador y empleado, al realizar una venta es necesario registrar y administrar los datos del cliente.
HU_08	Administración de proveedores	Como usuario administrador, al realizar una compra es necesario registrar y administrar los datos del proveedor.

Tabla 4.2 Historias de usuario parte 2

ID	Título	Descripción
HU_09	Generación de reportes de venta	Como administrador y empleado, es necesario poder generar un reporte de venta que permita visualizar las ventas dentro de un determinado intervalo de tiempo.
HU_10	Administración de usuarios	Como usuario administrador es necesario poder realizar la administración de los usuarios del sistema
HU_11	Generación de reportes de compra	Como administrador y empleado, es necesario poder generar un reporte de compra que permita visualizar las compras dentro de un determinado intervalo de tiempo.
HU_12	Visualización de notificaciones	Como administrador y empleado, es necesario poder visualizar las notificaciones sobre los productos que están por debajo de la cantidad mínima.

4.1.3 CASOS DE USO

Conforme a lo detallado en las historias de usuario, se han identificado únicamente dos actores en el prototipo: usuario administrador y usuario empleado; para cada usuario se han asignado funcionalidades correspondientes a su necesidad y su rol, Por lo mencionado anteriormente se ha generado el correspondiente diagrama de casos de uso.

En la Figura 4.12 se aprecian las interacciones de los usuarios con el prototipo, las cuales se detallan a continuación:

Interacción con el usuario empleado:

- Iniciar sesión, que incluye la verificación de credenciales y la asignación de funcionalidades de acuerdo con el rol.
- Buscar producto, que incluye la búsqueda de aceites, filtros, otros y vehículos.
- Gestionar comprobantes de venta, que incluye la generación, búsqueda y anulación.
- Gestionar reportes de venta, que incluye la generación, visualización y exportación.

- Administrar clientes, que incluye el ingreso, edición y eliminación.
- Visualizar notificaciones.

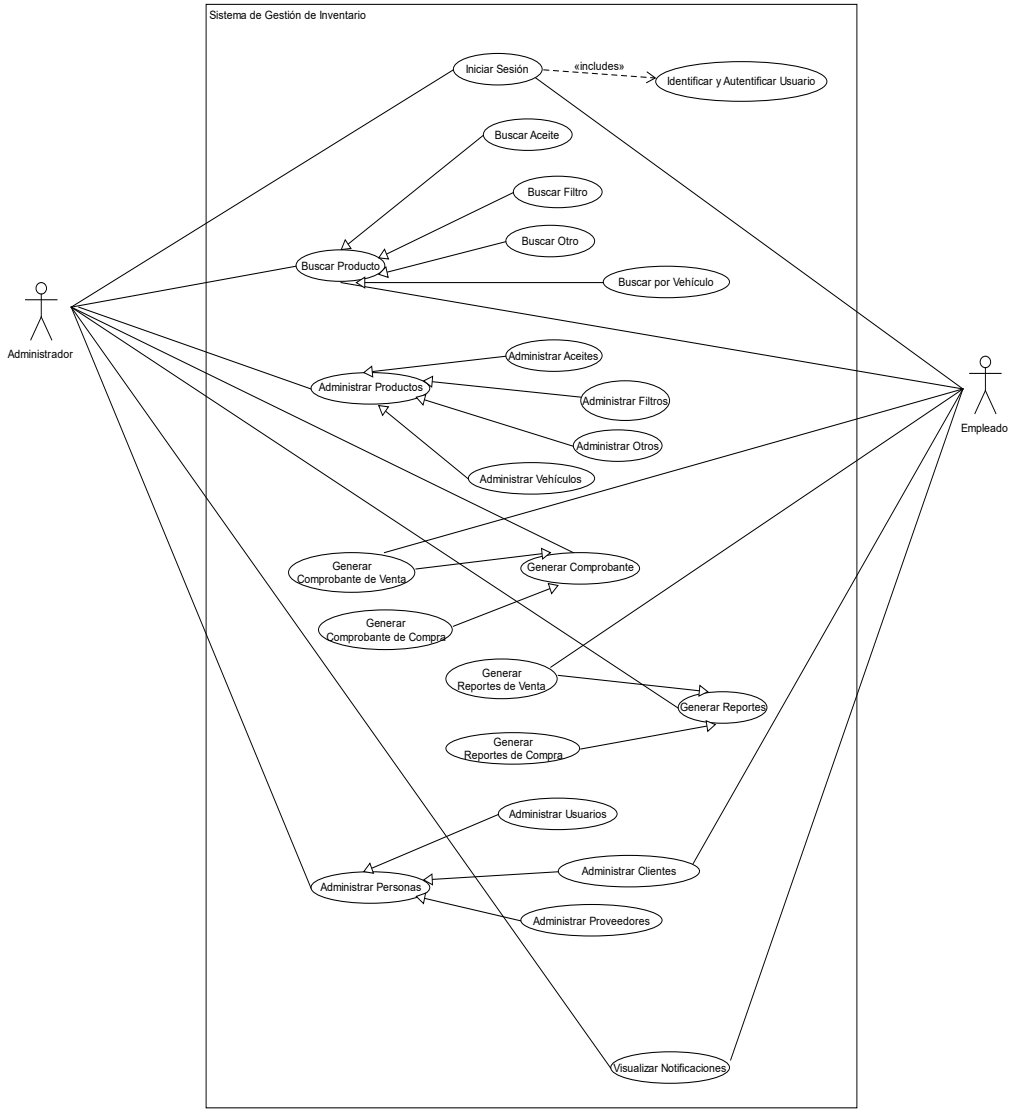


Figura 4.12 Diagrama de casos de uso

Interacción con el usuario administrador:

- Iniciar sesión, que incluye la verificación de credenciales y la asignación de funcionalidades de acuerdo con el rol.
- Buscar producto, que incluye la búsqueda de aceites, filtros, otros, y vehículos.
- Administrar productos, que incluye el CRUD de aceite, filtro, otros y vehículos.

- Gestionar comprobantes, que incluye la generación, búsqueda y anulación tanto de comprobantes de compra como de venta.
- Gestionar reportes, que incluye la generación, visualización y exportación anulación tanto de comprobantes de compra como de venta.
- Administrar personas, que incluye el ingreso, edición y eliminación de clientes, proveedores y usuarios.
- Visualizar notificaciones.

4.1.4 BACKLOG DE KANBAN

Las actividades necesarias para cumplir con los requerimientos de las historias de usuario fueron desglosadas en tareas específicas, las cuales se muestran en la Tabla 4.3.

Tabla 4.3 Tareas organizadas en el *backlog*

Tarea	
1	Iniciar de sesión mediante un correo y una contraseña.
2	Buscar productos mediante criterios de búsqueda.
3	Buscar productos para un vehículo en específico.
4	Ingresar, Editar, Visualizar y Eliminar productos.
5	Generar un comprobante de venta
6	Imprimir un comprobante de venta
7	Generar un comprobante de compra
8	Imprimir un comprobante de compra
9	Ingresar, Editar, Visualizar y Eliminar clientes.
10	Ingresar, Editar, Visualizar y Eliminar proveedores.
11	Generar reportes de venta diario, semanal y mensual
12	Exportar el reporte de venta
13	Generar reportes de compra diario, semanal y mensual
14	Imprimir el reporte de compra
15	Ingresar, Editar, Visualizar y Eliminar de usuarios.
16	Visualizar notificaciones.

4.2 DISEÑO

Una vez definidas las historias de usuario y organizadas las tareas que representan cada una de ellas, se procedió con la fase de diseño.

Nota: “La Lavadora y Lubricadora el Chino” durante el desarrollo del presente Trabajo de Titulación tuvo un cambio de nombre. Motivo por el cual la interfaz gráfica se muestra el nombre “Lavadora y Lubricadora Negritos”

4.2.1 ARQUITECTURA DEL PROTOTIPO

El prototipo se desarrollará mediante una arquitectura de cuatro niveles usando el modelo cliente – servidor, como se puede apreciar en la Figura 4.13.

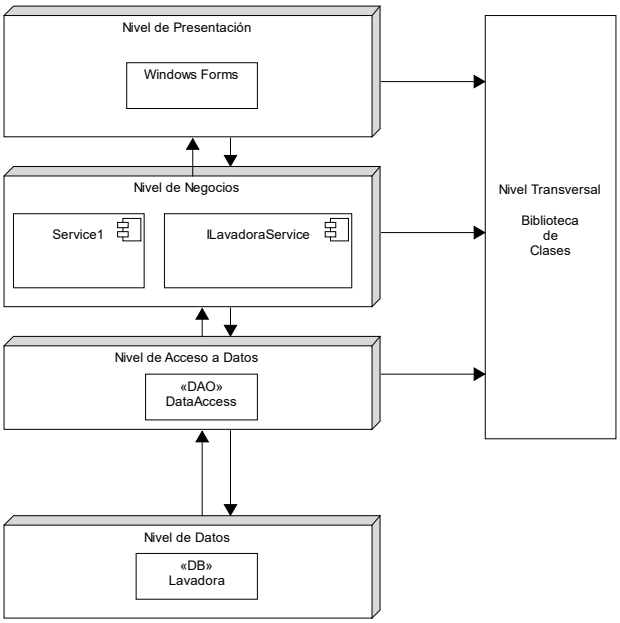


Figura 4.13 Arquitectura del prototipo

El prototipo cuenta con cuatro niveles principales y un nivel transversal:

- Nivel de datos: almacena la información.
- Nivel de acceso a datos: proporciona al servicio web el acceso al nivel de datos.
- Nivel de negocios: procesa la información recibida del cliente y gestiona su almacenamiento en la base de datos.
- Nivel de presentación: genera y visualiza la información.

- Nivel transversal: almacena datos temporales necesarios para el prototipo.

4.2.2 DISEÑO DE LA BASE DE DATOS

Para el desarrollo de la base de datos se generó un esquema relacional, en el que se pueden distinguir las entidades, atributos y sus respectivas relaciones. En el diagrama que se muestra en la Figura 4.14 se visualizan únicamente las entidades, relaciones, las llaves primarias y foráneas debido al limitado espacio disponible.

En el ANEXO B se encuentra el esquema relacional completo y los *scripts* para generar la base de datos se encuentra en el ANEXO C. Los *scripts* se encuentran divididos en cuatro partes: el primero contiene el código para la generación de tablas y relaciones, el segundo crea las vistas utilizadas en el prototipo, el tercero contiene todos los procedimientos almacenados y, por último, el cuarto contiene la información inicial para poblar las tablas base.

La tabla `Aceite` almacena la información necesaria para identificar a un aceite en específico y se relaciona con las tablas `TipoAceite`, `Presentacion`, `SAE`, `API` y `NotificacionAceite`, esta última permite generar una notificación automática cuando la cantidad de un producto está por debajo de su límite.

La tabla `Filtro` al igual que la tabla previa permite identificar a un filtro en específico y se relaciona con `TipoFiltro`, `CodigoFiltro`, `Vehículo` y `NotificacionFiltro` que sirve para generar la notificación automática.

La tabla `Producto` contiene información de los productos no contemplados en las tablas aceite y filtro; se relaciona con la tabla `NotificacionProducto` que sirve para generar la notificación automática.

La tabla `Vehículo` contiene la información requerida para identificar a un vehículo, y se relaciona con la tabla `Filtro` para de esta manera se logré identificar a que vehículo pertenece cada filtro.

La tabla `ComprobanteVenta` contiene información básica de una venta como la fecha, número de documento, total, subtotal, IVA y descuento; además incluye los productos adquiridos por lo que se relaciona con las tablas `Aceite`, `Filtro` y `Producto`, también gestiona información personal y de contacto del cliente y del empleado que realizó la venta, por lo que se relaciona con las tablas `Cliente` y `Usuario`.

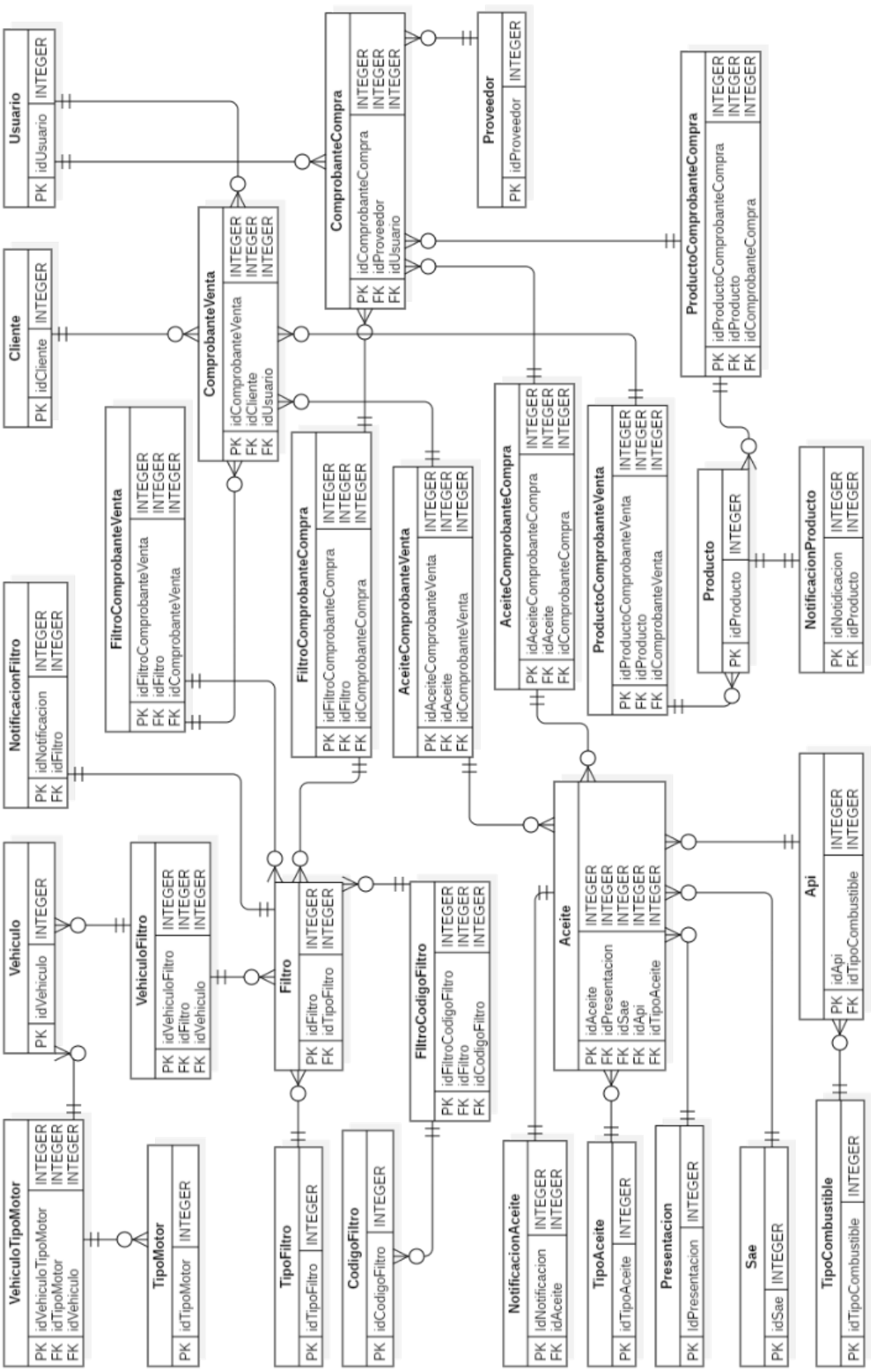


Figura 4.14 Esquema relacional de la base de datos

La tabla `ComprobanteCompra` contiene información básica de una compra como la fecha, número de documento, total, subtotal, IVA y descuento; además incluye los productos adquiridos por lo que también se relaciona con las tablas `Aceite`, `Filtro` y `Producto`; además administra la información personal y de contacto del proveedor y del administrador que realizó la compra, por lo que se relaciona con las tablas `Proveedor` y `Usuario`.

Las tablas de correlación `FiltroCodigoFiltro`, `VehículoFiltro`, `VehiculoTipoMotor`, `AceiteComprobanteCompra`, `FiltroComprobanteCompra`, `ProductoComprobanteCompra`, `AceiteComprobanteVenta`, `FiltroComprobanteVenta`, `ProductoComprobanteVenta` permiten romper las relaciones de "muchos-a-muchos".

4.2.3 DISEÑO DE CLASES

La Figura 4.15 representa las clases generadas en el servicio web que permiten atender las solicitudes que el cliente envía. Este diagrama se detalla en el ANEXO D.

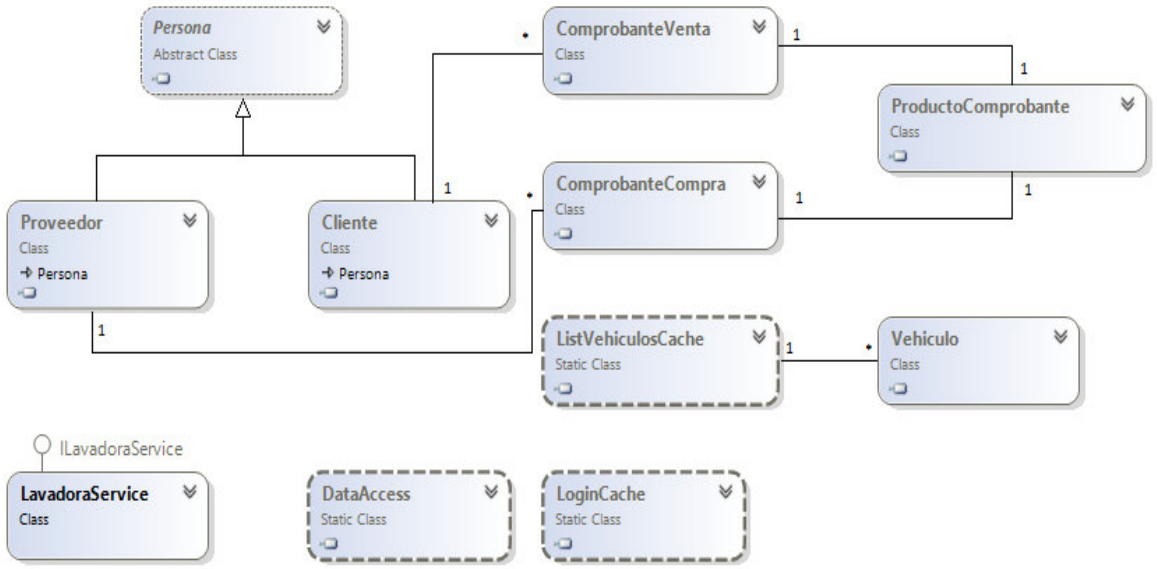


Figura 4.15 Diagrama de clases del servicio web

La clase `Persona` es abstracta y de ella heredan las clases `Cliente` y `Proveedor` utilizadas al momento de generar o recuperar un comprobante de venta, por lo cual se relacionan con las clases `ComprobanteVenta` y `ComprobanteCompra` respectivamente.

Además, la clase `ComprobanteCompra` se relaciona con la clase `ProductoComprobante` para gestionar los productos adquiridos.

La clase `ListaVehiculoCache` es una clase estática que se relaciona con la clase `Vehiculos`, permitiendo almacenar temporalmente una lista de estos. Adicionalmente la clase `LoginCache`, también estática, permite almacenar datos de inicio de sesión.

Para finalizar, la clase `LavadoraService` contiene los métodos necesarios para la comunicación entre el cliente y la clase `DataAccess`, la cual gestiona la información de la base de datos.

4.2.4 DIAGRAMAS DE ACTIVIDAD

En esta sección, se presentan las actividades que puede ejecutar un usuario administrador y un usuario empleado dentro del prototipo. Tanto para el usuario administrador como el usuario empleado, es necesario ingresar al sistema y verificar sus credenciales.

En la Figura 4.16 se puede observar que una vez verificada la identidad del usuario empleado este puede acceder a la búsqueda de productos, la misma que se puede hacer sobre productos específicos como aceites, filtros u otros, seleccionando el criterio de búsqueda que más se ajuste a sus necesidades.

El usuario también puede optar por generar un comprobante de venta, para lo cual escogerá los datos del cliente y los productos vendidos junto a su cantidad, adicionalmente podrá buscar comprobantes generados anteriormente a través de un identificador y de ser necesario podrá anularlos. Así mismo, el usuario empleado podrá optar por administrar los clientes registrados en el prototipo pudiendo agregarlos, editarlos y eliminarlos, de igual manera podrá generar un reporte de ventas de un rango especifico de fechas y finalmente podrá visualizar las notificaciones sobre productos que estén por debajo de la cantidad mínima.

El usuario administrador puede ejecutar las actividades del usuario empleado, además podrá optar por administrar vehículos y productos pudiendo ingresarlos, editarlos y eliminarlos, como se muestra en la Figura 4.17.

El usuario administrador puede gestionar los comprobantes de compra de forma similar al usuario empleado con los comprobantes de venta, finalmente el usuario administrador podrá optar por administrar usuarios (usuarios del sistema, clientes y proveedores).

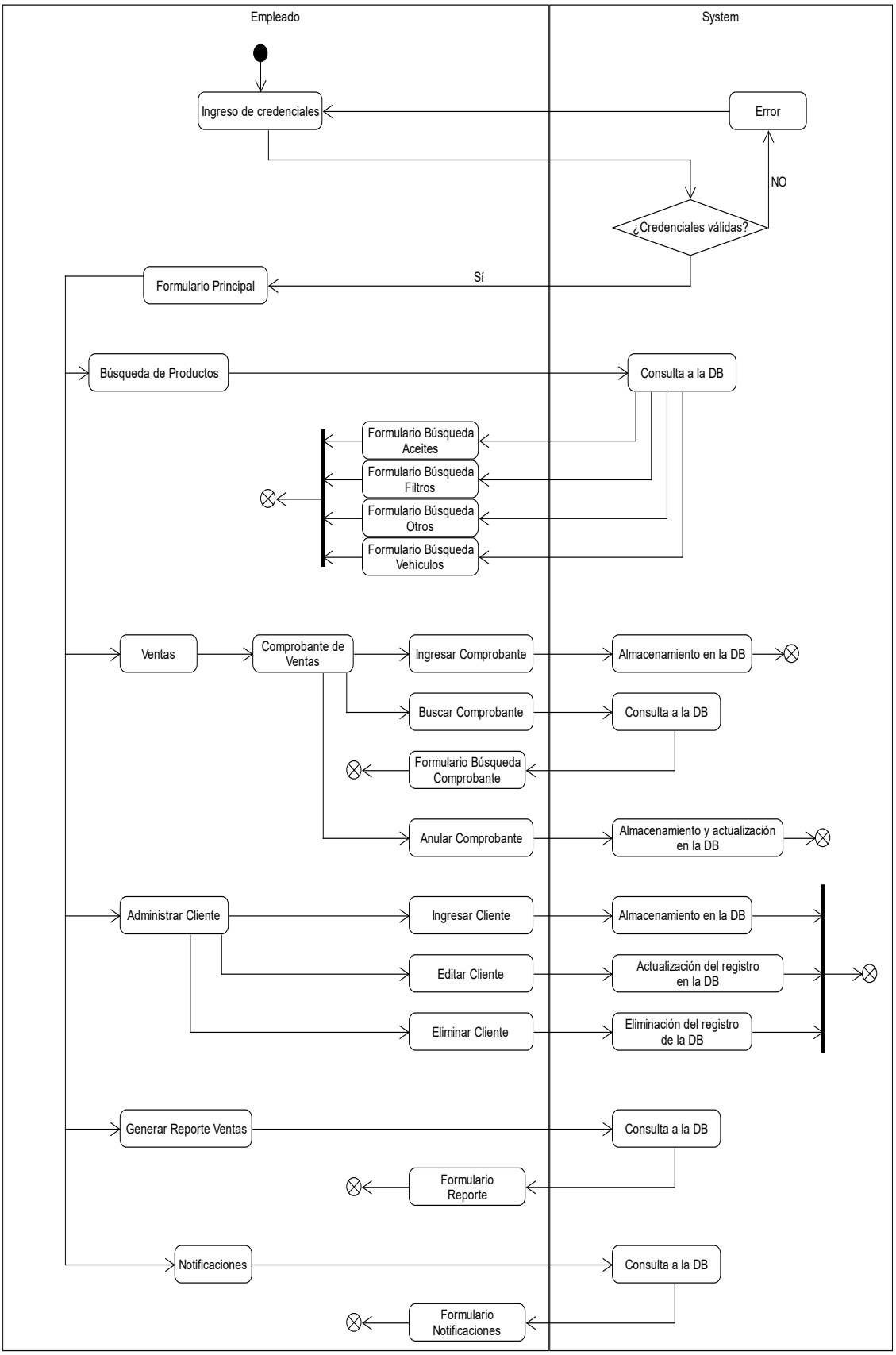


Figura 4.16 Diagrama de actividades del usuario empleado

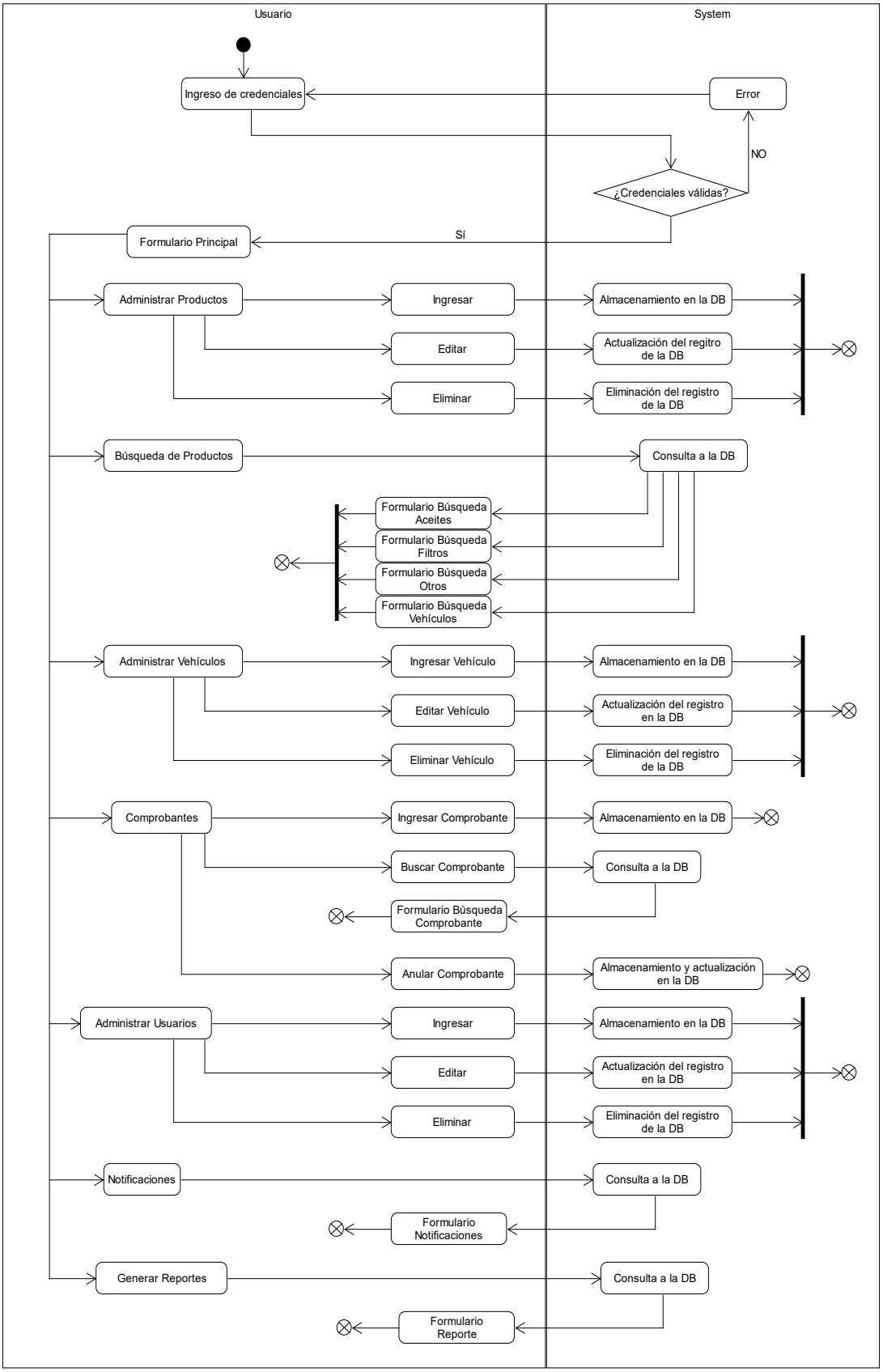


Figura 4.17 Diagrama de actividades del usuario administrador

Los usuarios pueden cerrar sesión en cualquier momento, además de ser requerido podrán realizar un cambio de usuario, como se visualiza en la Figura 4.18. Esto permite alternar entre el usuario administrador y el usuario empleado de manera ágil.

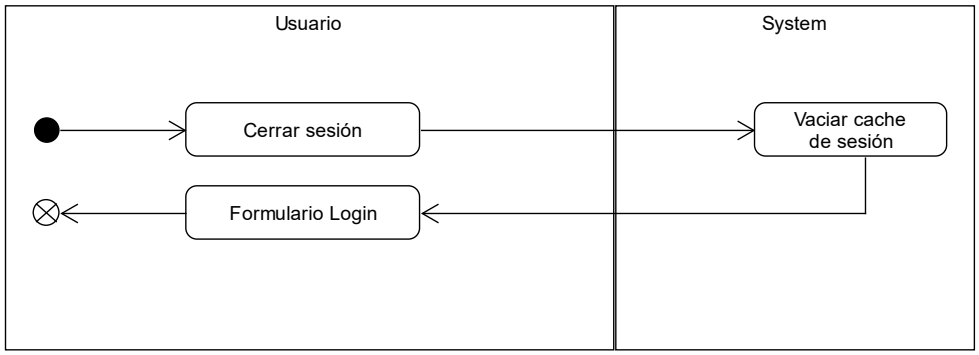


Figura 4.18 Diagrama de actividad de cierre de sesión

4.2.5 DIAGRAMAS DE SECUENCIA

Se elaboraron diagramas de secuencia para las siguientes operaciones: iniciar sesión, buscar de producto, ingresar producto, editar producto, eliminar producto, ingresar comprobante, buscar comprobante, anular comprobante, generar reporte, visualizar notificación.

En la Figura 4.19 se muestra la secuencia de acciones, los métodos y las interfaces que forman parte del proceso para ingresar un comprobante de venta dentro del prototipo.

Una vez que el usuario ingresa (empleado o administrador) al prototipo, se presentará la interfaz de usuario `GenerarComprobante`, la misma que solicita el ingreso del número de cedula del cliente con la que realiza una búsqueda en la tabla `Clientes`. Si el cliente existe, su información es desplegada en la interfaz de usuario y se solicita el ingreso del código de barras de los productos adquiridos, caso contrario se tiene que ingresar también los datos del cliente.

Al momento de ingresar un producto en la lista, como primer paso se procede a verificar en la tabla `Producto` su existencia, de ser el caso este se agrega al comprobante, en su defecto se envía una notificación al usuario; este proceso se repite cada vez que se deseé ingresar un producto a la lista. Para finalizar, se valida cada uno de los campos, adicionalmente si el cliente no existe en la base de datos, se procede a su ingreso conjuntamente con el comprobante, caso contrario, solo se realiza el ingreso del comprobante y se retorna un mensaje de éxito o error dependiendo el caso.

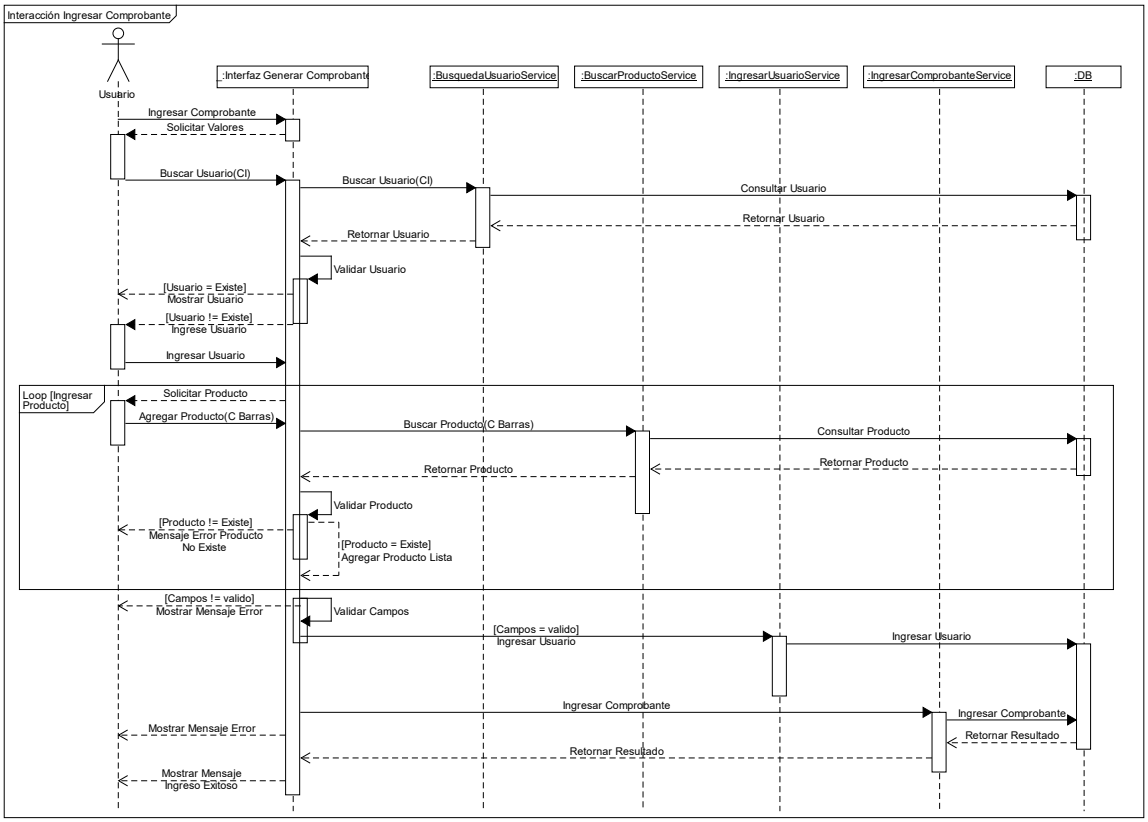


Figura 4.19 Diagrama de secuencia de ingresar un comprobante

Los diagramas de secuencia restantes mantienen la misma lógica del diagrama de la Figura 4.19, por lo que no serán explicados; sin embargo, se los ha incluido en el ANEXO E.

4.3 IMPLEMENTACIÓN

En esta sección se presentará un resumen de las tareas relacionadas con la implementación del prototipo.

4.3.1 CODIFICACIÓN DE LA BASE DE DATOS

Para la codificación de la base de datos se realizó un *script* basado en el esquema relacional, el cual se incluye en el ANEXO B.

En la Figura 4.20 se observa un fragmento del *script* que permite la creación de la tabla `ComprobanteVenta` haciendo uso de la sentencia `create table` (línea 161). En esta

tabla se registra la información necesaria al momento de realizar una venta, para ello cuenta con los siguientes campos: `numDocumento`, `subtotal`, `iva`, `total`, `estado` y `fechaVenta`. Los campos `idCliente` e `idUsuario` (líneas 163 y 164) se utilizan como llaves foráneas para la relación entre las tablas `Usuario` y `Cliente` con `ComprobanteVenta` respectivamente.

```
161 create table ComprobanteVenta (
162     idComprobanteVenta int PRIMARY KEY identity (1,1) not null,
163     idCliente int not null, --FK
164     idUsuario int not null, --FK
165     numDocumento varchar(20),
166     subtotal money not null,
167     iva money not null,
168     total money not null,
169     estado bit not null,
170     fechaVenta date not null
171 )
172 go
```

Figura 4.20 *Script* para la creación de tabla `ComprobanteCompra`

Con la finalidad de administrar las relaciones entre las tablas se utilizó la sentencia `add contraint` (línea 325), para ello fue necesario realizar cambios en la tabla mediante la sentencia `alter table` (línea 324), como se observa en la Figura 4.21.

```
324 alter table ComprobanteVenta
325 add constraint fk_ComprobanteVenta_Cliente
326 foreign key (idCliente) references Cliente(idCliente)
327 go
```

Figura 4.21 *Script* para la creación de la relación entre las tablas `ComprobanteVenta` y `Cliente`

Para poder visualizar datos de diferentes tablas en una misma consulta, se codificaron vistas de acuerdo a las necesidades del prototipo. Como se puede apreciar en la Figura 4.22, se realiza la creación de la vista `vw_VistaComprobantes` mediante la sentencia `create view` (línea 346), la cual contiene información de las tablas `ComprobanteVenta` y `Cliente`.

```
346 create view vw_VistaComprobantes
347 as
348 select idComprobanteVenta as ID, numDocumento, cv.estado, fechaVenta, c.nombre as Nombre, c.apellido as Apellido,
349 c.cedula as Cedula, c.telefono as Telefono, c.correo as Correo, c.direccion as Direccion,
350 subtotal, iva, total
351 from ComprobanteVenta cv
352 inner join Cliente c
353 on cv.idCliente = c.idCliente
354 go
```

Figura 4.22 *Script* para la creación de la vista `vw_VistaComprobantes`

Con el propósito de encapsular las sentencias SQL necesarias para realizar una tarea, se procedió a la creación de procedimientos almacenados. En la Figura 4.23 se observa que mediante el uso de la sentencia `create procedure` (línea 1168), se genera el procedimiento `sp_IngresarComprobanteVenta` y se logra agrupar las tareas necesarias para ingresar un comprobante, es decir buscar información del cliente, se almacena el comprobante de venta y finalmente se obtiene el id de comprobante ingresado para ser utilizado en el siguiente proceso.

```
1168 create procedure sp_IngresarComprobanteVenta
1169 @cedulaCliente varchar(20),
1170 @idUsuario int,
1171 @numDocumento varchar(20),
1172 @subtotal money,
1173 @iva money,
1174 @total money,
1175 @estado bit,
1176 @fechaVenta datetime
1177 as
1178 begin
1179 declare @idComprobante int
1180 declare @idCliente int
1181
1182 select @idCliente = idCliente from Cliente where cedula = @cedulaCliente
1183
1184 insert into ComprobanteVenta values
1185 (@idCliente, @idUsuario, @NumDocumento, @subtotal, @iva, @total,@estado, @fechaVenta)
1186 set @idComprobante = SCOPE_IDENTITY()
1187
1188 Select 'idComprobanteVenta' = @idComprobante
1189
1190 end
1191 go
```

Figura 4.23 *Script* para la creación del procedimiento almacenado `sp_IngresarComprobanteVenta`

4.3.2 INICIAR SESIÓN EN ELPROTOTIPO

Previo al ingreso al prototipo tanto el usuario empleado como el administrador debe iniciar sesión mediante la interfaz de *Login*. Como se logra apreciar en la Figura 4.24 la interfaz de *Login* es la misma para ambos.

Figura 4.24 Interfaz de *Login*

Para cada petición del cliente al servidor se crearon métodos con la etiqueta `OperationContract`, estos permiten conectar el nivel de presentación con el nivel de negocios en el lado del servicio web. En la Figura 4.25 se presenta la firma del método para inicio de sesión.

```
17 [OperationContract]
18 bool Login(string correo, string clave);
```

Figura 4.25 Declaración del método para inicio de sesión

Para cada contrato del prototipo se implementa un método que permita establecer una conexión entre el nivel de negocios y el nivel de acceso a datos (`DataAcces`). Para ello se implementa un método con la misma firma del método declarado como se aprecia en la Figura 4.26.

```
19        public bool Login(string correo, string clave)
20        {
21            return DataAccess.Login(correo, clave);
22        }
```

Figura 4.26 Implementación del método para inicio de sesión

La Figura 4.27 muestra el código desarrollado para realizar la validación de las credenciales (`correo` y `contraseña`) en la tabla `Usuarios` al momento de iniciar sesión. Para validar los campos, en primer lugar, se crea una conexión a la base de datos (línea 23), a continuación, se utiliza el procedimiento almacenado `sp_ValidarUsuario` (línea 26) el cual recibe como parámetros un correo y una clave, el procedimiento realiza la consulta y retorna los datos del usuario junto con un valor que indicará si el inicio de sesión fue exitoso o no; estos datos serán almacenados en la clase estática `LoginCache` para su posterior uso al momento de generar un comprobante.

```
19        public static bool Login(string correo, string clave)
20        {
21            //Conexion con la base de datos
22
23            using (SqlConnection con = new SqlConnection(conexion))
24            {
25                //Ejecucion de sentencia SQL
26                SqlCommand command = new SqlCommand("sp_ValidarUsuario", con);
27                con.Open();
28                command.CommandType = CommandType.StoredProcedure;
29                command.Parameters.AddWithValue("@correo", correo);
30                command.Parameters.AddWithValue("@clave", clave);
31                SqlDataReader reader = command.ExecuteReader();
32
33                //Lazo de recuperacion de objetos desde la base de datos
34                if (reader.HasRows)
35                {
36                    while (reader.Read())
37                    {
38                        LoginCache.idPersona = Convert.ToInt32(reader["idUsuario"]);
39                        LoginCache.nombre = reader["nombre"].ToString();
40                        LoginCache.apellido = reader["apellido"].ToString();
41                        LoginCache.correo = reader["correo"].ToString();
42                        LoginCache.rol = reader["rol"].ToString();
43                    }
44                    return true;
45                }
46                else
47                {
48                    return false;
49                }
50            }
51        }
```

Figura 4.27 Nivel de acceso a datos - *Login*

En la Figura 4.28 se visualiza la lógica del nivel de presentación. Como se puede apreciar en la línea 17, se procede a validar las credenciales del usuario mediante el método `Login` de la Figura 4.25, luego de lo cual, se procede a verificar el rol y se muestra la interfaz correspondiente (líneas 19 a 32).

```
if (cliente.Login(txtCorreo.Text, txtPassword.Text))
{
    if (cliente.LoginAdminitrador())
    {
        frmPrincipalP frmPrincipal = new frmPrincipalP();
        frmPrincipal.Show();
        frmPrincipal.FormClosed += Logout;
        this.Hide();
    }
    else
    {
        frmPrincipalU frmPrincipalU = new frmPrincipalU();
        frmPrincipalU.Show();
        frmPrincipalU.FormClosed += Logout;
        this.Hide();
    }
}
else
{
    lblMensajeError.Text = "       Usuario o Contraseña incorrecta \n       Intente de nuevo";
    lblMensajeError.Visible = true;

}
```

Figura 4.28 Nivel de presentación – Login

4.3.3 BUSCAR PRODUCTOS

Desde el panel de principal se puede acceder a los diferentes tipos de búsqueda como: aceites, filtros, otros y vehículos, cada uno con sus respectivos criterios de búsqueda. En la Figura 4.29 se muestra la interfaz de usuario correspondiente a la búsqueda de aceites, en la cual se puede realizar búsquedas usando como criterio el código de barras, la marca o la viscosidad. Adicionalmente, se define una opción para mostrar todos los aceites almacenados en la base de datos.

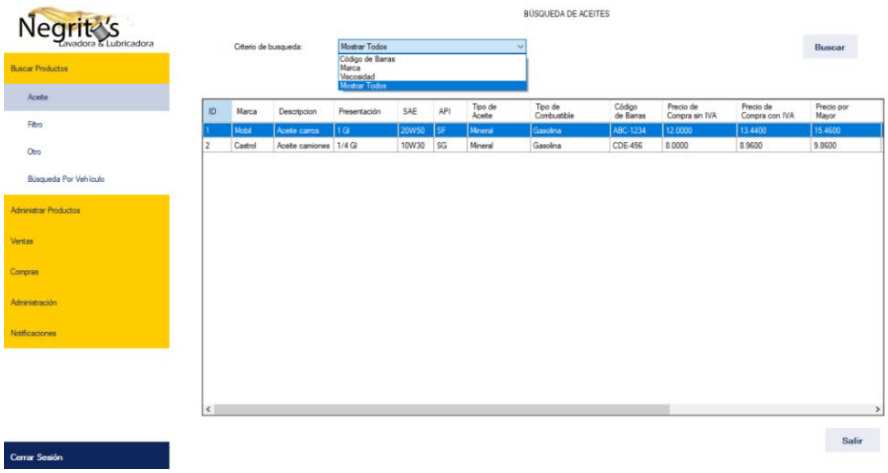


Figura 4.29 Interfaz de Búsqueda de Aceites

Previo a la implementación del método `BuscarAceiteMarca` en el `DataAccess`, se realiza la declaración e implementación del contrato. En la Figura 4.30 se observa que el

método tiene un parámetro de entrada `marca` de tipo `string` (línea 212), En el método se realiza la consulta mediante el proceso almacenado `sp_BuscarAceiteMarca` (línea 212) y retorna un objeto `DataTable` con los registros que coincidan con el valor de búsqueda (línea 227).

```
212 public static DataTable BuscarAceiteMarca(string marca)
213 {
214     DataTable dtAceites = new DataTable("Paging");
215 
216     //Conexion con la base de datos
217     using (SqlConnection con = new SqlConnection(conexion))
218     {
219         //Ejecucion de sentencia SQL
220         SqlCommand command = new SqlCommand("sp_BuscarAceiteMarca", con);
221         con.Open();
222         command.CommandType = CommandType.StoredProcedure;
223         command.Parameters.AddWithValue("@valor", marca);
224         SqlDataAdapter sqlData = new SqlDataAdapter(command);
225         sqlData.Fill(dtAceites);
226     }
227     return dtAceites;
228 }
```

Figura 4.30 Nivel de acceso a datos – `BuscarAceitesMarca`

En la Figura 4.31 se muestra el código que realiza la diferenciación entre los criterios de búsqueda existentes (líneas 39 a 59), dependiendo del criterio se ejecuta la correspondiente petición al servidor, el cual retorna un objeto del tipo `DataTable` que permite visualizar los registros en un control `DataGridView`.

```
37 if(cbxCriBusqueda.SelectedItem != null)
38 {
39     if (cbxCriBusqueda.SelectedItem.ToString().Equals("Código de Barras"))
40     {
41         DataTable aceites = cliente.BuscarAceiteCodigo(txtBusqueda.Text);
42         dgvAceites.DataSource = aceites;
43 
44     }
45     else if (cbxCriBusqueda.SelectedItem.ToString().Equals("Marca"))
46     {
47         DataTable aceites = cliente.BuscarAceiteMarca(txtBusqueda.Text);
48         dgvAceites.DataSource = aceites;
49     }
50     else if (cbxCriBusqueda.SelectedItem.ToString().Equals("Viscosidad"))
51     {
52         DataTable aceites = cliente.BuscarAceiteViscosidad(txtBusqueda.Text);
53         dgvAceites.DataSource = aceites;
54     }
55     else if (cbxCriBusqueda.SelectedItem.ToString().Equals("Mostrar Todos"))
56     {
57         DataTable aceites = cliente.ObtenerAceite();
58         dgvAceites.DataSource = aceites;
59     }
60 }
61 else
62 {
63     DialogResult dialogResult = MessageBox.Show("Seleccione un criterio de búsqueda", "Aviso", MessageBoxButtons.OK);
64 }
```

Figura 4.31 Nivel de presentación – `BuscarAceites`

4.3.4 ADMINISTRAR PRODUCTOS

Desde el panel principal se puede acceder a la administración de productos (aceite, filtro, otros y vehículos) y permite al usuario administrador en todos los casos ingresar, editar y eliminar un producto determinado.

En la Figura 4.32 se visualiza la interfaz de administración de aceites, específicamente el apartado de ingresar aceite, en la cual se observa las características generales de un aceite, así como los diferentes precios, algunos campos se calculan automáticamente en base al IVA o al porcentaje de venta.

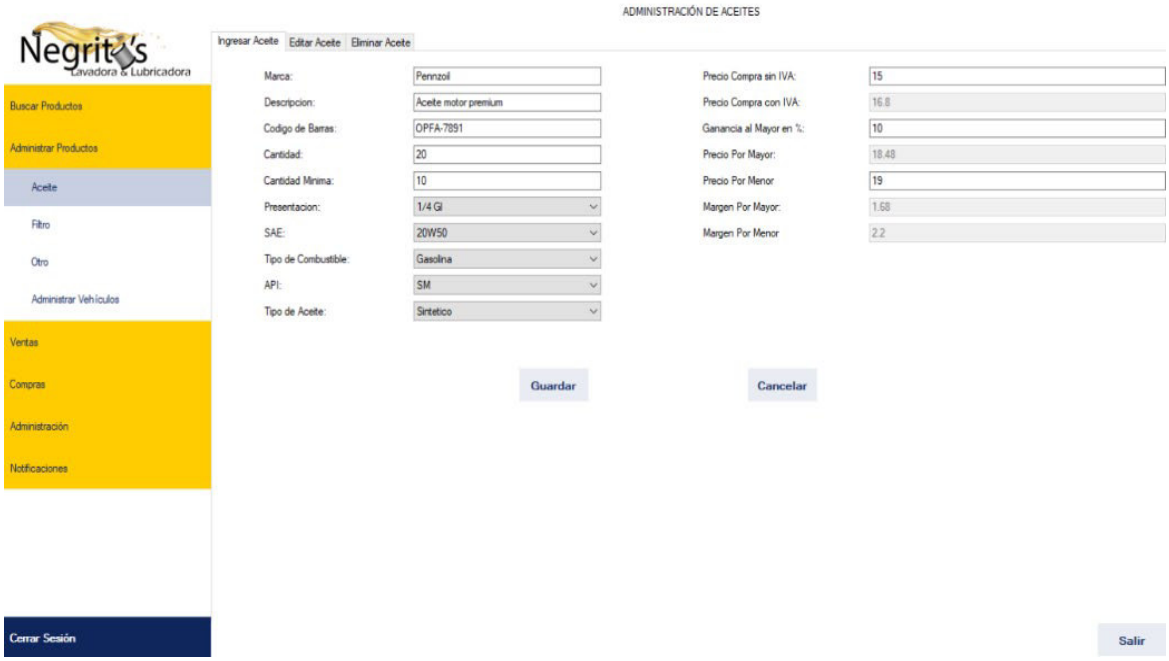


Figura 4.32 Interfaz de administración de aceites

En la Figura 4.33 se muestra el código correspondiente al método `IngresarAceite` (línea 55) del nivel de acceso a datos. Este método ejecuta el procedimiento almacenado `sp_IngresarAceite` (línea 66) que permite almacenar los datos ingresados por el usuario en la base de datos, adicionalmente este método retorna el número de líneas afectadas, siempre que este sea mayor a 1 se concluye que el registro se ha realizado exitosamente.

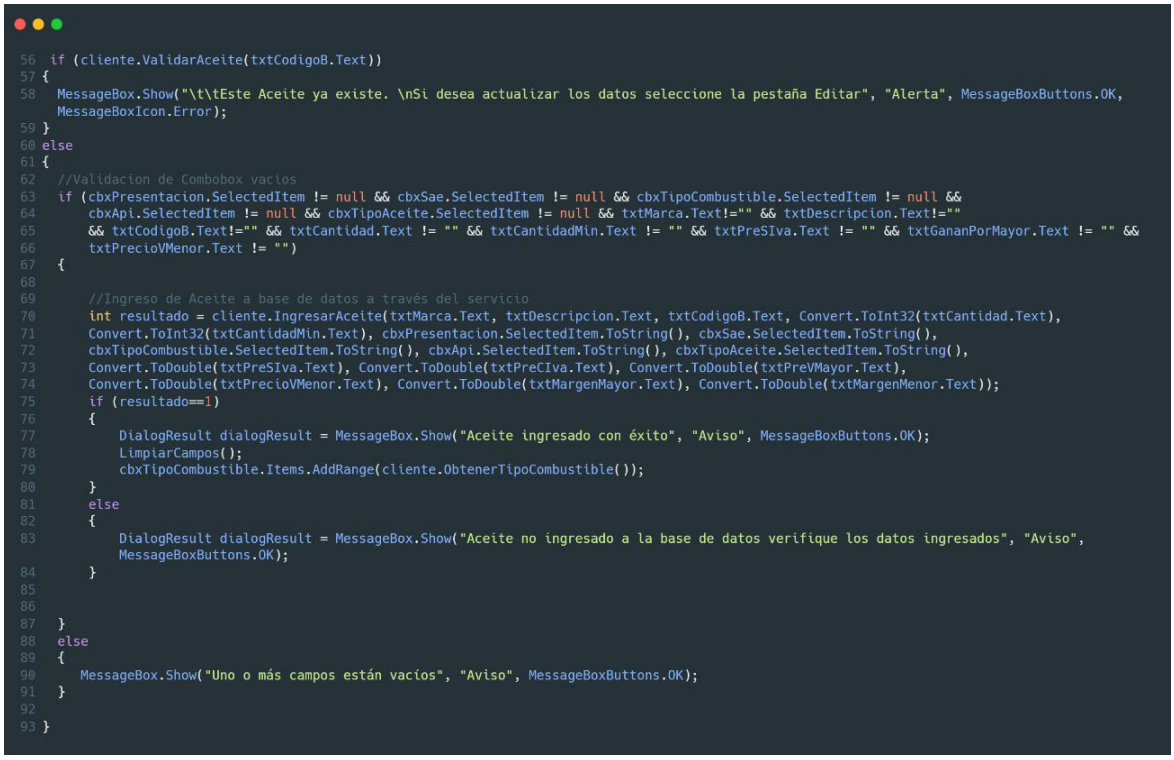
Con la finalidad de garantizar la disponibilidad del servicio y tener control sobre problemas o inconvenientes que puedan presentarse al realizar el ingreso, se ha añadido un bloque `try` (líneas 60 a 89) y `catch` (líneas 90 a 95), este indica de ser necesario que existe un error en el tipo de dato para que este se puede corregir.

```
55        public static int IngresarAceite(string marca, string descripcion, string codigoBarras, int cantidad,
56            int cantidadMin, string presentacion, string sae, string tipoCombustible, string api,
57            string tipoAceite, double precioSinIva, double precioConIva, double precioMayor, double precioMenor,
58            double margenMayor, double margenMenor)
59        {
60            try
61            {
62                //Conexion con la base de datos
63                using (SqlConnection con = new SqlConnection(conexion))
64                {
65                    //Ejecucion de sentencia SQL
66                    SqlCommand command = new SqlCommand("sp_IngresarAceite", con);
67                    con.Open();
68                    command.CommandType = CommandType.StoredProcedure;
69                    command.Parameters.AddWithValue("@presentacion", presentacion);
70                    command.Parameters.AddWithValue("@sae", sae);
71                    command.Parameters.AddWithValue("@api", api);
72                    command.Parameters.AddWithValue("@tipoAceite", tipoAceite);
73                    command.Parameters.AddWithValue("@tipoCombustible", tipoCombustible);
74                    command.Parameters.AddWithValue("@marca", marca);
75                    command.Parameters.AddWithValue("@descripcion", descripcion);
76                    command.Parameters.AddWithValue("@codigoBarras", codigoBarras);
77                    command.Parameters.AddWithValue("@precioCompraSinIva", precioSinIva);
78                    command.Parameters.AddWithValue("@precioCompraConIva", precioConIva);
79                    command.Parameters.AddWithValue("@precioVentaxMayor", precioMayor);
80                    command.Parameters.AddWithValue("@precioVentaxMenor", precioMenor);
81                    command.Parameters.AddWithValue("@margenxMayor", margenMayor);
82                    command.Parameters.AddWithValue("@margenxMenor", margenMenor);
83                    command.Parameters.AddWithValue("@cantidad", cantidad);
84                    command.Parameters.AddWithValue("@cantidadMinima", cantidadMin);
85                    int rowAfected = command.ExecuteNonQuery();
86
87                    return rowAfected;
88                }
89            }
90            catch (SqlException)
91            {
92                Console.WriteLine("Error en tipo de dato");
93                return 0;
94
95            }
96
97        }
```

Figura 4.33 Nivel de acceso a datos – IngresarAceite

En la Figura 4.34 se presenta el código correspondiente al nivel de presentación para el ingreso de un aceite, como se observa en la línea 56 la primera acción que se realiza es validar si el aceite ya se encuentra dentro del inventario, esto se hace en base al código de barras del producto.

De la línea 63 a la 66 se valida que cada uno de los campos estén llenos y no se presenten valores nulos; una vez finalizadas las validaciones se procede a ejecutar el método `IngresarAceite` (línea 70 a 74) que envía al servicio web los datos ingresados por el usuario, este retorna el número de filas afectadas, para validar el correcto ingreso del producto.



```
if (cliente.ValidarAceite(txtCodigoB.Text))
{
  MessageBox.Show("\t\tEste Aceite ya existe. \nSi desea actualizar los datos seleccione la pestaña Editar", "Alerta", MessageBoxButtons.OK,
  MessageBoxIcon.Error);
}
else
{
  //Validacion de Combobox vacios
  if (cbxPresentacion.SelectedItem != null && cbxSae.SelectedItem != null && cbxTipoCombustible.SelectedItem != null &&
      cbxApi.SelectedItem != null && cbxTipoAceite.SelectedItem != null && txtMarca.Text!="" && txtDescripcion.Text!=""
      && txtCodigoB.Text!="" && txtCantidad.Text != "" && txtCantidadMin.Text != "" && txtPreSIva.Text != "" && txtGananPorMayor.Text != "" &&
      txtPrecioVMenor.Text != "")
  {

      //Ingreso de Aceite a base de datos a través del servicio
      int resultado = cliente.IngresarAceite(txtMarca.Text, txtDescripcion.Text, txtCodigoB.Text, Convert.ToInt32(txtCantidad.Text),
      Convert.ToInt32(txtCantidadMin.Text), cbxPresentacion.SelectedItem.ToString(), cbxSae.SelectedItem.ToString(),
      cbxTipoCombustible.SelectedItem.ToString(), cbxApi.SelectedItem.ToString(), cbxTipoAceite.SelectedItem.ToString(),
      Convert.ToDouble(txtPreSIva.Text), Convert.ToDouble(txtPreCIva.Text), Convert.ToDouble(txtPreVMayor.Text),
      Convert.ToDouble(txtPrecioVMenor.Text), Convert.ToDouble(txtMargenMayor.Text), Convert.ToDouble(txtMargenMenor.Text));
      if (resultado==1)
      {
          DialogResult dialogResult = MessageBox.Show("Aceite ingresado con éxito", "Aviso", MessageBoxButtons.OK);
          LimpiarCampos();
          cbxTipoCombustible.Items.AddRange(cliente.ObtenerTipoCombustible());
      }
      else
      {
          DialogResult dialogResult = MessageBox.Show("Aceite no ingresado a la base de datos verifique los datos ingresados", "Aviso",
          MessageBoxButtons.OK);
      }


  }
  else
  {
     MessageBox.Show("Uno o más campos están vacíos", "Aviso", MessageBoxButtons.OK);
  }

}
```

Figura 4.34 Nivel de presentación – `IngresarAceite`

4.3.5 ADMINISTRAR COMPROBANTES

En la Figura 4.35 se aprecia la interfaz de administración de comprobantes de venta a la cual se puede acceder desde el panel principal, aquí se presentan los siguientes campos:

- Nº Documento: hace referencia a un documento físico (factura o nota de venta) de la PYME.
- Datos del cliente: pueden ser llenados automáticamente a partir de una consulta ingresando el número de cedula o manualmente si el cliente no se encuentra registrado.
- Cuadro de búsqueda de productos: permite agregar productos a la lista realizando una búsqueda a partir del código de barras y la cantidad de producto disponible.
- Lista de productos: panel donde se visualiza una descripción del producto junto con la cantidad y los precios.
- Cálculo de precios: panel donde se visualiza el subtotal, IVA, total y permite añadir un descuento.

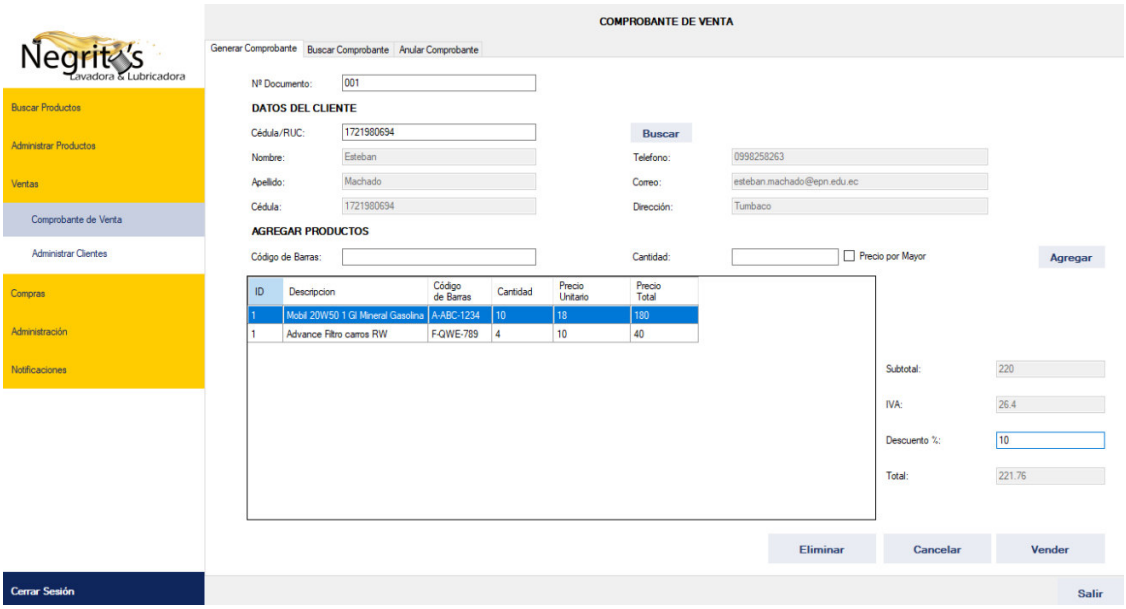


Figura 4.35 Interfaz de administración de comprobantes de venta

En la Figura 4.36 se presenta el método `IngresarComprobanteVenta` (línea 2013) del nivel de acceso a datos, como se puede apreciar el método acepta únicamente los parámetros propios de la tabla `ComprobanteVenta`, estos se ingresan a la base de datos mediante el procedimiento almacenado `sp_IngresarComprobanteVenta` (línea 2023).

Este método retorna un `idComprobanteVenta` (línea 2047) atributo que será utilizado en los métodos `IngresarAceiteComprobanteVenta`, `IngresarFiltroComprobanteVenta` e `IngresarProductoComprobanteVenta`.

En la Figura 4.37 se presenta el método `IngresarAceiteComprobanteVenta` (línea 2073) el cual acepta como parámetros el identificador del aceite, el identificador del comprobante y la cantidad a ser adquirida. A continuación, se ejecuta el procedimiento almacenado `sp_IngresarAceiteComprobanteVenta`, el cual se encarga de relacionar la lista de aceites adquiridos con su respectivo comprobante y realiza las operaciones necesarias para actualizar el inventario del producto.

Por último, en la Figura 4.38 se observa el código correspondiente al nivel de presentación, el cual envía los datos ingresados por el usuario por medio de los métodos antes mencionados. Como primer paso en la línea 2013 se envía al servicio web la información relacionada con el comprobante de venta, después se recorre la lista de productos mediante una sentencia `foreach` (líneas 2015 a 2041), donde se dividen los productos por su tipo (aceite, filtro, otro) para ejecutar el método correspondiente.

```
2013 public static int IngresarComprobanteVenta(string cedulaCliente, string numDocumento, double subtotal, double iva,
     double total, int estado, DateTime fechaVenta)
2014 {
2015     try
2016     {
2017         int idComprobante = 0;
2018 
2019         //Conexion con la base de datos
2020         using (SqlConnection con = new SqlConnection(conexion))
2021         {
2022             //Ejecucion de sentencia SQL
2023             SqlCommand command = new SqlCommand("sp_IngresarComprobanteVenta", con);
2024             con.Open();
2025             command.CommandType = CommandType.StoredProcedure;
2026             command.Parameters.AddWithValue("@cedulaCliente", cedulaCliente);
2027             command.Parameters.AddWithValue("@idUsuario", LoginCache.idPersona);
2028             command.Parameters.AddWithValue("@numDocumento", numDocumento);
2029             command.Parameters.AddWithValue("@subtotal", subtotal);
2030             command.Parameters.AddWithValue("@iva", iva);
2031             command.Parameters.AddWithValue("@total", total);
2032             command.Parameters.AddWithValue("@estado", estado);
2033             command.Parameters.AddWithValue("@fechaVenta", fechaVenta);
2034             SqlDataReader reader = command.ExecuteReader();
2035 
2036             //Lazo de recuperacion de objetos desde la base de datos
2037             if (reader.HasRows)
2038             {
2039                 while (reader.Read())
2040                 {
2041 
2042                     idComprobante = Convert.ToInt32(reader["idComprobanteVenta"]);
2043                 }
2044             }
2045         }
2046 
2047         return idComprobante;
2048     }
2049     catch (SqlException)
2050     {
2051         Console.WriteLine("Error en tipo de dato");
2052         return 0;
2053     }
2054 
2055 }
```

Figura 4.36 Nivel de acceso a datos – `IngresarComprobanteVenta`

```
2073 public static void IngresarAceiteComprobanteVenta(int idAceite, int idComprobanteVenta, int cantidad)
2074 {
2075     //Conexion con la base de datos
2076     using (SqlConnection con = new SqlConnection(conexion))
2077     {
2078         //Ejecucion de sentencia SQL
2079         SqlCommand command = new SqlCommand("sp_IngresarAceiteComprobanteVenta", con);
2080         con.Open();
2081         command.CommandType = CommandType.StoredProcedure;
2082         command.Parameters.AddWithValue("@idAceite", idAceite);
2083         command.Parameters.AddWithValue("@idComprobanteVenta", idComprobanteVenta);
2084         command.Parameters.AddWithValue("@cantidad", cantidad);
2085         command.ExecuteNonQuery();
2086     }
2087 }
```

Figura 4.37 Nivel de acceso a datos – `IngresarAceiteComprobanteVenta`

```
idComprobante = cliente.IngresarComprobanteVenta(txtCedulaBuscar.Text, txtNFactura.Text, Convert.ToDouble(txtSubtotal.Text),
Convert.ToDouble(txtIva.Text), Convert.ToDouble(txtTotal.Text), 0, DateTime.Now);

foreach (DataGridViewRow row in dgvProductosI.Rows)
{
    if ((row.Cells["dataGridViewTextBoxColumn26"].Value.ToString()).Contains("A-"))
    {
        cliente.IngresarAceiteComprobanteVenta(Convert.ToInt32(row.Cells[0].Value.ToString()), idComprobante,
Convert.ToInt32(row.Cells[3].Value.ToString()));
        if (cliente.ValidarMinAceite(Convert.ToInt32(row.Cells[0].Value.ToString())))
        {
            MessageBox.Show("Producto: " + row.Cells[1].Value.ToString() + " está próximo a agotarse", "Aviso", MessageBoxButtons.OK);
        }
    }
    if ((row.Cells["dataGridViewTextBoxColumn26"].Value.ToString()).Contains("F-"))
    {
        cliente.IngresarFiltroComprobanteVenta(Convert.ToInt32(row.Cells[0].Value.ToString()), idComprobante,
Convert.ToInt32(row.Cells[3].Value.ToString()));
        if (cliente.ValidarMinFiltro(Convert.ToInt32(row.Cells[0].Value.ToString())))
        {
            MessageBox.Show("Producto: " + row.Cells[1].Value.ToString() + " está próximo a agotarse", "Aviso", MessageBoxButtons.OK);
        }
    }
    if ((row.Cells["dataGridViewTextBoxColumn26"].Value.ToString()).Contains("P-"))
    {
        cliente.IngresarProductoComprobanteVenta(Convert.ToInt32(row.Cells[0].Value.ToString()), idComprobante,
Convert.ToInt32(row.Cells[3].Value.ToString()));
        if (cliente.ValidarMinProducto(Convert.ToInt32(row.Cells[0].Value.ToString())))
        {
            MessageBox.Show("Producto: " + row.Cells[1].Value.ToString() + " está próximo a agotarse", "Aviso", MessageBoxButtons.OK);
        }
    }
}
```

Figura 4.38 Nivel de presentación – `IngresarComprobante`

4.3.6 ADMINISTRAR USUARIOS

La interfaz de administración de usuarios puede ser accedida desde el panel principal por un usuario administrador, como se visualiza en la Figura 4.39. La interfaz cuenta con los campos necesarios para registrar un usuario, definir su rol (administrador o empleado) y gestionar su contraseña.

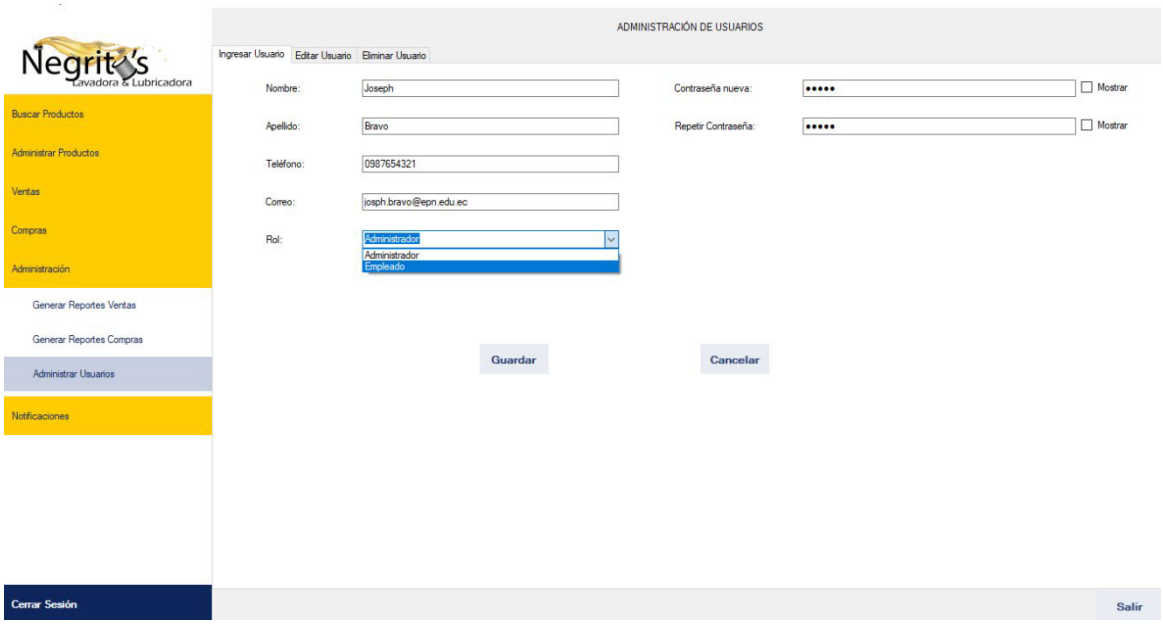


Figura 4.39 Interfaz de administración de usuarios

Tal como se muestra Figura 4.40 el método `IngresarUsuario` (línea 1716) del nivel de acceso a datos recibe los parámetros `nombre`, `apellido`, `telefono`, `correo`, `rol` y `clave`, y mediante el uso del procedimiento almacenado `sp_IngresarUsuario` (línea 1722) los registra en la base de datos.

```
1716 public static void IngresarUsuario(string nombre, string apellido, string telefono, string correo, string rol, string clave)
1717 {
1718   //Conexion con la base de datos
1719   using (SqlConnection con = new SqlConnection(conexion))
1720   {
1721       //Ejecucion de sentencia SQL
1722       SqlCommand command = new SqlCommand("sp_IngresarUsuario", con);
1723       con.Open();
1724       command.CommandType = CommandType.StoredProcedure;
1725       command.Parameters.AddWithValue("@nombre", nombre);
1726       command.Parameters.AddWithValue("@apellido", apellido);
1727       command.Parameters.AddWithValue("@telefono", telefono);
1728       command.Parameters.AddWithValue("@correo", correo);
1729       command.Parameters.AddWithValue("@rol", rol);
1730       command.Parameters.AddWithValue("@clave", clave);
1731       command.ExecuteNonQuery();
1732
1733   }
1734 }
```

Figura 4.40 Nivel de acceso a datos – `IngresarUsuario`

En la Figura 4.41 se verifica la coincidencia de los campos `txtClaveNueva` y `txtCRepetir` con la finalidad de asegurar que la contraseña ingresada en ambos campos es igual. Una vez verificadas las contraseñas, se procede a ejecutar la petición de ingreso del usuario (línea 65).

```
63 if (txtCRepetir.Text.Equals(txtClaveNueva.Text))
64 {
65   cliente.IngresarUsuario(txtNombre.Text, txtApellido.Text, txtTelefono.Text, txtCorreo.Text, cbxRol.SelectedItem.ToString(), txtCRepetir.Text);
66   DialogResult dialogResult = MessageBox.Show("Usuario ingresado con éxito", "Aviso", MessageBoxButtons.OK);
67   LimpiarCampos();
68 }
69 else
70 {
71   DialogResult dialogResult = MessageBox.Show("Las contraseñas no coinciden", "Aviso", MessageBoxButtons.OK);
72   txtClaveNueva.Clear();
73   txtCRepetir.Clear();
74 }
```

Figura 4.41 Nivel de presentación – `IngresarUsuario`

4.3.7 GENERAR REPORTES

Desde el panel principal se puede acceder a la interfaz de generación de reportes, en la Figura 4.42 se muestra el reporte generado y los diferentes intervalos de tiempo que se pueden utilizar.

Adicionalmente, se cuenta con dos controles `DateTimePicker` mediante los cuales el usuario podrá seleccionar el rango de fechas a consultar de acuerdo con su necesidad. Este reporte tal y como se visualiza en el panel puede ser exportado en formato PDF, Exel y Word.

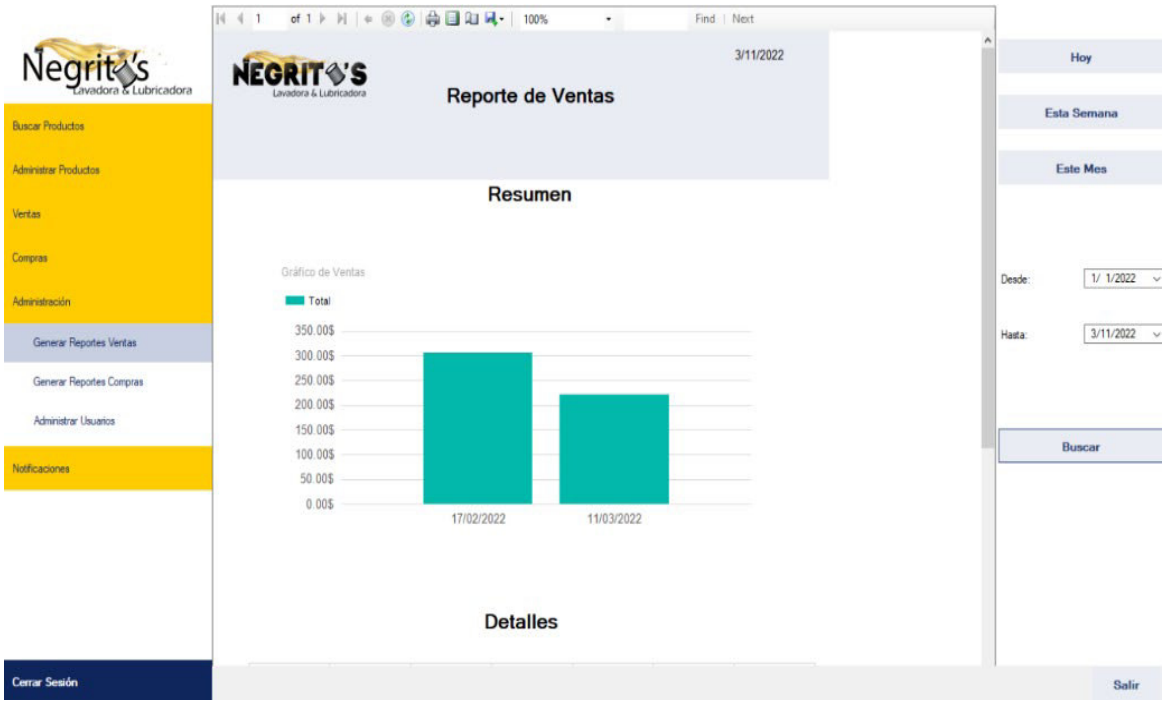


Figura 4.42 Interfaz para gestionar reportes

Para poder realizar los reportes, es necesario realizar una consulta a la tabla `ComprobanteVenta` o `ComprobanteCompra` de la base de datos, para lo cual será necesario construir la consulta indicando el rango de fechas en las que se requiere generar el reporte.

En la Figura 4.43 en la línea 2216 se observa el método `BuscarComprobanteRangoFecha` el cual obtiene todos los atributos de un comprobante a partir del procedimiento almacenado `sp_BuscarComprobanteRangoFecha` (línea 2225) y retorna una lista de comprobantes que se encuentran dentro de las fechas definidas (línea 2257).

Para que se visualicen los datos en el reporte, se utiliza el componente `ComprobanteVentaBlindingSourece`, como se muestra en línea 36 de la Figura 4.44 el cual recibe como datos de entrada la lista de reportes generados en las fechas definidas por el usuario.

```
2216 public static List<ComprobanteVenta> BuscarComprobanteRangoFecha(string fechaInicio, string fechaFin)
2217 {
2218     List<ComprobanteVenta> comprobantesV = new List<ComprobanteVenta>();
2219
2220
2221     //Conexion con la base de datos
2222     using (SqlConnection con = new SqlConnection(conexion))
2223     {
2224         //Ejecucion de sentencia SQL
2225         SqlCommand command = new SqlCommand("sp_BuscarComprobanteRangoFecha", con);
2226         con.Open();
2227         command.CommandType = CommandType.StoredProcedure;
2228         command.Parameters.AddWithValue("@fechaInicio", fechaInicio);
2229         command.Parameters.AddWithValue("@fechaFin", fechaFin);
2230         SqlDataReader reader = command.ExecuteReader();
2231
2232         //Lazo de recuperacion de objetos desde la base de datos
2233         if (reader.HasRows)
2234         {
2235             while (reader.Read())
2236             {
2237                 ComprobanteVenta comprobante = new ComprobanteVenta();
2238                 comprobante.ID = Convert.ToInt32(reader["ID"]);
2239                 comprobante.NumDocumento = reader["numDocumento"].ToString();
2240                 comprobante.Estado = Convert.ToInt32(reader["estado"]);
2241                 comprobante.FechaVenta = Convert.ToDateTime(reader["fechaVenta"].ToString());
2242                 comprobante.Nombre = reader["Nombre"].ToString();
2243                 comprobante.Apellido = reader["Apellido"].ToString();
2244                 comprobante.Cedula = reader["Cedula"].ToString();
2245                 comprobante.Telefono = reader["Telefono"].ToString();
2246                 comprobante.Correo = reader["Correo"].ToString();
2247                 comprobante.Direccion = reader["Direccion"].ToString();
2248                 comprobante.Subtotal = Convert.ToDouble(reader["subtotal"]);
2249                 comprobante.Iva = Convert.ToDouble(reader["iva"]);
2250                 comprobante.Total = Convert.ToDouble(reader["total"]);
2251                 comprobantesV.Add(comprobante);
2252             }
2253
2254         }
2255
2256     }
2257     return comprobantesV;
2258 }
```

Figura 4.43 Nivel de acceso a datos – `BuscarComprobanteRangoFecha`

```
32 DateTime fechaInicio = dtpFechaInicio.Value;
33 DateTime fechaFin = dtpFechaFin.Value;
34
35
36 ComprobanteVentaBindingSource.DataSource =
   cliente.BuscarComprobanteRangoFecha(fechaInicio.ToString("yyyy'-'MM'-'dd"),
   fechaFin.ToString("yyyy'-'MM'-'dd"));
37 this.rpvVentas.RefreshReport();xtCRepetir.Clear();
```

Figura 4.44 Nivel de presentación - `GenerarReporte`

4.3.8 VISUALIZAR NOTIFICACIONES

La Figura 4.45 presenta la interfaz de notificaciones, a la cual el usuario puede acceder desde el menú principal, esta cuenta con un panel donde se visualiza una descripción del

producto en escasez, el stock actual, el stock mínimo y la fecha en la que se generó la notificación.

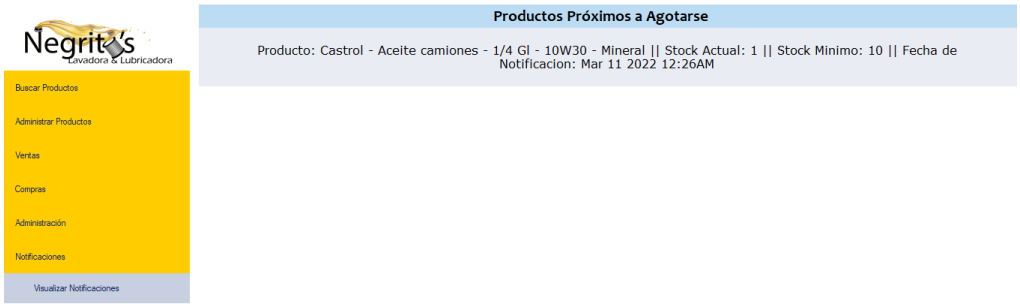


Figura 4.45 Interfaz de notificaciones

En la Figura 4.46 se presente el código del método `ValidarMinAceite`, el cual recibe de la interfaz de usuario el ID del producto. A continuación, realiza una consulta a la base de datos obteniendo los valores de cantidad mínima y cantidad actual (líneas 2730 a la 2749), a partir de esto determina si el producto está en escasez (líneas 2751 a la 2761) y genera una notificación mediante el método `NotificacionAceite` presentado en la Figura 4.47.

```
2723  public static bool ValidarMinAceite(int id)
2724          {
2725              int cantidadActual = 0;
2726              int cantidadMin = 0;
2727
2728
2729              //Conexion con la base de datos
2730              using (SqlConnection con = new SqlConnection(conexion))
2731              {
2732                  //Ejecucion de sentencia SQL
2733                  SqlCommand command = new SqlCommand("sp_ValidarMinAceite", con);
2734                  con.Open();
2735                  command.CommandType = CommandType.StoredProcedure;
2736                  command.Parameters.AddWithValue("@idAceite", id);
2737                  SqlDataReader reader = command.ExecuteReader();
2738
2739                  //Lazo de recuperacion de objetos desde la base de datos
2740                  if (reader.HasRows)
2741                  {
2742                      while (reader.Read())
2743                      {
2744
2745                          cantidadActual = Convert.ToInt32(reader["cantidad"].ToString());
2746                          cantidadMin = Convert.ToInt32(reader["cantidadMinima"].ToString());
2747                      }
2748                  }
2749              }
2750
2751              if (cantidadActual <= cantidadMin)
2752              {
2753                  NotificacionAceite(id, cantidadActual, cantidadMin, DateTime.Now);
2754
2755                  return true;
2756              }
2757              else
2758              {
2759                  EliminarNotificacionAceite(id);
2760                  return false;
2761              }
2762
2763          }
2764
2765  end
2766  go
```

Figura 4.46 Nivel de acceso a datos – `ValidarMinAceite`

```
2596 public static void NotificacionAceite(int idAceite, int cantidadActual, int cantidadMinima, DateTime fechaNotificacion)
2597 {
2598     //Conexion con la base de datos
2599     using (SqlConnection con = new SqlConnection(conexion))
2600     {
2601         //Ejecucion de sentencia SQL
2602         SqlCommand command = new SqlCommand("sp_NotificacionAceite", con);
2603         con.Open();
2604         command.CommandType = CommandType.StoredProcedure;
2605         command.Parameters.AddWithValue("@idAceite", idAceite);
2606         command.Parameters.AddWithValue("@cantidadActual", cantidadActual);
2607         command.Parameters.AddWithValue("@cantidadMinima", cantidadMinima);
2608         command.Parameters.AddWithValue("@fechaNotificacion", fechaNotificacion);
2609
2610         command.ExecuteNonQuery();
2611
2612     }
2613 }
```

Figura 4.47 Nivel de acceso a datos – `NotificacionAceite`

Finalmente, se obtienen todas las notificaciones de productos generadas a partir del método `ObtenerNotificaciones` como se puede ver en la Figura 4.48 (línea 29). Se llama a este método cada vez que se despliega la interfaz de notificaciones.

```
29 DataTable notificaciones = cliente.ObtenerNotificaciones();
30 dgvNotificaciones.DataSource = notificaciones;
```

Figura 4.48 Nivel de presentación – Notificaciones

5. RESULTADOS Y DISCUSIÓN

En este capítulo se presentan los resultados de las pruebas de funcionamiento y aceptación.

Como parte de las pruebas de funcionamiento, se realizó el ingreso de información en cada una de las interfaces para verificar su correcto funcionamiento, así también se validó que la información sea almacenada correctamente en la base de datos y que no existan campos vacíos cuando estos sean obligatorios. Además, se corroboró que se puedan generar comprobantes, que la cantidad de productos se sume o reste del stock actual y que se pueda imprimir dicho comprobante.

Adicionalmente, se realizó una encuesta de aceptación la cual fue aplicada a 5 empleados de la PYME para validar el correcto funcionamiento del prototipo.

Al final, se procesó las respuestas obtenidas y se realizaron cambios y mejoras al prototipo tales como: el cambio del logo de la interfaz principal y se mejoró el aspecto del botón cerrar sesión.

5.1 PRUEBAS DE FUNCIONAMIENTO

5.1.1 INICIAR SESIÓN

En la interfaz de *Login* se agregaron controles visuales que guían al usuario en el proceso de autenticación. Como se observa en la Figura 5.1, se muestra una guía en los cuadros de ingreso de credenciales, donde se muestra el dato que el usuario debe ingresar ya sea el usuario/correo o la contraseña; cuando uno de estos campos está vacío, se muestra un mensaje de advertencia.

Figura 5.1 Validación de campos en la interfaz de *Login*

En el caso de que el usuario ingrese credenciales no validas, se muestra un mensaje de error, como se aprecia en la Figura 5.2. De esta manera se le informa al usuario que su proceso de autenticación ha fallado.

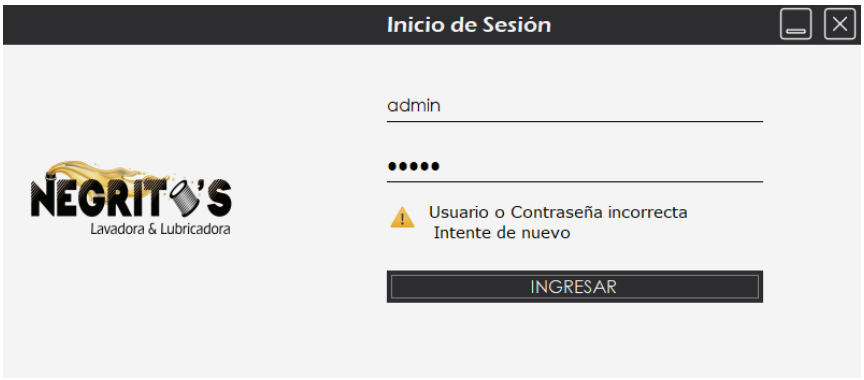


Figura 5.2 Inicio de sesión fallido

Si el usuario ingresa sus credenciales de manera correcta el prototipo presentará el menú principal, el mismo que dispondrá de diferentes opciones dependiendo de su rol. En la Figura 5.3 se presenta la interfaz principal para el usuario administrador.

Figura 5.3 Interfaz principal del usuario administrador

5.1.2 BUSCAR PRODUCTOS

Para corroborar el funcionamiento, se realizó una búsqueda por código de barras con el valor ABC, obteniendo como resultado el aceite que se muestra en la lista. Si la búsqueda no encuentra coincidencias, no se visualizará ningún valor en la lista. En la Figura 5.4 se

presenta el resultado de la prueba, como se aprecia en la interfaz se presentan los detalles del aceite encontrado.

Figura 5.4 Interfaz de búsqueda de aceites

5.1.3 BUSCAR PRODUCTOS POR VEHÍCULO

Para probar este interfaz de búsqueda se seleccionó el vehículo “marca: Chevrolet, modelo: Corsa, año: 2000 – 2006, motor: 1.6”, dando como resultado para este vehículo el producto que se observa en la lista de la Figura 5.5. En el caso de que uno o más campos se encuentren vacíos el prototipo presenta un mensaje de error.

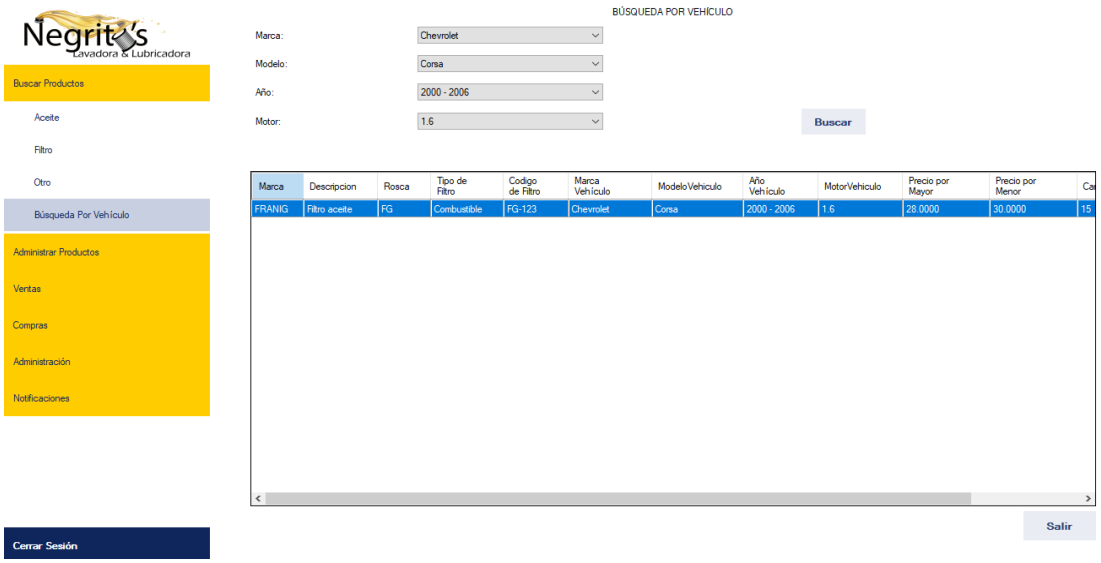


Figura 5.5 Interfaz de búsqueda por vehículo

5.1.4 INGRESAR PRODUCTO

Para probar el ingreso de un aceite es necesario llenar todos los campos del formulario, como se puede apreciar en la Figura 5.6. Si la operación tuvo éxito se muestra un mensaje que informa el éxito del proceso como se puede ver en la Figura 5.7. Finalmente, en la Figura 5.8 se muestra el registro ingresado en la base de datos.

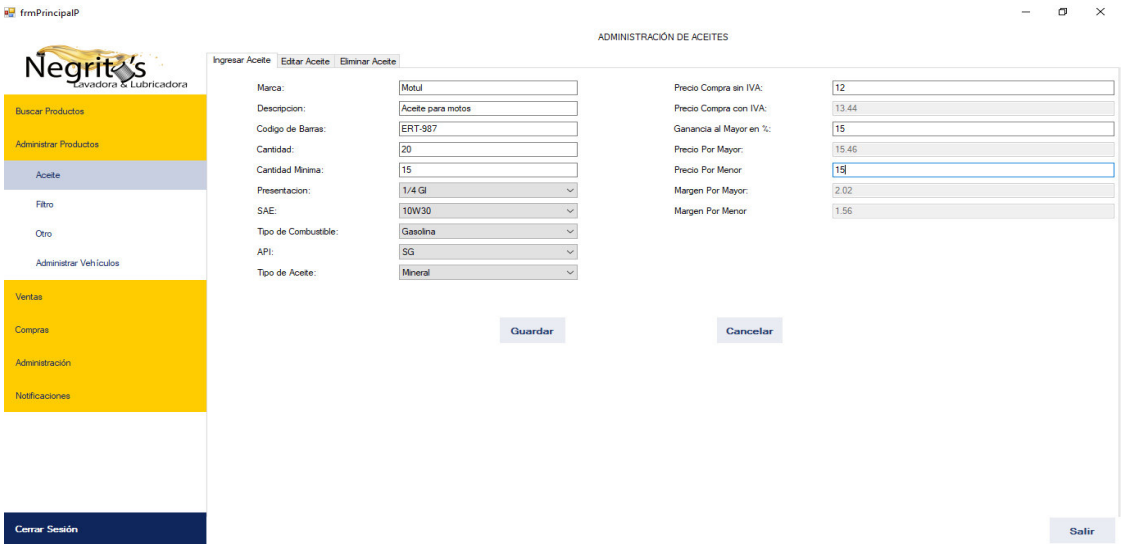


Figura 5.6 Interfaz de ingresar aceite

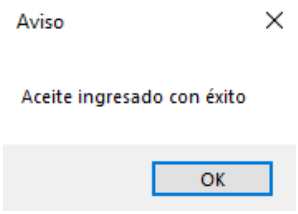
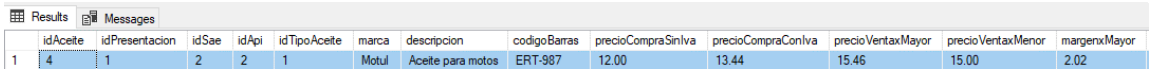


Figura 5.7 Mensaje informativo al ingresar un aceite



	idAceite	idPresentacion	idSae	idApi	idTipoAceite	marca	descripcion	codigoBarras	precioCompraSinIva	precioCompraConIva	precioVentaxMayor	precioVentaxMenor	margenxMayor
1	4	1	2	2	1	Motul	Aceite para motos	ERT-987	12.00	13.44	15.46	15.00	2.02

Figura 5.8 Aceite ingresado a la base de datos

5.1.5 INGRESAR VEHÍCULOS

Para realizar el ingreso de un vehículo, es necesario dirigirse a la interfaz de administrar vehículos, la cual se presenta en la Figura 5.9. Se ingresó un vehículo de una marca no

ingresada al prototipo, por lo cual es necesario ingresar todos los campos de forma manual. A continuación, al momento de presionar el botón `Guardar` se despliega un mensaje que informa el éxito del proceso, como se observa en la Figura 5.10. Para finalizar la prueba, se muestra el registro ingresado correctamente en la Figura 5.11.

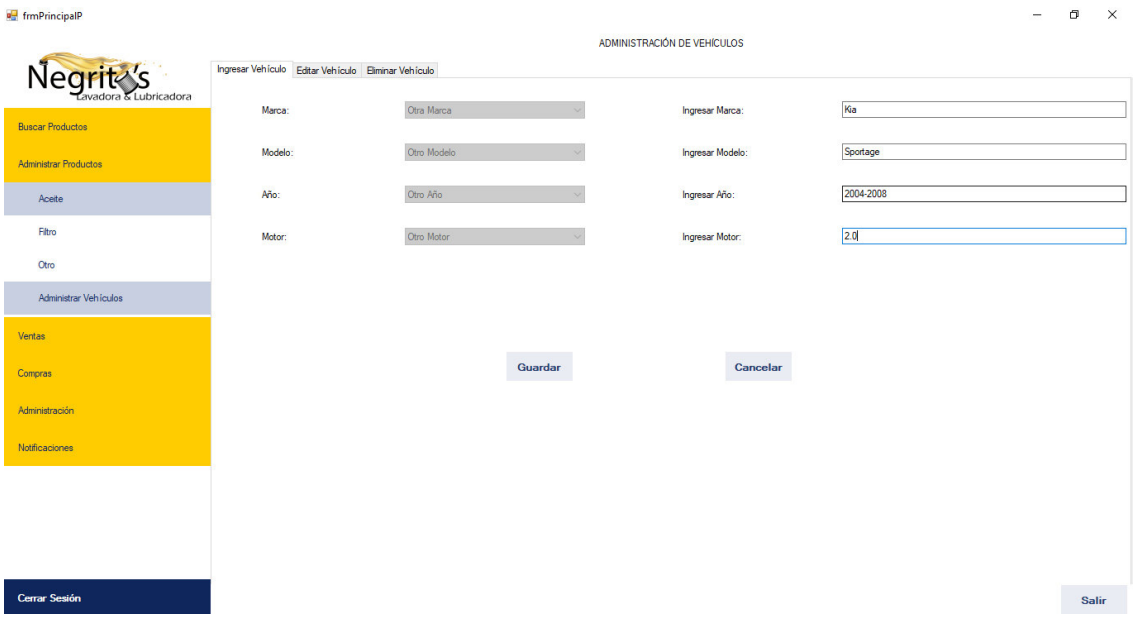


Figura 5.9 Interfaz de ingreso de vehículos

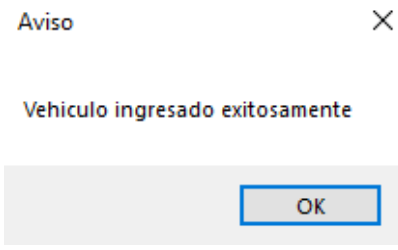
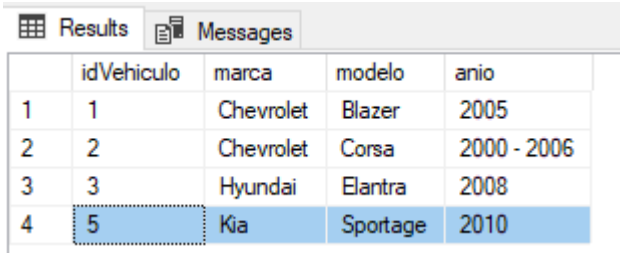


Figura 5.10 Mensaje informativo al ingresar un vehículo



	idVehiculo	marca	modelo	anio
1	1	Chevrolet	Blazer	2005
2	2	Chevrolet	Corsa	2000 - 2006
3	3	Hyundai	Elantra	2008
4	5	Kia	Sportage	2010

Figura 5.11 Vehículo ingresado a la base de datos

5.1.6 GENERAR COMPROBANTE DE VENTA

En la Figura 5.12 se observa la interfaz para generar un comprobante de venta, en esta se corroboró que es posible llenar la información de un cliente existente a partir de su número de cédula. Adicionalmente, se puede observar el ingreso de productos y el cálculo automático de los valores: subtotal, IVA y total.

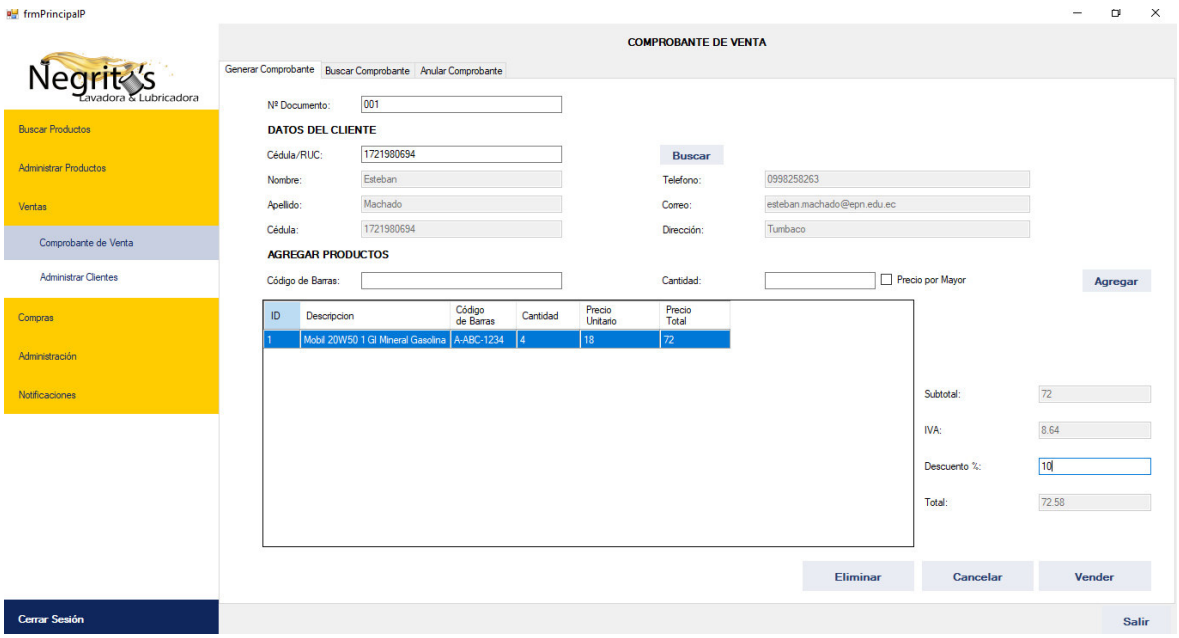


Figura 5.12 Interfaz de ingreso de comprobante de venta

Existen varias validaciones al momento de ingresar un comprobante. Entre las cuales se encuentran: validación de campos, registro previo del cliente, existencia del producto con la cantidad requerida. En la Figura 5.13 se visualiza el mensaje de advertencia que muestra el prototipo cuando la cantidad ingresada supera el *stock* disponible.

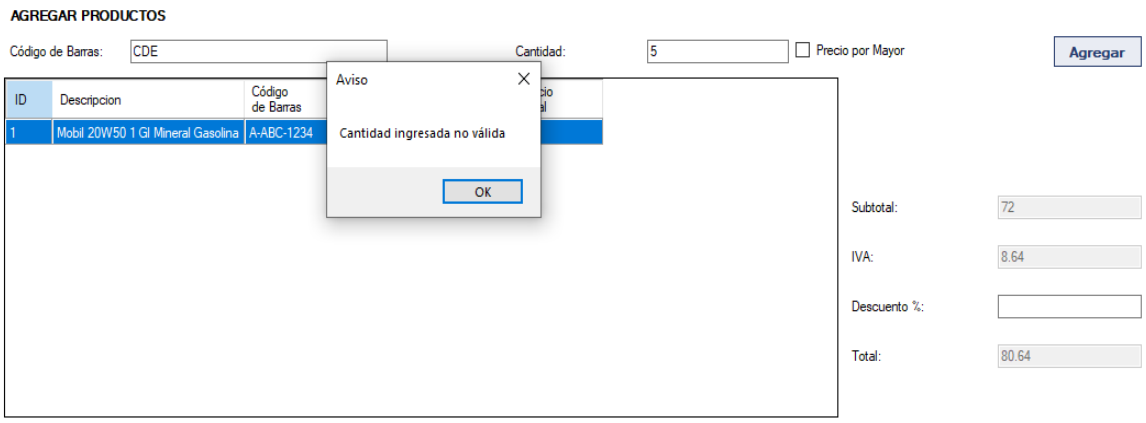
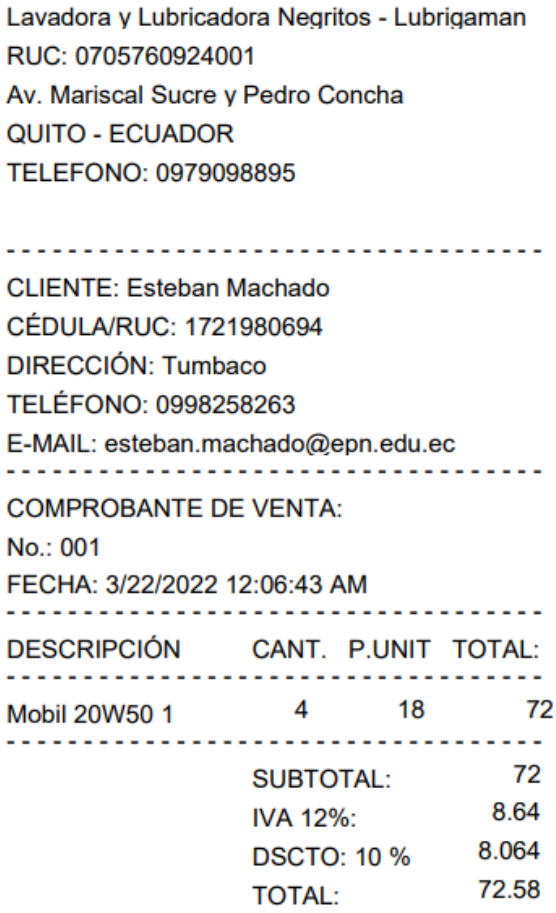


Figura 5.13 Comprobación de cantidad disponible de un producto

Una vez ingresados los datos del cliente y los productos solicitados en el comprobante de venta y posteriormente almacenados en la base de datos, se genera un comprobante imprimible, en el cual se incluyen todos los datos correspondientes a la venta y la información de contacto de la PYME, como se puede apreciar en la Figura 5.14.


Lavadora y Lubricadora Negritos - Lubrigaman
RUC: 0705760924001
Av. Mariscal Sucre y Pedro Concha
QUITO - ECUADOR
TELEFONO: 0979098895

CLIENTE: Esteban Machado
CÉDULA/RUC: 1721980694
DIRECCIÓN: Tumbaco
TELÉFONO: 0998258263
E-MAIL: esteban.machado@epn.edu.ec

COMPROBANTE DE VENTA:
No.: 001
FECHA: 3/22/2022 12:06:43 AM

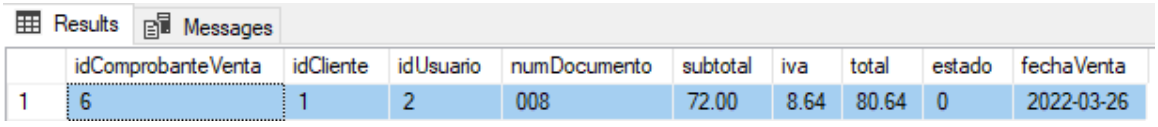
DESCRIPCIÓN	CANT.	P.UNIT	TOTAL:
Mobil 20W50 1	4	18	72

SUBTOTAL:	72
IVA 12%:	8.64
DSCTO: 10 %	8.064
TOTAL:	72.58

Figura 5.14 Comprobante de compra imprimible

5.1.7 ANULAR COMPROBANTE DE VENTA

En la Figura 5.15 se muestra el comprobante de venta a ser anulado, así mismo, se puede apreciar que el `estado` se encuentra con un valor de 0, lo que indica que no ha sido anulado previamente.


Results | Messages

	idComprobanteVenta	idCliente	idUsuario	numDocumento	subtotal	iva	total	estado	fechaVenta
1	6	1	2	008	72.00	8.64	80.64	0	2022-03-26

Figura 5.15 Estado del comprobante previo a la anulación

Para la anulación de un comprobante de venta, se dispone de la interfaz de usuario que se muestra en la Figura 5.16. Para realizar la anulación es necesario buscar el comprobante mediante el `ID`. Para esta prueba, se tomó como referencia el comprobante de venta con `ID = 6`.

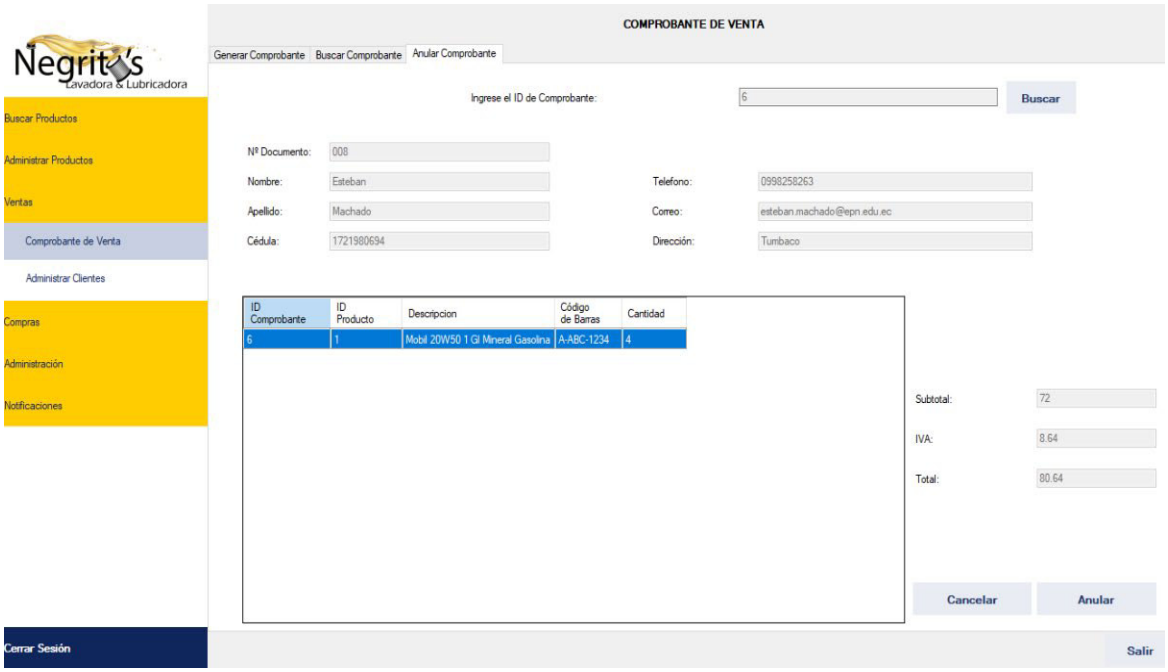


Figura 5.16 Interfaz de anulación de comprobante de venta

Una vez realizada la anulación, se puede observar en la Figura 5.17 que el comprobante ha cambiado de estado, y se ha generado un nuevo comprobante con valores negativos para que no exista incongruencias al momento de generar un reporte.

Results Messages

	idComprobanteVenta	idCliente	idUsuario	numDocumento	subtotal	iva	total	estado	fechaVenta
1	6	1	2	008	72.00	8.64	80.64	1	2022-03-26
2	7	1	2	008	-72.00	-8.64	-80.64	1	2022-03-26

Figura 5.17 Estado del comprobante posterior a la anulación

5.1.8 INGRESAR CLIENTES

La Figura 5.18 muestra la interfaz de ingreso de clientes, en esta se ingresa la información relevante del cliente. En los campos cédula y el teléfono se corrobora que solo se ingresen números, como se puede observar en la Figura 5.19

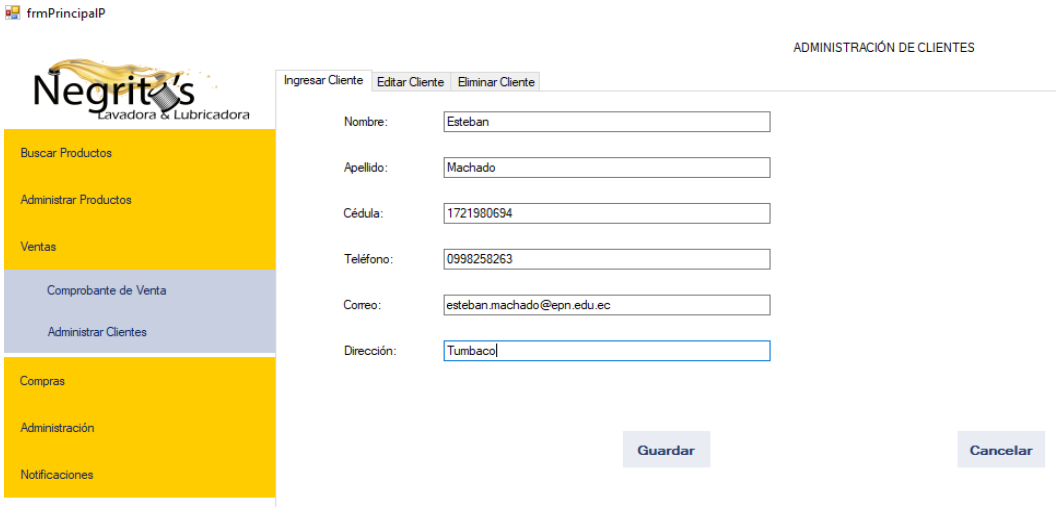


Figura 5.18 Interfaz de ingreso de clientes

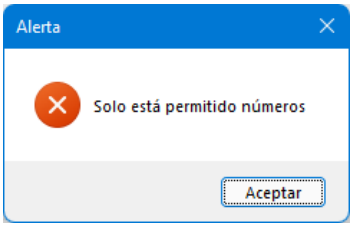


Figura 5.19 Validación de campos numéricos

Una vez ingresados correctamente los datos del cliente, se procede a guardar los datos en la base. Para corroborar que el ingreso del registro fue exitoso, se realiza una consulta en la base de datos, la misma se puede visualizar en la Figura 5.20

Results | Messages

	idCliente	nombre	apellido	telefono	correo	cedula	direccion
1	1	Esteban	Machado	0998258263	esteban.machado@epn.edu.ec	1721980694	Tumbaco
2	2	Consumidor	Final	0000	0000	0000	0000

Figura 5.20 Ingreso del cliente en la base de datos

5.1.9 GENERAR REPORTES

Para la generación de reportes se corroboró que desde la interfaz se pueda visualizar las ventas generadas en el periodo seleccionado (del 1/1/2022 al 3/22/2022), la Figura 5.21 muestra el reporte de ventas del año 2022.

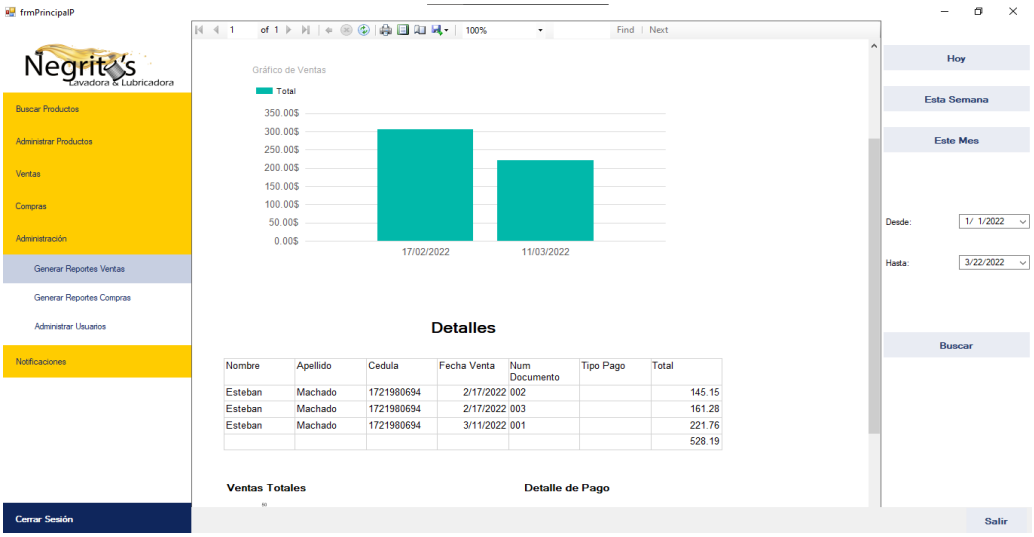


Figura 5.21 Generación de reportes

Adicionalmente se comprobó que se pueda exportar el reporte en un archivo de formato PDF o en un archivo de formato Excel. La Figura 5.22 muestra el reporte generado en formato PDF.

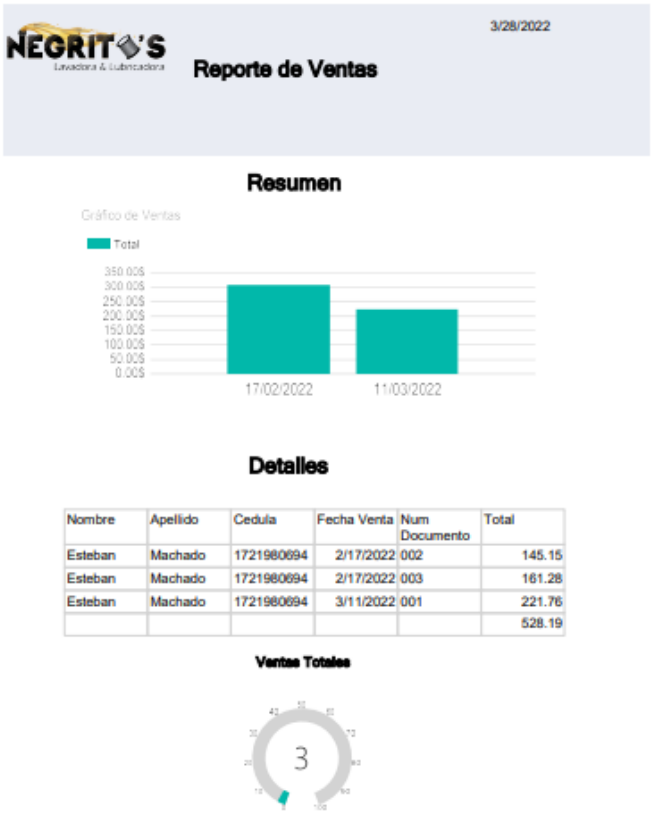


Figura 5.22 Reporte de ventas en formato PDF

5.1.10 VISUALIZAR NOTIFICACIONES

La interfaz que se muestra en la Figura 5.23 presenta las notificaciones generadas por la escasez de un producto, en estas se puede apreciar información sobre el producto, el stock actual y el stock mínimo. Para esta prueba se redujo el stock actual a 1 del producto Castrol 10W30 el cual cuenta con una cantidad mínima de 10.

Figura 5.23. Interfaz de notificaciones

Se puede visualizar la notificación con una consulta en la base de datos Figura 5.24

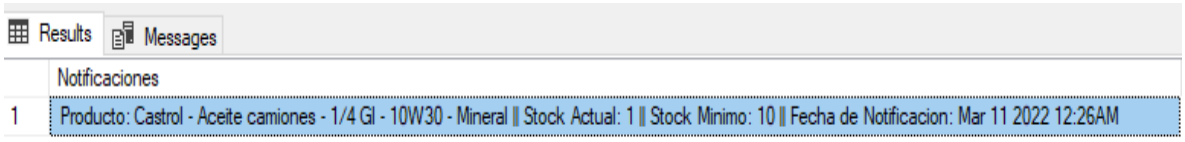


Figura 5.24 Notificaciones en la base de datos

Finalmente, en la **Tabla 5.1** se puede apreciar el cumplimiento de lo establecido previamente en las historias de usuario, se verificó si cada una de ellas cumple con lo descrito en las Tabla 4.1 y Tabla 4.2.

Tabla 5.1 Verificación de cumplimiento con las historias de usuario.

ID	Título	Realizado
HU_01	Autenticación de usuarios	OK
HU_02	Búsqueda de productos	OK
HU_03	Búsqueda de productos por vehículo	OK
HU_04	Administración de productos	OK
HU_05	Generación de comprobantes de venta	OK
HU_06	Generación de comprobantes de compra	OK
HU_07	Administración de clientes	OK
HU_08	Administración de proveedores	OK
HU_09	Generación de reportes de venta	OK
HU_10	Administración de usuarios	OK
HU_11	Generación de reportes de compra	OK
HU_12	Visualización de notificaciones	OK

5.2 PRUEBAS DE ACEPTACIÓN

Se realizaron pruebas de aceptación a cinco usuarios de la PYME, los cuales probaron cada una de las funcionalidades del prototipo. Luego de lo cual, se realizó una encuesta para validar si el prototipo cumple con los requerimientos establecidos. En la Tabla 5.2 se puede apreciar los resultados obtenidos, estos ayudaron a la detección y corrección de errores encontrados. La encuesta generada se encuentra en el ANEXO F.

Tabla 5.2 Resultado de las pruebas de aceptación

Pregunta	**Respuesta**	
	Si	**No**
¿Pudo iniciar sesión con su usuario/correo y contraseña?	100%	0%
¿Pudo ejecutar la búsqueda de un producto?	80%	20%
¿Pudo ejecutar la búsqueda de productos por vehículo?	100%	0%
¿Pudo ingresar un nuevo producto?	100%	0%
¿Pudo ingresar un nuevo vehículo?	100%	0%
¿Pudo generar un comprobante de venta correctamente?	100%	0%
¿Pudo anular un comprobante de venta correctamente?	100%	0%
¿Pudo ingresar un nuevo cliente?	100%	0%
¿Pudo generar un reporte correctamente?	100%	0%
¿Pudo visualizar las notificaciones?	100%	0%

5.3 CORRECCIÓN DE ERRORES

Las modificaciones y correcciones fueron realizadas una vez procesados los datos obtenidos de las pruebas de aceptación. Los resultados obtenidos muestran que no existió ningún problema al momento de probar el prototipo, sin embargo, se sugirió mejorar la forma en que se realizan las búsquedas y ciertos detalles estéticos en la interfaz del usuario. Las correcciones realizadas son las siguientes:

- En los procedimientos almacenados de búsqueda se agregó la sentencia `like '%'+@valor+'%'` para de esta manera permitir al usuario realizar búsquedas ingresando únicamente un fragmento del nombre, marca, código de barras, etc.
- Cambio del logo de la empresa en el formulario principal.
- Mejoras en la descripción de los mensajes de error.
- Cambio de aspecto en el botón de cerrar sesión.

- Cambios y mejoras en el aspecto físico de la generación de reportes de ventas y compras. Adicionalmente, se ajustó los tamaños de impresión al formato A4.

En la Figura 5.25 se muestra el cambio de logo realizado a la interfaz principal. Se aprecia que existió un cambio en la fuente del logotipo que va acorde a los de la PYME.

La Figura 5.26 muestra los cambios realizados al botón `Cerrar Sesión`, en el cual se agregó un icono y se centró el texto para que el botón se más intuitivo.

Figura 5.25 Cambios de Logo en la interfaz principal

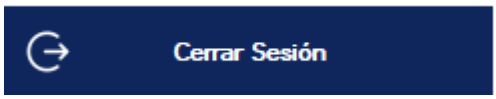


Figura 5.26 Modificaciones al botón `Cerrar Sesión`

6. CONCLUSIONES Y RECOMENDACIONES

En este capítulo se exponen las conclusiones y recomendaciones obtenidas como resultado del desarrollo del presente Trabajo de Titulación.

6.1 CONCLUSIONES

- Al concluir el presente Trabajo de Titulación se dispone de un prototipo de sistema de gestión de inventarios el cual cuenta con tres componentes: una base de datos que almacena los registros de la aplicación en SQL Server, un servicio web desarrollado en WCF encargado de atender las peticiones del cliente y una aplicación de escritorio desarrollada en *Windows Forms* que interactúa con el usuario.

- En base a elementos tales como el esquema relacional de la base de datos y los diagramas de clases, actividad y secuencia se logró diseñar los componentes necesarios para el presente trabajo de titulación.

- La implementación del prototipo a nivel de negocios se basó en el desarrollo de contratos e interfaces, los cuales fueron implementados en el lenguaje de programación C#, así mismo para el nivel de acceso a datos se crearon vistas y procedimientos almacenados para facilitar la visualización y manipulación de datos en la aplicación.

- Este prototipo permite mantener un control del inventario de productos, la gestión de clientes, proveedores y empleados, generar comprobantes imprimibles de compra y venta, visualizar notificaciones de productos en escasez, generar y exportar reportes a formato PDF, XLSX y DOCX.

- El uso de roles de usuario permitió segmentar las funciones del usuario empleado y administrador limitado el acceso a interfaces no autorizadas.

- Se probó cada una de las funcionalidades del prototipo y se realizaron encuestas de aceptación, a partir de las cuales, se realizaron mejoras funcionales en el proceso de búsqueda y en la interfaz de usuario.

6.2 RECOMENDACIONES

- Se recomienda utilizar el presente Trabajo de Titulación para administrar el inventario y llevar un registro de las compras y ventas realizadas en una empresa tipo lavadora y lubricadora.
- Sería posible realizar una mejora en la apariencia estética de las notificaciones mediante la instalación de uso del control `Microsoft.Toolkit.Uwp.Notificacations`.
- Se recomienda, para trabajos futuros, revisar otras tecnologías de comunicación entre un servicio web y una aplicación, ya que cada mensaje enviado mediante SOAP incluye un *overhead* considerable originado por el uso de XML.
- Se recomienda utilizar regiones para separar el código presente dentro de un mismo formulario, de esta manera, es posible identificar las líneas de código pertenecientes a una sección.
- En la implementación del servicio web se recomienda el uso de un certificado de seguridad otorgado por una entidad certificadora para proteger las comunicaciones entre la aplicación de escritorio y el servicio web.
- Para futuros proyectos, se recomienda emplear el *frawework* .NET Core desarrollado también por Microsoft debido a que este es multiplataforma y puede ser implementado en servidores Linux
- La creación de clases para los modelos donde se requiere realizar validaciones ayudaría a simplificar los cambios en la aplicación de escritorio en el caso que se realicen cambios en la base de datos.

7. REFERENCIAS BIBLIOGRÁFICAS

[1] O. Blancarte, Introducción a la Arquitectura de Software, Ciudad de México: Independently published, 2020.

[2] C. De La Torre, U. Zorrilla, M. Ramos y J. Calvarro, Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0, Pontevedra: Krasis Press, 2010.

[3] M. Perez, Microsoft SQL Server 2008 R2. Motor de base de datos y administración, Madrid: RC Libros, 2011.

[4] M. Berger, "Data Access Object Pattern," 30 Octubre 2005. [En línea]. Available: http://max.berger.name.s3.amazonaws.com/research/silenus/print/dao.pdf. [Último acceso: 7 Abril 2021].

[5] J. Löwy y M. Montgomery, Programming WCF Services, Boston: O`Retlly Media , 2015.

[6] N. Pathak, Pro WCF 4 Practical Microsoft SOA Implementation, California: Apress, 2011.

[7] B. Mwikali, "Comparing Graphical User Interface (GUI) and Command Line Interface (CLI)," Section, 6 Abril 2021. [En línea]. Available: https://www.section.io/engineering-education/comparing-graphical-user-interface-gui-and-command-line-interface-cli/. [Último acceso: 10 Agosto 2021].

[8] E. Tello-Leal, C. Sosa y D. Tello-Leal, "Revisión de los sistemas de control de versiones utilizados en el desarrollo de software," vol. 3, nº 1, 2012.

[9] S. Chacon y B. Staub, Pro GIT, Apress, 2021.

[10] M. Tsitoara, Beginning Git and GitHub, Antananarivo: Apress, 2020.

[11] E. Yepez y K. Armijos, APLICACIÓN DE LA METODOLOGÍA KANBAN EN EL DESARROLLO DEL SOFTWARE PARA GENERACIÓN, VALIDACIÓN Y ACTUALIZACIÓN DE REACTIVOS, INTEGRADO AL SISTEMA INFORMÁTICO DE CONTROL ACADÉMICO UNACH, Riobamba: UNIVERSIDAD NACIONAL DE CHIMBORAZO, 2020.

[12] M. Perez, Guía Comparativa de Metodologías Ágiles, Segovia: Universidad de Valladolid.

[13] H. kniberg y M. Skarin, Kanban y Scrum - Obteniendo lo mejor de ambos, Estados Unidos de América: C4media Inc., 2010.

[14] "Qué es Kanban: Definición, Características y Ventajas," Kanbanize, 2021. [En línea]. Available: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban. [Último acceso: 6 Agosto 2021].

[15] I. Baquero, "¿Qué es Trello?," Ascenso, 13 Octubre 2017. [En línea]. Available: https://ascenso.org/categoria/actualidad-digital/que-es-trello/. [Último acceso: 9 Agosto 2021].

[16] G. Myers, C. Sandler y T. Badgett, The Art Of The Software Testing, New Jersy: John Wiley & Sons, Inc. , 2012.

8. ANEXOS

ANEXO A

1. Seleccione cuáles son los productos más vendidos

☐ Aceite

☐ Filtros

☐ Otro

2. Seleccione las características generales de un producto

☐ Marca

☐ Descripción

☐ Código de Barras

☐ Numero de Lote

☐ Precio de Venta al por Mayor y Menor

☐ Fecha de Fabricación

☐ Precio de Compra

☐ Margen de venta al por Mayor y Menor

☐ Cantidad

☐ Cantidad Mínima

3. Seleccione las características específicas del producto Aceite

☐ Presentación (gal, ¼ gal, etc.)

☐ SAE (20W50, 10W30, etc.)

☐ API (SN, SF, etc.)

☐ Lugar de Fabricación (Nacional/Importado)

☐ Tipo de Aceite (Mineral, Sintético, Semisintético)

☐ Tipo de Combustible (Gasolina, Diesel)

4. Seleccione las características específicas del producto Filtro

☐ Tipo de Filtro (Aire, Aceite, Combustible, Cabina)

☐ Rosca (M20x1.5, M3/4x16, etc.)

☐ Medidas

☐ Código de Filtro (AO3614, PH966, etc.)

5. ¿Qué características considera necesarias de un vehículo para su mantenimiento?

☐ Marca

☐ Modelo

☐ Color

☐ Año

☐ Motor

6. ¿Qué campos considera necesarios en un comprobante de venta?

☐ Datos Cliente

☐ Lista Productos (Aceite, Filtro, Otros)

☐ Fecha

☐ Subtotal

☐ IVA

☐ Total

☐ Descuento

7. ¿Qué datos considera necesarios de un cliente?

☐ Nombre

☐ Apellido

☐ Cédula

☐ Nacionalidad

☐ Dirección

☐ Teléfono

☐ Correo

8. ¿Qué campos considera necesarios en un comprobante de compra?

☐ Datos del Proveedor

☐ Lista de Productos (Aceite, Filtros, Otros)

☐ Fecha

☐ Subtotal

☐ IVA

☐ Total

☐ Descuento

9. ¿Qué datos considera necesarios de un proveedor?

☐ Nombre

☐ Apellido

☐ RUC

☐ Régimen (Sociedad, Persona Natural)

☐ Empresa

☐ Teléfono

☐ Correo

☐ Dirección

10. ¿Desearía la opción de anular un comprobante, en caso de que este haya sido generado por error?

o Sí

o No

11. ¿Qué rango de fechas sería adecuado para la generación de reportes de compra o venta?

☐ Diario

☐ Semanal

☐ Mensual

☐ Trimestral (3 meses)

☐ Semestral (6 meses)

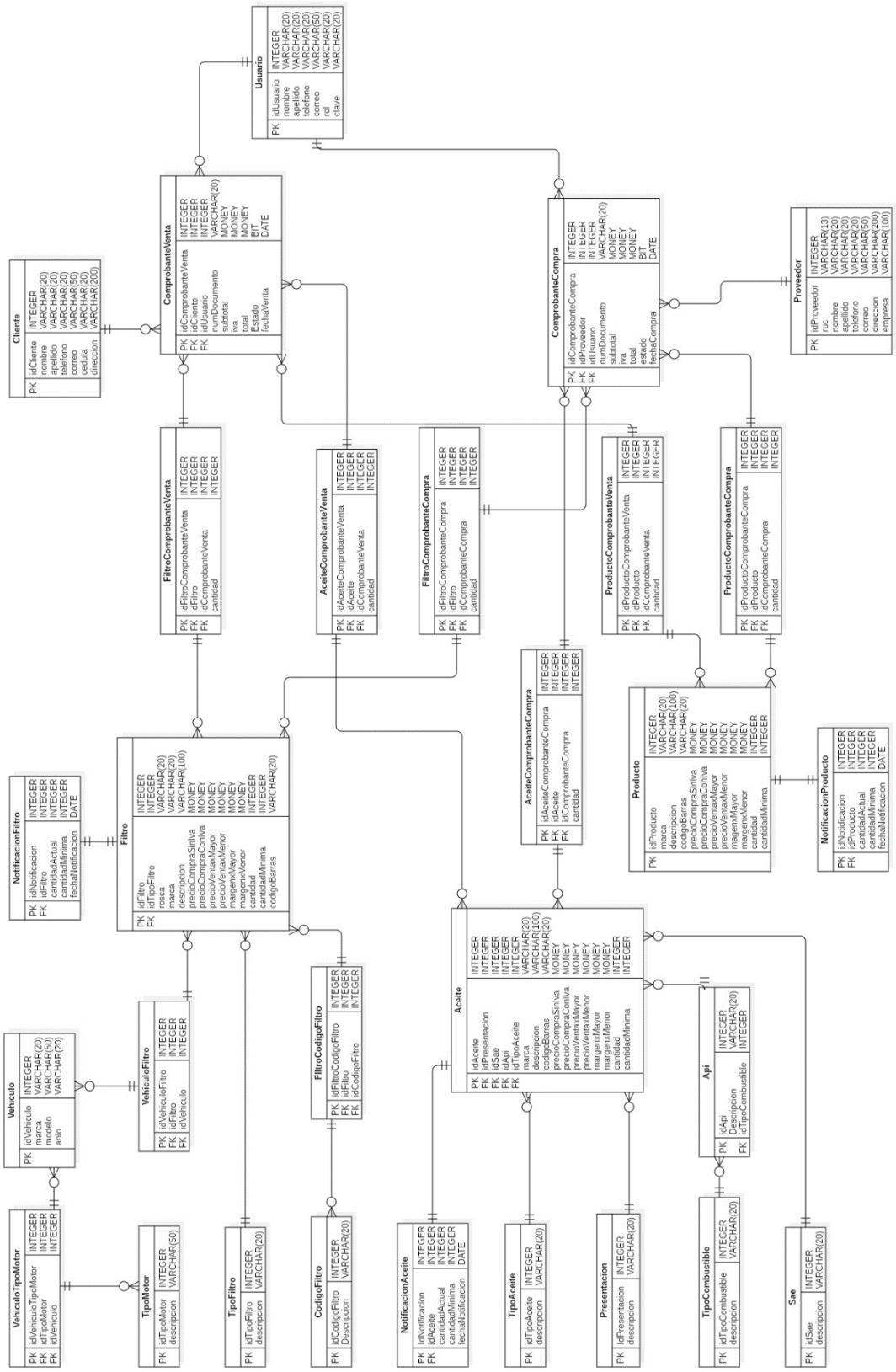
12. ¿Desearía que se le notifique cuando un producto está por agotarse?

o Sí

o No

ANEXO B

Esquema relacional de la base de datos.

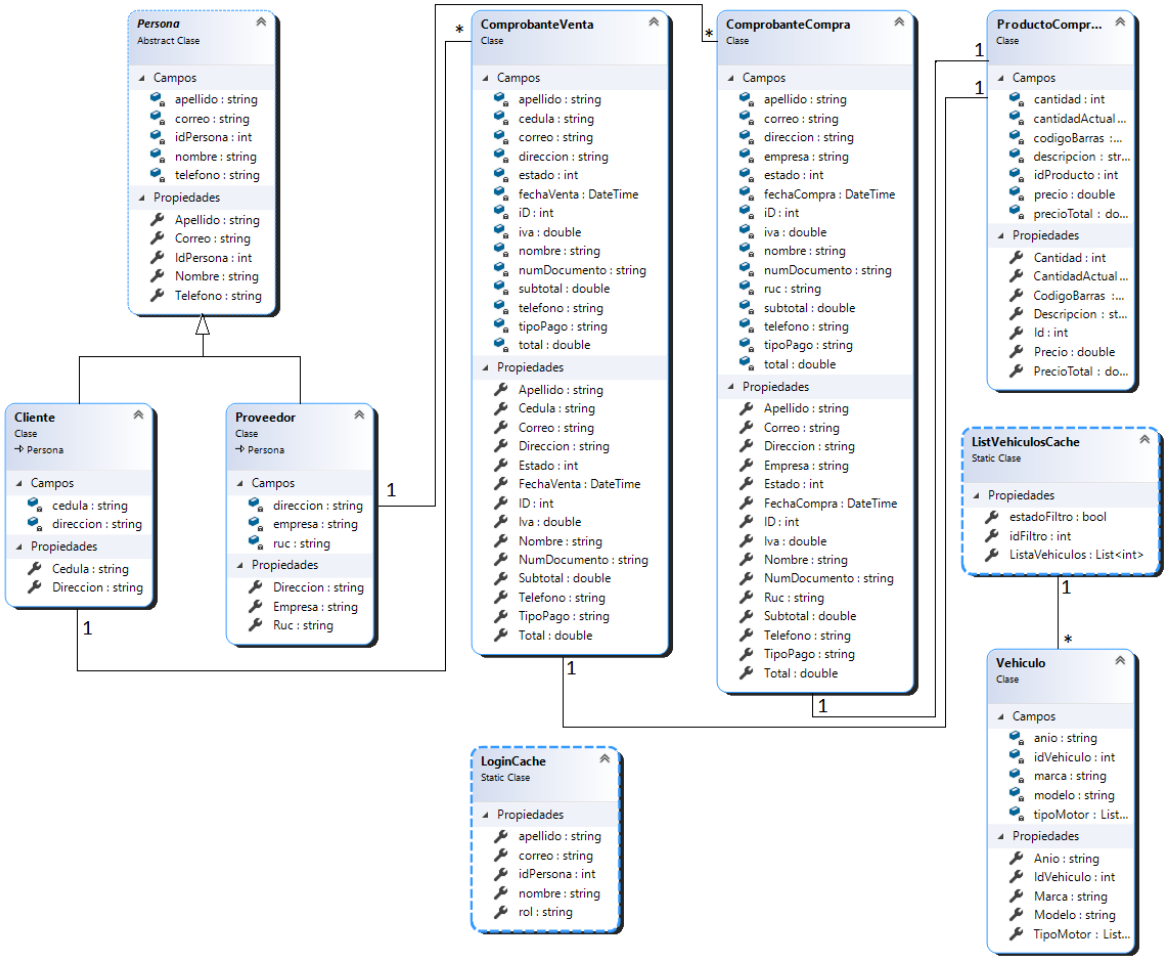


ANEXO C

Este anexo se adjunta de manera digital y contiene el script en SQL para la generación de la base de datos y el código del prototipo.

ANEXO D

En este anexo se encuentra el diagrama de clases generado para este prototipo.



ANEXO E

En este anexo se encuentran los diagramas de secuencia generados para este prototipo.

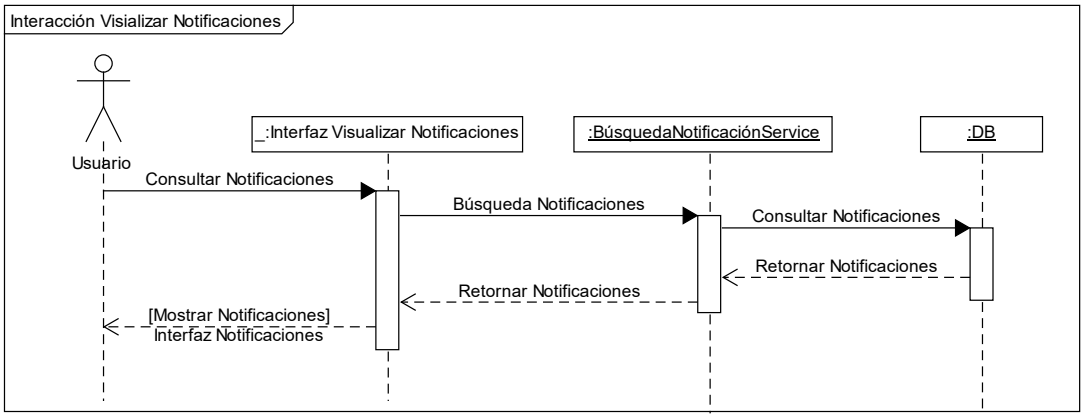


Figura E.1 Diagrama de secuencia de visualizar notificaciones

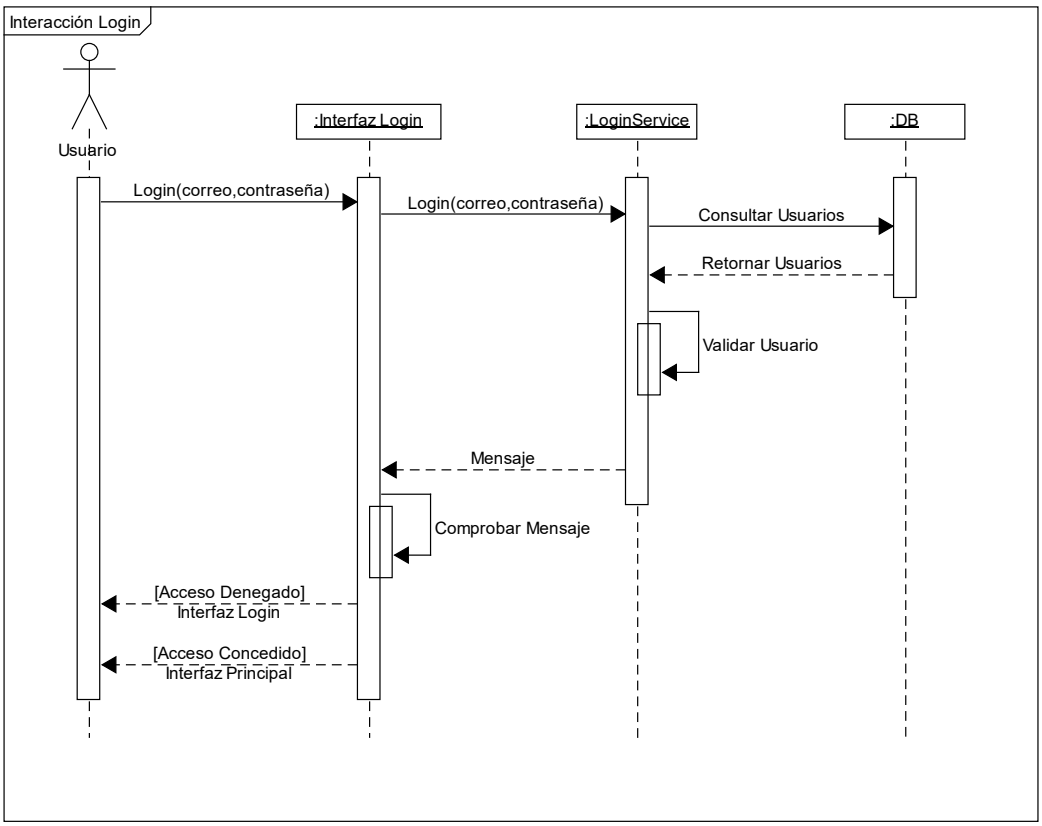


Figura E.2 Diagrama de secuencia de Login

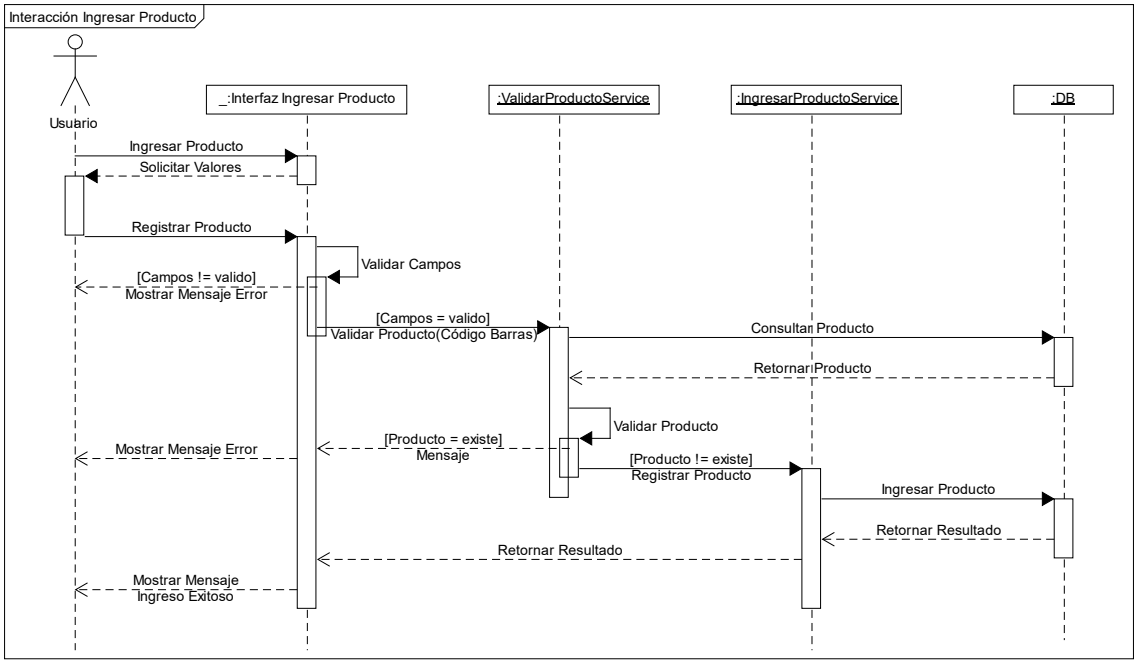


Figura E.3 Diagrama de secuencia de ingresar producto.

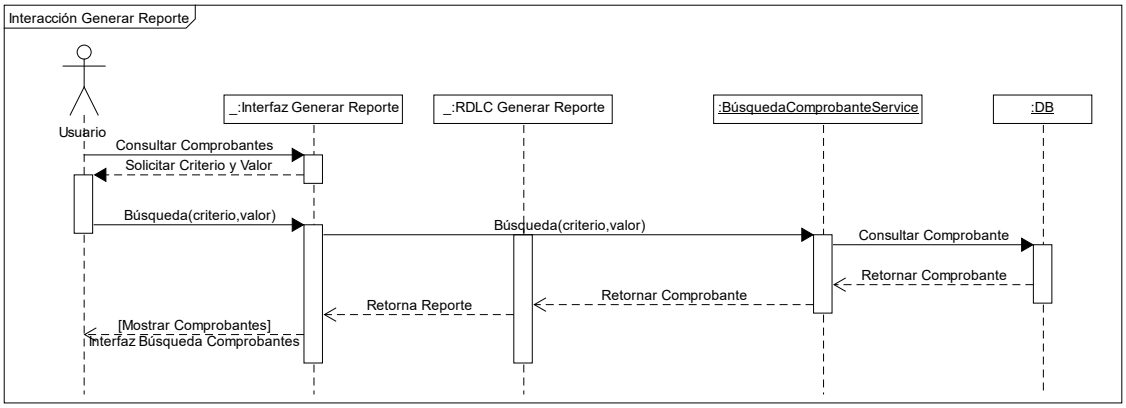


Figura E.4 Diagrama de secuencia de generar reporte

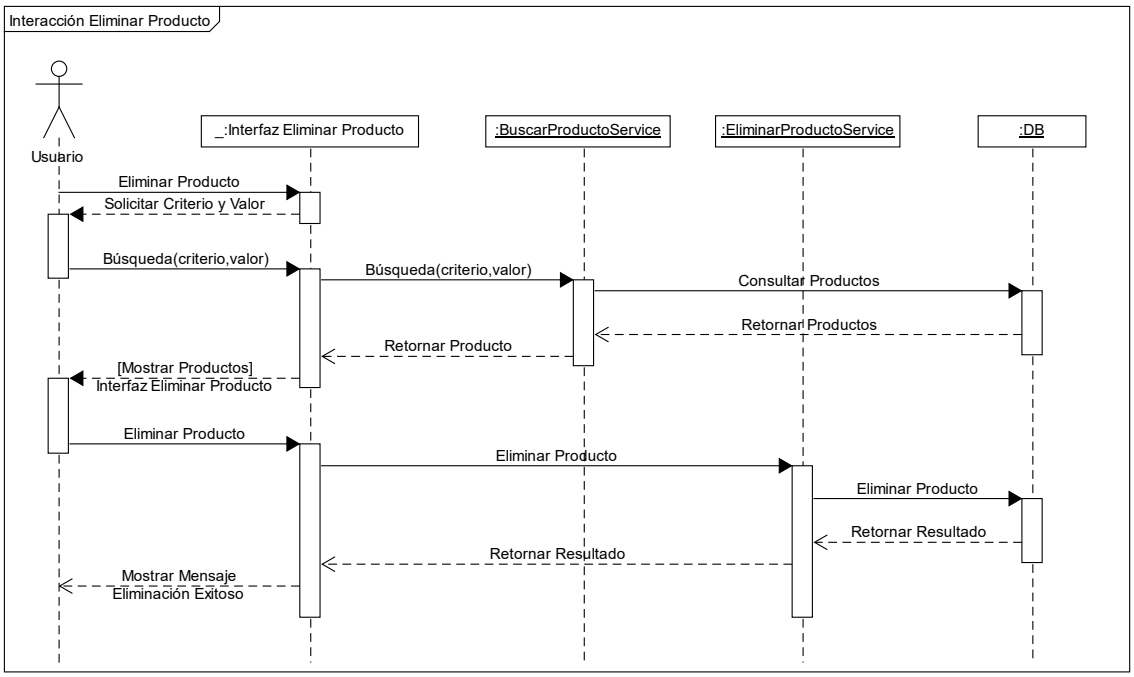


Figura E.5 Diagrama de secuencia de eliminar producto

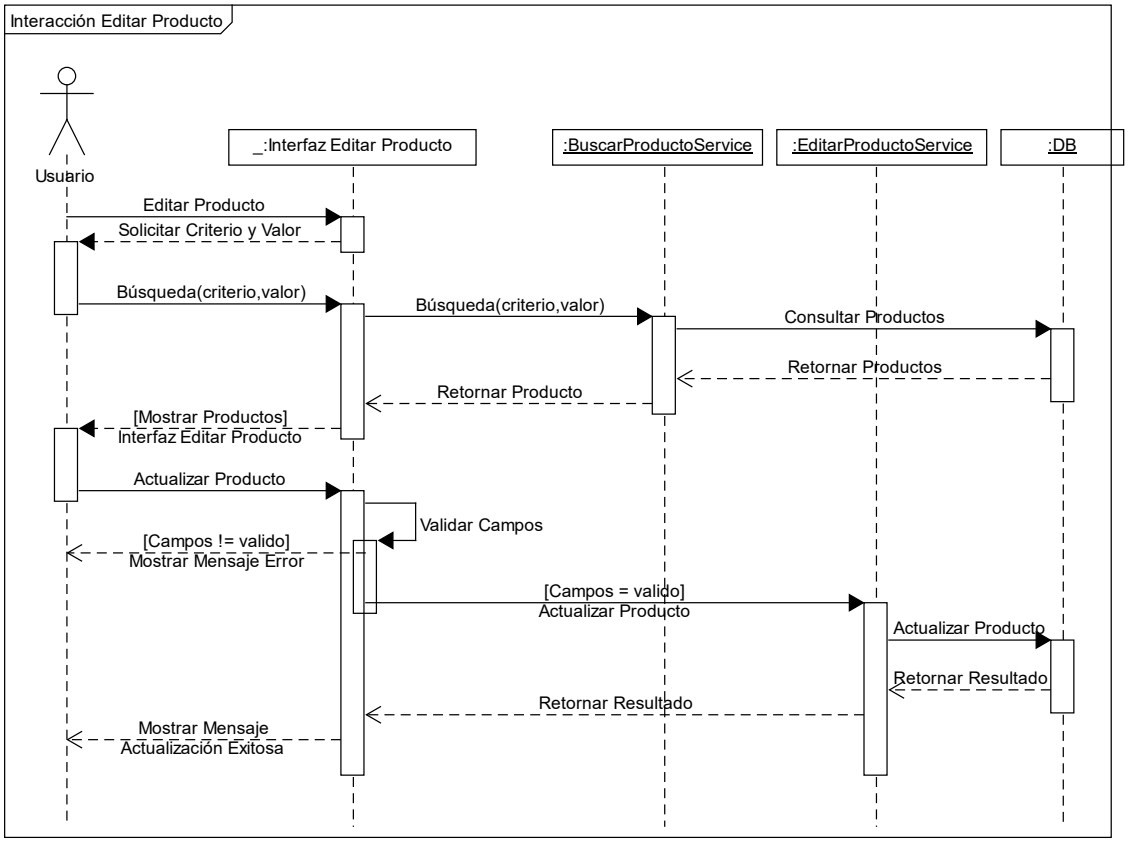


Figura E.6 Diagrama de secuencia de editar producto

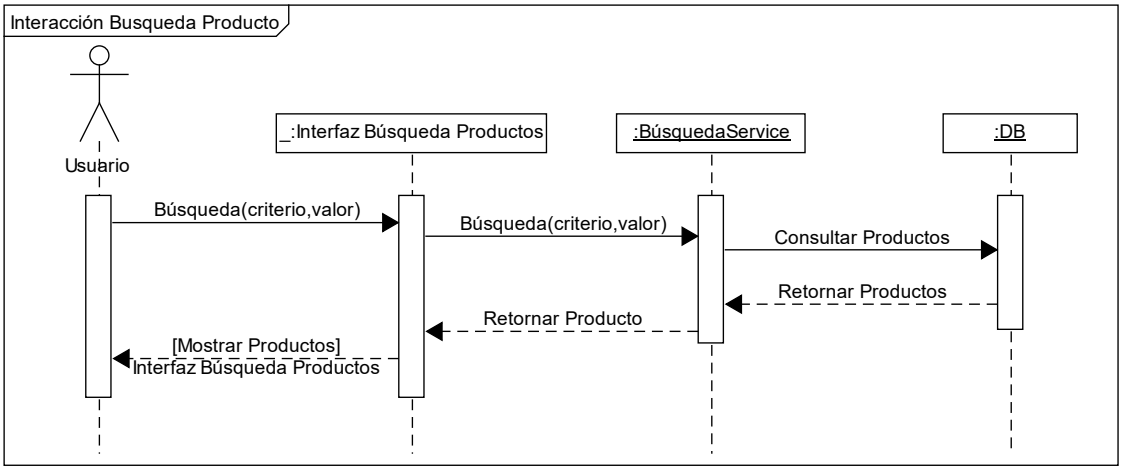


Figura E.7 Diagrama de secuencia de búsqueda de productos

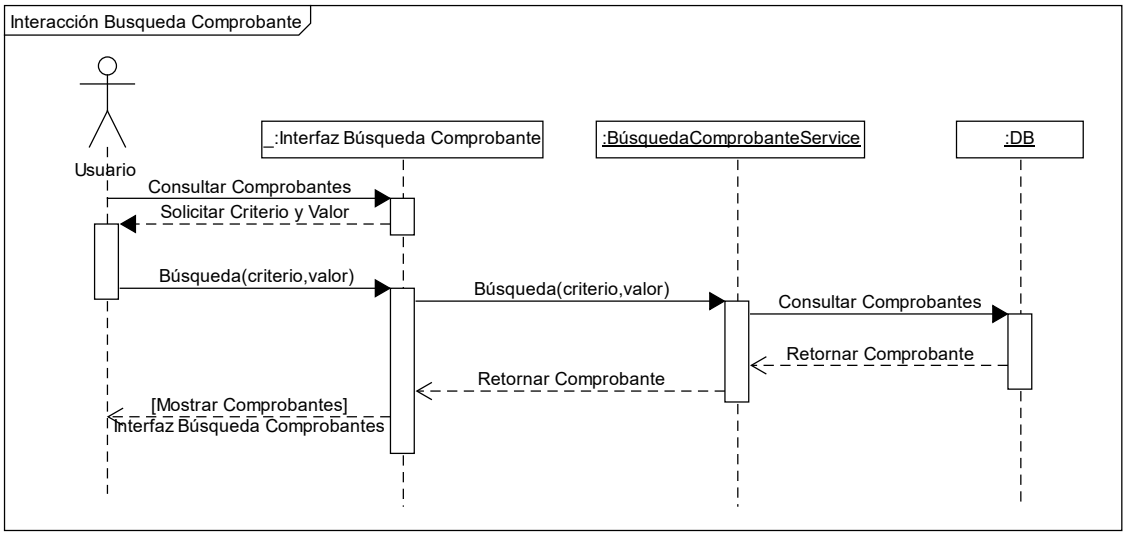


Figura E.8 Diagrama de secuencia de búsqueda de comprobante

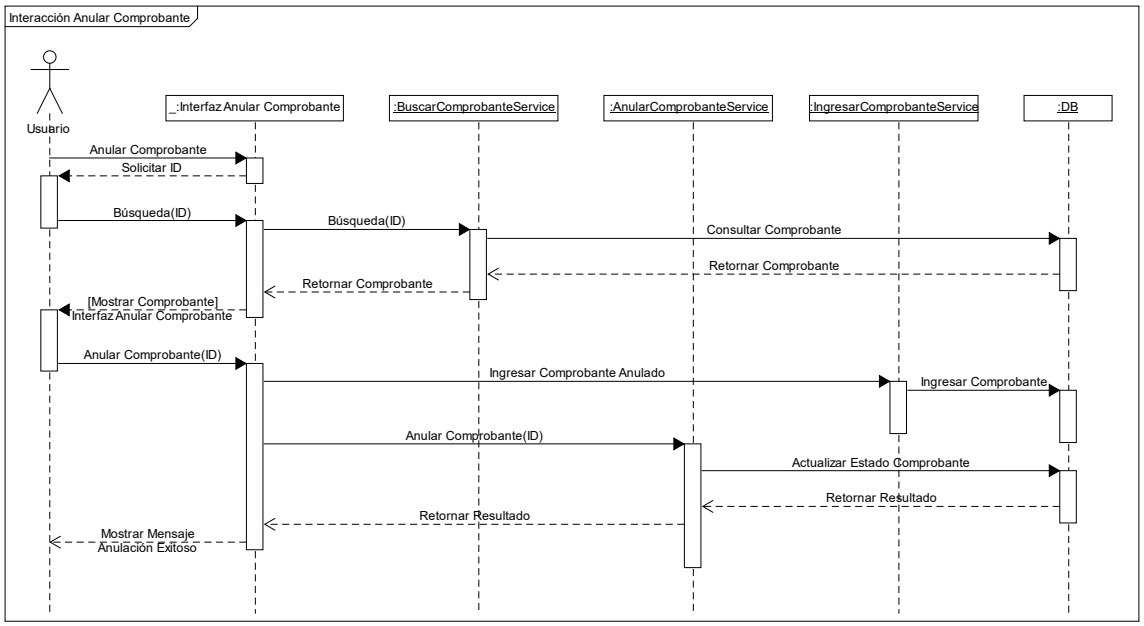


Figura E.9 Diagrama de secuencia de anular comprobante

ANEXO F

1. **¿Pudo iniciar sesión con su usuario/correo y contraseña?**
    - Sí
    - No
2. **¿Pudo ejecutar la búsqueda de un producto?**
    - Sí
    - No
3. **¿Pudo ejecutar la búsqueda de productos por vehículo?**
    - Sí
    - No
4. **¿Pudo ingresar un nuevo producto?**
    - Sí
    - No
5. **¿Pudo ingresar un nuevo vehículo?**
    - Sí
    - No
6. **¿Pudo generar un comprobante de venta correctamente?**
    - Sí
    - No
7. **¿Pudo anular un comprobante de venta correctamente?**
    - Sí
    - No
8. **¿Pudo ingresar un nuevo cliente?**
    - Sí
    - No

9. **¿Pudo generar un reporte correctamente?**
    - Sí
    - No
10. **¿Pudo visualizar las notificaciones?**
    - Sí
    - No

ANEXO G

Este anexo se adjunta de manera digital y contiene el manual de instalación del prototipo.

ANEXO H

Este anexo se adjunta de manera digital y contiene el manual de usuario del prototipo.

ORDEN DE EMPASTADO