UNIVERSIDAD HISPANOAMERICANA

ESCUELA DE INFORMÁTICA

PROYECTO DE TESINA PARA OPTAR EL GRADO DE BACHILLERATO EN INGENIERÍA INFORMÁTICA

TÍTULO DEL PROYECTO

Propuesta de implementación de aplicación web para comercio electrónico de la empresa ProLine.

Sustentante: Luiscarlos Martínez Zambrano.

Tutor: Pedro Ignacio Leiva Chinchilla.

Marzo 2023.

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

DECLARACIÓN JURADA

DECLARACIÓN JURADA

Yo, Luiscarlos Martinez Zambrano, de documento de identidad DIMEX número 186200546207, estudiante de Ingeniería en Informática, bajo la fe del juramento y conociendo las sanciones con que la ley castiga el falso testimonio, declaro que el presente trabajo de investigación es fruto de mi autoría y no de la duplicación de información ajena presentada como propia, ya sea de forma parcial o total, proveniente de libros, documentos o artículos, impresos o no, físicos o virtuales. Toda cita se ha destacado entre comillas o se ha parafraseado, y toda referencia se ha consignado como corresponde. Es todo.

CARTAS DE APROBACIÓN DEL TUTOR Y LECTOR

CARTA DEL TUTOR

San José, 23 de febrero del 2023

Ing. María Isabel Losilla Barrientos.
Facultad de Computación
Universidad Hispanoamericana

Estimada señora:

El estudiante Luiscarlos Martínez Zambrano, cédula de identidad número 186200546207 (DIMEX), me ha presentado, para efectos de revisión y aprobación, el trabajo de investigación denominado "Propuesta de implementación de aplicación web para comercio electrónico de la empresa ProLine", el cual ha elaborado para optar por el grado académico de Bachillerato en Ingeniería Informática.

En mi calidad de tutor, he verificado que se han hecho las correcciones indicadas durante el proceso de tutoría y he evaluado los aspectos relativos a la elaboración del problema, objetivos, justificación; antecedentes, marco teórico, marco metodológico, tabulación, análisis de datos; conclusiones y recomendaciones.

De los resultados obtenidos por el postulante, se obtiene la siguiente calificación:

a)	ORIGINAL DEL TEMA	10%	0%
b)	CUMPLIMIENTO DE ENTREGA DE AVANCES	20%	20%
C)	COHERENCIA ENTRE LOS OBJETIVOS, LOS INSTRUMENTOS APLICADOS Y LOS RESULTADOS DE LA INVESTIGACIÓN	30%	25%
d)	RELEVANCIA DE LAS CONCLUSIONES Y RECOMENDACIONES	30%	25%
e)	CALIDAD, DETALLE DEL MARCO TEÓRICO	10%	10%
	TOTAL		80%

En virtud de la calificación obtenida, se avala el traslado al proceso de lectura.

Atentamente,

Mag. Pedro Ignacio Leiva Chinchilla
1-1394-0453

CARTA DE LECTOR

San José,

Universidad Hispanoamericana
Sede Llorente
Carrera de Informática

Estimado señor

El estudiante Luiscarlos Martínez Zambrano, cédula de identidad 186200546207, me ha presentado para efectos de revisión y aprobación, el trabajo de investigación denominado "Propuesta de implementación de aplicación web para comercio electrónico de la empresa ProLine. ".

He revisado y he hecho las observaciones relativas al contenido analizado, particularmente lo relativo a la coherencia entre el marco teórico y análisis de datos, la consistencia de los datos recopilados y la coherencia entre éstos y las conclusiones; asimismo, la aplicabilidad y originalidad de las recomendaciones, en términos de aporte de la investigación.

Por consiguiente, este trabajo cuenta con mi aval para ser presentado en la defensa pública.

Atte.

Randall Vargas Villalobos

Firmado digitalmente por Randall Vargas Villalobos
Fecha: 2023.04.17 23:14:17 -06'00'

Firma
Randall Vargas Villalobos
Cédula: 1-1140-0113

AUTORIZACIÓN PARA CENIT

UNIVERSIDAD HISPANOAMERICANA
CENTRO DE INFORMACION TECNOLOGICO (CENIT)
CARTA DE AUTORIZACIÓN DE LOS AUTORES PARA LA CONSULTA, LA REPRODUCCION PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DE LOS TRABAJOS FINALES DE GRADUACION

San José, 18 de abril de 2023

Señores:
Universidad Hispanoamericana
Centro de Información Tecnológico (CENIT)

Estimados Señores:

El suscrito (a) Luiscarlos Martínez Zambrano, con número de identificación DIMEX 186200546207, autor del trabajo de graduación titulado "Propuesta de implementación de aplicación web para comercio electrónico de la empresa ProLine", presentado y aprobado en el año 2023 como requisito para optar por el título de Bachillerato en Ingeniería Informática; autorizo al Centro de Información Tecnológico (CENIT) para que con fines académicos, muestre a la comunidad universitaria la producción intelectual contenida en este documento.

De conformidad con lo establecido en la Ley sobre Derechos de Autor y Derechos Conexos N° 6683, Asamblea Legislativa de la República de Costa Rica.

Cordialmente,

Firma y Documento de Identidad

ANEXO 1 (Versión en línea dentro del Repositorio)
LICENCIA Y AUTORIZACIÓN DE LOS AUTORES PARA PUBLICAR Y PERMITIR LA CONSULTA Y USO

Parte 1. Términos de la licencia general para publicación de obras en el repositorio institucional

Como titular del derecho de autor, confiero al Centro de Información Tecnológico (CENIT) una licencia no exclusiva, limitada y gratuita sobre la obra que se integrará en el Repositorio Institucional, que se ajusta a las siguientes características:

a) Estará vigente a partir de la fecha de inclusión en el repositorio, el autor podrá dar por terminada la licencia solicitándolo a la Universidad por escrito.

b) Autoriza al Centro de Información Tecnológico (CENIT) a publicar la obra en digital, los usuarios puedan consultar el contenido de su Trabajo Final de Graduación en la página Web de la Biblioteca Digital de la Universidad Hispanoamericana

c) Los autores aceptan que la autorización se hace a título gratuito, por lo tanto, renuncian a recibir beneficio alguno por la publicación, distribución, comunicación pública y cualquier otro uso que se haga en los términos de la presente licencia y de la licencia de uso con que se publica.

d) Los autores manifiestan que se trata de una obra original sobre la que tienen los derechos que autorizan y que son ellos quienes asumen total responsabilidad por el contenido de su obra ante el Centro de Información Tecnológico (CENIT) y ante terceros. En todo caso el Centro de Información Tecnológico (CENIT) se compromete a indicar siempre la autoría incluyendo el nombre del autor y la fecha de publicación.

e) Autorizo al Centro de Información Tecnológica (CENIT) para incluir la obra en los índices y buscadores que estimen necesarios para promover su difusión.

f) Acepto que el Centro de Información Tecnológico (CENIT) pueda convertir el documento a cualquier medio o formato para propósitos de preservación digital.

g) Autorizo que la obra sea puesta a disposición de la comunidad universitaria en los términos autorizados en los literales anteriores bajo los límites definidos por la universidad en las “Condiciones de uso de estricto cumplimiento” de los recursos publicados en Repositorio Institucional.

SI EL DOCUMENTO SE BASA EN UN TRABAJO QUE HA SIDO PATROCINADO O APOYADO POR UNA AGENCIA O UNA ORGANIZACIÓN, CON EXCEPCIÓN DEL CENTRO DE INFORMACIÓN TECNOLÓGICO (CENIT), EL AUTOR GARANTIZA QUE SE HA CUMPLIDO CON LOS DERECHOS Y OBLIGACIONES REQUERIDOS POR EL RESPECTIVO CONTRATO O ACUERDO.

DEDICATORIA

Este proyecto lo dedico en primer lugar a mi esposa, cuyo amor me refina, purifica, y me impulsa a progresar y mejorar.

En segundo lugar, a mi mami, mi abuelo y mis tías; esto representa la realización de un sueño que compartimos desde mi infancia.

En tercer lugar, a mis sobrinos Matías, Annabella y Charlotte, que son el futuro de nuestra familia y les debemos impulsarlos con todo nuestro amor para tener una realidad mejor.

AGRADECIMIENTO

Doy gracias a mi Padre Celestial, que me dio la vida y me dotó de habilidades y talentos, poniendo a mi disposición todo lo que necesito para progresar.

Agradezco a mi mami, mi abuelo y mis tías por todo lo que hicieron para darme la formación académica que garantizaría mi futuro.

Gracias especiales doy a mi esposa por compartir nuestras cargas juntos, y el amor, paciencia, motivación, y amonestaciones que me ha dado a lo largo de este camino.

También doy gracias a mi hermano Guillermo y mi compañero Byron. Su ayuda en brindarme un poco de su experiencia y orientación en esta última etapa probó ser decisiva.

ABREVIATURAS

Las abreviaturas utilizadas en este documento se definen a continuación:

- C#: C Sharp.
- HTML: Hyper Text Markup Language.
- ID: Identification.
- IDE: Integrated Development Environment
- SQL: Structured Query Language.
- UML: Unified Modeling Language.
- CSS: Cascade Style Sheets.
- UNAM: Universidad Nacional Autónoma de México.
- IEEE: Institute of Electrical and Electronics Engineers.
- S.A.: Sociedad Anónima.
- PYME: Pequeña y mediana empresa.
- UNED: Universidad Estatal a Distancia.
- E-commerce: Comercio electrónico.

RESUMEN

El proyecto presentado en este documento es una propuesta de implementación de una aplicación web para comercio electrónico de la empresa ProLine CR.

Representa también la puesta en práctica del conocimiento y las habilidades obtenidas a lo largo de la carrera de Ingeniería Informática de la Universidad Hispanoamericana, particularmente en lo relacionado a la ingeniería de software y su ciclo de vida. Temas como recolectar datos y analizarlos con miras a diseñar una solución de software, el diseño de esta incluyendo los elementos físicos, lógicos y visuales, el desarrollo de la solución de software haciendo uso de técnicas de programación, llegando hasta la realización de pruebas sobre la solución brindada y la apropiada documentación orientada al usuario final; se ven de manifiesto en el presente proyecto, que se encuentra dividido en siete partes:

- Capítulo I: Problema del proyecto. Contiene los antecedentes del problema, la justificación del proyecto y la definición del problema, los objetivos, el alcance y las limitaciones; y un cronograma de actividades.
- Capítulo II: Marco teórico. En este capítulo se da cuenta de toda la base teórica necesaria para desarrollar el proyecto, y para que el lector también comprenda los datos del problema a investigar y la solución brindada.
- Capítulo III: Marco metodológico. Se da cuenta de los métodos a usarse en el proyecto, incluyendo el tipo y enfoque de la investigación, las fuentes, lo relacionado a la recolección de datos, las variables y el diseño de la investigación.
- Capítulo IV: Diagnóstico de la situación actual. En este capítulo se muestra la aplicación de las técnicas de recolección de información, y se realiza el análisis de los datos recolectados y el diseño de la solución a partir de dicho análisis.

- Capítulo V: Propuesta del proyecto. Este capítulo consiste en el desarrollo de la solución que se propone implementar como parte de la investigación. Se detallan todas las partes del software desarrollado como solución.
- Capítulo VI: Conclusiones y recomendaciones. Brindan una reseña de lo que se alcanzó con el proyecto, y también ideas de otros elementos no abarcados durante este.
- Capítulo VII: Apéndices. Son documentos separados de éste, creados durante el desarrollo del proyecto.

CAPÍTULO I: PROBLEMA DEL PROYECTO

El primer capítulo de esta tesina servirá como introducción a los aspectos históricos de la compañía ProLine CR S.A., conocida como “ProLine CR” o “ProLine”. Entre los aspectos a desarrollar en este capítulo se encontrará, además, la definición de la problemática de la empresa, los objetivos, el alcance y las limitaciones del proyecto.

1.1 ANTECEDENTES Y JUSTIFICACIÓN DEL PROYECTO

En esta sección del documento se dará la información general de la empresa y datos de estrategia organizacional como: misión, visión y objetivos. Se indicará también el negocio al que se dedica y se describirá su historia.

1.1.1. MARCO DE REFERENCIA EMPRESARIAL Y CONTEXTUAL

El proyecto de desarrollo de software se llevará a cabo en la empresa ProLine CR S.A. Toda la información obtenida de la empresa es obtenida a través de una conversación con el Gerente General, Carlos Eduardo Martínez.

Nombre de la empresa: ProLine CR.

Año de fundación: 2020.

Estrategia

Misión: Según indica el Gerente General: “Brindar el mejor servicio posible a un precio justo a los clientes.” (Martínez, C., comunicación personal, 4 de abril de 2022).

Visión: Según indica el Gerente General: “Satisfacer todas las necesidades de nuestros clientes al tener la vanguardia en tecnología en nuestros productos.” (Martínez, C., comunicación personal, 4 de abril de 2022).

Objetivos: Según indica el Gerente General: “Ser líderes en ventas al mayor y a al detalle de equipos y accesorios para vehículos automotrices con el mejor respaldo y sello de garantía del país.” (Martínez, C., comunicación personal, 4 de abril de 2022).

Organización

Gerente General: Carlos Eduardo Martínez. Accionista Principal: Raury Fernández.

Negocio al que se dedica

Según indica el Gerente General: “Venta al detalle y al mayor de accesorios audio visuales para vehículos automotrices e instalación de los mismos.” (Martínez, C., comunicación personal, 4 de abril de 2022).

Historia de la organización

Según indica el Gerente General: “La empresa comenzó con un capital producto del rompimiento de una sociedad anterior, de dos años de antigüedad y dedicada a las mismas actividades.” (Martínez, C., comunicación personal, 4 de abril de 2022).

Tendencia del mercado

En Costa Rica existen muchas tiendas que ofrecen servicios similares a ProLine, aunque la mayoría tienden a la diversificación, es decir, no solo a la venta e instalación de radios con pantallas Android para carros, sino también la venta de accesorios, alarmas, y otras instalaciones de dispositivos no relacionados con el audio.

Una búsqueda rápida en los buscadores de internet más populares (Ejemplo, Google) arroja como resultado principalmente tiendas muy grandes que no son especializadas y que venden este tipo de accesorios sin ofrecer su instalación, por ejemplo, Gollo o ADNtienda. Los resultados de competidores directos son pocos, y sus páginas web lucen desactualizadas y poco

atractivas en su mayoría, con poca información o con muchos otros productos y servicios. En el apéndice 1 se incluyen algunos de los resultados de dichas búsquedas.

Contar con una aplicación o página web actualizada que permita a sus clientes realizar compras en línea le permitiría a ProLine competir con otras empresas en cuanto a presencia en la web, calidad de servicio y rotación de mercancías. Ayudaría a ProLine a igualar el campo con otras empresas más grandes, en la competencia por lograr la atención de potenciales clientes.

También, la aplicación o página web permitiría mejorar la comunicación entre la empresa y los clientes actuales que realizan compras al mayor al proporcionar un medio de comunicación adicional disponible las 24 horas del día, y mejorar el proceso de compra/venta al poder mostrar los productos disponibles en tiempo real y hacer reservaciones de mercancía sin intervención directa de parte de los colaboradores de la empresa.

1.1.2. Justificación del Proyecto

Desde inicios de 2020 se ha experimentado un aumento masivo de ventas a través del comercio en línea por todo el mundo, debido a las restricciones impuestas por la pandemia de COVID-19 y Costa Rica no es ajena a esto.

De acuerdo con el diario El Financiero, la cantidad de envíos de Correos de Costa Rica creció abruptamente desde que la pandemia inició, en gran medida como consecuencia del aumento en las transacciones de comercio electrónico producidas localmente (Cordero Pérez, 2021) Durante el 2020 realizaron el envío de 1,2 millones de paquetes, 9 de cada 10 desde el inicio de la pandemia. No todos son producto del comercio en línea, pero para comparar, en 2017 el 10% fue por ese motivo, mientras que en 2020 subió a 25%. Esta tendencia continuó en aumento durante el año 2021, y no hay razones para estimar un decrecimiento en los próximos años.

De manera que, para competir y aumentar las ventas, cualquier empresa puede beneficiarse de una página web que permita entregar información actualizada, agradable y fácil de entender. De acuerdo con la revista Forbes, en Costa Rica se registró un crecimiento del 48% de las ventas en línea durante el año 2020, indicando también que el 18,1% de la población costarricense hizo regularmente compras en línea, tendencia que se mantiene en aumento (Forbes Staff, 2021). Además, reseñan que alrededor de apenas el 10% de las empresas registradas como PYMES cuentan con presencia en la web. Esto corresponde con lo mencionado anteriormente acerca de la relativamente poca cantidad de páginas web encontradas en buscadores de empresas del mismo ramo que ProLine.

Un artículo publicado en la Revista Nacional de Administración de la UNED destaca la importancia del comercio electrónico como parte de una buena planificación de negocios, y una página web es uno de los recursos que sirven para llevar a cabo acciones estratégicas que permitan su aplicación (Canossa, 2019). En particular se menciona cómo puede impulsar algunos de los aspectos fundamentales para aumentar el valor de una empresa, los cuales define como: Marca, innovación, comunicación efectiva, conveniencia y garantía.

Con tantos medios de comunicación, o canales, puede ser complicado mantener el control de la comunicación entre la empresa y el cliente, poniendo en peligro la satisfacción del cliente con el servicio brindado. Es allí donde entra en juego el concepto de "Omnicanalidad", que consiste en una estrategia para usar todos los canales de comunicación de una empresa de manera sincronizada e integrada. La información que se recopila en una página o aplicación web y la manera en que conecta varios canales de comunicación puede servir para alcanzar la Omnicanalidad. De acuerdo con el blog de ZenDesk (da Silva, 2021) algunas de las ventajas pueden ser:

- Promover la fidelización de los clientes.
- Aumentar las ventas.
- Mejorar la reputación de la marca.
- Aumentar la facturación.

De acuerdo a proyecciones realizadas por Google, las estrategias omnicanal crecerán 90% entre 2020 y 2025 y representarán un 75% del crecimiento de las ventas en ese periodo (Aramburu y Gómez, 2021). Esto no significa que las tiendas físicas desaparecerán, sino que la manera de comprar está transformándose en una experiencia más amplia que incluye casi en igual medida la compra en línea con la compra en un lugar físico. Google cita también un estudio donde se muestra que al 70% de los consumidores latinoamericanos no les importa dónde comprar, si online u offline, siempre que consigan lo que están buscando.

Por todo lo mencionado anteriormente, queda claro que ProLine se beneficiará de manera considerable al tener una aplicación web que consista en una página web actualizada, con información de sus productos y servicios y la posibilidad de concretar compras en línea. Es de carácter estratégico para la empresa porque aumentará su presencia en la web, tendrá un impacto económico al ayudarla a captar más clientes y al tener un mejor aprovechamiento del tiempo de su recurso humano.

1.2. DEFINICIÓN DEL PROBLEMA

En esta sección se dará una descripción de la problemática actual de la empresa ProLine.

1.2.1 Problemática

A lo largo de la historia se puede observar, a través de muchos ejemplos, que con los avances de las tecnologías las sociedades tienen tendencia a adoptar dichos cambios. Algunas veces los métodos y prácticas que se tienen en un momento dado son desplazados al llegar una innovación

tecnológica, en otros casos son mejorados, y en otros, dichas innovaciones no necesariamente desplazan la tecnología del momento, sino que pueden representar una adición a las opciones disponibles.

En el caso de comunicar e informar, por ejemplo, las sociedades pasaron de depender exclusivamente de la comunicación boca a boca cuando se desarrolló la escritura. Para el caso de la escritura, la invención de la imprenta dio un gran impulso a la misma, abaratando los costos de producción y haciendo posible que grandes masas de población tuvieran acceso a más material escrito, entre otras cosas. Con respecto a lo que la imprenta como innovación significó, Eisenstein, E. (1994) dice lo siguiente:

> Ahora que los libros eran más baratos y abundantes, un estudiante aplicado tenía la posibilidad de abarcar, gracias a la lectura privada, un volumen de materia mayor del que debían dominar o podían desear conocer sus mismos compañeros o, lo que es más, los maestros de la "época pretipográfica". Ya no fue imprescindible convertirse en sabio errante para poder consultar esta o aquella obra. Las siguientes generaciones de sabios sedentarizados ya no fueron tan propensas a absorberse en un único texto y a dedicar todas sus energías a explicarlos con el mayor lujo de detalles. Terminaba la época del glosador y del comentarista y daba comienzo a una nueva época en que se intensificaron las referencias cruzadas entre un libro y otro.

Luego de la invención de la imprenta, algunos otros avances tecnológicos con un gran impacto en la comunicación y difusión de la información fueron: la radio, la televisión, las computadoras, el internet y los teléfonos inteligentes.

Las tecnologías de información y comunicación han tenido mucho impulso en la última década, así como la infraestructura de redes que permiten su uso masivo en la población. Los avances

en dicha tecnología han permitido abaratar costos, de manera que la mayoría de las personas tienen acceso constante a un dispositivo con acceso a internet.

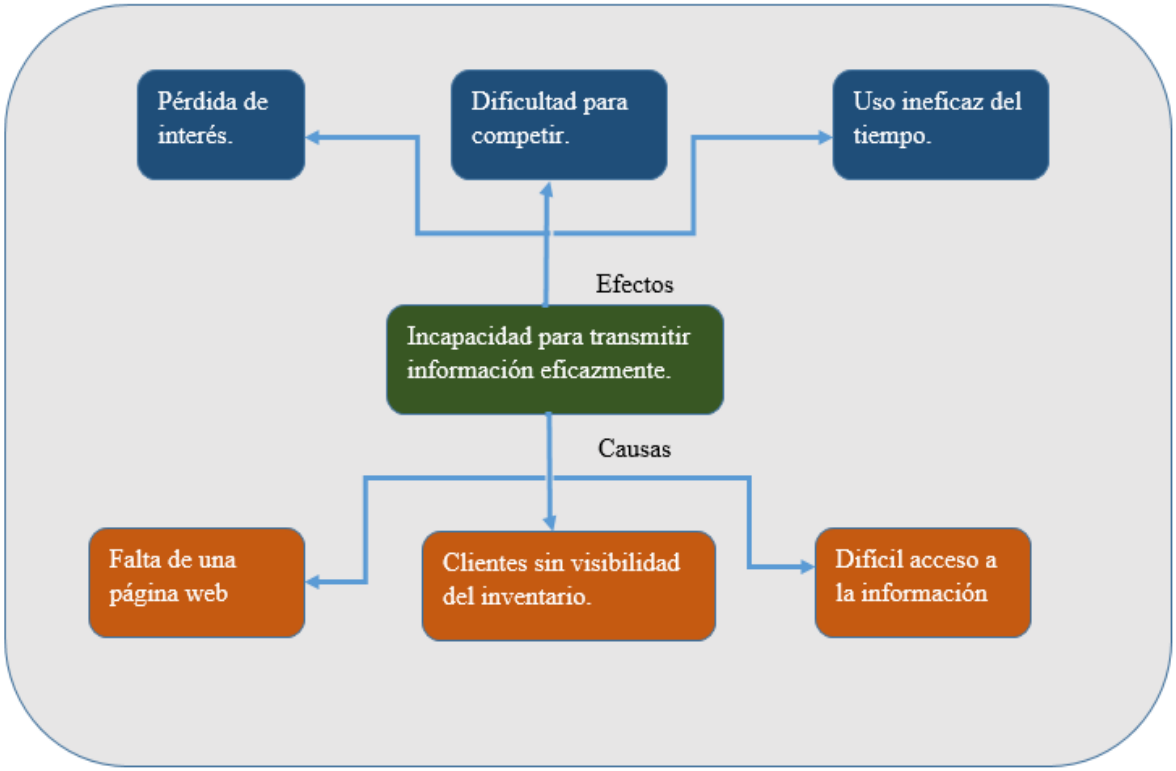
Esto permite difundir y comunicar información de manera más asequible, prácticamente al instante y con alcances no limitados a una ubicación geográfica. Así se han convertido en una herramienta indispensable de las operaciones comerciales para muchas organizaciones, al proveer una ventaja competitiva sobre las empresas que no las usan debido a que aumentan el alcance que tienen para comunicar lo que ofrecen, y llegar a su población objetivo a través del medio que más utilizan actualmente para recibir información.

La empresa ProLine no posee una página web, y actualmente maneja perfiles en redes sociales como Facebook, Instagram y WhatsApp. De esta manera le es difícil comunicar rápida y eficazmente información clave a sus clientes (por ejemplo, acerca de sus productos), ya que se debe escribir individualmente a cada persona que le contacte, repitiendo la misma información y corriendo el riesgo de cometer errores en este proceso. Además, la inversión de tiempo que este nivel de comunicación exige y la necesidad de brindar información personalmente también se dificulta fuera del horario de trabajo. Esto también impacta en otros procesos, como la facturación y la instalación de los productos vendidos, que deben ser interrumpidos constantemente para responder a los mensajes recibidos.

Esto tiene un impacto en los clientes y potenciales clientes, que al no recibir respuestas inmediatas para la información que buscan tienden a desmotivarse, o a continuar buscando información, por lo que eventualmente consiguen productos similares en páginas web de competidores. También ocasiona un desgaste en el personal de la empresa, porque que debe estar constantemente respondiendo mensajes y llamadas para brindar información repetitiva, interrumpiendo otras actividades a las que también deben dedicarse.

Figura 1.

Diagrama Causa-Efecto.



La imagen anterior (Figura 1. Diagrama Causa-Efecto) muestra la problemática encontrada en la empresa ProLine descrita en una frase: incapacidad para transmitir información eficazmente. Las causas de esta problemática son la falta de una página web, la poca visibilidad que los clientes tienen del inventario existente de productos y el difícil acceso que los clientes tienen a la información de precios y horarios, ya que dependen de comunicarse directamente y dentro de los horarios laborales para recibir dicha información.

Los efectos que esto produce son: Pérdida de interés de parte de los clientes, dificultad para competir con otras empresas del sector (especialmente si son más grandes) y un uso menos eficaz del tiempo por parte de los colaboradores de la empresa, pues deben atender múltiples

solicitudes de brindar la misma información una y otra vez limitados por el tiempo de horario laboral y por las varias actividades que deben realizar simultáneamente.

1.2.2 Problema General

¿Cómo implementar una aplicación web que permita la comunicación eficaz y constante con los clientes y potenciales clientes de la empresa ProLine sin necesitar la intervención directa de sus colaboradores?

1.2.3 Problemas Específicos

1.2.3.1. ¿Cuáles son los requerimientos de la empresa ProLine CR para implementar una aplicación web que les permita mostrar información constante de productos y servicios disponibles a sus clientes y potenciales clientes, incluyendo la realización de pedidos en línea, en tiempo real?

1.2.3.2. ¿Cuál será el diseño del software y la base de datos del proyecto para que cumpla con todos los requisitos planteados?

1.2.3.3. ¿Cómo se implementará la aplicación web?

1.2.3.4. ¿De qué manera se verificará y validará el correcto funcionamiento y el cumplimiento de los requerimientos de la aplicación web?

1.2.3.5. ¿Cómo se logrará que los colaboradores de la empresa aprendan a usar la aplicación web, mantengan actualizada su información y colaboren en la introducción de mejoras?

1.3 OBJETIVOS

Los objetivos del proyecto se describen a continuación.

1.3.1 Objetivo General

Entregar una propuesta de software a la empresa ProLine para implementar una aplicación web que se ajuste a sus requerimientos mejorando su comunicación con clientes y potenciales clientes.

1.3.2 Objetivos específicos

1.3.2.1. Determinar los requerimientos de una aplicación web para la empresa ProLine CR con la finalidad de definir las características de la aplicación a implementar.

1.3.2.2. Elaborar el diseño lógico y físico del software y de la base de datos, usando el resultado del análisis de los requisitos recopilados para llevar a cabo una implementación exitosa que satisfaga todos los requisitos planteados.

1.3.2.3. Implementar una aplicación web con base en el diseño hecho según los requisitos recopilados para permitir una mejor comunicación con los clientes y potenciales clientes de ProLine CR.

1.3.2.4. Determinar y ejecutar las pruebas de la aplicación web junto a los usuarios que lo utilizarán, para verificar y validar el correcto funcionamiento de la aplicación, así como el cumplimiento de los requerimientos.

1.3.2.5. Redactar el manual de usuario final llevando a cabo la documentación enfocada al uso de la aplicación web para que los colaboradores de ProLine CR se capaciten en el uso y mantenimiento de dicha aplicación.

1.4 ALCANCE Y LIMITACIONES

A continuación, se muestran los alcances y limitaciones del proyecto.

1.4.1 Alcances del Proyecto

Los alcances del proyecto se definen de la siguiente manera:

- Levantamiento de requerimientos para la aplicación web. Se llevarán a cabo entrevistas con los stakeholders, a partir de las cuales se elaborará la documentación de los casos de uso y las historias de usuario.
- Diseño de la página web: Diagramas de flujo, modelado de la base de datos, diseño lógico, diseño físico y diseño de las interfaces de usuario. Todo enmarcado en el cumplimiento de las necesidades anteriormente descritas por los usuarios.
- Implementación de la página: Llevar a la acción lo diseñado, teniendo reuniones regulares con los stakeholders para mostrar el progreso y realizar ajustes. Al final de esta etapa la aplicación estará funcional. Inicialmente los módulos a implementar (Existe la posibilidad de agregar más módulos de acuerdo con las necesidades de los usuarios) serán los siguientes:
    - Página de inicio: Este módulo mostrará la información general de la empresa incluida la información de contacto, algunos de sus productos y la opción de inicio de sesión para administradores de la página.
    - Módulo de productos: Este módulo mostrará en detalle la información de los productos excepto las cantidades disponibles, que estarán visibles solo para los usuarios administradores.
    - Control de usuarios: Este módulo es de acceso para los administradores de la página. Será para la creación y edición de los usuarios registrados como administradores.
    - Mantenimiento del catálogo de productos: Este módulo será solo para los administradores de la página. En él podrán agregar, editar y eliminar los productos para mostrar a los usuarios de la página.

    - Módulo de carro de compras: Este módulo es para que los usuarios puedan verificar y confirmar sus pedidos.
    - Módulo de pedidos: Este módulo es para que los administradores de la página puedan observar los pedidos que los usuarios han realizado, darlos por completados o cancelarlos según sea el caso.
- Ejecución de pruebas para verificar y validar el correcto funcionamiento de la aplicación web, así como el cumplimiento de los requerimientos.
- Elaboración y entrega de manual para usuario final con todas las instrucciones de uso de la aplicación web.

1.4.2 Limitaciones del Proyecto

No se cuenta con un presupuesto que permita el uso de herramientas de pago para servicios especializados de diseño de páginas web o el uso de plantillas muy estilizadas, por lo cual todos los recursos a usar serán de uso gratuito o donados, con la excepción de la adquisición del nombre de dominio. Por este motivo, además, se contará con una base de datos de pequeño tamaño, por lo cual existirá un límite de información que se pueden colocar para mostrar, así como de otros recursos multimedia como videos, fotografías y artículos informativos.

1.5 CRONOGRAMA DE ACTIVIDADES

La tabla a continuación (Tabla 1. Cronograma de actividades) muestra el cronograma de actividades, indicando el nombre de cada actividad, la fecha de inicio y finalización, así como la cantidad de días que cada actividad tomará para su realización.

Tabla 1.

Cronograma de actividades.

Actividad	Fecha de inicio	Fecha de finalización	Cantidad de días
Capítulo I. Problema del Proyecto	13/6/2022	20/6/2022	8
Antecedentes y justificación del problema	13/6/2022	14/6/2022	2
Definición del problema	15/6/2022	16/6/2022	2
Objetivos del Proyecto	17/6/2022	18/6/2022	2
Alcance y limitaciones	19/6/2022	20/6/2022	2
Cronograma	20/6/2022	20/6/2022	1
Capítulo II. Marco Teórico	21/6/2022	6/7/2022	23
Metodología	21/6/2022	23/6/2022	3
Métodos para levantamiento de requerimientos	23/6/2022	26/6/2022	3
Técnicas para diseño lógico y físico de software y bases de datos	26/6/2022	28/6/2022	3
Prácticas para implementar software para e-commerce	28/6/2022	30/6/2022	3
Técnicas para probar y validar software	1/7/2022	3/7/2022	3
Modelos para elaboración de manuales de usuario	4/7/2022	6/7/2022	3
Capítulo III. Marco Metodológico	7/7/2022	13/7/2022	7
Tipo de investigación	7/7/2022	8/7/2022	2
Fuentes de información	8/7/2022	9/7/2022	2

Técnicas y herramientas de recolección de datos	9/7/2022	10/7/2022	2
Variables	10/7/2022	11/7/2022	2
Diseño de la investigación	11/7/2022	12/7/2022	2
Matriz de coherencia	12/7/2022	13/7/2022	2
Capítulo IV. Diagnóstico	14/7/2022	19/7/2022	6
Descripción de la situación actual	14/7/2022	16/7/2022	3
Recolección y análisis de datos	16/7/2022	19/7/2022	4
Capitulo V. Diseño e implementación del Proyecto	20/7/2022	12/10/2022	85
Diseño de la propuesta	20/7/2022	25/7/2022	6
Implementación del Proyecto	26/7/2022	1/10/2022	68
Pruebas del Proyecto	2/10/2022	5/10/2022	4
Documentación del Proyecto	6/10/2022	12/10/2022	7
Capítulo VI. Conclusiones y recomendaciones	13/10/2022	15/10/2022	3
Conclusiones	13/10/2022	14/10/2022	2
Recomendaciones	15/10/2022	15/10/2022	1

La tabla a continuación (Tabla 2. Cronograma del diseño e implementación) muestra el cronograma del diseño e implementación del Proyecto. Se da nombre a las diferentes fases del proyecto, las subfases dentro de las fases, las actividades a realizarse dentro de las subfases, en qué semana inician y terminan dichas actividades (escrito en texto y también indicado en un diagrama). En total son 12 semanas, de conformidad con las fechas indicadas en el cronograma de actividades.

Tabla 2.

Cronograma de diseño e implementación.

Fase	Sub-Fase	Actividades	Semanas											
			1	2	3	4	5	6	7	8	9	10	11	12
1. Planificación y análisis del sistema	1.1 Recopilación de requerimientos	1.1.1 Entrevistas												
		1.1.2 Análisis de requerimientos												
	1.2 Registro de requerimientos	1.2.1 Validación de requerimientos												
		1.2.2 Documentación de requerimientos												
2. Diseño del sistema	2.1 Diseño lógico del sistema	2.1.1 Elaboración de diagramas de casos de uso												
		2.1.2 Elaboración de diagramas de secuencias de casos de uso												
		2.2.1 Definición de módulos del sistema												
	2.2 Diseño físico del sistema	2.2.1 Plataforma donde se implementará												
		2.2.2 Modelaje de la base de datos												
3. Implementación	3.1 Desarrollo de la base de datos	3.1.1 Construcción de la base de datos												
	3.2 Implementación del software	3.2.1 Implementación de los módulos del sistema												
4. Pruebas del sistema y manual de usuario	4.1 Pruebas del sistema	4.1.1 Elaboración del documento de pruebas del sistema												
		4.1.2 Ejecución de las pruebas del sistema												
	4.2 Manual de usuario	4.2.1 Elaboración del manual de usuario												

CAPÍTULO II: MARCO TEÓRICO

En este capítulo se da cuenta del sustento teórico para la manera en que se desarrollará el proyecto, definiendo conceptos básicos relacionados con la ingeniería de software y la solución de software a implementarse. Se dividirá en 7 partes: Metodología, métodos para levantamiento de requerimientos y análisis, diseño, e-commerce, desarrollo, técnicas para probar y validar software, y el soporte de software.

2.1 METODOLOGÍA

La elaboración de software no es muy diferente a la de cualquier producto que se desee elaborar, desde cierto punto de vista. Si se quiere entregar un producto de calidad, debe seguirse una serie de pasos para asegurar la rentabilidad de un producto que satisfaga (o supere) las expectativas del consumidor. Y si se quiere entregar ese mismo producto muchas veces, esa serie de pasos debe ser secuencial y de efectividad comprobada. Para lograr esto se han introducido la ciencia y el método científico a muchos campos de producción, creándose así disciplinas de ingeniería aplicada a dichos campos. Igualmente, en el desarrollo de software se ha presentado también la necesidad de aplicar la ingeniería, creándose así la ingeniería de software.

Según Blum (1996), “la ingeniería de software es la aplicación de herramientas, métodos y disciplinas para la producción y mantenimiento de soluciones automáticas a problemas del mundo real”. Aplicando la ingeniería de software, se definen las herramientas, los métodos y el proceso con los cuales se desarrolla la solución de software, con un enfoque en la calidad y la posibilidad de mejora continua. Cada uno de estos elementos se definirán a continuación, iniciando por la metodología a usarse para el desarrollo de este proyecto.

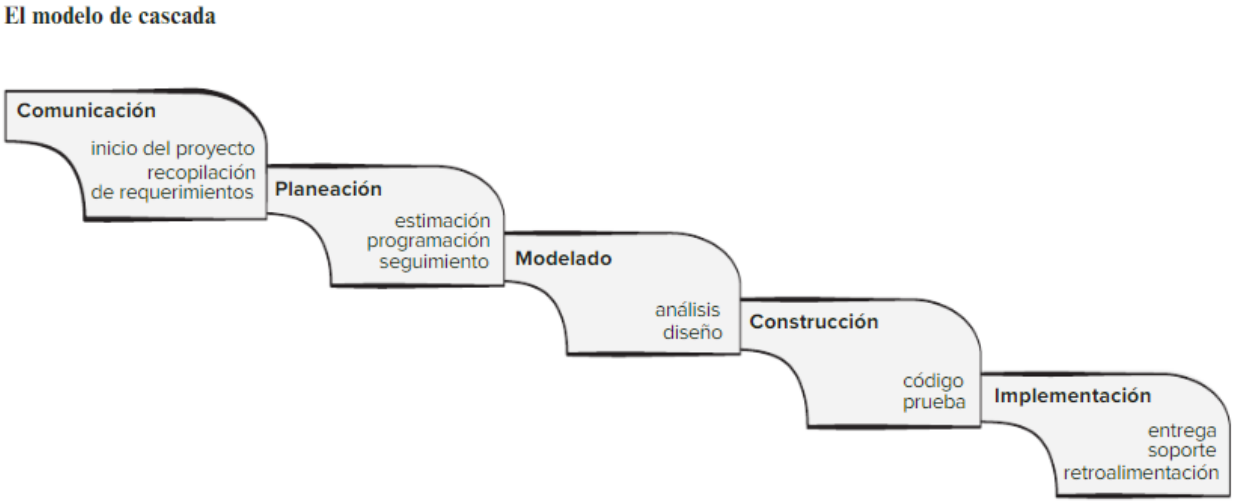
2.1.1 Metodología en cascada

El uso de una metodología para el desarrollo de software permite definir "un conjunto predefinido de elementos del proceso y un flujo de trabajo del proceso predecible" (Maxim y Pressman (2021, p. 25). De esta manera se tienen un orden y una serie de pasos secuenciales.

La metodología en cascada define las diferentes fases del proyecto y éstas se llevan a cabo una tras otra sin dar vuelta atrás. Según Maxim y Pressman (2021, p.25) este modelo es ideal cuando se conocen muy bien los requerimientos y éstos son razonablemente estables. Es por este motivo que se usará dicha metodología en este proyecto. Como se aprecia en la imagen a continuación (Figura 2) ellos definen las siguientes etapas de este modelo: Comunicación, planeación, modelado, construcción e implementación.

Figura 2.

El modelo de cascada.



Fuente: Maxim y Pressman (2021, p.26)

Hilard (2020) define siete etapas en este modelo:

- Análisis (análisis de las necesidades).
- Diseño.

- Planificación (cargas, presupuesto, plazos).
- Realización.
- Verificación (test y pruebas).
- Puesta en producción.
- Mantenimiento.

Para los fines de este proyecto, se usará el modelo propuesto por Maxim y Pressman, debido a que es una simplificación del modelo propuesto por Hilard.

La etapa de comunicación es la que da inicio al proyecto. En ésta se lleva a cabo la recopilación de requerimientos por parte del o los clientes. Se conversa con las partes interesadas por medio de reuniones, entrevistas o medios de comunicación escrita, para entender sus necesidades y delimitar el alcance del proyecto, así como algunas normas.

Le sigue la etapa de planeación, en la cual se proyecta el costo que el proyecto tendrá, se establece una línea de tiempo y un cronograma de actividades. Se establecen los roles necesarios y quiénes los ocuparán.

Luego viene la etapa de modelado, en la cual se validan los requerimientos de los usuarios a través del análisis de los mismos. Se elabora la documentación correspondiente a partir de los mismos y se diseñan los aspectos visuales del producto final. Se elaboran diagramas para la rápida comprensión de los requerimientos, sirviendo como guía a medida que se avance en el proyecto. En esta etapa se define también el modelo de bases de datos a ser utilizado, con las entidades y relaciones correspondientes (En caso de ser necesaria una base de datos con un modelo entidad-relación).

En la etapa de construcción se desarrolla la solución como tal, a través de la elaboración de código y la implementación de software a través de las herramientas escogidas, siguiendo las

indicaciones y el cronograma establecido en las etapas anteriores. Se realizan pruebas para encontrar fallos y resolverlos antes de la puesta en producción, así como para verificar que los requerimientos sean cumplidos.

La última etapa es la implementación, en la cual se entrega el producto, ya funcional, al cliente. En general, la vida útil de la solución transcurrirá en esta etapa, en la cual se da soporte para garantizar un buen funcionamiento. Se recibe retroalimentación para introducir mejoras y correcciones.

El motivo por el cual se ha escogido esta metodología es porque el proyecto tiene objetivos muy bien definidos en los cuales se contempla una corta duración, que serán desarrollados por una sola persona por lo cual no se necesitará de una coordinación con un equipo de trabajo. Además de tratarse de una metodología sencilla de aplicar.

Debido a la naturaleza de secuencia lineal de esta metodología, la fase más importante es la de diseño porque es la que define el resto del trabajo, basándose en el previo análisis de las necesidades descritas. El proyecto se entregará únicamente hasta haber completado todas las demás fases del mismo. Para los fines de este proyecto y de acuerdo con su alcance, no se llegará a las fases de puesta en producción y mantenimiento, por tratarse de una propuesta de implementación.

2.1.2 Metodologías ágiles.

Las metodologías ágiles para el desarrollo de software son un conjunto de alternativas a la metodología en cascada y otras derivadas. Una parte importante de su definición se dio con el Manifiesto por el Desarrollo Ágil de Software, conocido también en inglés como “The Agile Manifesto”. Éste consiste en un documento público, en el cual un grupo de desarrolladores de software emite una declaración afirmando estar “descubriendo formas mejores de desarrollar

software tanto por nuestra propia experiencia como ayudando a terceros.” (Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... Thomas, D. (2001).

El Manifiesto por el Desarrollo Ágil de Software se encuentra en su totalidad en el anexo 1. Los principios allí declarados son las bases para las metodologías ágiles que han surgido desde entonces. De acuerdo con Anand, R. V., & Dinakaran, M. (2016), entre las más populares se encuentran:

- Scrum.
- Kanban.
- Extreme Programming.
- Lean Software Development.
- Crystal.

Anand, R. V., & Dinakaran, M. (2016) también comentan que la metodología Scrum es de las más populares, y específicamente es muy utilizada para el desarrollo de sistemas y productos complejos. A continuación, se da una descripción básica de la metodología Scrum. Para una descripción exhaustiva, véase el Anexo 2 (La Guía de Scrum).

2.1.2.1. Metodología Scrum.

La metodología Scrum fue desarrollada inicialmente por Ken Schwaber y Jeff Sutherland, quienes publicaron una guía oficial en el año 2010, que se encuentra en constante actualización y se puede encontrar de manera gratuita en su sitio web oficial, scrum.org (Así como otros recursos de ayuda para el uso de dicha metodología).

De acuerdo con Schwaber y Sutherland (2020), se define Scrum como “un marco de trabajo liviano que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptativas para problemas complejos.”. En este marco de trabajo se definen algunos roles:

- Scrum Master: Responsable de la aplicación de la guía de Scrum. Ayuda a comprender los principios allí descritos, y es el principal responsable de la efectividad del equipo. Bajo su liderazgo se guía a los demás para la adopción y adherencia a la metodología Scrum.
- Product Owner: Es el principal responsable de la definición del Product Backlog, en el cual se encuentra una lista de lo que se necesita hacer para mejorar el producto, así como el Objetivo del Producto. El Product Owner es el principal responsable de que el trabajo se realice, ejerciendo su liderazgo directamente sobre el Scrum Team.
- Scrum Team: Equipo compuesto por un Scrum Master, un Product Owner, y varios Developers, en el cual no hay jerarquías ni otras subdivisiones. Todos sus miembros poseen las habilidades necesarias para añadir valor al producto, y son autogestionados. El Scrum Team es el encargado de desarrollo del producto en su totalidad, trabajando de acuerdo con el Product Backlog, y en una serie de tareas definidas que aportan valor incremental, llamadas Sprints. Suelen componerse como máximo de 10 personas y puede haber más de un Scrum Team de ser necesario, pero siempre con el mismo Product Owner.
- Developers: Los desarrolladores. Forman parte del Scrum Team, y trabajan aplicando los valores de Scrum: Compromiso, Foco, Franqueza, Respeto y Coraje. Son profesionales capaces de llevar a cabo las labores de desarrollo, pero con capacidades de planificación, comunicación y análisis, lo que los convierte en profesionales integrales.

Scrum se compone de eventos iterativos, que van marcando el progreso de un proyecto. Cada evento tiene un propósito específico, y se enlistan a continuación:

- Sprint: Eventos consistentes de duración fija (un mes o menos). Funcionan como contenedores para todos los demás eventos, que ocurren siempre dentro de un Sprint. Dentro de un Sprint se busca alcanzar el Objetivo del Sprint, y la consecución de los Objetivos de los Sprints busca lograr el Objetivo del Producto. Siempre se comienza un Sprint inmediatamente al terminar el anterior.
- Sprint Planning: Da inicio al Sprint, y define el trabajo a realizar al responder a las preguntas: ¿Por qué?, ¿Qué? y ¿Cómo?. Se lleva a cabo de manera colaborativa por el Scrum Team, bajo la dirección del Product Owner. Puede llegar a durar como máximo ocho horas, y da como resultado un Sprint Backlog que se compone de: el Objetivo del Sprint, los elementos del Product Backlog seleccionados y el plan para entregarlos.
- Daily Scrum: Evento diario de unos 15 minutos de duración. En él, el Scrum Team inspecciona y planifica el trabajo diario. Sin embargo, de ser necesario, en cualquier otro momento del día se pueden realizar ajustes a lo planificado.
- Sprint Review: Es el penúltimo evento del Sprint, de máximo cuatro horas de duración. Consiste en una sesión de trabajo en la cual el Sprint Team y otros interesados revisan el resultado del Sprint y el progreso hacia el Objetivo del Producto, analizan los cambios del entorno y pueden hacer ajustes al Product Backlog.
- Sprint Retrospective: Sirve para analizar todo lo ocurrido durante el Sprint e incorporar cambios para el siguiente, que sirvan para aumentar la calidad y efectividad. Este evento da fin al Sprint, y tiene una duración máxima de tres horas.

Además, Scrum posee tres artefactos, y "representan trabajo o valor. Están diseñados para maximizar la transparencia de la información clave." (Schwaber y Sutherland. 2020). Cada uno

contiene un compromiso, con el cual se busca medir el progreso de manera transparente y manteniendo el enfoque. Se describen a continuación:

- Product Backlog: Contiene el Objetivo del Producto, que a su vez es el compromiso, y describe un estado futuro del producto. Contiene además una lista de lo que se necesita hacer para elaborar y mejorar el producto. Cada elemento puede tener una descripción, un orden y tamaño, entre otros detalles, que pueden ser refinados a lo largo del proceso. Estos elementos se seleccionan durante los eventos de Sprint Planning, y se van marcando como Terminados cuando se cumplen los requisitos establecidos.
- Sprint Backlog: Contiene el Objetivo del Sprint (El por qué), un conjunto de elementos seleccionados del Product Backlog (Qué), y un plan de acción para entregar un Increment (Cómo). Se puede actualizar a medida que avanza el Sprint, sin modificar nunca el Objetivo del Sprint, ya que éste representa el compromiso del mismo.
- Increment: Representa un avance hacia la consecución del Objetivo del Producto y siempre debe ser utilizable para agregar valor. Un Increment se suma a los Increments anteriores y es verificado para asegurarse que todos los Increments funcionen juntos. Puede crearse más de un Increment dentro de un Sprint, y lo que determina si un trabajo realizado se considera o no como un Increment es que su estatus se marque como Terminado, que es el compromiso del Increment. La Definición de Terminado es una descripción de lo que se debe cumplir y se compone de medidas de calidad y de otros estándares establecidos por la organización o en su defecto, por el Scrum Team.

La metodología Scrum es ampliamente utilizada no solamente en procesos de ingeniería de software, sino también en otras disciplinas.

2.2 MÉTODOS PARA LEVANTAMIENTO DE REQUERIMIENTOS Y ANÁLISIS

Antes de iniciar la realización del proyecto, es importante comprender los requerimientos que el cliente tiene, para poder satisfacer sus necesidades y evitar realizar algo que no tenga el impacto deseado y por tanto represente un costo con poco beneficio. En esta etapa del proyecto suelen participar los ingenieros de software, gerentes, analistas e incluso otras partes interesadas como los mismos clientes y usuarios finales del software. Éstos últimos son realmente quienes marcan la pauta con respecto a los requerimientos, pues son quienes tendrán interacción con el software a realizar y conocen sus necesidades mejor que nadie. Las interacciones con ellos se conocen como "Negociación", y a partir de la información que ellos brindan se elaboran las "Historias de usuario".

La información recolectada en las interacciones con las personas anteriormente mencionadas es analizada y convertida en diagramas que permitan crear una ruta de acción clara, con objetivos y metas definidos dentro de un marco de tiempo, de manera que el flujo de trabajo pueda llevarse ordenadamente y evitar salirse de los requerimientos establecidos.

2.2.1 Historias de usuario o escenarios de casos de uso.

Larman (2004) Define como "Actores" aquello que tiene comportamiento dentro del sistema, como las personas que harán uso del sistema (Definidas por un rol) y otros sistemas. Larman (2004) define también como "Escenario" una secuencia de acciones e interacciones específicas entre los actores y el sistema. La historia de usuario es un relato de parte del usuario que describe entonces lo que éste desea que ocurra como resultado de su interacción con el sistema. Aunque es en sí de un formato libre, suele ser importante definir algunos elementos esenciales en la historia de usuario.

Cockburn (2001) sugiere una plantilla para la definición de las historias de usuario (También llamados “Escenarios de caso de uso”) con los siguientes elementos (a la izquierda, el nombre del elemento; a la derecha, una breve descripción del mismo) mostrados en la tabla a continuación:

Tabla 3.

Plantilla para la definición de historias de usuario.

Número	Se da un número al caso de uso.
Caso de uso	Se da un nombre al caso de uso.
Actor principal	Se define quién interactúa para lograr la función requerida.
Meta en contexto	Lo que el actor principal quiere lograr.
Precondiciones	Lo que debe ser cierto antes de iniciar el caso de uso.
Desencadenante	Lo que motiva al actor principal a iniciar el caso de uso.
Escenario	La secuencia de acciones que ocurren de manera ideal.
Excepciones	La secuencia de acciones que ocurren cuando no se cumple el escenario ideal.
Prioridad	Importancia que tiene el caso de uso para el proyecto.
Disponibilidad	Momento durante las fases de implementación o desarrollo en que estará disponible.
Frecuencia de uso	Con qué frecuencia los usuarios harán uso del caso de uso indicado.

Canal para el actor	Medio por el cual el actor interactúa con el sistema.

Fuente: Cockburn (2001).

2.2.2 Diagrama de casos de uso

Con las historias de usuario o escenarios de casos de uso listos, se comienza un proceso de diagramación para facilitar la comprensión de los requerimientos. Generalmente se usa un modelo llamado UML para la elaboración de diagramas. Pressman y Maxim (2021) explican lo siguiente: “UML, el lenguaje de modelado unificado se desarrolló para apoyar su trabajo. El UML contiene una notación robusta para el modelado y desarrollo de sistemas orientados a objetos, y se ha convertido en un estándar de facto en la industria para modelar software de todo tipo.”

Pressman y Maxim (2021) explican que un diagrama de casos de uso en UML incluye todos los casos de uso, más no las descripciones de éstos o sus detalles, así como todos los actores y sus relaciones con los casos de uso. Los elementos de un diagrama de casos de uso se simbolizan de la siguiente manera:

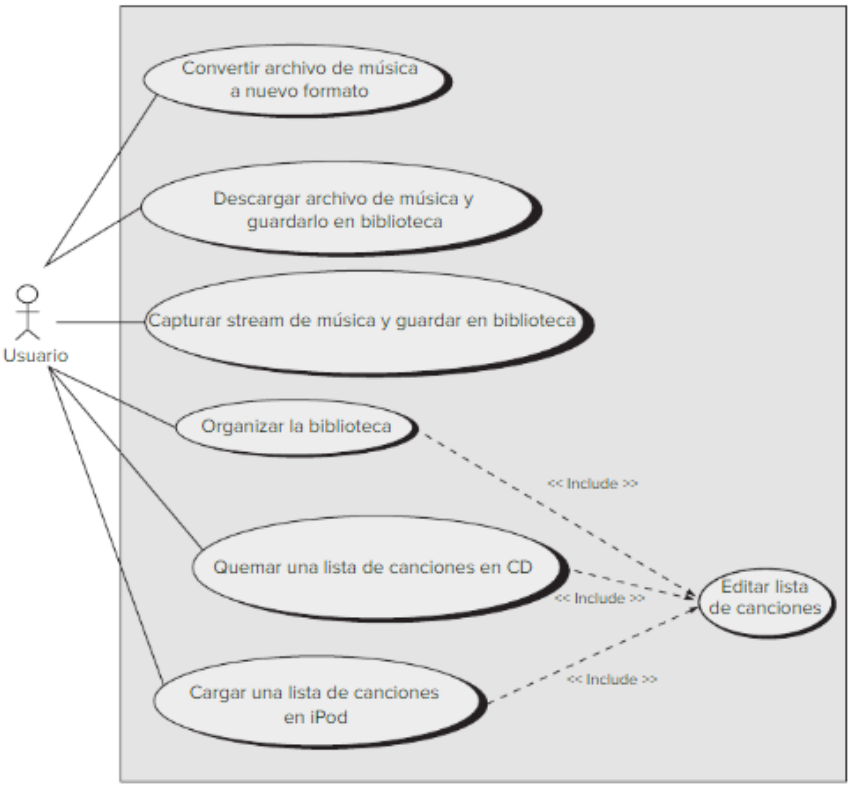
- Actor: Una figura humana hecha con un círculo representando la cabeza y varios palitos representando el tronco y las extremidades.
- Sistema: Un rectángulo vertical dentro del cual se colocan los casos de uso. Los actores van fuera de él. De esta manera se representan visualmente los límites del sistema y que los actores están fuera de él.
- Caso de uso: Se representa con un óvalo para cada uno. Solo lleva el nombre del caso de uso.

- Línea de conexión sólida: Se usa una línea que conecta a los actores y los casos de uso con los que interactúa. Lleva una flecha en el extremo que apunta hacia el óvalo del caso de uso.
- Línea de conexión punteada: Esta línea se usa para conectar casos de uso que están relacionados entre sí. Por ejemplo, cuando hay una acción que se realiza en varios casos de uso, se puede incluir como un caso de uso en sí misma, y se usa la línea punteada con la palabra “Include” para indicar qué casos de uso incluyen dicha acción.

La imagen a continuación (Figura 3. Diagrama de caso de uso) muestra un ejemplo de un diagrama UML de casos de uso. Se aprecia un usuario (Representado con un círculo y líneas rectas para representar cabeza, tronco y extremidades) y líneas rectas conectando al usuario a óvalos que se encuentran dentro de un rectángulo. Cada uno de los óvalos es un caso de uso. Se aprecia cómo tres de esos casos de uso se conectan a otro caso de uso a través de una línea recta punteada, ya que es una serie de acciones que se repiten en esos tres casos de uso.

Figura 3.

Diagrama de caso de uso.



Fuente: Pressman y Maxim (2021)

2.2.3 Análisis

Después de contar con todos los requerimientos ya documentados, éstos se deben revisar en búsqueda de inconsistencias, ambigüedades o faltantes para resolverlos preferiblemente antes de continuar con la siguiente fase. Es posible que durante las otras fases sea necesario entablar conversaciones nuevamente con los actores, para esclarecer asuntos que no hayan sido detectados en el análisis de requerimientos.

Una de las maneras más comunes de clasificar los requerimientos es identificarlos como uno de dos tipos: funcionales y no funcionales. De acuerdo con Sommerville (2011) los requerimientos funcionales son declaraciones del servicio que el sistema debe proveer, cómo el sistema debe reaccionar ante la recepción de datos en particular, y cómo el sistema debe comportarse en

situaciones determinadas. En algunos casos, incluso se declara lo que el sistema no debe hacer ante determinadas situaciones.

Sommerville (2011) también indica que los requerimientos no funcionales son restricciones en los servicios o funciones ofrecidos por el sistema. Esto incluye restricciones de tiempo, del proceso de desarrollo y otras restricciones impuestas por estándares. Generalmente las restricciones no funcionales se aplican al sistema como un todo, en vez de a una función o servicio específico.

Es muy importante mencionar que la inclusión de algunos requerimientos puede significar agregar otros requerimientos adicionales que no se habían previsto, para garantizar su cumplimiento.

Según Pressman y Maxim (2021) se debe realizar una validación de los requerimientos para asegurarse que éstos estén "sin ambigüedades; que se hayan detectado y corregido las inconsistencias, omisiones y errores, y que los productos de trabajo se conformen a las normas establecidas para el proceso, el proyecto y el producto.". Hay varias maneras de lograr esto, de acuerdo con Sommerville (2011) entre las que se encuentran:

- Revisión de requerimientos: Los requerimientos son analizados sistemáticamente por parte de un panel de revisores en búsqueda de errores e inconsistencias.
- Prototipado: En este tipo de validación, un modelo ejecutable del sistema en cuestión es demostrado a los usuarios finales y clientes. Así ellos pueden experimentar con el modelo para verificar si cumple con sus requerimientos.
- Generación de prueba-caso: Los requerimientos deben ser susceptibles a ser probados. En dichas pruebas pueden revelarse problemas en los requerimientos. Cuando son

puestos a prueba, si la misma resulta imposible o muy difícil de diseñar, significa que el requerimiento será muy difícil de implementar por lo cual debería ser reconsiderado.

2.3 DISEÑO

A la etapa de análisis le sigue la etapa de diseño. Es de suma importancia, porque toma los resultados del análisis y los transforma en el insumo a ser utilizado para la construcción o elaboración de la solución de software. En la mayoría de los casos hace uso de diagramas para mostrar la información ya analizada de una manera más fácilmente entendible. Todas las características del sistema se deben diseñar, así como la manera en que se muestran al usuario final.

Usualmente puede dividirse en dos partes: El diseño lógico y el diseño físico. El diseño lógico consiste en representaciones abstractas del software para definir sus funcionalidades, componentes y relaciones. El diseño físico toma los elementos del diseño lógico y los traduce en acciones a realizar para lograr la implementación del software.

Hay muchos elementos a considerar en esta etapa, de los cuales se describirán solo algunos.

2.3.1 Diagrama de secuencia

Un diagrama de secuencia muestra las interacciones entre los actores y el sistema y sus componentes. Se realiza con UML, y muestra la secuencia de interacciones de un caso de uso. Sommerville (2011) explica los elementos de este diagrama:

En la parte superior izquierda se indican los objetos y los actores, y a partir de allí se dibujan líneas punteadas horizontalmente. Las interacciones entre los objetos se indican con flechas. Los rectángulos con las líneas punteadas indican el tiempo de vida de un objeto. Las interacciones se leen de arriba abajo y de izquierda a derecha. Las anotaciones en las flechas

indican las llamadas a objetos, sus parámetros y sus valores de retorno. También es posible incluir dentro de un recuadro posibles escenarios alternativos.

La Figura 4, diagrama de secuencia, muestra un ejemplo de lo explicado por Sommerville (2011).

La recepcionista médica activa el método ViewInfo en una instancia P de la clase de objeto PatientInfo, que proporciona el identificador del paciente, PID. P es un objeto de interfaz de usuario, que se muestra como un formulario que muestra información del paciente.

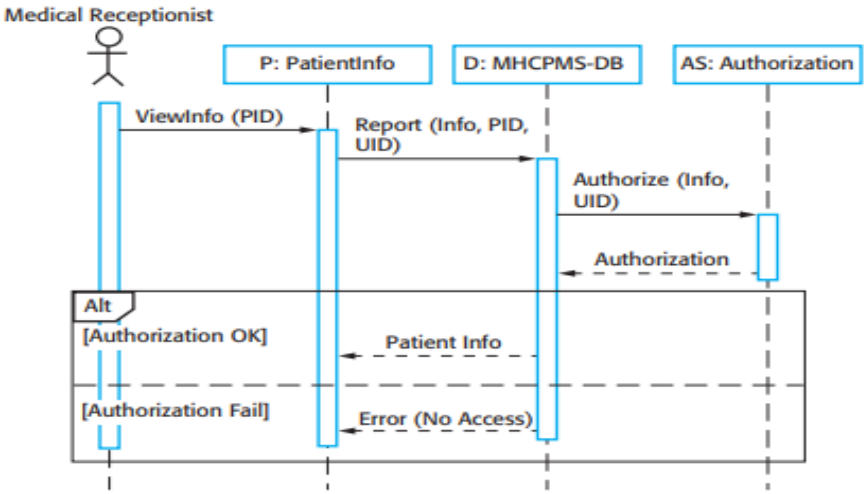
La instancia P llama a la base de datos para devolver la información requerida, proporcionando el identificador de la recepcionista para permitir el control de seguridad (en esta etapa, no importa de dónde viene este UID).

La base de datos comprueba con un sistema de autorización que el usuario está autorizado para esta acción.

Si está autorizado, se devuelve la información del paciente y un formulario en la pantalla del usuario se rellena. Si falla la autorización, se devuelve un mensaje de error.

Figura 4.

Diagrama de secuencia.



Fuente: Sommerville (2011).

2.3.2 Modelado basado en eventos

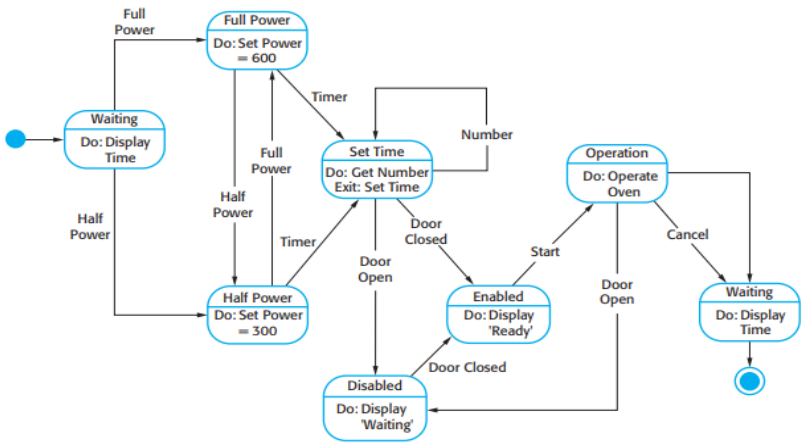
El modelado basado en eventos muestra cómo un sistema responde a eventos internos y externos. Asume que el sistema tiene una cantidad limitada de estados y que los eventos son la causa de la transición entre un estado y otro. No se muestra el flujo de datos entre un estado y otro, pero pueden incluir información de los procesos realizados en cada estado. (Sommerville, 2021).

Usando UML, los estados se representan en cuadros con bordes redondeados. Pueden incluir una breve descripción de las acciones que se toman en dicho estado. Cada estado se conecta con un estímulo o evento, indicado con una flecha, y que desencadena el paso de un estado a otro. El estado inicial y el estado final pueden indicarse con círculos.

En la Figura 5 (Modelado basado en eventos) se muestra un ejemplo de lo descrito anteriormente. Se describe el funcionamiento de un microondas muy sencillo. Inicia en el estado de espera, en el cual se muestra en una pantalla la hora. Se está a la espera de que se seleccione potencia completa o mitad de potencia, estados en los cuales se establece una potencia de 600 o de 300, respectivamente. En ambos eventos se puede pasar de uno al otro, y se está a la espera de que se determine luego el temporizador, cuya entrada es un número marcado en un panel numérico, y con este número se establece el tiempo de funcionamiento en segundos. Si la puerta está abierta, el funcionamiento no se inicia (Por razones de seguridad). Si la puerta está cerrada, se indica que el microondas está listo, tras lo cual inicia su operación. En este estado, el microondas está funcionando, y está a la espera de tres eventos que pueden detener el funcionamiento: Que se abra la puerta, que se presione el botón de cancelar o que el temporizador llegue a 0. Al final, llega nuevamente al estado inicial de espera.

Figura 5.

Modelo basado en eventos.



Fuente: Sommerville (2011).

2.3.3 Diseño de interfaces de usuario

La experiencia del usuario (Conocida también como UX) es en general toda interacción que el usuario tiene con el sistema. Se debe ir diseñando desde las primeras etapas del ciclo de vida del proyecto y no hasta el final, para asegurar el éxito del mismo. Se toman en cuenta todos los elementos con los que el usuario interactuará de manera directa, las entradas y salidas de información, la usabilidad, el diseño visual y la apariencia del sistema.

Las interfaces de usuario son el espacio visual en el cual el usuario interactúa con el sistema. Es lo que se le muestra en pantalla. De acuerdo con Pressman y Maxim (2021) “comienza con el diseño de la pantalla y procede hacia una consideración de los esquemas de color globales; tipos de fuente, tamaños y estilos; el uso de medios suplementarios (como audio, video y animación) y todos los demás elementos estéticos”.

Pressman y Maxim (2021) mencionan tres reglas de oro a seguir durante el diseño de interfaces para garantizar una óptima experiencia del usuario:

1. Dejar al usuario el control.

2. Reducir la carga de memoria del usuario.
3. Hacer la interfaz consistente.

Con respecto a la primera regla, dejar al usuario el control, ésta consiste en varias cosas, como dar al usuario varias opciones para conseguir un resultado, de acuerdo con sus preferencias. Por ejemplo, para llevar a cabo una acción, el usuario podría usar un comando del teclado, o presionar un botón haciendo clic en él. También consiste en permitir al usuario deshacer sus acciones e interrumpir lo que está haciendo; en no mostrar información innecesaria como detalles técnicos; y en dar al usuario una interacción directa con los elementos en pantalla.

La segunda regla, reducir la carga de memoria del usuario, afirma que mientras menos se exija al usuario recordar un proceso, menos propenso a errores será el sistema (Pressman y Maxim, 2021). Para lograr esta reducción en la carga de memoria del usuario, se sugiere proveerle pautas visuales que permitan reconocer acciones pasadas; establecer valores predeterminados con sentido para el usuario promedio; usar metáforas del mundo real en los elementos de las interfaces para facilitar la asociación de los elementos que forman parte de un proceso; si se incluyen comandos del teclado para llevar a cabo acciones, dichos comandos deben resultar intuitivos (Como usar la primera letra de la palabra cuya acción realiza el comando); y la información debe presentarse de manera progresiva y jerárquica, mostrando detalles de un elemento sólo hasta después que el usuario muestre interés en el mismo.

La tercera regla, hacer la interfaz consistente, trata principalmente en mantener el estilo visual de los elementos en todas las interfaces, respetando sus reglas de diseño. Los elementos en las interfaces deben tener descripciones que ayuden al usuario a identificar las transiciones que sus acciones pueden provocar. También se deben mantener modelos interactivos que existan en versiones anteriores, en la medida de lo posible, para evitar confusiones. Al cumplir con esta

regla de oro, los usuarios pueden reconocer cada página como parte de un conjunto en común dentro del sitio web.

Además de estas tres reglas de oro, existen prácticas recomendadas para lograr que las páginas web sean atractivas visualmente mientras cumplen sus objetivos. Beaird, J. (2007) indica que usualmente hay dos puntos de vista al momento de realizar un diseño considerado bueno: el enfoque en la funcionalidad y el enfoque en la estética. Él agrega que debe evitarse ir a los extremos en estos enfoques, y que se deben maximizar ambos enfoques para llegar a las personas y mantener su interés.

Una página o aplicación web puede tener muchos diferentes elementos, arreglados de muchas maneras diferentes. Beaird, J. (2007) propone una anatomía de una página web, compuesta de una serie de elementos que no deben faltar para un buen diseño visual, los cuales se describen a continuación:

- Bloques contenedores: Sirven para colocar y organizar el contenido de la página web, permitiendo que dicho contenido se adapte a la pantalla del dispositivo que se utiliza para visualizarlo. Algunos ejemplos de contenedores dentro de HTML son las etiquetas de body, de div, o inclusive una tabla.
- Logotipo: El logotipo de la empresa debe estar visible constantemente en todos sus materiales, incluyendo sus páginas web. En este último caso, debe colocarse en la parte superior de todas las páginas del sitio. Esto refuerza la identidad y posiciona la marca.
- Navegación: Es esencial que el sitio web sea de fácil navegación, y se espera que en la parte superior de las páginas se encuentren los controles para navegar por el sitio web.
- Contenido: La parte central del sitio web. Es aquello que se busca transmitir o comunicar, y debe ser el enfoque principal del diseño. Se compone principalmente de

texto, ilustraciones, elementos multimedia y otros elementos con los que el usuario puede interactuar.

- Pie de página: Conocido en inglés como footer. Se coloca en la parte inferior de la página, y suele contener información sobre los derechos de autor, datos de contacto, información legal, y algunos enlaces a otras secciones o recursos.
- Espacio en blanco: Es cualquier área de la página que no contiene texto o ilustraciones. Tiene como propósito mantener un balance visual para evitar sobrecargar el ojo del usuario.

El contenido de una página web suele organizarse en grillas o cuadrículas, a lo que en inglés se refiere como grid. HTML facilita dicha organización, dividiendo el espacio en rectángulos de la misma medida, habilitando cada rectángulo para ser utilizado como contenedor para los elementos de la página.

Beaird, J. (2007) hace referencia a una estructura que debe seguirse dentro de la cuadrícula de la página, llamada la regla de los tercios. Esta regla consiste en dividir la cuadrícula en tres partes iguales, y en dividir el contenido en bloques. Una recomendación en cuanto a los tamaños de los bloques es tener dos tamaños diferentes, siendo uno del doble del tamaño que el otro. Entonces se pueden colocar estos bloques distribuidos de la siguiente manera:

- Tres bloques pequeños, uno al lado del otro.
- Un bloque pequeño y un bloque grande, uno al lado del otro.

Cuando se usa un encabezado como contenedor del logotipo y/o de los elementos de navegación del sitio, éste puede ocupar el espacio de tres bloques pequeños, ya que ocupa toda la parte superior de la página.

El uso de la regla de los tercios facilita también la distribución del contenido de una manera simétrica, en el caso de la recomendación anterior. También existen otras proporciones fijas para los tamaños de los bloques que pueden usarse para otros tipos de distribución con un balance asimétrico, como tener tres tamaños diferentes de boques, por ejemplo.

Existen otros principios para el diseño de los elementos visuales de una aplicación web. En un blog publicado por la reconocida empresa Adobe, Silveira, D. (2021) enumera los principios para el diseño, que son:

- Balance: Consiste en dividir el espacio utilizando un eje vertical y otro horizontal, para organizar el contenido. La regla de los tercios se utiliza para mantener el balance.
- Contraste: Trata del uso de colores y texturas para marcar diferencias en los elementos visuales, y con esto se pretende resaltar algunos o dar mayor claridad. Por ejemplo, al aplicar un color más vistoso a un botón, para que el usuario pueda ubicarlo más fácilmente.
- Énfasis: Consiste en hacer resaltar un elemento en específico sobre todos los demás para dirigir hacia ese lugar la atención del usuario. Por ejemplo, al hacer un botón más grande que otros botones cercanos.
- Proporción: Es la escala que cada elemento de la página tiene, lo que permite identificar cómo se relacionan dichos elementos, dando una sensación de orden y facilitando la interpretación de lo que se muestra. Por ejemplo, los encabezados antes de un párrafo suelen ser un poco más grandes, y ayudan a entender que se pasa de un tema al otro.
- Movimiento: Consiste en ordenar los elementos de acuerdo con una jerarquía visual, manteniendo un hilo entre los mismos, lo cual permite entregar información ordenada al usuario a medida que éste se mueve a través de la página. Esto puede lograrse, por

ejemplo, al colocar primero los elementos que explican las bases de los que se pretende mostrar luego.

- Patrón: A medida que se agregan más elementos a un sitio, se debe buscar que exista la repetición de ciertos patrones visuales para mantener la unidad y la cohesión de todos los elementos del sitio. Esto evita que el usuario se confunda, y pueda identificar siempre los diferentes elementos como parte de un grupo.
- Unidad: Se logra al aplicar correctamente todos los demás principios anteriormente descritos, manteniendo la cohesión visual de todos los elementos de un sitio a pesar de sus diferencias en colores, tamaños o estilos. Para lograr la unidad, es necesario hacer una revisión de todos los demás principios.

2.3.4 Diseño de la base de datos

Los programas y aplicaciones de software generalmente deben ser capaces de guardar información, o de encontrar información necesaria para funcionar y lograr sus objetivos. Esta información por lo general se guarda en una base de datos, desde donde puede ser obtenida también. La base de datos suele ser almacenada de manera independiente y no dentro del programa que hace uso de esta, como tal. Marqués-Andrés (2011) define que "Una base de datos es un conjunto de datos almacenados en memoria externa que están organizados mediante una estructura de datos."

De acuerdo a Silberschatz, Sudarshan y Korth (2014) la base de datos posee tres niveles de abstracción que permiten una interacción más sencilla para los administradores de bases de datos y para los usuarios. Dichos niveles de abstracción son:

- Nivel físico: Es el nivel más bajo. Describe cómo se guardan los datos. El nivel físico describe en detalle las estructuras de datos complejas de bajo nivel. (Silberschatz, Sudarshan y Korth, 2014, p. 3)
- Nivel lógico: Es el nivel de abstracción que sigue al físico. Describe qué datos se almacenan en la base de datos y qué relaciones existen entre esos datos. (Silberschatz, Sudarshan y Korth, 2014) Por este motivo los administradores de bases de datos suelen utilizar este nivel para interactuar con la base de datos, ya que no tienen necesidad de preocuparse por la complejidad del cómo están estructurados los datos físicamente.
- Nivel de vistas: Es el nivel más elevado de abstracción y solo muestra parte de la base de datos. Existe con la finalidad de simplificar la interacción entre algunos usuarios y la base de datos. Una base de datos puede tener múltiples vistas.

Existen varias maneras de describir los datos y sus relaciones, a lo cual se llama modelo de datos. En este proyecto se usará el modelo entidad-relación, que consiste en representar elementos del mundo real y sus características, y maneras en las que se relacionan.

Una entidad es la representación de un objeto o cosa, y sus características se conocen como atributos. Estos atributos deben tener un valor, el cual puede ser de muchos tipos diferentes, por ejemplo, numérico o una o varias palabras. Si se toma por ejemplo un carro, la entidad sería el carro en sí, sus atributos pueden ser el color, la cantidad de puertas, la marca, el modelo, el año en que se fabricó, y la tracción, por mencionar algunos. Y en el caso del atributo del color, su valor podría ser una palabra, y por mencionar algunos ejemplos, esta palabra puede ser “verde”, “rojo” o “azul”. Para cada entidad se espera que existan varias instancias, por lo cual, en el ejemplo de la entidad carro se espera que existan varios carros con valores variados en cada uno de sus atributos con respecto de una instancia a otra. Por regla general, todas las instancias de

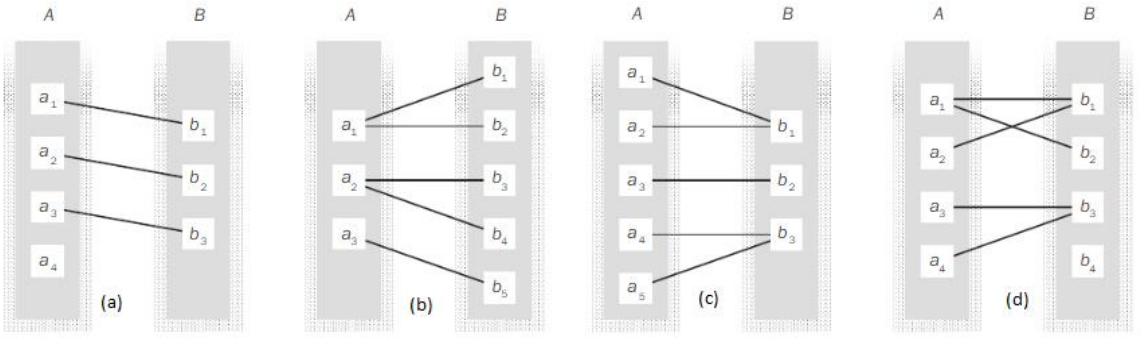
una entidad poseen un atributo ID o identificador, y con frecuencia es un valor numérico único cuya función es posibilitar su distinción de otras instancias del mismo tipo de entidad. Los conjuntos de entidades iguales se agrupan en una tabla.

Las entidades y sus atributos pueden relacionarse entre sí. Entonces, siguiendo con el ejemplo anterior, podría existir una tabla que agrupe todas las entidades de tipo carro, y otra tabla que agrupe los colores que existen para pinturas de carro. Así, un carro puede estar pintado de azul, blanco, rojo, o negro, por mencionar solo algunos colores, que serían parte de los valores de un atributo en la tabla de colores.

Existen también algunas restricciones que pueden encontrarse en las relaciones entre entidades. Una de ellas se conoce como cardinalidad, y esta "expresa el número de entidades a las que otra entidad se puede asociar mediante un conjunto de relaciones." (Silberschatz, Sudarshan y Korth, 2014, p. 121). En la Figura 6 (Correspondencia de cardinalidades) se aprecian los cuatro tipos de correspondencias que hay. El ejemplo a muestra una cardinalidad tipo uno a uno, en la cual cada entidad de A se asocia solo con una entidad de B, y viceversa. En el ejemplo b se muestra una cardinalidad tipo uno a varios, en la cual cada entidad de A puede asociarse a cualquier cantidad de entidades de B, pero cada entidad de B solo puede asociarse con una entidad de A. En el ejemplo c puede observarse el tipo de cardinalidad de varios a uno, que es similar a la anterior, pero en este caso cada entidad de A solo se pude relacionar con una entidad de B, y cada entidad de B se puede relacionar con cero o muchas entidades de A. Y el ejemplo d muestra el tipo de cardinalidad varios a varios, en la que cada entidad de A se puede relacionar cualquier número de veces con cada entidad de B, y viceversa.

Figura 6.

Correspondencia de cardinalidades.



Fuente: Silberschatz, Sudarshan y Korth, 2014.

A una instancia de una entidad con frecuencia se le llama tupla. Cada instancia o tupla debe ser única, no puede haber dos iguales, es decir, que posean todos sus atributos con valores idénticos. Para solucionar esto se hace uso de llaves o claves, en este caso la llave primaria. Como indica Marqués-Andrés, M. (2011) "se denomina clave primaria de una relación a aquella clave candidata que se escoge para identificar sus tuplas de modo único." Esta clave o llave es de hecho un atributo de la entidad que se escoge para dicho fin. Se debe conocer el contexto de lo que se representa en la entidad para escoger adecuadamente la llave primaria. Por ejemplo, en la entidad carro no se debería escoger el atributo color como llave primaria, ya que es un valor que se espera sea repetido o compartido por varias tuplas. En cambio, podría asignarse un valor numérico como identificador (ID) que sea único para cada tupla y escoger ese atributo como llave primaria. Además, existen llaves foráneas, ajenas o secundarias. Según describe Marqués-Andrés, M. (2011) "Una clave ajena es un atributo o un conjunto de atributos de una relación cuyos valores coinciden con los valores de la clave primaria de alguna otra relación (puede ser la misma). Las claves ajenas representan relaciones entre datos." Entonces en el ejemplo de la entidad carro, el atributo color se convierte en una llave secundaria porque sus posibles valores existen en la entidad color.

Como se ha mencionado anteriormente, la base de datos generalmente es parte de un conjunto y existe de manera independiente del código de las aplicaciones o sistemas que hacen uso de ella. Por este motivo, es necesario contar con una herramienta para su creación, gestión y seguridad. A las aplicaciones usadas para esto se les conoce como Sistemas de Gestión de Bases de Datos (SGBD). El uso de un SGBD permite la abstracción de los datos que contiene y su estructura, de manera que los usuarios no deben preocuparse por dicha estructura para su manipulación. Marqués-Andrés, M. (2011) dice lo siguiente acerca de los sistemas de bases de datos: "separan la definición de la estructura física de los datos de su estructura lógica, y almacenan esta definición en la base de datos. Todo esto es gracias a la existencia del SGBD, que se sitúa entre la base de datos y los programas de aplicación."

El SGBD también permite la implementación de mecanismos de seguridad para proteger la integridad de la base de datos y la información que contiene. Algunos de estos mecanismos consisten en:

- Validación de credenciales de acceso.
- Registro de accesos y de cambios realizados.
- Respaldos de información y sistemas de recuperación, para usarse en casos de fallos o pérdidas.
- Un diccionario para los usuarios de la base de datos, con una descripción de esta.
- Un sistema para mantener la integridad y consistencia de los datos.
- Un sistema para permitir y controlar el acceso por parte de múltiples usuarios de manera simultánea.

Para manipular los datos, es necesario utilizar código. Dependiendo de lo que se quiera hacer, dicho código utiliza un lenguaje determinado. Silberschatz, A., Sudarshan, S., Korth, H. F. (2014) definen los siguientes:

- Lenguaje de definición de datos (DDL por sus siglas en inglés): Se utiliza para definir los esquemas de las bases de datos, para especificar propiedades de los datos, la estructura de almacenamiento y los métodos de acceso; y también puede servir para reforzar restricciones de consistencia de datos.
- Lenguaje de manipulación de datos (DML por sus siglas en inglés): Se utiliza para la recuperación, inserción, borrado y modificación de los datos en una base de datos. Se separan en dos tipos: procedimentales y no procedimentales. Lo que los diferencia principalmente es que los lenguajes procedimentales requieren que el usuario especifique qué datos quiere obtener y cómo obtenerlos, mientras que los lenguajes no procedimentales requieren de parte del usuario únicamente los datos que quiere obtener y el sistema determina la manera más eficiente de obtenerlos. El lenguaje más utilizado es de tipo no procedimental, llamado lenguaje de consulta estructurada (SQL por sus siglas en inglés), utilizado para bases de datos relacionales como el modelo Entidad-Relación mencionado anteriormente.

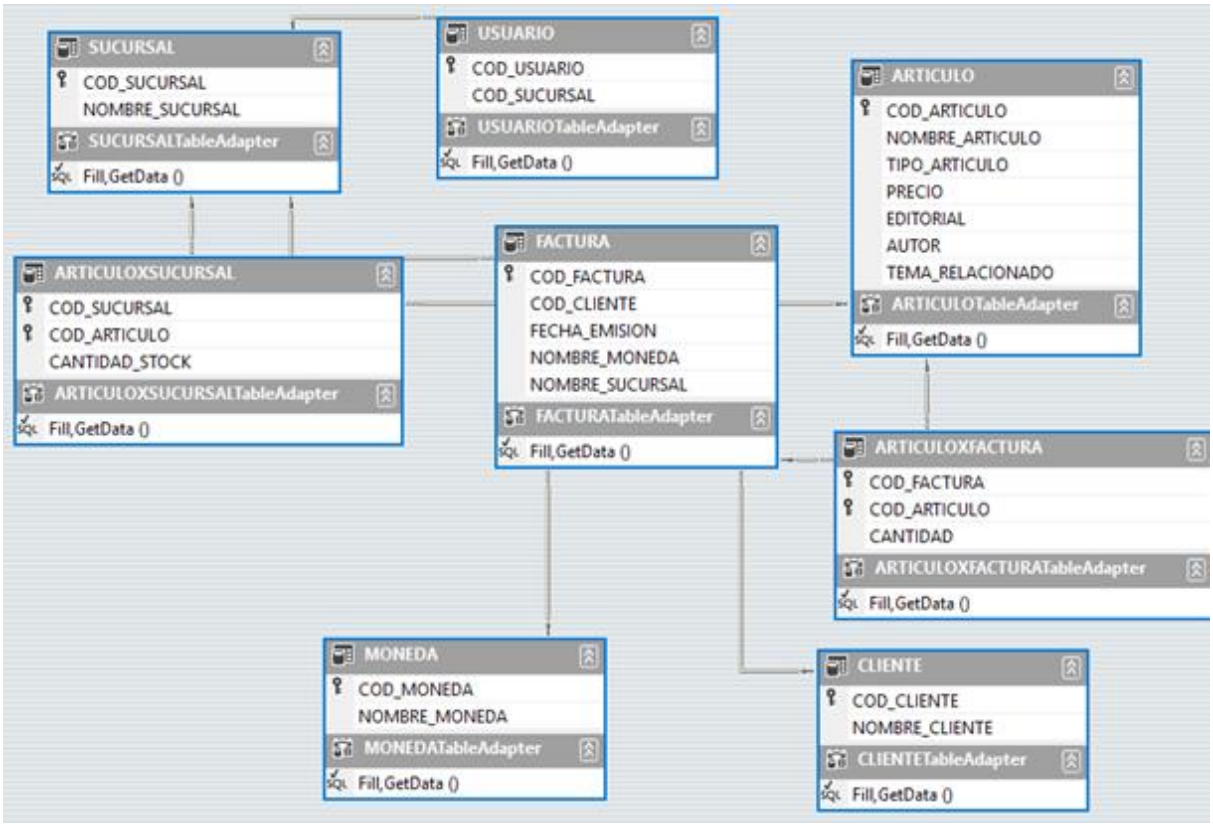
Para representar la estructura de una base de datos y las propiedades de sus tablas se utiliza un diagrama. Su uso es casi una norma, ya que permite la identificación visual de la estructura de la base de datos, sus entidades y relaciones, de manera que un usuario inexperto pueda entenderla.

Muchos sistemas de gestión de base de datos poseen integrada una herramienta para realizar dichos diagramas de manera automática. En la ilustración a continuación (Figura 7, diagrama

de base de datos) se aprecia un diagrama de base de datos. Dicho diagrama contiene ocho tablas, cada una representando una entidad: Usuario, sucursal, articulosxsucursal, factura, articulo, articuloxfactura, moneda y cliente. Cada tabla muestra los atributos de la entidad que representa, identificando la llave principal con un ícono de una llave al lado izquierdo del primer atributo de la tabla, o de los primeros dos atributos de la tabla cuando se trata de una llave compuesta; también muestra los métodos almacenados de dicha tabla (En este caso, todas tienen al menos un método usado para recuperar los datos de la tabla). Por ejemplo, la tabla factura posee los siguientes atributos: Cod_factura (Para almacenar el código de la factura), cod_cliente (Para almacenar el código del cliente que se asocia a dicha factura), fecha_emisión, nombre_moneda, nombre_sucursal. Esta tabla se relaciona con otras tablas, lo que se representa con una línea continua que tiene una llave al final, demostrando así que hay una llave foránea, siendo posible identificar cuál atributo es la llave foránea debido a que en ambas existe dicho atributo. Por ejemplo, en la tabla factura se observa el atributo cod_cliente, y una línea que la conecta con la tabla cliente, donde se observa dicho atributo como la llave principal de dicha tabla. Esto mismo se repite con esta misma tabla, así como con otras.

Figura 7.

Diagrama de base de datos.



Fuente: propia.

2.3.4.1 Tipos de bases de datos

Existen básicamente dos tipos de bases de datos: Relacionales y no relacionales, dentro de los cuales existen muchas clasificaciones a su vez dependiendo del modelo de base de datos. Marqués-Andrés, M. (2011) define lo siguiente: “En el modelo relacional los datos se describen como un conjunto de tablas con referencias lógicas entre ellas, mientras que en los modelos jerárquico y de red, los datos se describen como conjuntos de registros con referencias físicas entre ellos”.

Por lo tanto, en los modelos jerárquico y de red, cuando se tiene un registro y se desea relacionarlo con otro, se debe agregar un campo en el primero con la información de dónde se encuentra físicamente ubicado el segundo. A este campo se le llama puntero, ya que apunta hacia la dirección donde se encuentra el otro registro. Esto representa una limitante, y una

vulnerabilidad también, debido a que si los datos son movidos físicamente de un lugar a otro (por un cambio de disco duro, por ejemplo) es necesario actualizar todos los punteros.

Estos dos modelos de bases de datos son de primera generación, y fueron ampliamente utilizados hasta la aparición del modelo de base de datos relacional, cuya principal ventaja sobre los anteriores es la separación de la capa física y la capa lógica, haciendo posible que su estructura lógica sea diferente a su estructura física, simplificando también su uso ya que es más sencillo trabajar enfocándose en la estructura lógica.

Algunos tipos de modelos de datos relacionales son:

- Relacional: es el más utilizado, y consiste en organizar los datos en tablas con filas y columnas, con cada tabla representando una entidad diferente y una relación.
- Jerárquico: En este modelo se organizan los datos en una estructura de tipo árbol, con relaciones de padre-hijo entre los elementos de datos.
- De red: Permite relaciones más complejas entre los datos, con cada registro teniendo múltiples dueños y miembros.
- Orientado a objetos: Sigue los mismos principios de la programación orientada a objetos, combinándolos con los principios del modelo relacional. Permite representar datos con propiedades y métodos.
- Entidad-Relación: Representa datos como entidades y las relaciones entre ellas. Es útil para visualizar estructuras de datos complejas y sus relaciones antes de llevar a cabo una implementación.

Los modelos de datos no relacionales son de origen más reciente, y buscan adaptarse a las nuevas tendencias y necesidades en desarrollo de software. Uno de los líderes en el desarrollo

de bases de datos, MongoDB (2018) indica que las bases de datos no relacionales se pueden clasificar en los siguientes tipos:

- Modelo de documentos: Los datos se almacenan en documentos en lugar de tablas. Usualmente utilizan una estructura parecida a JSON, por lo que es orientada a objetos. Cada documento posee varios campos, con sus propios valores y relaciones entre campos de otros documentos, organizados de manera jerárquica.
- Modelo de grafos: Utiliza estructuras de grafos, con nodos, bordes y propiedades para representar los datos, que son modelados como una red de relaciones entre los elementos.
- Modelo de llave-valor: Cada registro se guarda con una clave y su valor. La información se accede únicamente a través de las claves, y no se refuerzan estructuras o reglas.
- Modelo orientado a columnas: Los datos son almacenados en un mapa multidimensional, y cada registro puede poseer una cantidad diferente de columnas. Estas columnas pueden agruparse por familias. Se accede a ellos a través del identificador de la familia.

Los esquemas de bases de datos no relacionales son más flexibles que los relacionales, al poseer menos reglas y estar orientados a facilitar la escalabilidad y los cambios. Mientras tanto, los esquemas de bases de datos relacionales son preferibles para el registro de transacciones y para realizar consultas complejas de datos que tienen relaciones entre sí.

2.4 E-COMMERCE

La llegada de tecnología digital y su adopción por parte de la mayor parte de la población mundial ha hecho posible realizar muchas tareas de maneras que antes no era posible. Por ejemplo, permite que personas en lugares diferentes puedan trabajar colaborativamente en una

misma tarea, cuando dichas personas se conectan por medio del internet a alguna aplicación para procesar palabras, y escriben al mismo tiempo, siendo todo lo que hacen visible para todas las partes.

El sector del comercio ha adoptado también la tecnología para ofrecer nuevas maneras de llevarse a cabo la actividad comercial, de formas que hasta hace un par de décadas no era posible.

Laudon, K. C., y Guercio Traver, C. (2014) definen el comercio electrónico (Conocido también como e-commerce, del inglés "Electronic Commerce") de la siguiente manera: "Uso de internet, la web y aplicaciones de software para hacer negocios. Dicho de manera más formal, comprende las transacciones comerciales digitales que ocurren entre organizaciones, entre individuos, y entre organizaciones e individuos". Esto permite a las empresas y a los comerciantes utilizar espacios digitales como vitrina para sus productos y servicios. Pero más allá de poder mostrar el producto que ofrecen a una audiencia tal vez mayor a la que tiene por ejemplo la vitrina de una tienda en una calle, también ofrece la oportunidad de recibir dinero y de coordinar la entrega del producto o servicio (Cuando éste sea algo físico, como un par de zapatos por mencionar un ejemplo). Esto puede resultar muy conveniente para las personas que están familiarizadas con la tecnología digital, y que se sienten más cómodas observando un producto de su interés a través de sus pantallas que trasladándose físicamente al lugar donde se ofrece. De hecho, hace posible que dos personas en extremos opuestos del mundo puedan realizar transacciones comerciales al instante, algo que, si bien es posible realizar con otras tecnologías, es sin dudas una solución más completa.

Al igual que cualquier actividad comercial, el e-commerce se vale de modelos de comercio para definir sus actividades y llegar a ser exitoso. Laudon, K. C., y Guercio Traver, C. (2014)

explican que generalmente las empresas se apegan a uno de los modelos, pero en ocasiones pueden realizar una combinación de varios, que son: el modelo de publicidad, el modelo de suscripción, el modelo de cobro por transacción, el modelo de ventas y el modelo de afiliación. Para los fines de este proyecto, se hablará únicamente del modelo de ventas.

El modelo de ventas en e-commerce consiste en "vender bienes, información o servicios a los clientes." (Laudon, K. C., y Guercio Traver, C. 2014) y sigue los mismos modelos o enfoques que el comercio tradicional. Por ejemplo, dependiendo del tipo de cliente se puede clasificar en B2B (Del inglés "Business to Business", o negocio a negocio), en B2C (Del inglés "Business to Customer", o negocio a cliente) o en B2G (Del inglés "Business to Government", o negocio a gobierno). Cuando el objetivo es vender a otras empresas, entonces es un modelo B2B. Cuando el objetivo es vender directamente al consumidor final, entonces el modelo es B2C. Y cuando el cliente es un gobierno o entidad pública, entonces es un modelo B2G. Cada uno de ellos posee también modelos diferentes dentro de su clasificación.

En el modelo B2C, de acuerdo con Laudon, K. C., y Guercio Traver, C. (2014) los modelos principales son: Tienda minorista en línea, proveedor de comunidades, proveedor de contenido, portal, corredor de transacciones, generador de mercado y proveedor de servicios. Y dentro del modelo para B2C de tienda minorista en línea, existen cuatro variaciones: comerciante virtual (Por ejemplo, Amazon), Bricks-and-clicks (Negocios con presencia física y virtual. Por ejemplo, Walmart), comerciante por catálogo (Por ejemplo, Bloomingdale's) y directo del fabricante (Por ejemplo, Dell). Estas cuatro variaciones tienen como modelo de ingresos la venta de bienes.

Una empresa que desee hacer uso del e-commerce tendrá que decir también en qué medios digitales tendrá presencia. Laudon, K. C., y Guercio Traver, C. (2014) indican que existen

“cuatro tipos de presencia en el comercio electrónico: sitios web, correo electrónico, medios sociales y medios fuera de línea.” En cuanto a sitios web, se realiza siempre la distinción del tipo de dispositivo objetivo para la página web: computadora, teléfono inteligente o tableta. Y aunque hoy en día existen métodos para que una página web sea responsiva, es decir, se adapte al tamaño de la pantalla del dispositivo por lo cual es funcional y utilizable en los tres tipos de dispositivos objetivos, es importante tomar en cuenta el dispositivo más utilizado por los usuarios objetivos o clientes y potenciales clientes, de manera que el diseño dé prioridad a dicho tipo de dispositivo. En el caso de Costa Rica, el teléfono inteligente es el dispositivo de mayor uso para realizar transacciones electrónicas.

Un estudio realizado por el personal del Centro de Investigación Observatorio del Desarrollo de la Universidad de Costa Rica (2022), para el Ministerio de Economía, Industria y Comercio (MEIC) durante el año 2022 muestra que el 48.5% de las personas en Costa Rica realizaron algún tipo de compra por un medio electrónico, y que el medio más utilizado es el teléfono inteligente o celular, representando el 65.3% de los dispositivos usados para realizar dichas compras, seguido de la computadora con un 32,8%. También se menciona que el 65,8% de las personas se enteran de la publicidad por medio de las redes sociales. Este tipo de información es de importancia para planificar una estrategia adecuada de presencia en la web por parte de una empresa, y hacer uso del e-commerce como un medio para alcanzar sus objetivos.

Como se describió anteriormente, los requerimientos para un sistema de información son definidos en su mayoría por el cliente, pero Laudon, K. C., y Guercio Traver, C. (2014) indican que para un sitio típico de comercio electrónico deben existir al menos diez objetivos de negocio básicos con una funcionalidad para cada uno de ellos, y sus respectivos requerimientos. La tabla a continuación (Tabla 4. Objetivos de negocio, funcionalidad del sistema y requerimientos de

información para un sitio típico de comercio electrónico) los describe, empezando de izquierda a derecha con los objetivos de negocio, luego con la funcionalidad del sistema necesaria para alcanzar dicho objetivo, y por último con el requerimiento de información necesario para dicha funcionalidad:

Tabla 4.

Objetivos de negocio, funcionalidad del sistema y requerimientos de información para un sitio típico de comercio electrónico.

Objetivos de negocio	Funcionalidad del sistema	Requerimientos de información
Mostrar los bienes	Catálogo digital	Catálogo con texto y gráficos dinámicos
Proporcionar información del producto (contenido)	Base de datos de los productos	Descripción del producto, cantidad en existencia, niveles de inventario
Personalizar y/o adecuar el producto	Seguimiento del cliente en el sitio	Registro de la visita de cada cliente; capacidad de extracción de datos para identificar rutas comunes y respuestas adecuadas de los clientes
Involucrar a los clientes en conversaciones	Blog en el sitio	Software con funcionalidad de respuesta a blogs y comunidades

Ejecutar una transacción	Sistema de carrito de compras y sistema de pagos	Pago seguro mediante tarjeta de crédito; opciones de pago múltiples
Acumular información sobre los clientes	Base de datos de clientes	Nombre, dirección, teléfono y correo electrónico de todos los clientes; registro de clientes en línea
Proporcionar soporte posterior a la venta al cliente	Base de datos de ventas	ID del cliente, producto, fecha, pago, fecha de envío
Coordinar marketing y publicidad	Servidor de anuncios, servidor de correo electrónico, director de campaña y director de banners	Registro de comportamiento en el sitio de prospectos y clientes enlazados a un correo electrónico, banners y campañas
Comprender la eficacia del marketing	Sistema de seguimiento y reportes en el sitio	Número de visitantes únicos, páginas visitadas, productos comprados, identificados por campaña de marketing
Proporcionar vínculos hacia la producción y los proveedores	Sistema de administración de inventario	Niveles de productos e inventario, identificación y contacto de proveedor, datos de cantidad de pedidos por producto

Fuente: Laudon, K. C., y Guercio Traver, C. (2014).

2.5 DESARROLLO

Para el desarrollo de una aplicación web es posible hacer uso de una variedad de tecnologías, y todas ellas hacen uso de diferentes leguajes de programación para su implementación. También existen diversos patrones o técnicas a seguir para la formar la estructura del código del software. Esta sección menciona algunos lenguajes de programación, marcos de trabajo o frameworks y modelos de arquitectura usados para el desarrollo de aplicaciones web.

2.5.1 Lenguajes de programación

Las computadoras utilizan impulsos eléctricos para llevar a cabo sus acciones y los lenguajes de programación son el medio para brindarles las instrucciones necesarias para que lleven a cabo dichas acciones. "Programación es el proceso de análisis, diseño, implementación, prueba y depuración de un algoritmo, a partir de un lenguaje que compila y genera un código fuente ejecutado en la computadora" (Universidad Nacional Autónoma de México, 2017).

Existen dos tipos generales de lenguajes de programación: de alto nivel y de bajo nivel. Los de bajo nivel tienen contacto directo con el hardware, pero no siguen una estructura natural para el ser humano, por lo cual existen los lenguajes de alto nivel, que poseen una sintaxis y una semántica más adaptadas a la forma natural de hablar del ser humano. Existen también los llamados "compiladores" y los "intérpretes", programas encargados de traducir instrucciones entre ambos tipos de lenguajes.

Los lenguajes de programación de alto nivel tienen varias clasificaciones, y en ocasiones un lenguaje puede encajar en más de una clasificación. De acuerdo con la UNAM, algunas de las clasificaciones más comúnmente aceptadas son:

- Lenguajes imperativos.

- Lenguajes declarativos.
- Lenguajes orientados a objetos.
- Lenguajes orientados al problema.
- Lenguajes naturales.

Según el Instituto de Ingenieros Eléctricos y Electrónicos, o IEEE según sus siglas en inglés (Cass, Stephen. IEEE, 2022) entre los lenguajes de programación de alto nivel, los 10 más usados son:

1. Python.
2. C.
3. C++.
4. C#.
5. Java.
6. SQL.
7. JavaScript.
8. R.
9. HTML.
10. TypeScript.

Muchos lenguajes de programación se mantienen en constante cambio y actualización para servir mejor a las tendencias de desarrollo actuales y facilitar la innovación.

Cada lenguaje puede tener características diferentes. Tomando como ejemplo C#, a continuación se muestran algunas de sus características según Microsoft (2023) quienes desarrollaron dicho lenguaje:

- Es orientado a objetos, por lo cual “la unidad de proceso es el objeto y en él se incluyen los datos (variables) y operaciones que actúan sobre ellos” (UNAM, 2017)
- Posee un recolector de elementos no usados para desocupar la memoria no utilizada.
- Sus objetos pueden aceptar valores nulos.
- Cuenta con control de excepciones.
- Permite el uso de expresiones lambda, para la creación de funciones anónimas y el uso de técnicas de programación funcional.
- Posee una sintaxis llamada LINQ, que permite trabajar con datos de cualquier origen usando consultas estandarizas.
- Es compatible con operaciones asincrónicas.

2.5.2 Frameworks para el desarrollo

Un framework (inglés para marco de trabajo) se utiliza para facilitar el desarrollo de software. Según Pressman, Roger (2011) un framework no es un patrón de arquitectura, sino más bien un esqueleto con una colección de elementos prediseñados que pueden ser usados para entregar soluciones específicas. Son mini-arquitecturas reutilizables, con clases o funcionalidades preestablecidas que colaboran entre sí.

Generalmente, cada framework es elaborado para ser utilizado con un conjunto específico de lenguajes de programación, por lo cual existen también una amplia variedad.

De acuerdo con el sitio web Stack Overflow (2022), que sirve de referencia para una comunidad mundial de desarrolladores, una encuesta a más de 80.000 desarrolladores a nivel mundial reveló que los 6 frameworks para desarrollo web más utilizados son:

1. React.js.
2. jQuery.

3. Express.
4. Angular.
5. Vue.js.
6. ASP.NET Core.

React, JQuery, Angular, Vue y Express son frameworks principalmente para el lenguaje JavaScript, ampliamente utilizado en el desarrollo de aplicaciones web. ASP.NET permite el uso de varios lenguajes, pero principalmente C#. En la encuesta citada anteriormente .NET Framework fue escogido como el framework que más se uso para otras tecnologías de desarrollo además de desarrollo web.

Cada framework tiene sus propias características. A continuación se citan algunas de las características de ASP.NET Core para mencionar un ejemplo, de acuerdo a lo reseñado por Microsoft (2023), quien desarrolló dicho framework:

- Es gratuita y de código abierto.
- Se puede utilizar en sistemas operativos Android, Apple, Linux y Windows.
- Todos los años, en noviembre, se lanzan nuevas versiones de .NET.
- La base de sus aplicaciones es la CLR (Common Language Runtime) que se encarga de la administración de la memoria, hardware y sistema operativo.
- Admite tres lenguajes de programación: C#, F# y Visual Basic.
- Sus aplicaciones se compilan en un lenguaje intermedio.
- Tiene una amplia colección de bibliotecas, permitiendo implementaciones para muchos propósitos en general.
- Posee un administrador de paquetes llamado NuGet, por medio del cual se pueden agregar más bibliotecas a un proyecto dado.

- Proporcionan clases y servicios útiles que escribir código seguro, usar criptografía e implementar la seguridad basada en roles.

2.5.3 Modelos de arquitectura

De acuerdo con Pressman, R. (2011) la arquitectura de software es “la estructura u organización de los componentes del programa (módulos), la manera en que estos componentes interactúan, y la estructura de los datos que son usados por los componentes.”.

El tener un modelo o patrón de arquitectura puede resultar muy útil al momento de llevar a cabo el diseño de una solución de software, pero desde el punto de vista más técnico, con respecto a la implementación o desarrollo del código como tal al brindar estructura y orden. Esto permite a su vez resolver problemas y desafíos que puedan encontrarse durante las etapas de desarrollo y de soporte técnico, porque los desarrolladores pueden ver ejemplos de otros proyectos y casos de uso que pueden adaptar a las necesidades de sus propios proyectos.

Dado que un sistema de software puede abordar una gran cantidad de problemáticas, poder segmentar la estructura del software para resolver cada uno de los asuntos pertinentes facilita las tareas de los desarrolladores. Según Pressman, R. (2011) para los desarrolladores puede ser más fácil separar por componentes asuntos concernientes al manejo de datos, la seguridad, los métodos y las clases, la interoperabilidad con otros programas de software, por mencionar algunos.

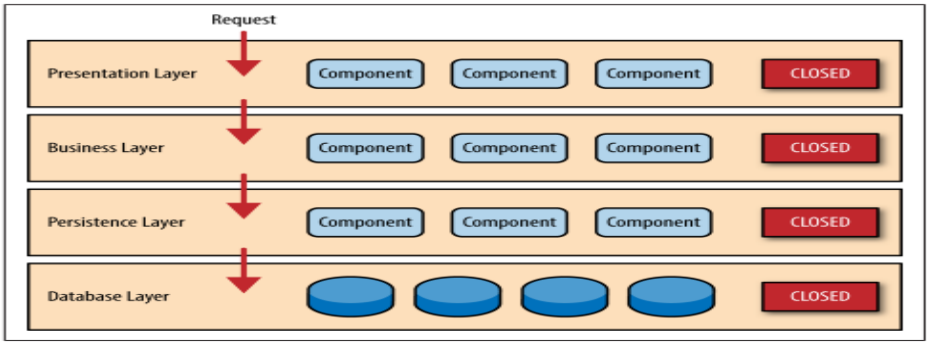
Richards, M. (2015) menciona que entre los modelos de arquitectura más usados se encuentran los siguientes:

- Arquitectura en capas: En este modelo el software se escribe en diferentes componentes de manera horizontal, llamados capas, y cada uno de ellos desempeñan un rol específico dentro de la aplicación. Generalmente posee al menos tres capas: Capa de presentación,

capa de negocio y capa de datos; éstas a su vez pueden subdividirse en capas adicionales según sea conveniente. Suele ser muy utilizado para código Java. A continuación, en la figura 8 se observa un diagrama de una arquitectura en capas, en la cual se muestran cuatro capas con sus respectivos componentes: Las capas de presentación, de negocios, de persistencia y de base de datos; y se señala el recorrido de una solicitud a través de las capas en su orden respectivo. Tienen un símbolo de "cerrado" para denotar la restricción de acceso de la solicitud, la cual debe pasar de una capa a la próxima, en orden.

Figura 8.

Arquitectura en capas.



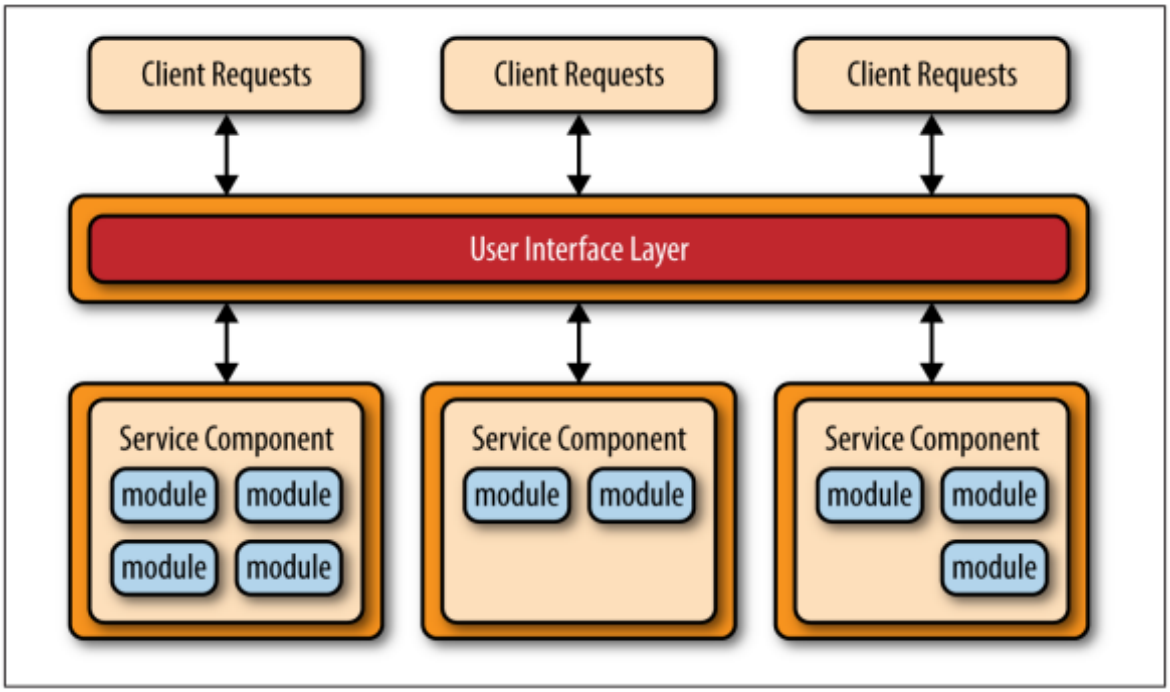
Fuente: Richards, M. (2015).

- Arquitectura basada en eventos: Es un modelo o patrón que hace uso de componentes desacoplados entre sí y cumplen un propósito individual, que reciben eventos de manera asíncrona para procesarlos. Se enfoca en permitir una fácil escalabilidad. Consiste en dos principales topologías: Productores y consumidores de eventos. Los productores detectan los eventos y los envían a los consumidores, quienes los procesan de manera asíncrona.

- Microkernel: En este modelo existe un núcleo central o kernel, de menor tamaño y encargado de las funciones esenciales para el funcionamiento del sistema. Luego, hay espacios para adjuntar otros módulos, encargados del resto de las funciones del software, y procesados en el lado del usuario. Esto permite la adición de módulos intercambiables.
- Arquitectura de microservicios: Consiste en desarrollar diferentes componentes o unidades, cada una enfocada en realizar tareas específicas, con la capacidad de comunicarse entre sí por medio de una interfaz definida. Estas unidades pueden a su vez subdividirse de acuerdo con las necesidades del programa. A continuación, la figura 9 muestra un ejemplo de arquitectura de microservicios, en el cual se aprecian cómo las solicitudes de los clientes pasan a través de una interfaz definida, y luego son distribuidas a los contenedores específicos de acuerdo a los servicios que éstos brindan.

Figura 9.

Arquitectura de microservicios.

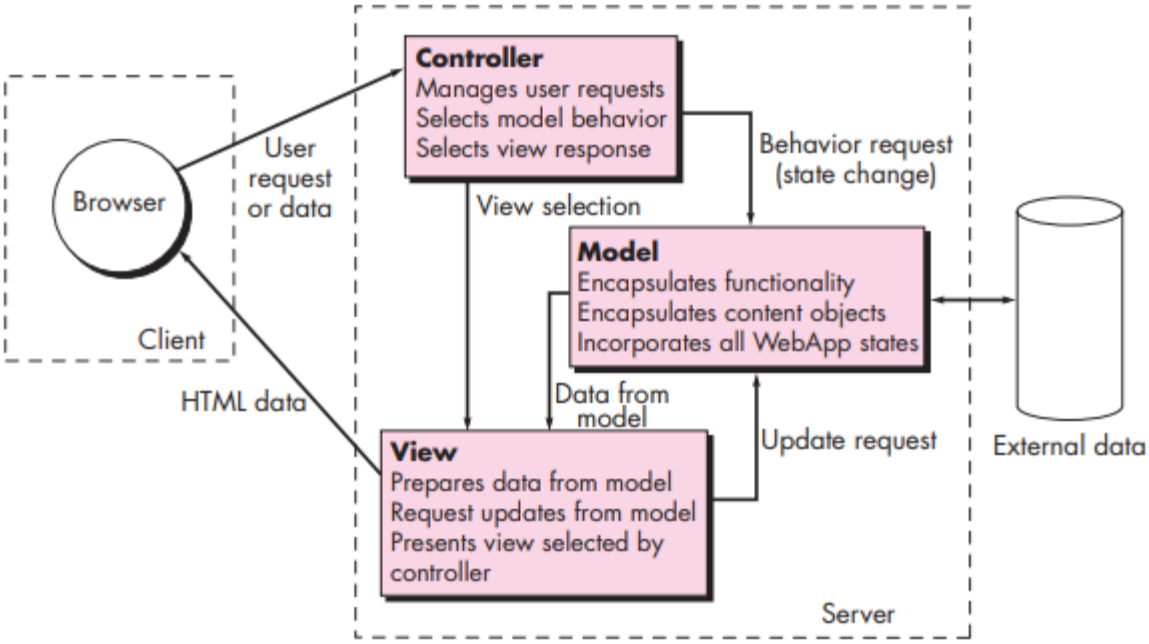


Fuente: Richards, M. (2015).

- Modelo-Vista-Controlador: También conocido como MVC. Consiste en un diseño de tres capas: El modelo, encargada de gestionar todas las conexiones a los datos de la aplicación (Almacenados interna o externamente); la vista, encargada de todas las funciones relacionadas a las interfaces del usuario y su presentación; y el controlador, que sirve de intermediario entre las dos anteriores, y se encarga también de gestionar los accesos. Pressman, R. (2011) recomienda su uso para aplicaciones web, ya que simplifica la implementación y mejora la reutilización. La figura 10 (a continuación) muestra un ejemplo de la arquitectura MVC, en el cual al usuario se le muestra una vista, tras lo que lleva a cabo una solicitud, manejada por el controlador. El controlador entonces envía una solicitud al modelo, el único capaz de poder acceder directamente a la base de datos, y pasa la información para ser mostrada en la siguiente vista seleccionada por medio del controlador en base a la última acción del usuario.

Figura 10.

Modelo-Vista-Controlador.



Fuente: Pressman, R. (2011).

2.6 TÉCNICAS PARA PROBAR Y VALIDAR SOFTWARE

Como cualquier producto que se lanza al mercado, un producto de software debe ser sometido a pruebas antes de ser totalmente entregado. En la etapa de pruebas se compara lo que se está entregando contra los requerimientos recibidos durante las primeras etapas del proyecto. Esto es una manera de asegurarse de cumplir con dichos requerimientos antes de la entrega final del software. Idealmente, los documentos de requerimientos y especificaciones no se deben reescribir en la etapa de pruebas, pues si sucede, esto es fuente de retrasos y de conflictos (Hilard, Vincent. 2020).

Al igual que todas las demás etapas del desarrollo de software, la etapa de pruebas debe ocurrir de acuerdo con un plan. Hilard, Vincent (2020) indica que se debe crear un plan de test, y menciona lo siguiente: “El plan de test es el documento que precisa la estrategia adoptada para efectuar los test. En él se especifica qué funcionalidades de la aplicación se probarán, su orden,

así como su sucesión.”. Idealmente este plan se redacta al mismo tiempo que el documento de especificaciones funcionales, para aprovechar el momento en que se describen con mayor precisión y detalle las funcionalidades del software y así fijar las reglas que se seguirán para comprobar dichas funcionalidades.

Cada prueba o test debe realizarse con un objetivo claro. En el plan de test se puede incluir una plantilla a seguirse durante cada prueba realizada, y puede incluir datos como la acción a realizarse, el resultado esperado y el resultado obtenido. En el caso de ser requeridos datos específicos para ser introducidos durante una prueba, esto debe especificarse también, por ejemplo, al probar un formulario de inscripción este debe validar que se ingrese un correo electrónico con un formato válido, así como ingresar el nombre y el apellido, y en caso de no cumplirse el requerimiento entonces se mostrará un mensaje de error. También en una misma prueba pueden darse diferentes escenarios, los cuales también deben se especificados. La tabla 5, plantilla de plan de test (a continuación) muestra lo descrito anteriormente, teniendo como encabezados de la tabla la referencia, acción, resultado esperado, resultado obtenido y estado. También muestra información de ejemplo, agrupada por casos de test (Cada caso describe un error que se busca encontrar) y escenarios dentro de dicho caso (Las diferentes instancias en las que el error que se busca puede aparecer).

Tabla 5.

Plantilla de plan de test.

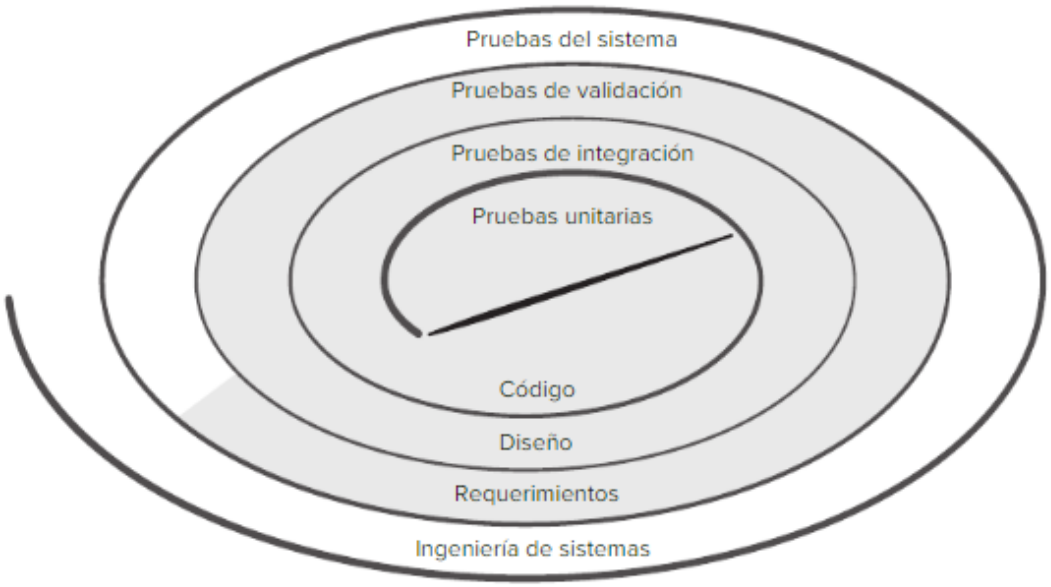
Nombre: Pruebas unitarias del módulo Información de contacto				
Referencia	Acción	Resultado esperado	Resultado obtenido	Estado
Caso test 5.1	Campos vacíos			

Escenario 1	Dejar un campo vacío y presionar el botón de enviar mensaje.	Se muestra un mensaje de error indicando que hay un campo vacío.	Se vacían los campos, pero no muestra un mensaje de error.	No ok
Caso test 5.2	Correo no cumple con el formato			
Escenario 1	Introducir un texto en un formato diferente al de un correo electrónico.	Se muestra un mensaje de error indicando que se debe escribir un correo en el formato correcto	El mensaje de error se muestra apropiadamente	Ok
…	…	…	…	Ok / No ok
...	...	...	...	Ok / No ok
Realizado por:	Luiscarlos Martínez		Resultado:	Parcialmente exitosa
Observaciones:	Se deben revisar los data annotations para cuando los campos están vacíos.			

Además, las pruebas ocurren idealmente dentro de una estrategia, similar a la llevada a cabo hasta ese punto. Esta estrategia contempla etapas, así como el desarrollo también tiene sus etapas. La Figura 11, llamada Estrategia de pruebas (Se muestra al final de este párrafo) indica un ejemplo de las etapas en el desarrollo y de las etapas en la estrategia de pruebas. Se comparan a una espiral, en la que se empieza desde afuera con la aplicación de la ingeniería de software, que conduce a la obtención de los requerimientos, luego de lo cual se analizan y se lleva a cabo el diseño de la solución, tras lo cual se procede con la elaboración del código. Y al final de la elaboración del código, la espiral comienza a ir en sentido contrario, empezando por las pruebas unitarias, las pruebas de integración, pruebas de validación y por último las pruebas de sistema.

Figura 11.

Estrategia de pruebas.



Fuente: Pressman, R.S. y Maxim, B.R. (2021).

Las pruebas unitarias se concentran en cada unidad del código fuente, como en los métodos, los objetos, las clases, o cualquier otra parte del código. Luego las pruebas de integración tienen como enfoque el diseño y la construcción de la arquitectura del software. Les siguen las pruebas de validación, que se concentran en validar que los requerimientos se cumplan, enfrentando cada requerimiento al software y observando su cumplimiento. Finalmente, las pruebas del sistema sirven para evaluar el software y otros elementos como un conjunto, con mayor amplitud que las etapas anteriores (Pressman, R. S., Maxim, B. R. 2021).

Existen dos tipos de enfoque en las pruebas: Pruebas de caja blanca y pruebas de caja negra. Dentro de ambos enfoques existen diferentes estrategias y planes que se pueden llevar a cabo para conseguir el objetivo: Buscar errores del sistema (O comprobar la falta de errores).

Las pruebas de caja blanca son “una filosofía de diseño de casos de prueba que utiliza la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba.” (Pressman, R. S., Maxim, B. R. 2021). Las pruebas diseñadas bajo este concepto garantizan que se hayan usado al menos una vez todas las rutas independientes de cada módulo, que se ejecuten todas las decisiones lógicas en ambos valores (Verdadero y falso), que se ejecuten todos los ciclos llegando a sus límites, y que se validen las estructuras de datos internas. Las pruebas de caja negra se enfocan en lo establecido dentro de los requerimientos funcionales. Pressman, R.S. y Maxim, B.R. (2021) indican que las pruebas de caja negra buscan encontrar errores en lo siguiente: “(1) funciones incorrectas o faltantes, (2) errores de interfaz, (3) errores en las estructuras de datos o en el acceso a bases de datos externas, (4) errores de comportamiento o rendimiento y (5) errores de inicialización y terminación.” Suponen un complemento a las pruebas de caja blanca y buscan encontrar otros tipos de errores en el sistema, prestando especial atención a si el sistema cumple con los requerimientos funcionales.

A lo anteriormente descrito se le llama “Verificación y Validación”. Verificar consiste en un conjunto de tareas que aseguran que el software implemente de manera correcta sus funciones, mientras que validar consiste en otro conjunto de tareas que buscan asegurar el cumplimiento de los requerimientos del cliente (Pressman, R. S., Maxim, B. R. 2021). La validación suele suceder en presencia del cliente, pudiendo incluso ser necesario que el mismo cliente lleve a cabo las pruebas que considere pertinente y pueda darse por satisfecho con el proyecto que le es entregado.

2.7 SOPORTE DEL SOFTWARE

La última fase del ciclo de vida del software consiste en mantener su buen funcionamiento mientras está en uso. Una manera de lograr esto es por medio del mantenimiento del software,

y Hilard, Vincent (2020) define que dar mantenimiento es “efectuar modificaciones en el sitio sin que estas alteren su buen funcionamiento o sus funcionalidades básicas. Tiene como objetivo rectificar anomalías y tener en cuenta las evoluciones solicitadas por los usuarios.”. También indica que hay dos tipos de mantenimiento: correctivo y evolutivo.

Sin embargo, Pressman, R. S., Maxim, B. R. (2021) definen cuatro tipos de mantenimiento: Correctivo, adaptativo, perfectivo y preventivo.

El mantenimiento correctivo consiste, como su nombre indica, en corregir fallos del sistema de manera reactiva. El mantenimiento adaptativo consiste en modificar reactivamente el software para mantenerlo utilizable, aún en entornos que cambian. El mantenimiento perfectivo sucede cuando de manera proactiva se modifica el software con el objetivo de introducir nuevas funcionalidades o mejoras. Por último, el mantenimiento preventivo es similar al correctivo, pero sucede de manera proactiva, es decir, antes que los usuarios finales encuentren las fallas que se pretenden erradicar.

Para asegurarse que las tareas de mantenimiento se software se lleven a cabo de la mejor manera posible, es indispensable contar con una documentación adecuada, llevada a cabo durante las etapas anteriores del proceso de desarrollo. Cada una de ellas incluyen la elaboración y uso de diferentes documentos. Por ejemplo, en la etapa de requerimientos se necesita un instrumento para registrar las historias de usuario, lo cual se convierte en documentación útil para el análisis. Igualmente, en la fase de pruebas se elaboran documentos donde se establecen los parámetros a probarse y se registran los resultados (Plan de test). Pero una vez finalizado y entregado, el proyecto queda en manos de los usuarios quienes deben aprender a utilizar la solución de software que les es entregada. Una manera de ayudar a los futuros usuarios de la solución a poder aprender a usar el software es elaborar manuales de usuario.

Los manuales de usuario son parte de la documentación del proyecto. Su objetivo es lograr que los usuarios finales entiendan cómo hacer uso de la solución, sepan cómo resolver problemas comunes en su uso que no requieran de cambios en el código, y que puedan sacar el máximo provecho de la solución. Los manuales de usuario incluyen la documentación con los pasos para la instalación, cuando sea necesario instalar algo, y los requerimientos mínimos de hardware y software para usar el sistema.

CAPÍTULO III: MARCO METODOLÓGICO

Este capítulo explica la manera de realizar el proyecto y la forma en que se obtuvieron los datos necesarios, indicando los instrumentos y técnicas empleados para su obtención.

3.1 TIPO DE INVESTIGACIÓN

El tipo de investigación para este proyecto es la investigación de campo. Consiste en una investigación aplicada para definir, analizar, interpretar y proponer una solución a un problema. En este proyecto la recolección de datos para la identificación y análisis de una problemática se llevó a cabo en la empresa ProLine CR, y la solución propuesta se le da a dicha empresa. Esto encaja con la definición dada por Pimienta Prieto, J. H., Estrada Coronado y R. M., de la Orden Hoz, A. (2018).:

> Esta modalidad de investigación consiste en recabar la información a partir del análisis directo del entorno y de la realidad circundante. Para efectuarla es necesario acudir al espacio y contexto específicos en que tiene lugar el fenómeno por investigar y entonces obtener los datos. Los datos son llamados primarios, porque son recabados directamente de los informantes, ya sea por medio de entrevistas, aplicación de cuestionarios, encuestas o mediante el registro de la observación. (p.10)

3.1.1 Enfoque de la investigación

Esta investigación ha sido realizada bajo el enfoque cualitativo. Muñoz Razo, C. (2015) define lo siguiente:

> Son tesis cuya investigación se fundamenta más en estudios descriptivos, interpretativos e inductivos (que van de lo particular a lo general) y se utilizan para analizar una realidad social empleando un enfoque subjetivo, con el propósito de explorar, entender, interpretar y describir el comportamiento de la realidad en estudio, no necesariamente para comprobarla, sino para describirla e interpretarla. (p.30)

De acuerdo con lo anteriormente citado, en este proyecto la solución brindada a la problemática descrita no es comprobada posteriormente, y se da como una interpretación de lo que puede usarse para resolver dicho problema. Esta interpretación se lleva a cabo no solo por el autor del proyecto sino también por las personas que forman parte de la empresa ProLine CR a través de la validación de la herramienta de software recibida como parte de este proyecto. Esto también encaja con lo mencionado por Muñoz Razo, C. (2015):

> El planteamiento del problema parte de una variedad de concepciones y experiencias de una realidad que se busca entender e interpretar, pero no comprobar. Si bien se establece una hipótesis de trabajo, ésta no necesariamente se comprueba con datos de medición numérica o con interpretación estadística. Sin embargo, con los resultados obtenidos, es posible interpretar y explicar la realizada estudiada. (p.30)

Otro aspecto del enfoque cualitativo es que los datos que se recopilan no son de naturaleza numérica por lo cual se toma en cuenta la opinión de los involucrados. De acuerdo con Muñoz Razo, C. (2015) "De esta manera, se pretende entender la realidad que se estudia a través de esas aportaciones subjetivas, o bien, por medio de interpretaciones, también subjetivas, que de ellas hace el propio investigador."

3.2 FUENTES DE LA INFORMACIÓN

Las fuentes de la información son los medios de los cuales se extrae conocimiento tanto teórico como práctico para el proyecto, y sobre la cual sostener todo lo que se lleva a cabo en el mismo. Se definen las fuentes primarias, las fuentes secundarias y los sujetos de información.

3.2.1 Fuentes primarias

Las fuentes primarias están "integradas por testimonios o documentos originales, periódicos, documentos oficiales, anales, memorias o diarios." (Pimienta Prieto, J. H., Estrada Coronado,

R. M., de la Orden Hoz, A., 2018). Todas estas fuentes son de primera mano y no son el resultado del análisis o interpretación de otros autores.

Para este proyecto se utilizaron las siguientes fuentes primarias:

- Libros de texto:
    - De programación de software.
    - De ingeniería de software y planificación de proyectos.
    - De análisis y diseño de sistemas de información.
    - De sistemas de bases de datos.
    - De comercio electrónico o e-commerce.
    - De proyectos de investigación.
- Revistas universitarias.
- Entrevistas a los interesados en el proyecto.

3.2.2 Fuentes secundarias

Las fuentes secundarias suelen tomar como referencia otras investigaciones o fuentes, y elaborar sus conclusiones e interpretaciones a partir de aquellas. De acuerdo con Muñoz Razo, C. (2015) "La investigación que utiliza información de segunda mano tiene la ventaja de que está más documentada, pues toma varias fuentes para complementar y se apoya en la seriedad metodológica.".

En este proyecto las fuentes secundarias utilizadas fueron:

- Periódicos especializados en finanzas (En su versión digital).
- Artículos de investigación de empresas en los sectores de tecnología, productividad y finanzas.

3.2.3 Sujetos de información

Los sujetos de información son todas las personas que fueron contactadas con el fin de recopilar información útil para la realización del proyecto.

Para este proyecto los sujetos de información fueron los siguientes:

- Gerente General de la compañía. Es el encargado también de todas las operaciones de la compañía.
- Accionista principal de la compañía. A cargo principalmente de la información financiera y legal.

3.3 TÉCNICAS Y HERRAMIENTAS DE RECOLECCIÓN DE DATOS

Las técnicas de recolección de datos son las diferentes maneras de recolectar la información necesaria para desarrollar el proyecto de investigación. Las herramientas que se utilizan para recolectar estos datos pueden ser materiales escritos bien estructurados como formularios y plantillas, o pueden ser incluso conversaciones informales con las personas que tienen influencia en el proyecto y su transcripción.

Pimienta Prieto, J. H., Estrada Coronado, R. M. y de la Orden Hoz, A.(2018) definen lo siguiente:

> Las técnicas de investigación son procedimientos diversos, esenciales para la investigación científica, por medio de las cuales es posible recabar y organizar la información. Toda técnica de investigación debe cumplir con los siguientes objetivos:
>
> • Aportar estrategias para reunir y organizar la información.
>
> • Permitir el manejo y procesamiento de los datos reunidos.

- Brindar elementos para orientar el proceso de construcción de conocimientos con base en dicha información. Tanto el método como el diseño sobre los cuales se guía la investigación determinarán el tipo de técnicas que serán aplicadas.

A continuación, se describen las técnicas seleccionadas para la recolección de datos dentro de este proyecto.

3.3.1 Entrevista

Las entrevistas son conversaciones que se llevan a cabo entre el investigador y otras personas de interés para la elaboración del proyecto. Permiten conocer la opinión de las personas involucradas en la problemática que se espera solucionar o estudiar por medio del proyecto. En ocasiones también pueden servir para encontrar asesoría en la elaboración del proyecto por parte de especialistas en temas que van más allá de lo que el investigador domina pero que resultan esenciales para conseguir los objetivos propuestos.

De acuerdo con Pimienta Prieto, J. H., Estrada Coronado, R. M. y de la Orden Hoz, A. (2018) la entrevista es "Basada en una serie de preguntas que el investigador formula de manera directa a una o varias personas, o bien, conversa con ellas, con la finalidad de conocer su opinión y experiencia acerca del tema o problema estudiado."

Durante la elaboración de este proyecto se llevaron a cabo entrevistas con el gerente general de la empresa ProLine CR, encargado de las operaciones diarias de la empresa, así como con la accionista principal, encargada de cierta información financiera y legal. Durante estas entrevistas se hicieron algunas preguntas específicas preparadas con anticipación con el propósito de conocer los requerimientos necesarios para diseñar la solución de software a proponerse.

3.3.2 Historias de usuario

Como se describió en 2.2.1, las historias de usuario se elaboran en una plantilla. La información allí contenida se extrae de conversaciones con los diferentes interesados en el proyecto, los cuales pueden ser usuarios finales, administradores, encargados de dar soporte, e inclusive la gerencia o los patrocinantes del proyecto.

En este proyecto las historias de usuario se elaboraron siguiendo la plantilla antes descrita en 2.2.1, la tabla 3 (Plantilla para la definición de historias de usuario).

3.3.3 Prototipado

El prototipado consiste en la puesta en marcha de una versión inicial del software que aún está en desarrollo. Como se explicó en 2.2.3, el objetivo de esta puesta en marcha es mostrar el software a los usuarios y a otros interesados para que puedan dar sus impresiones de lo que se ha realizado hasta el momento.

Como parte del proceso de validación, se toman los comentarios de dichas personas para realizar correcciones, mejoras y afinar detalles que idealmente no se salgan del presupuesto planificado. De hecho, según Sommerville (2011) el uso de prototipos en fases tempranas del desarrollo permite a los usuarios probar funciones específicas del software antes de que el equipo de trabajo se comprometa en la producción de características de alto costo. Por tanto, los prototipos deben ser de rápida elaboración y de bajos costos para ser viables.

En este proyecto el prototipo realizado permitió validar dos partes importantes para el usuario y para el éxito de la solución brindada: las interfaces visuales del usuario, y los controles para agregar nueva mercancía al inventario.

3.3.4 Testing

De acuerdo con lo descrito en 2.5, esta etapa del desarrollo consiste en la ejecución de pruebas en el software antes de su entrega final, principalmente para detectar fallos o requerimientos aún no satisfechos.

Según con el plan de testing realizado durante este proyecto, se realizaron varias pruebas para validar el buen funcionamiento de los diferentes elementos de la solución brindada. La plantilla propuesta en 2.5 (Tabla 5, Plantilla de plan de test) se utilizó para planificar las pruebas y para registrar los resultados de dichas pruebas.

3.4 VARIABLES DE LA INVESTIGACIÓN

De acuerdo con la Oficina de Integridad de Investigación (ORI por sus siglas en inglés) (Sin fecha) la investigación debe tener como propósito describir "los cambios que ocurren de manera natural en el mundo o que son causados debido a una manipulación. Las variables son nombres que damos a las variaciones que deseamos explicar."

Las variables a observar se desprenden de los objetivos propuestos en 1.3.2.

A continuación, se muestran en la tabla 6 (Variables) las variables de este proyecto. En esta tabla se incluyen los objetivos, la variable asociada que se desprende de cada objetivo, y la descripción de dicha variable.

Tabla 6.

Variables.

Objetivo Específico	Variable asociada	Descripción
Determinar los requerimientos de una aplicación web para la empresa ProLine CR con la	Requerimientos de la empresa ProLine para la aplicación web.	Las necesidades que tiene la empresa ProLine dictan las características de la solución

finalidad de definir las características de la aplicación a implementar.		propuesta. Éstas se documentan y analizan.
Elaborar el diseño lógico y físico del software y de la base de datos, usando el resultado del análisis de los requisitos recopilados para llevar a cabo una implementación exitosa que satisfaga todos los requisitos planteados.	Diseño lógico y físico del software y la base de datos.	El diseño previo a la implementación, el cual permite el desarrollo de la misma, de manera que se satisfagan todos los requerimientos.
Implementar una aplicación web con base en el diseño hecho según los requisitos recopilados para permitir una mejor comunicación con los clientes y potenciales clientes de ProLine CR.	Aplicación web para mejorar la comunicación con clientes y potenciales clientes de la empresa.	La aplicación web implementada y funcional, de acuerdo con lo establecido en la fase de diseño.
Determinar y ejecutar las pruebas de la aplicación web junto a los usuarios que lo utilizarán, para verificar y validar el correcto funcionamiento de la aplicación web, así como el cumplimiento de los requerimientos.	Pruebas y validación de la aplicación web.	Ejecución de pruebas de acuerdo con el plan de testing y validación por parte de los usuarios antes de la entrega final.

Redactar el manual de usuario final llevando a cabo la documentación enfocada al uso de la aplicación web para que los colaboradores de ProLine CR se capaciten en el uso y mantenimiento de esta.	Manual de usuario con instrucciones de uso para los usuarios finales.	El manual de usuario, que proporciona ayuda para el usuario y sirve como fuente de consulta a medida que se familiariza con la solución y la usa para cumplir sus objetivos.

3.5 DISEÑO DE LA INVESTIGACIÓN

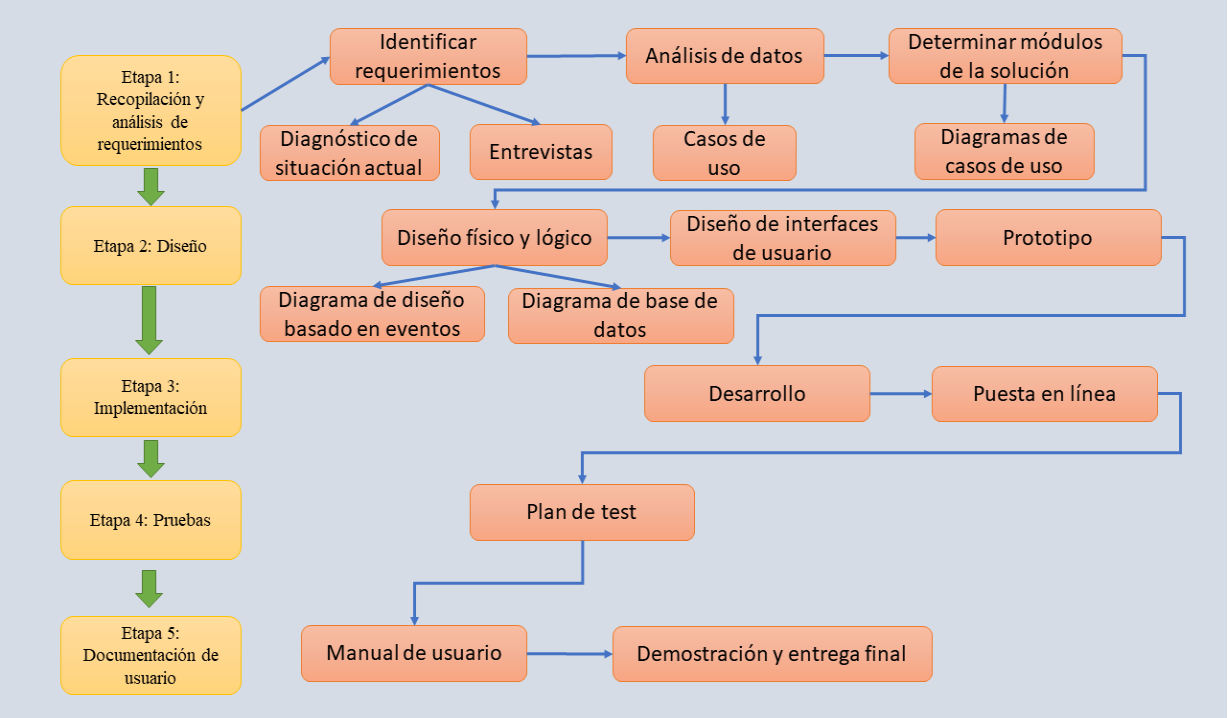
El diseño de la investigación recoge todos los elementos de esta y los muestra en orden. Cada una de las partes de la investigación ocurren de conformidad con lo diseñado. De acuerdo con la Oficina de Integridad de la Investigación (Sin fecha) “el diseño de la investigación ofrece los componentes y el plan para llevar a cabo el estudio de manera satisfactoria. El diseño de la investigación es la “columna vertebral” del protocolo de investigación.”.

Este proyecto de investigación consta de cinco etapas, descritas de manera resumida en el gráfico a continuación, la figura 12 (Diseño de la investigación). Dichas etapas son: Recopilación y análisis de requerimientos, diseño, implementación, pruebas y documentación de usuario.

La metodología utilizada en este proyecto es la metodología en cascada, de acuerdo con lo descrito en 2.1.1.

Figura 12.

Diseño de la investigación.



3.5.1 Recopilación y análisis de requerimientos

Durante esta etapa se inicia con la planificación de las entrevistas a los futuros usuarios del sistema, luego se realizan dichas entrevistas y se elaboran las historias de usuario. Después de recopilada, la información es analizada.

Las herramientas utilizadas fueron:

- Entrevistas: se entrevistó al Gerente General y a la accionista principal de la empresa ProLine CR, quienes son los usuarios principales del sistema. Ellos describieron la problemática que enfrenta la empresa y también describieron la mayoría de los requerimientos que un sistema tendría que cubrir para solventar dicha problemática.
- Plantilla de historia de usuario: La información recopilada en las entrevistas fue organizada haciendo uso de una plantilla de historia de usuario, y desde ese punto comienza el análisis de dicha información. Fue necesario hacer un uso repetido de la plantilla, de acuerdo con los diversos requerimientos identificados.

- Diagramas de caso de uso: Como parte del análisis de requerimientos se efectuaron los diagramas de caso de uso a partir de la información recopilada. Esto sirvió como referencia rápida durante las siguientes etapas.

3.5.2 Diseño

Durante la etapa de diseño se tomaron los resultados del análisis de requerimientos para realizar el diseño lógico de la solución de software: Se determinó qué hará la aplicación, cuál arquitectura se usará, cuáles interfaces se usarán junto con su estructura y sus elementos visuales, y se realizó el flujo de eventos dentro de dichas interfaces.

Luego se realizó el diseño físico: modelado de la base de datos con su respectivo diagrama. Se escogieron el lenguaje de programación y el framework a utilizar. Las herramientas usadas fueron:

- Modelado basado en eventos: Se elaboraron diagramas para los eventos que ocurren en las vistas dentro de la aplicación, que permitieron tener una visualización de los diferentes elementos con los que los usuarios interactúan y los resultados de dichas interacciones.
- Diagrama de modelo de base de datos: Se realizó un diagrama para el diseño de la base de datos.

3.5.3 Implementación

Esta etapa del proyecto consistió en la elaboración del código para la aplicación web. Las herramientas utilizadas fueron:

- Visual Studio 2022: Es un entorno de desarrollo integrado (IDE) que permite la edición de código y cuenta con herramientas para su implementación. Se utilizó para escribir y editar el código fuente de la aplicación, escrito en C# y HTML (véase 2.5.1), en el

framework .NET Core de Microsoft (véase 2.5.2) que facilita la implementación de aplicaciones web con la arquitectura Modelo-Vista-Controlador (véase 2.5.3).

- Servicio de hosting de SmarterASP: Hosting es un servicio que consiste en almacenar el código y los archivos de una aplicación web en servidores, para ejecutarlos y darles acceso a los usuarios de la web a dicha aplicación. Para este proyecto se utilizaron los servicios de hosting de la empresa SmarterASP.
- phpMyAdmin: Es una herramienta en línea para administrar bases de datos y ejecutar sentencias en SQL. Utiliza MySql como sistema de gestión de bases de datos, y se utiliza a través de un navegador web. El hosting de SmarterASP utiliza phpMyAdmin para administrar las bases de datos.

3.5.4 Pruebas

La etapa de pruebas se lleva a cabo después del desarrollo o la implementación de la solución de software. De acuerdo con lo explicado en 2.5, se llevaron a cabo diferentes pruebas para verificar no solo la integridad de la solución de software, sino también el cumplimiento de los requerimientos planteados en la fase inicial del proyecto. Las herramientas y técnicas utilizadas fueron:

- Plantilla de plan de test: Se describe anteriormente en 2.6, y se muestra un ejemplo en la tabla 5 (Plantilla de plan de test). Se utilizó la plantilla para elaborar los planes de test y seguir la planificación durante las pruebas, así como para registrar el resultado de las mismas.
- Pruebas unitarias: Se realizaron revisiones y depuración del código fuente y sus partes.
- Pruebas de integración: Se revisó que el diseño fuera funcional y cumpliera con los estándares propuestos en la fase de diseño.

- Pruebas de validación: Se tomaron los requerimientos recopilados en la fase inicial del proyecto y se validó junto con los usuarios el cumplimiento de los mismos.
- Pruebas del sistema: Las últimas pruebas en realizarse, en las cuales se evaluaron todas las partes del sistema y se verificó su buen funcionamiento.

3.5.5 Documentación de usuario

Última etapa del proyecto. En esta, se elaboró la documentación final para realizar la entrega formal de la solución de software a los usuarios y consistió en un manual de usuario. Este manual describe las funciones del sistema y muestra una guía paso a paso de cómo usarlo. El manual de usuario contiene imágenes obtenidas de la aplicación, junto con explicaciones de cada parte con la que el usuario tiene interacción.

Las partes del manual de usuario son:

- Portada.
- Índice.
- Introducción.
- Contenido.

El contenido se divide en los siguientes temas:

- La descripción detallada de cada sección o módulo de la aplicación. Por ejemplo, la sección de inventario, con los pasos a seguir en sus diferentes vistas para realizar las tareas necesarias y una descripción de lo que se puede hacer en cada vista.
- Qué hacer en caso de problemas. Si bien la solución brindada pasó por un proceso de verificación y se entregó con total funcionalidad, es posible que a veces los usuarios encuentren inconvenientes derivados de los cambios en la tecnología usada o por elementos que escapan de su control (Por ejemplo, cambios en el hosting). En esta

sección del documento hay soluciones para los problemas más comunes y sencillos que se previó puedan surgir.

Las herramientas utilizadas fueron:

- Microsoft Word: Es un procesador de palabras muy utilizado para la elaboración de documentos. El manual se elaboró usando esta aplicación. Así mismo el enlace al archivo del manual se encuentra en la aplicación web, de manera que los usuarios pueden acceder al mismo en cualquier momento.
- Google Drive: Servicio de almacenamiento de archivos en la nube, proporcionado por Google. Se utilizó para guardar el manual de usuario, y facilitar su actualización a medida que la aplicación web sufra cambios en el futuro.

3.6 MATRIZ DE COHERENCIA

La matriz de coherencia es un instrumento para mostrar de manera organizada y resumida el proceso metodológico utilizado durante el desarrollo de este proyecto. Se muestra en una tabla (Tabla 7, Matriz de coherencia) cada objetivo junto con el o los entregables producidos para lograr dicho objetivo, la fase de la metodología del proyecto en que se trabaja para alcanzar el objetivo, los instrumentos utilizados en dicha fase y los temas del Marco Teórico que se relacionan con la consecución de dicho objetivo.

Tabla 7.

Matriz de coherencia.

Objetivo	Entregable	Fase de la metodología del proyecto	Instrumentos	Temas relacionados

Determinar los requerimientos de una aplicación web para la empresa ProLine CR con la finalidad de definir las características de la aplicación a implementar.	·Documento con las historias de usuarios y los diagramas de caso de uso.	Recopilación y análisis de requerimientos.	· Plantilla de historia de usuario. · Diagrama de caso de uso.	· Métodos para levantamiento de requerimientos y análisis. · Historias de usuario. · Diagramas de caso de uso.
Elaborar el diseño lógico y físico del software y de la base de datos, usando el resultado del análisis de los requisitos recopilados para llevar a cabo una implementación exitosa que satisfaga todos los requisitos planteados.	· Diagramas de diseño basado en eventos. · Diagrama de modelo de base de datos.	Diseño.	· Diagramas de diseño basado en eventos. · Diagramas de modelo de base de datos.	· Modelado basado en eventos. · Diseño de base de datos. · Diseño de interfaces de usuario.

Implementar una aplicación web con base en el diseño hecho según los requisitos recopilados para permitir una mejor comunicación con los clientes y potenciales clientes de ProLine CR.	· Prototipo de la aplicación web. · Aplicación web.	Implementación.	· Aplicación web hospedada en SmarterASP y accesible usando internet.	· E-commerce. · Lenguajes de programación. · Frameworks para el desarrollo. · Modelos de arquitectura.
Determinar y ejecutar las pruebas de la aplicación web junto a los usuarios que lo utilizarán, para verificar y validar el correcto funcionamiento de la aplicación web, así como el cumplimiento de los requerimientos.	·Documento con las plantillas del plan de test incluyendo los resultados de las pruebas.	Pruebas y validación.	· Plantillas de plan de test.	· Técnicas para probar y validar software

Redactar el manual de usuario final llevando a cabo la documentación enfocada al uso de la aplicación web para que los colaboradores de ProLine CR se capaciten en el uso y mantenimiento de esta.	· Manual de usuario final.	Documentación.	·Documento del manual de usuario final.	· Soporte de software.

CAPITULO IV DIAGNÓSTICO DE LA SITUACIÓN ACTUAL

Este capítulo (Capítulo IV) detalla la situación actual de la empresa ProLine, describiendo la problemática y proporcionando la información que sirvió para la propuesta de la solución a dicha problemática.

Muñoz Razo, C. (2015) menciona que después de identificar la problemática, "el siguiente paso es emitir un diagnóstico actualizado sobre la problemática encontrada y el funcionamiento del sistema, identificando todos los aspectos que rodean al fenómeno observado"

Toda la información fue obtenida haciendo uso de los instrumentos de recolección de datos mencionados en los capítulos anteriores y se anexan al final de este documento. Éstos sirven para realizar el análisis posterior, y el diseño de la solución de software que sigue a la etapa de análisis, las cuales se incluyen en este capítulo. La técnica utilizada para recolectar información fue la entrevista, y se llevaron a cabo varias entrevistas con los colaboradores de la empresa.

4.1 DESCRIPCIÓN DE LA SITUACIÓN ACTUAL

Esta sección del proyecto muestra la situación actual de la empresa, describe su problemática y la información brindada por los colaboradores de la empresa ProLine CR.

4.1.1 Diagnóstico operativo

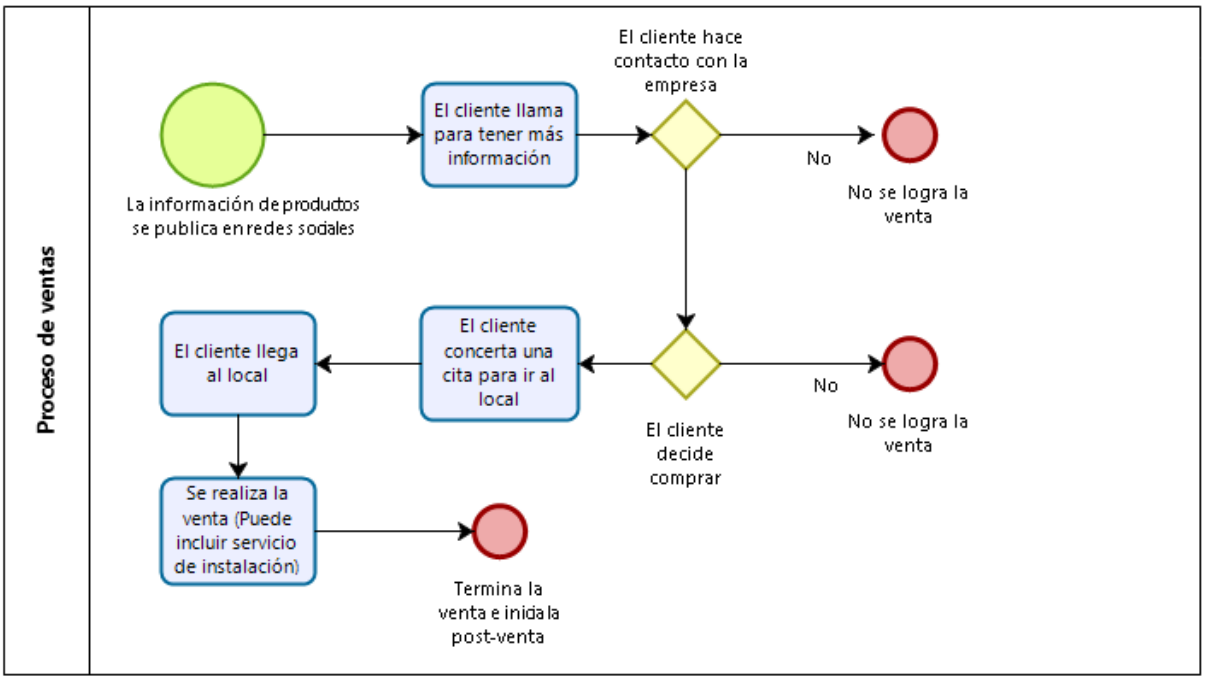
La información descrita en este apartado se obtuvo a través de entrevistas con los colaboradores de la empresa, realizadas de manera no estructurada sino más bien por medio de conversaciones informales.

ProLine es una entidad comercial, y tal como se describe en 1.1.1, se dedica a la venta de sistemas de sonido y el servicio de instalación de éstos. A continuación, se describe el flujo del proceso de ventas de la empresa en la Figura 13 (Proceso de ventas). Se muestra cómo el proceso inicia cuando el cliente encuentra la información publicada en redes sociales, y luego se comunica por el mismo medio para tener más información de los productos y en caso de querer

realizar la compra, concertar una cita en el local para retirar el producto o llevar a cabo la instalación de este, tras lo cual la venta culmina e inicia el proceso postventa.

Figura 13.

Proceso de ventas.



El control de pedidos y de inventarios se lleva a cabo en una hoja de cálculo almacenada en la computadora que hay en el local, por lo que este control solo puede llevarse a cabo cuando se está en el local. El gerente general es también el encargado de todo el proceso de ventas y de realizar las instalaciones y cuenta con un ayudante para realizar algunas instalaciones. No hay un horario de trabajo fijo, ya que depende de las citas que los clientes tengan, o de la llegada de nueva mercancía para ser acomodada dentro del local.

El gerente general suele atender las solicitudes de más información por parte de clientes y potenciales clientes desde donde se encuentre (Ya que no siempre se encuentra en el local) a través de varios canales de comunicación: Facebook, Instagram, WhatsApp y llamadas directas a la línea telefónica de la empresa. En ocasiones hay intentos de comunicación en horas en las

que no se está laborando por lo que no se logra transmitir información rápidamente, entorpeciendo la comunicación y retrasando el proceso de venta. Cuando esto último ocurre, en ocasiones la venta se pierde porque el posible cliente busca más información en otros lugares y se compromete con otros vendedores.

La empresa también vende productos al por mayor a otras empresas ubicadas fuera del territorio del Gran Área Metropolitana. Cuando el gerente general está en negociaciones con clientes que compran al por mayor, a veces no está seguro del inventario que posee y si no está en el local, entonces no puede consultar el archivo que maneja en la computadora del local únicamente para tener certeza de los productos en existencia, lo cual ha generado retrasos en el proceso de ventas o en el peor de los casos, malentendidos con los clientes.

4.1.2 Diagnóstico técnico

La empresa cuenta con una computadora con conexión a internet en el local donde principalmente se lleva a cabo su actividad comercial. Es en este local donde se encuentra casi toda la mercancía, se realizan las instalaciones de reproductores y sistemas de sonido en los vehículos de los clientes y donde se lleva a cabo la mayor parte del trabajo relacionado a publicar información acerca de los productos ofrecidos en redes sociales.

El gerente general cuenta también con un teléfono celular con conexión a internet, desde donde atiende las solicitudes de información por parte de clientes y potenciales clientes.

En el domicilio del gerente general él utiliza una computadora personal con conexión a internet en ocasiones para llevar a cabo tareas relacionadas a las operaciones comerciales de la empresa.

4.1.3 Diagnóstico de percepción

La información descrita en este apartado se obtuvo por medio de una entrevista con el gerente general de la empresa y con la accionista principal.

La problemática de la empresa gira en torno a la capacidad de brindar información a los clientes y potenciales clientes en todo momento. Es precisamente esa la necesidad que se quiere abordar por parte de la empresa, particularmente en poder mostrar la información de los productos disponibles e incluso poder completar un pedido sin necesidad de interactuar directamente con los clientes, de acuerdo con la respuesta obtenida en la primera pregunta de la entrevista.

También quieren cambiar la situación actual en la cual se ven obligados a responder mensajes que reciben por WhatsApp o llamadas telefónicas a cualquier hora, dado que es la única manera que tienen de concretar un pedido de mercancía. Les gustaría entregar a los clientes la posibilidad de que realicen compras en línea, para reducir la dependencia de comunicación directa y personal a todas horas.

4.2 RECOLECCIÓN Y ANÁLISIS DE DATOS

Una vez establecida la posibilidad de proponer una herramienta de software como solución a la problemática antes descrita, la primera fase propuesta en este proyecto consiste en la recopilación y análisis de requerimientos, de manera que podría dividirse en dos etapas: En primer lugar, la recopilación de información y, en segundo lugar, el análisis de la información recopilada.

Dentro de los objetivos planteados, el primer objetivo se desarrolla en esta sección: terminar los requerimientos de una aplicación web para la empresa ProLine CR con la finalidad de definir las características de la aplicación a implementar. Esta sección corresponde a la fase de recopilación y análisis de requerimientos. La variable para analizar en esta fase es: Requerimientos de la empresa ProLine para la aplicación web.

4.2.1 Recopilación de requerimientos

En este apartado se presentan los resultados obtenidos a través de la entrevista realizada a los colaboradores de ProLine, llevada a cabo en una sola sesión. La totalidad de la entrevista se encuentra en el apéndice 2. Los requerimientos obtenidos se resumen de la siguiente manera:

- Mostrar un catálogo en línea.
- Acceso a través de internet para dar mantenimiento al catálogo, restringido a los administradores.
- Posibilidad de realizar pedidos de manera remota y sin asistencia.
- Recolectar información del usuario para la realización del pedido.
- Posibilidad de ser contactados de diferentes maneras a través de la información brindada en la plataforma.

4.2.2 Análisis de los datos

A partir de los datos recopilados durante la entrevista se realizó un análisis de los requerimientos y se utilizó la plantilla indicada anteriormente en 2.2.1. Cada caso de uso se muestra y se describe a continuación:

Escenario de caso de uso 1: Inicio de sesión.

En el siguiente escenario de caso de uso se describe el requerimiento de tener un módulo para iniciar sesión, de manera que los usuarios administradores puedan tener acceso a las funciones exclusivas para ellos.

Tabla 8.

Escenario de caso de uso 1.

Número	1
Caso de uso	Inicio de sesión

Actor principal	Administrador.
Meta en contexto	Validar las credenciales para iniciar sesión como administrador y tener acceso a las áreas únicas para los administradores.
Precondiciones	El usuario existe en la base de datos del sistema.
Desencadenante	El administrador desea dar mantenimiento al catálogo de productos, revisar los pedidos o dar mantenimiento a los usuarios.
Escenario	El administrador escribe su nombre de usuario y contraseña para iniciar sesión.
Excepciones	Se pide contactar a un administrador para solicitar ayuda.
Prioridad	Alta.
Disponibilidad	Desde el prototipo.
Frecuencia de uso	Media.
Canal para el actor	Página web.

Escenario de caso de uso 2: Mantenimiento de catálogo

En el siguiente escenario de caso de uso se describe el requerimiento para un módulo donde se pueda dar mantenimiento al catálogo de productos, únicamente accesible para los administradores del sitio.

Tabla 9.

Escenario de caso de uso 2.

Número	2
Caso de uso	Mantenimiento del catálogo.

Actor principal	Administrador.
Meta en contexto	Crear un perfil de cada producto con información pertinente o darles mantenimiento.
Precondiciones	Se valida el usuario administrador por medio de un usuario y una contraseña.
Desencadenante	Agregar un nuevo producto o modificar información de los productos existentes.
Escenario	El administrador observa el catálogo de productos y puede agregar un producto nuevo o ingresar al perfil de los productos existentes.
Excepciones	Se pide al usuario iniciar sesión como administrador.
Prioridad	Alta.
Disponibilidad	Desde el prototipo.
Frecuencia de uso	Alta.
Canal para el actor	Página web.

Escenario de caso de uso 3: Catálogo

En este escenario de caso de uso se indica el requerimiento de un catálogo de productos para que los clientes puedan ver la información de estos y seleccionar productos para que sean colocados en el carro de compras.

Tabla 10.

Escenario de caso de uso 3.

Número	3
Caso de uso	Catálogo.
Actor principal	Cliente.
Meta en contexto	Tener acceso a los perfiles de cada producto con información pertinente.
Precondiciones	Ninguna.

Desencadenante	Querer información de los productos que ofrece la empresa y/o realizar una compra.
Escenario	El cliente observa el catálogo de productos y puede agregar el producto a al carro de compras.
Excepciones	Ninguna.
Prioridad	Alta.
Disponibilidad	Desde el prototipo.
Frecuencia de uso	Alta.
Canal para el actor	Página web.

Escenario de caso de uso 4: Carro de compras.

En este escenario de caso de uso se indica el requerimiento del carro de compras. En este módulo los clientes pueden visualizar los productos seleccionados para la compra, sacarlos del carro de compras o finalizar la compra. También podrán llenar un formulario con la información para la facturación y seleccionar el modo de pago. De esta manera se completa el pedido.

Tabla 11.

Escenario de caso de uso 4.

Número	4
Caso de uso	Carro de compras.
Actor principal	Cliente.
Meta en contexto	Hacer un pedido de mercancía y/o servicio de instalación e ingresar la información para la facturación.
Precondiciones	Debe existir al menos un producto seleccionado para ser comprado.

Desencadenante	El cliente desea formalizar una compra.
Escenario	El cliente seleccionó el o los productos que desea comprar y se dispone a realizar la compra. Ingresa la información para la facturación.
Excepciones	Si no hay ningún producto seleccionado se pide que se seleccione al menos uno. Si algún campo del formulario para la facturación está vacío se pide llenarlo.
Prioridad	Alta.
Disponibilidad	Desde el prototipo.
Frecuencia de uso	Media.
Canal para el actor	Página web.

Escenario de caso de uso 5: Información de contacto.

En este escenario de caso de uso se indica la necesidad de un módulo donde se muestre a los clientes y potenciales clientes la información de contacto de la empresa.

Tabla 12.

Escenario de caso de uso 5.

Número	5
Caso de uso	Información de contacto
Actor principal	Cliente.
Meta en contexto	Encontrar la información de contacto de la empresa.
Precondiciones	Ninguna.

Desencadenante	Querer saber dónde se encuentra el local y/o tener los datos de contacto.
Escenario	El cliente entra a ver la información de contacto.
Excepciones	Ninguna.
Prioridad	Alta.
Disponibilidad	Desde el prototipo
Frecuencia de uso	Media.
Canal para el actor	Página web.

Escenario de caso de uso 6: Pedidos.

En este escenario de caso de uso se muestra el requerimiento para un módulo donde se pueden observar todos los pedidos realizados por los clientes. Es de acceso exclusivo para los usuarios administradores.

Tabla 13.

Escenario de caso de uso 6.

Número	6
Caso de uso	Pedidos.
Actor principal	Administrador.
Meta en contexto	Revisar todos los pedidos y tener acceso información editable relacionada a cada pedido.
Precondiciones	Se valida el usuario administrador por medio de un usuario y una contraseña.
Desencadenante	Se agregó un nuevo pedido o se desea modificar información de los pedidos existentes.

Escenario	El administrador observa una lista de los pedidos que los clientes han realizado y editar la información de los mismos, incluyendo un área para comentarios.
Excepciones	Se pide al usuario iniciar sesión como administrador.
Prioridad	Alta.
Disponibilidad	Entrega final.
Frecuencia de uso	Media.
Canal para el actor	Página web.

Escenario de caso de uso 7: Mantenimiento de usuarios.

Este escenario de caso de uso describe el requerimiento de un módulo para dar mantenimiento a los usuarios registrados en el sistema, que son administradores.

Tabla 14.

Escenario de caso de uso 7.

Número	7
Caso de uso	Mantenimiento de usuarios
Actor principal	Administrador.
Meta en contexto	Revisar los usuarios existentes y poder modificar sus credenciales de acceso o eliminarlos.
Precondiciones	Se valida el usuario administrador por medio de un usuario y una contraseña.
Desencadenante	El administrador desea agregar o modificar los otros usuarios existentes.

Escenario	El administrador observa una lista de los demás usuarios y puede modificar sus credenciales de acceso o eliminarlos.
Excepciones	Se pide al usuario iniciar sesión como administrador.
Prioridad	Alta.
Disponibilidad	Entrega final.
Frecuencia de uso	Baja.
Canal para el actor	Página web.

4.2.3 Diagrama de casos de uso

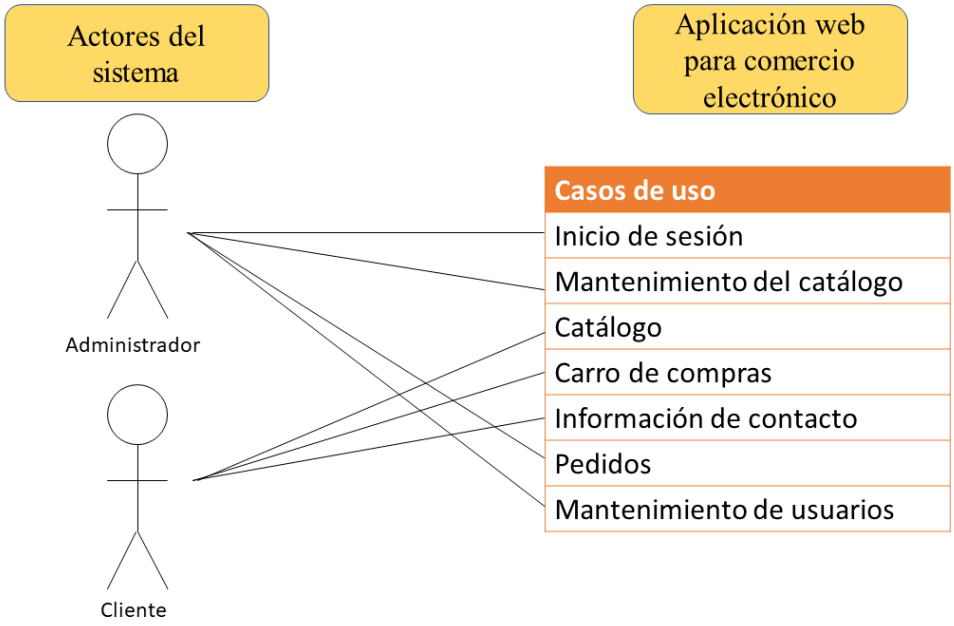
A partir del análisis realizado por medio de los escenarios de casos de uso, es posible determinar dos roles de usuario: El rol de administrador y el rol de cliente. Los usuarios administradores son los que pueden acceder a todos los módulos, incluyendo los módulos que requieren de validación de credenciales: mantenimiento de catálogo, pedidos y mantenimiento de usuarios.

El rol de cliente puede entrar únicamente a los módulos de catálogo, carrito de compras e información de contacto. Si bien puede tener acceso al módulo de inicio de sesión, al no contar con las credenciales de acceso no podrá autenticarse haciendo uso de su función.

Lo descrito anteriormente se puede visualizar de manera rápida en el diagrama de casos de uso, realizado a partir de lo descrito en 2.2.2. y que se ilustra a continuación en la figura 14:

Figura 14.

Diagrama de casos de uso.



4.3 DISEÑO

En esta fase del proyecto se toman los resultados del análisis de requerimientos para realizar el diseño de la solución de software, lo que servirá como marco de trabajo para el desarrollo e implementación de la solución.

Dentro de los objetivos del proyecto, el segundo objetivo se desarrolla en esta sección Elaborar el diseño lógico y físico del software y de la base de datos, usando el resultado del análisis de los requisitos recopilados para llevar a cabo una implementación exitosa que satisfaga todos los requisitos planteados.

La fase que se lleva a cabo en esta sección es la de diseño, y la variable observada es: Diseño lógico y físico del software y la base de datos.

4.3.1 Diagramas de diseño basado en eventos

Cada caso de uso descrito en los escenarios de caso de uso de la sección anterior se desarrolla en un módulo, y para cada módulo se muestran a continuación los diagramas de diseño basado

en eventos, siguiendo la técnica descrita en 2.3.2. Por tanto, se determina la implementación de los siguientes módulos:

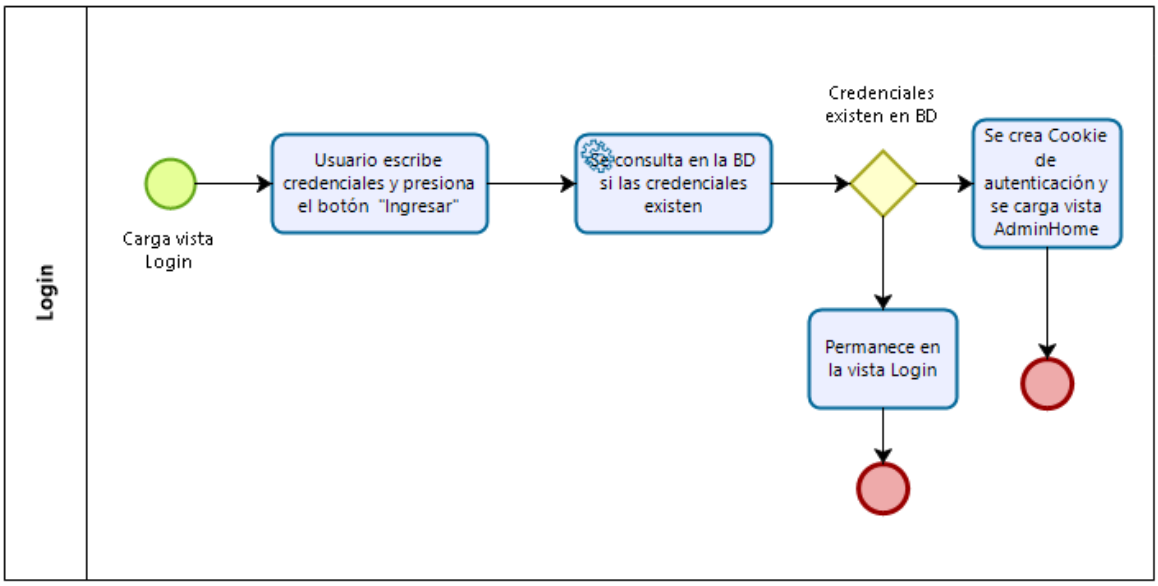
- Inicio de sesión.
- Mantenimiento de catálogo.
- Catálogo.
- Carro de compras.
- Información de contacto.
- Pedidos.
- Mantenimiento de usuario.

4.3.1.1 Diagrama de módulo 1: Inicio de sesión.

A continuación, la figura 15 representa el diagrama de flujo para el módulo 1: Inicio de sesión. Éste inicia cuando se carga la vista Login, en la cual hay un formulario que recoge las credenciales de acceso. El usuario escribe un nombre de usuario y una contraseña, que se validan en la base de datos y si existen entonces se cargan la vista de inicio de los usuarios administradores, llamada AdminHome. De lo contrario, se permanece en la vista Login.

Figura 15.

Diagrama de inicio de sesión.

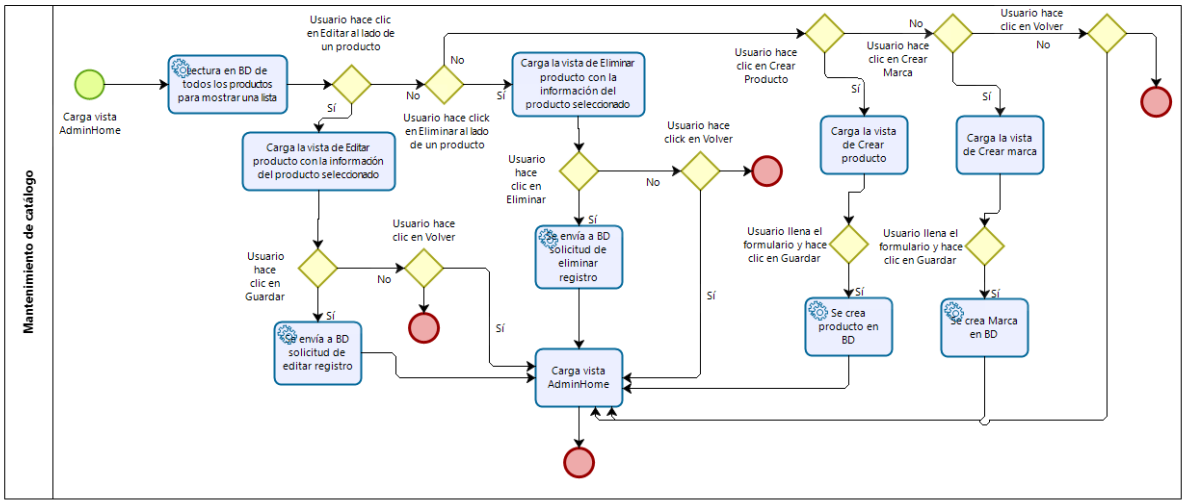


4.3.1.2 Diagrama de módulo 2: Mantenimiento de catálogo.

A continuación, la figura 16 representa el diagrama de flujo para el módulo 2: Mantenimiento de catálogo. Cuando la vista carga, el usuario administrador observa una lista de todos los productos que tiene almacenados, y tiene dos opciones para cada producto: editar y eliminar. Al presionar alguno de ellos, se carga la vista de editar producto o de eliminar producto, respectivamente. También puede crear productos nuevos, y marcas nuevas. Las marcas son un parámetro para los productos.

Figura 16.

Diagrama de mantenimiento de catálogo.

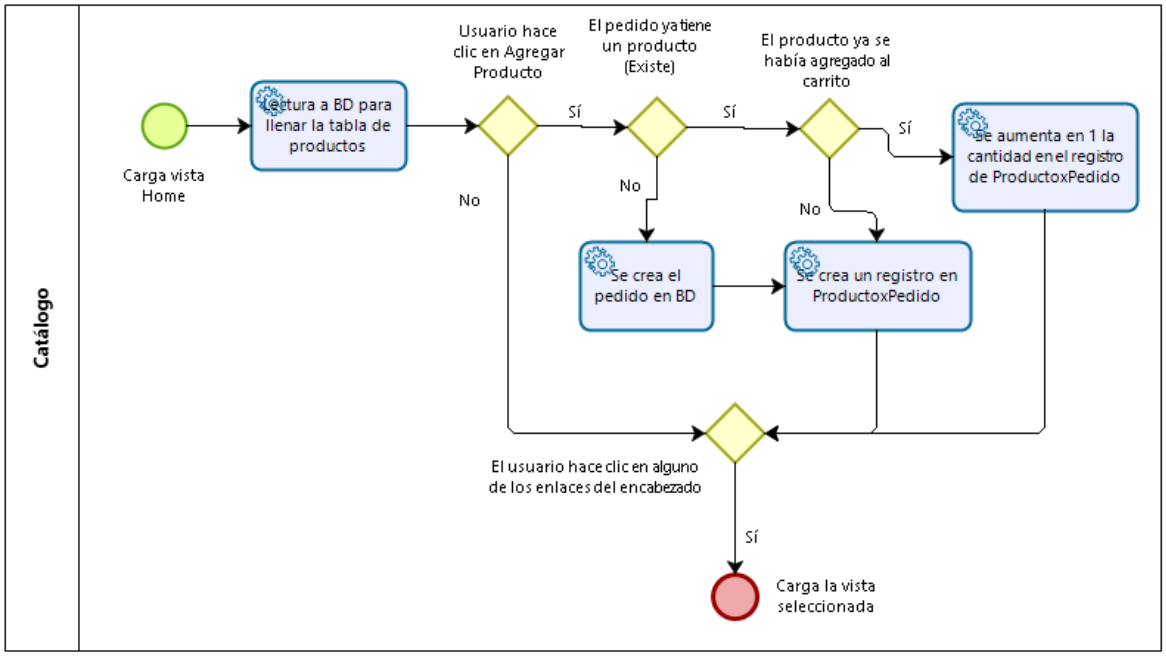


4.3.1.3 Diagrama de módulo 3: Catálogo.

A continuación, la figura 17 muestra el diagrama de flujo para el módulo 3: Catálogo. Cuando carga la vista Home, se hace una consulta a la base de datos para leer los productos y ubicarlos en una tabla. El usuario tiene la opción de agregar productos al carrito, o escoger otra vista. Si agrega un producto por primera vez, se crea dos registros en la base de datos: uno en la tabla de Pedido, y otro en la tabla de ProductoxPedido. Si el usuario vuelve a agregar un producto diferente, sólo se crea un registro en la tabla de ProductoxPedido. Pero si el usuario agrega nuevamente un producto que ya ha agregado al carrito, entonces se suma 1 al campo de cantidad en el registro correspondiente de la tabla ProductoxPedido.

Figura 17.

Diagrama de Catálogo.

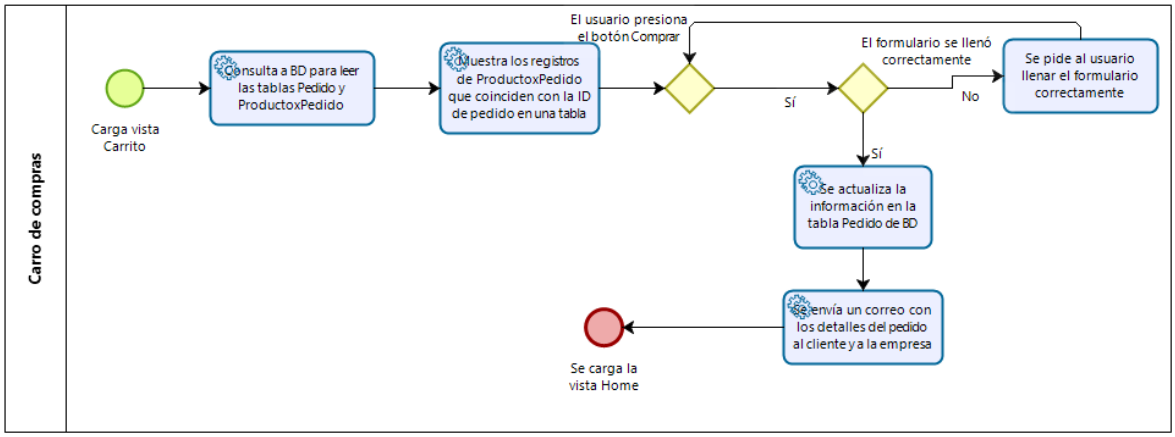


4.3.1.4 Diagrama de módulo 4: Carro de compras.

En la próxima imagen, figura 18, se muestra el diagrama para el módulo 4, carro de compras. Al cargar la vista se hace una lectura a la base de datos y se cargan en una lista todos los registros de la tabla ProductoxPedido que coinciden con la ID del pedido. También en esta vista hay un formulario para que el cliente complete la información del pedido. Si se llena correctamente, se actualiza la información en la base de datos y se envía un correo tanto al cliente como al administrador de la empresa con los detalles del pedido. Si no se llena correctamente, se le pide al cliente que lo haga.

Figura 18.

Diagrama de Carro de compras.

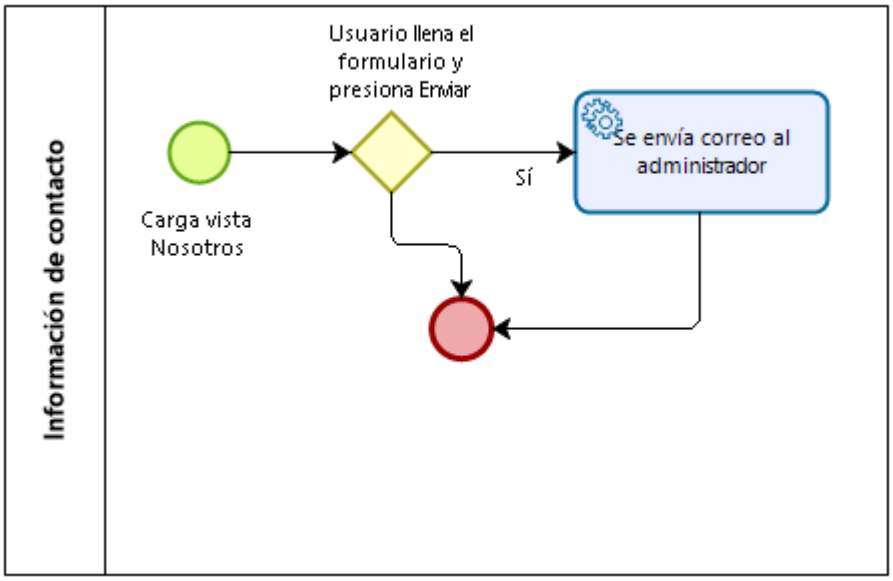


4.3.1.5 Diagrama de módulo 5: Información de contacto

Este módulo cuenta con un formulario, que, si el cliente llena y presiona el botón de enviar, se envía por correo electrónico al administrador de la empresa. Esto se ilustra en la figura 15, a continuación.

Figura 19.

Diagrama de información de contacto.

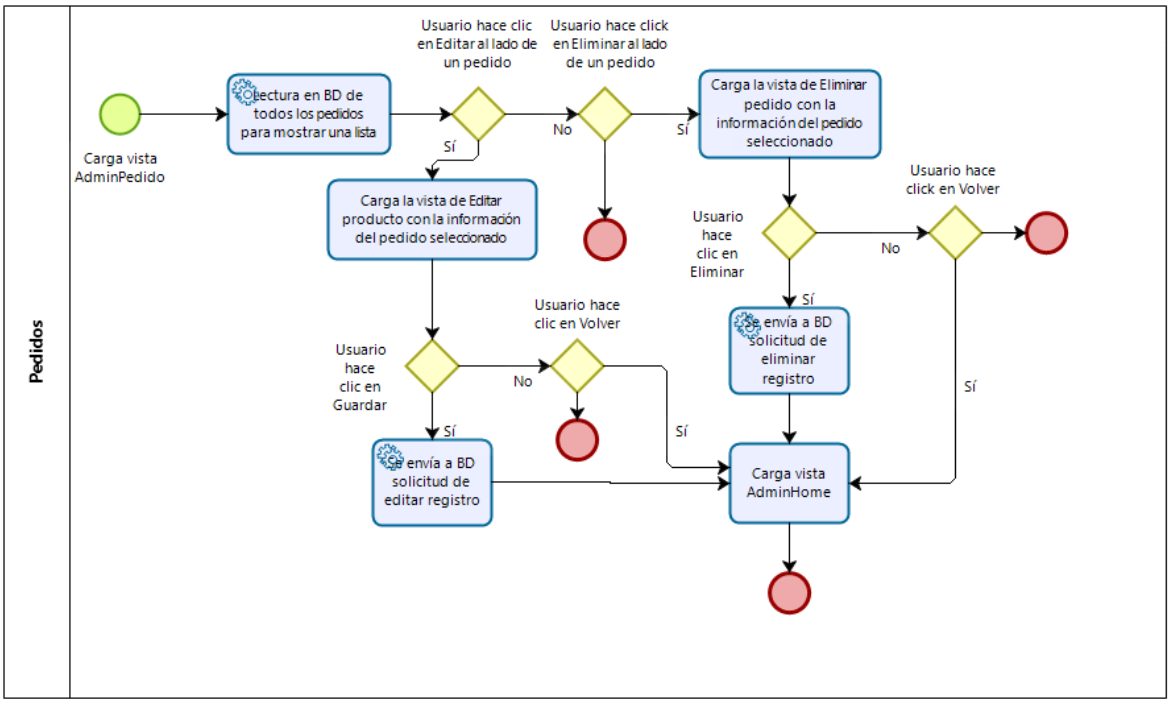


4.3.1.6 Diagrama de módulo 6: Pedidos.

En este módulo se presenta una tabla al usuario administrador, donde puede ver todos los pedidos que se han realizado. Puede editar los pedidos, editarlos y eliminarlos. A continuación, se muestra esto en la figura 20:

Figura 20.

Diagrama de Pedidos.

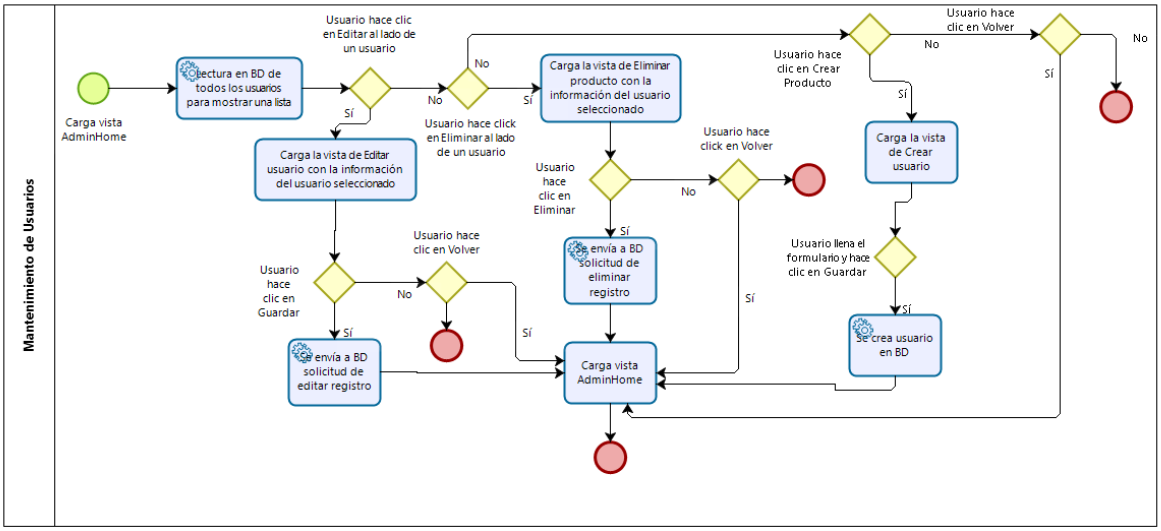


4.1.3.7 Diagrama de módulo 7. Mantenimiento de usuarios.

Similar al módulo anterior, en este módulo se muestran al usuario administrador todos los usuarios existentes en la base de datos, con la posibilidad de crear, editar y eliminar usuarios. La figura 21 (A continuación) muestra lo descrito:

Figura 21.

Diagrama de Usuarios.



4.3.2 Modelo de base de datos

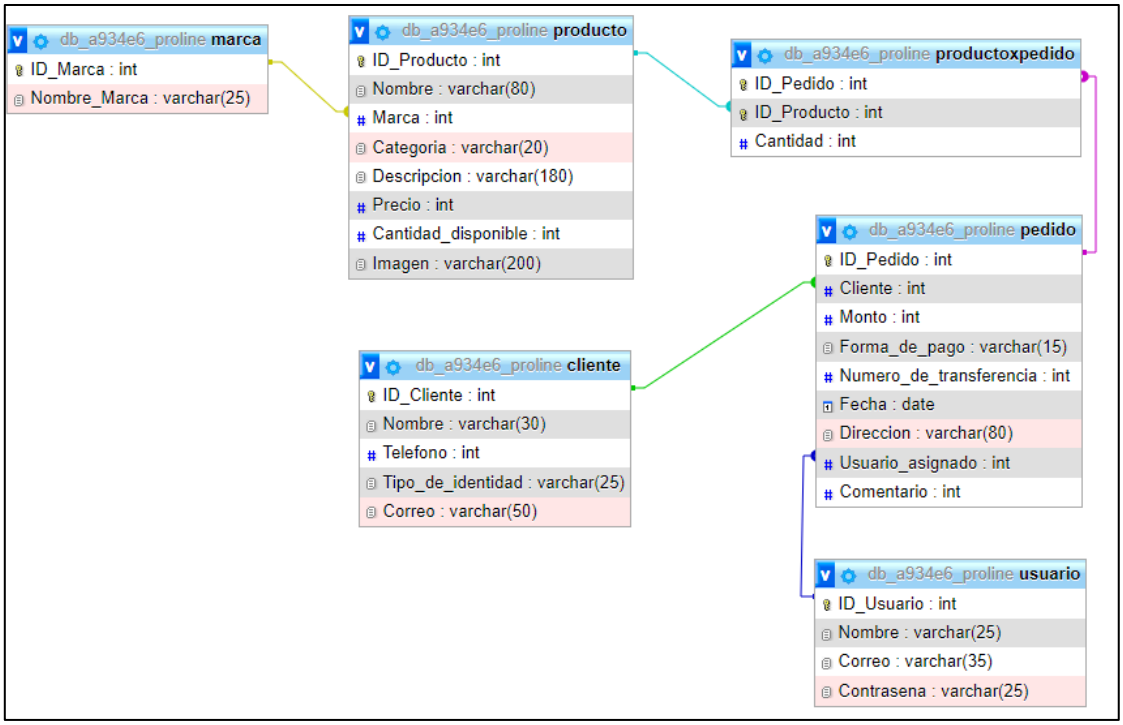
Para la solución de software que se presenta en este proyecto se hace uso de un modelo de base de datos relacional (Véase 2.3.4), MySql, debido a que son más aptas para el registro de transacciones, una de las necesidades que se busca abordar en este proyecto, y a que no se proyectan grandes cambios a la estructura de los datos a lo largo del tiempo.

Antes de crear la base de datos, es necesario llevar a cabo su diseño haciendo uso de la información recopilada y analizada en la primera fase del proyecto.

Una manera de visualizar fácilmente el diseño de una base de datos es por medio de un diagrama de modelo de base de datos. A continuación, la figura 22 muestra el diagrama de modelo base de datos de la base de datos a usada en este proyecto, donde se pueden apreciar las tablas, sus campos y tipos de campo, sus llaves primarias, las relaciones con otras tablas y los campos que permiten un valor nulo.

Figura 22.

Diagrama de modelo de base de datos.



Las tablas de la base de datos son:

- Usuario: Almacena la información de los usuarios administradores. Sus campos son para almacenar la siguiente información: ID usuario, nombre, correo y contraseña. La llave primaria es ID_Usuario, que es autoincrementable.
- Pedido: Almacena la información de un pedido realizado por un cliente. Sus campos son para almacenar la siguiente información: ID pedido, cliente, monto, forma de pago, número de transferencia, fecha, usuario asignado y comentario. La llave primaria es ID_Pedido, que es autoincrementable, y posee dos llaves foráneas: ID_Usuario y ID_Cliente.
- Cliente: Almacena la información de los clientes. Sus campos son para almacenar la siguiente información: ID del cliente, nombre, teléfono, tipo de identificación y correo electrónico. La llave primaria es ID_Cliente, que será proporcionada por el cliente cuando llene un formulario y será su número de cédula de identidad.

- ProductoXPedido: Almacena la información de cada producto que se hace en un pedido. Para lograr esto, usa una llave compuesta de los campos ID_Pedido y ID_Producto, que son llaves foráneas. Además, almacena la cantidad que se pide de un producto.
- Producto: Almacena la información de los productos disponibles. Sus campos son para almacenar la siguiente información: ID producto, nombre, marca, categoría, descripción, precio, cantidad disponible e imagen. La llave primaria es ID_Producto, que es autoincrementable, y posee una llave foránea: ID_marca.
- Marca: Almacena la información de las marcas de los productos. Sus campos son para almacenar la siguiente información: ID de marca y nombre. La llave primaria es ID_Marca, que es autoincrementable.

CAPÍTULO V: PROPUESTA DEL PROYECTO

Este capítulo describe el desarrollo de la aplicación web, de acuerdo con los requerimientos recopilados y analizados en 4.2 (Recolección y análisis de datos), siguiendo el diseño propuesto en 4.3 (Diseño).

Las secciones de este capítulo son: Desarrollo y fase de pruebas.

5.1 DESARROLLO

Esta sección describe todo lo relacionado al desarrollo de la solución diseñada. Se muestran los diferentes elementos de la aplicación, y fragmentos del código que los componen.

Dentro de los objetivos del proyecto, el tercer objetivo se desarrolla en esta sección: Implementar una aplicación web con base en el diseño hecho según los requisitos recopilados para permitir una mejor comunicación con los clientes y potenciales clientes de ProLine CR.

La fase que se lleva a cabo en esta sección es la de implementación, y la variable observada es: Aplicación web para mejorar la comunicación con clientes y potenciales clientes de la empresa.

Se utilizó la metodología en cascada, descrita en 2.1.1, debido a que se conocen muy bien los requerimientos, alcances y limitaciones del proyecto, sin que éstos sean de gran complejidad.

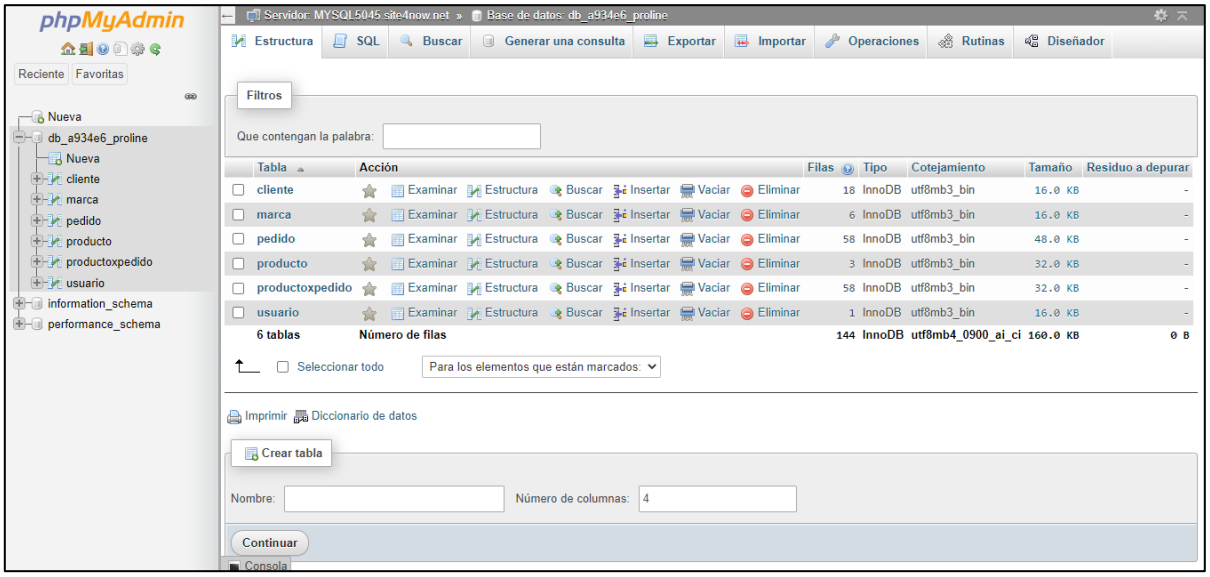
5.1.1 Base de datos

Según los requerimientos dados y el diseño propuesto se elaboró la base de datos, tal como se muestra en 4.3.2. Este modelo se creó en el hosting de SmarterASP donde la aplicación está hospedada, y se gestionó utilizando la herramienta phpMyAdmin que está integrada en el servicio de hosting, tal como se menciona en 3.5.3.

La figura 23 muestra la interfaz de usuario que tiene la consola de phpMyAdmin, en la cual se aprecia la base de datos usada en el proyecto, llamada db_a934e6_proline en sus servidores, además se observan algunas de las opciones disponibles para administrar la base de datos, junto con las tablas existentes y algunas de las opciones que pueden realizarse sobre esas tablas.

Figura 23.

Vista de phpMyAdmin.



A continuación, la figura 24 muestra un fragmento del código SQL utilizado para la creación de la base de datos que se ejecutó en phpMyAdmin. Son un conjunto de sentencias DDL (véase 2.3.4).

Figura 24.

Fragmento de Script DDL.

```
-- --------------------------------------------------------
-- Estructura de tabla para la tabla `Cliente`

CREATE TABLE `Cliente` (
  `ID_Cliente` int(30) NOT NULL,
  `Nombre` varchar(30) COLLATE utf8_bin NOT NULL,
  `Telefono` int(15) NOT NULL,
  `Tipo_de_identidad` varchar(25) COLLATE utf8_bin NOT NULL,
  `Correo` varchar(50) COLLATE utf8_bin NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

-- --------------------------------------------------------
-- Estructura de tabla para la tabla `Pedido`

CREATE TABLE `Pedido` (
  `ID_Pedido` int(30) NOT NULL,
  `Cliente` int(25) DEFAULT NULL,
  `Monto` int(25) DEFAULT NULL,
  `Forma_de_pago` varchar(15) COLLATE utf8_bin DEFAULT NULL,
  `Numero_de_transferencia` int(35) DEFAULT NULL,
  `Fecha` date DEFAULT NULL,
  `Direccion` varchar(80) COLLATE utf8_bin DEFAULT NULL,
  `Usuario_asignado` int(15) DEFAULT NULL,
  `Comentario` int(100) DEFAULT NULL
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

-- --------------------------------------------------------
-- Estructura de tabla para la tabla `Producto`
--

CREATE TABLE `Producto` (
  `ID_Producto` int(15) NOT NULL,
  `Nombre` varchar(80) COLLATE utf8_bin NOT NULL,
  `Marca` varchar(30) DEFAULT NULL,
  `Categoria` varchar(20) DEFAULT NULL,
  `Descripcion` varchar(180) COLLATE utf8_bin DEFAULT NULL,
  `Precio` int(20) NOT NULL,
```

5.1.2 Configuración

Entre los requerimientos recopilados está demostrada la necesidad de que esta solución de software sea accesible a los clientes de la empresa, y la internet es el medio ideal para brindar el acceso necesario.

Para lograr que la solución sea accesible por medio de internet, se hizo uso del framework .NET Core con el lenguaje de programación C# y se usó el IDE Visual Studio 2022, de acuerdo con lo mencionado en 3.5.3. Así, se crearon varias clases de forma automática, por ejemplo una clase llamada Program.cs, donde se encuentra una parte muy importante de la configuración para la aplicación; otra clase llamada appsettings.json, donde se indican los hostings a los cuales la aplicación va a permitir el acceso; y las clases que están dentro de los paquetes instalados por medio de NuGet (véase 2.5.2) que permiten la adición de clases y métodos al proyecto.

A continuación, se muestran fragmentos de código de algunas partes encargadas de funciones importantes para cumplir con los requerimientos y el diseño propuesto.

5.1.2.1 Contexto de base de datos

Las dos imágenes siguientes, figuras 25 y 26, muestran un fragmento del código de la clase Program.cs y la clase appsettings.json, respectivamente. La línea 14 de la clase Program.cs contiene el código del método necesario para la inyección de dependencia para el contexto de base de datos. Esto permite usar una instancia de la base de datos y crear objetos con propiedades iguales a las tablas de la base de datos, para almacenar la información leída de la base de datos o a almacenarse o editarse en la base de datos. Al usar la inyección de dependencia no es necesario introducir el código relacionado con la conexión a la base de datos cada vez que se use, y si esta conexión cambiara, solo será necesaria cambiarla una única vez. El string de conexión a la base de datos se encuentra en la clase appsettings.json.

Figura 25.

Fragmento del código en Program.cs

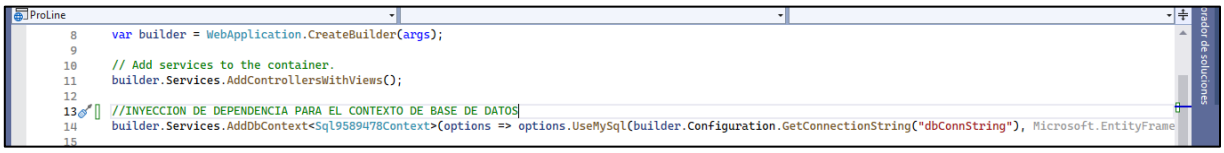
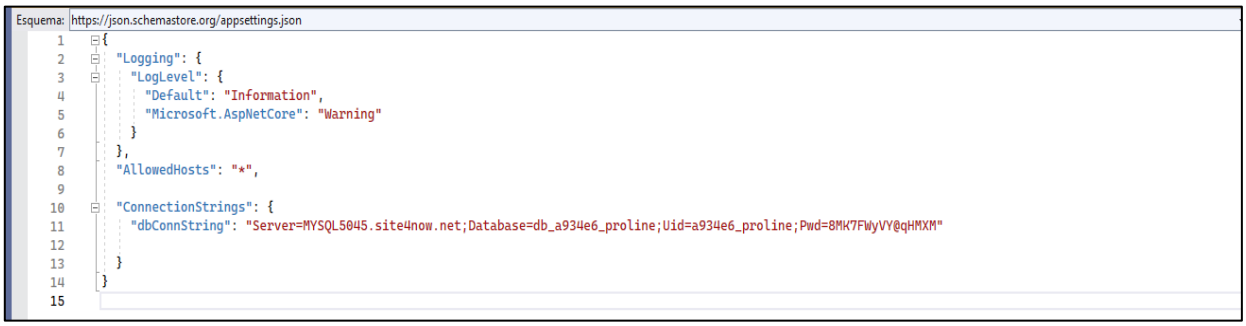


Figura 26.

Clase appsettings.json.



Al usar un contexto de bases de datos por medio de Entity Framework se pueden crear de forma ágil los métodos para la lectura y escritura sobre la base de datos. Para ello se instalaron los paquetes NuGet desarrollados por Microsoft llamados EntityFrameworkCore.SqlServer, EntityFrameworkCore.Tools, y Pomelo. EntityFrameworkCore.MySql.

A través de sus métodos se crearon un modelo para cada tabla y el modelo de la base de datos completa. Las imágenes a continuación, figura 27 y figura 28, muestran las carpetas dentro de la carpeta de Modelos donde se aprecian las clases creadas para representar las tablas de la base de datos, y el código en la clase Cliente.cs, que es similar al código en las demás clases de la carpeta Models, respectivamente.

Figura 27.

Carpetas de la clase Models.

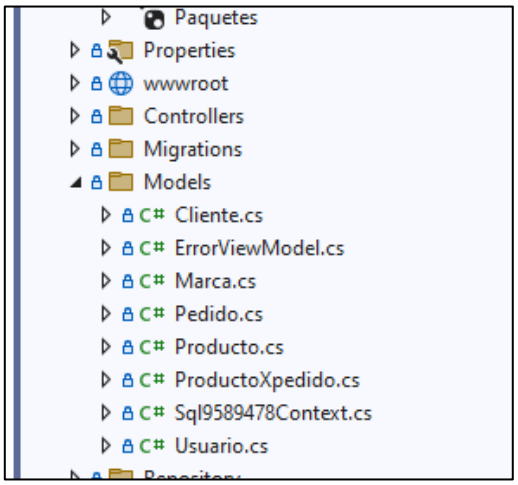


Figura 28.

Clase Cliente.cs

```
using System;
using System.Collections.Generic;

namespace ProLine.Models;

3 referencias
public partial class Cliente
{
    5 referencias
    public int IdCliente { get; set; }

    1 referencia
    public string Nombre { get; set; } = null!;

    1 referencia
    public int Telefono { get; set; }

    1 referencia
    public string TipoDeIdentidad { get; set; } = null!;

    1 referencia
    public string Correo { get; set; } = null!;

    1 referencia
    public virtual ICollection<Pedido> Pedidos { get; } = new List<Pedido>();
}
```

5.1.2.2 Operaciones de lectura y escritura en la base de datos.

Todos los métodos para consultar la base de datos y realizar lecturas y escrituras utilizados en los módulos de la aplicación son iguales, con la excepción de las tablas a las que hacen referencia y los conjuntos de datos que se manipulan.

En un controlador se escriben los métodos que usarán sus vistas asociadas, que generalmente son: Index, Create, Edit y Delete. Dependiendo de lo que el usuario quiera hacer, se cargará una vista u otra.

Las vistas de Index cargan los datos contenidos en la tabla correspondiente en la base de datos, y los muestran en una tabla HTML. También tienen enlaces para crear nuevos registros en la tabla específica, y para editar o eliminar los registros.

A continuación, la figura 29 muestra el código del controlador de Marcas para la vista Index. Se puede observar el uso del método ToListAsync para trasladar la información a una tabla HTML, previamente cargada en una variable de contexto de base de datos (_context). Luego, la figura 30 muestra el código de la vista correspondiente:

Figura 29.

Código para la vista Index en controlador de Marcas.

```
namespace ProLine.Controllers
{
    1 referencia
    public class MarcasController : Controller
    {
        private readonly Sql9589478Context _context;

        0 referencias
        public MarcasController(Sql9589478Context context)
        {
            _context = context;
        }

        // GET: Marcas
        3 referencias
        public async Task<IActionResult> Index()
        {
            return View(await _context.Marcas.ToListAsync());
        }
```

Figura 30.

Código de la vista Index en Marcas.

```
@model IEnumerable<ProLine.Models.Marca>

@{
    ViewData["Title"] = "Index";
    Layout = "~/Views/Shared/_Layout_admin.cshtml";
}
<div class="marginTop">

<h2>Marcas</h2>

<p>
    <a asp-action="Create">Crear nueva</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.NombreMarca)
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.NombreMarca)
            </td>
            <td>
                <a asp-action="Edit" asp-route-id="@item.IdMarca">Edit</a> |
                <a asp-action="Delete" asp-route-id="@item.IdMarca">Delete</a>
            </td>
        </tr>
}
    </tbody>
</table>
</div>
```

A continuación, la figura 31 muestra el código del controlador de Marcas para la vista Create. Se puede observar que se valida si todos los parámetros recogidos en el formulario de la vista son adecuados, usando el método ModelState.IsValid. En caso de ser verdadero, entonces se usa el método Add para agregar el registro en la variable de contexto _context, que luego es

guardada en la base de datos con el método SaveChangesAsync. Luego, la figura 32 muestra la vista correspondiente.

Figura 31.

Código para la vista Create en controlador de Marcas.

```
50
51          // POST: Marcas/Create
52          // To protect from overposting attacks, enable the specific properties you want to bind to.
53          // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
54          [HttpPost]
55          [ValidateAntiForgeryToken]
            0 referencias
56          public async Task<IActionResult> Create([Bind("IdMarca,NombreMarca")] Marca marca)
57          {
58              if (ModelState.IsValid)
59              {
60                  _context.Add(marca);
61                  await _context.SaveChangesAsync();
62                  return RedirectToAction(nameof(Index));
63              }
64              return View(marca);
65          }
66
67          // GET: Marcas/Edit/5
            0 referencias
68          public async Task<IActionResult> Edit(int? id)
69          {
```

Figura 32.

Código de la vista Create en Marcas.

```
1   @model ProLine.Models.Marca
2
3   @{
4       ViewData["Title"] = "Create";
5       Layout = "~/Views/Shared/_Layout_admin.cshtml";
6   }
7   <div class="marginTop">
8
9   <h2>Crear Marca nueva</h2>
10
11  <h4>Marca</h4>
12  <hr />
13  <div class="row">
14      <div class="col-md-4">
15          <form asp-action="Create">
16              <div asp-validation-summary="ModelOnly" class="text-danger"></div>
17              <div class="form-group">
18                  <label asp-for="NombreMarca" class="control-label"></label>
19                  <input asp-for="NombreMarca" class="form-control" />
20                  <span asp-validation-for="NombreMarca" class="text-danger"></span>
21              </div>
22              <div class="form-group">
23                  <input type="submit" value="Create" class="btn btn-default" />
24              </div>
25          </form>
26      </div>
27  </div>
28
29  <div>
30      <a asp-action="Index">Ver todas las marcas</a>
31  </div>
32  </div>
33  @section Scripts {
34      @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
35  }
36
```

A continuación, la figura 33 muestra el código del controlador de Marcas para la vista Edit. Al igual que con Create, valida que el modelo que recibe tenga todos los campos de manera correcta, pero además valida también que la ID exista a través del método MarcaExists, para

que en caso de que el usuario administrador la haya cambiado, evitar la creación de un registro duplicado pero con un ID diferente. Luego, la figura 34 muestra la vista correspondiente.

Figura 33.

Código para la vista Edit en el controlador de Marcas.

```
82
83          // POST: Marcas/Edit/5
84          // To protect from overposting attacks, enable the specific properties you want to bind to.
85          // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
86          [HttpPost]
87          [ValidateAntiForgeryToken]
            0 referencias
88          public async Task<IActionResult> Edit(int id, [Bind("IdMarca,NombreMarca")] Marca marca)
89          {
90              if (id != marca.IdMarca)
91              {
92                  return NotFound();
93              }
94
95              if (ModelState.IsValid)
96              {
97                  try
98                  {
99                      _context.Update(marca);
100                     await _context.SaveChangesAsync();
101                 }
102                 catch (DbUpdateConcurrencyException)
103                 {
104                     if (!MarcaExists(marca.IdMarca))
105                     {
106                         return NotFound();
107                     }
108                     else
109                     {
110                         throw;
111                     }
112                 }
113                 return RedirectToAction(nameof(Index));
114             }
115             return View(marca);
116         }
117
```

Figura 34.

Código de la vista Edit de Marcas.

```
@model ProLine.Models.Marca

@{
    ViewData["Title"] = "Edit";
    Layout = "~/Views/Shared/_Layout_admin.cshtml";
}
<div class="marginTop">
<h2>Edit</h2>

<h4>Marca</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="IdMarca" />
            <div class="form-group">
                <label asp-for="NombreMarca" class="control-label"></label>
                <input asp-for="NombreMarca" class="form-control" />
                <span asp-validation-for="NombreMarca" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Ver todas las marcas</a>
</div>
</div>
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

A continuación, la figura 35 muestra el código del controlador de Marcas para la vista Delete. Ésta valida que el registro que recibe por parámetro no sea nulo, para evitar un error en la base de datos, y luego ejecuta una búsqueda para encontrar el registro con la ID proporcionada y eliminarlo con el método Remove. Luego, la figura 36 muestra la vista correspondiente.

Figura 35.

Código para la vista Delete en el controlador de Marcas.

```
135
136             // POST: Marcas/Delete/5
137             [HttpPost, ActionName("Delete")]
138             [ValidateAntiForgeryToken]
                0 referencias
139             public async Task<IActionResult> DeleteConfirmed(int id)
140             {
141                 if (_context.Marcas == null)
142                 {
143                     return Problem("Entity set 'Sql9589478Context.Marcas'  is null.");
144                 }
145                 var marca = await _context.Marcas.FindAsync(id);
146                 if (marca != null)
147                 {
148                     _context.Marcas.Remove(marca);
149                 }
150
151                 await _context.SaveChangesAsync();
152                 return RedirectToAction(nameof(Index));
153             }
154
                1 referencia
155             private bool MarcaExists(int id)
156             {
157               return _context.Marcas.Any(e => e.IdMarca == id);
158             }
159         }
160     }
161
```

Figura 36.

Código de la vista Delete de Marcas.

```
@model ProLine.Models.Marca

@{
    ViewData["Title"] = "Delete";
    Layout = "~/Views/Shared/_Layout_admin.cshtml";
}
<div class="marginTop">
<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Marca</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.NombreMarca)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.NombreMarca)
        </dd>
    </dl>

    <form asp-action="Delete">
        <input type="hidden" asp-for="IdMarca" />
        <input type="submit" value="Delete" class="btn btn-default" /> |
        <a asp-action="Index">Back to List</a>
    </form>
</div>
</div>
```

5.1.2.3 Validación del usuario administrador

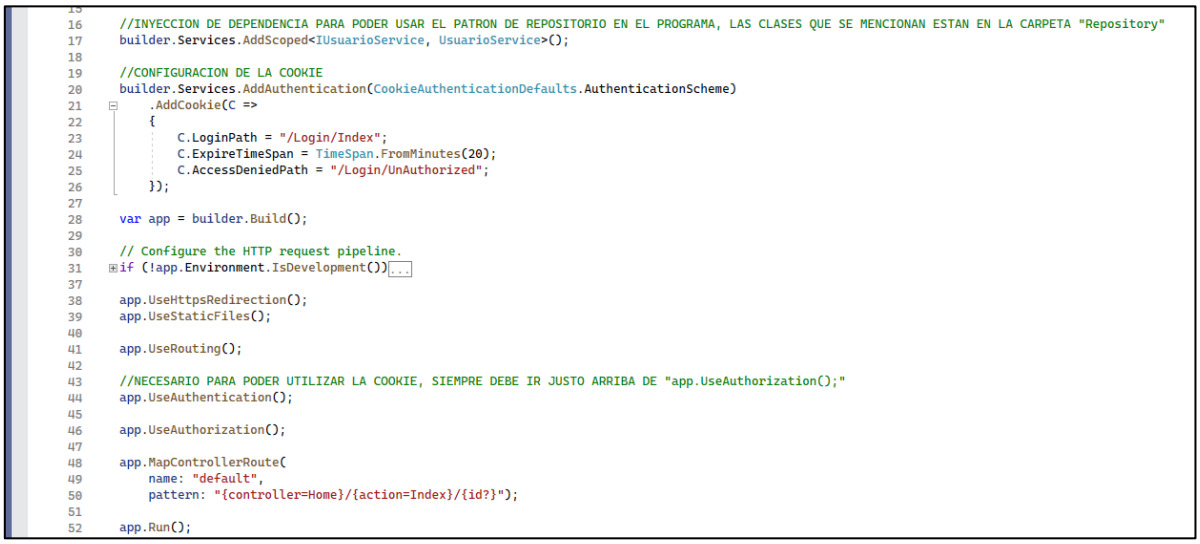
Para realizar la validación del usuario administrador se hizo uso de la autenticación por medio de cookies. Haciendo uso de los métodos predeterminados en Microsoft.AspNetCore.Authentication.Cookies, se crea una cookie después de validar en la base de datos la existencia de las credenciales de usuario a través de lo que se describirá a continuación.

Las vistas de administrador solicitan que esta cookie exista para poder cargarse, de lo contrario se carga la vista de Login. La inyección de dependencia permite que las consultas a la base de datos para realizar la validación se implementen en todas las vistas sin tener que escribir cada vez los parámetros de las clases involucradas. Para ello se crearon dos carpetas en una carpeta llamada Repository: Interfaces y Services; cada una con un archivo: IUsuarioService.cs en Interfaces y UsuarioService.cs en Services.

La imagen a continuación, figura 37, muestra el código en Program.cs que se encarga de la implementación de la autenticación:

Figura 37.

Validación de usuario en Program.cs.



```
//INYECCION DE DEPENDENCIA PARA PODER USAR EL PATRON DE REPOSITORIO EN EL PROGRAMA, LAS CLASES QUE SE MENCIONAN ESTAN EN LA CARPETA "Repository"
builder.Services.AddScoped<IUsuarioService, UsuarioService>();

//CONFIGURACION DE LA COOKIE
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(C =>
    {
        C.LoginPath = "/Login/Index";
        C.ExpireTimeSpan = TimeSpan.FromMinutes(20);
        C.AccessDeniedPath = "/Login/UnAuthorized";
    });

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())...

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

//NECESARIO PARA PODER UTILIZAR LA COOKIE, SIEMPRE DEBE IR JUSTO ARRIBA DE "app.UseAuthorization();"
app.UseAuthentication();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

Las próximas dos imágenes, figuras 38 y 39, muestran el código de las clases IUsuarioService.cs y UsuarioService.cs, respectivamente. En la clase IUsuarioService.cs se crea una lista de tipo Usuario donde luego se almacenarán todos los usuarios de la base de datos, y un objeto de tipo Usuario donde se almacenará el usuario que se recuperará luego a partir de las credenciales escritas por el usuario. En la clase UsuarioService.cs se carga todo el contexto de base de datos, y se envía la información de la tabla de Usuarios a la lista creada con las propiedades establecidas en IUsuarioService.cs; luego se recorre esa lista pidiendo retornar el usuario que coincida en los campos correo y contraseña. Si no existe, retorna un valor nulo.

Figura 38.

Clase IUsuarioService.cs

```
using ProLine.Models;

namespace ProLine.Repository.Interfaces
{
    4 referencias
    public interface IUsuarioService
    {
        2 referencias
        List<Usuario> ListaUsuarios();
        2 referencias
        Usuario ValidarUsuario(string _correo, string _contrasena);
    }
}
```

Figura 39.

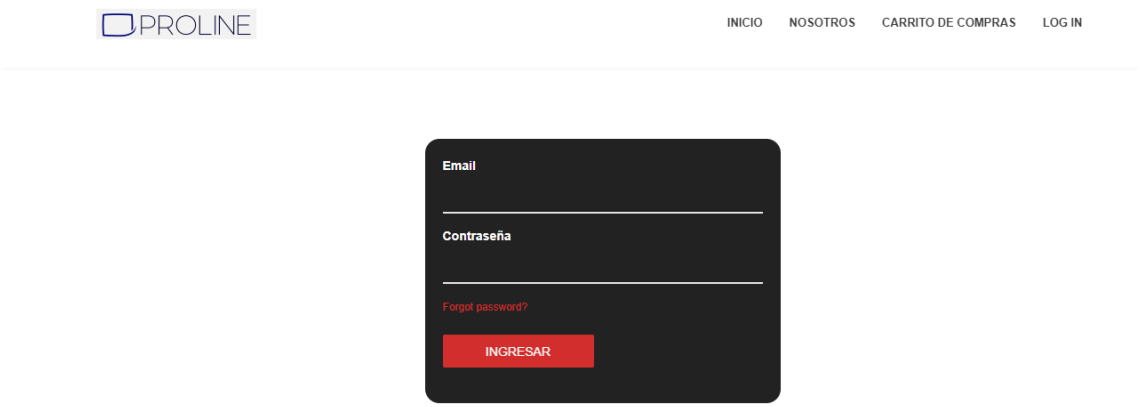
Clase UsuarioService.cs

```
using ProLine.Models;
using ProLine.Repository.Interfaces;

namespace ProLine.Repository.Services
{
    2 referencias
    public class UsuarioService : IUsuarioService
    {
        //ACA IMPLEMENTAMOS LA INYECCION DE DEPENDENCIA UTILIZANDO EL CONTEXTO CREADO CON EF CORE
        private readonly Sql9589478Context _context;
        0 referencias
        public UsuarioService(Sql9589478Context context)
        {
            _context = context;
        }

        2 referencias
        public List<Usuario> ListaUsuarios()
        {
            return _context.Usuarios.ToList();
        }

        2 referencias
        public Usuario ValidarUsuario(string _correo, string _contrasena)
        {
            return ListaUsuarios().Where(Usu => Usu.Correo == _correo && Usu.Contrasena == _contrasena).FirstOrDefault();
        }
    }
}
```

En la vista de Login, que se muestra en la figura 40 (a continuación) hay un formulario con dos campos: Usuario y contraseña. En el primer campo deben escribir un correo electrónico, y en el segundo, la contraseña respectiva.

Figura 40.

Vista de Login.



Estos dos son textos son capturados y usados como los parámetros que se usan para validar si el usuario existe en la base de datos. A continuación se muestra la figura 41, donde se observa el código en el controlador del Login. Es acá donde se crea la cookie de autenticación si la variable usu tiene un valor asignado después de usar el método ValidarUsuario (Lo que significa que hubo una coincidencia en la base de datos) descrito anteriormente:

Figura 41.

Fragmento del controlador del login.

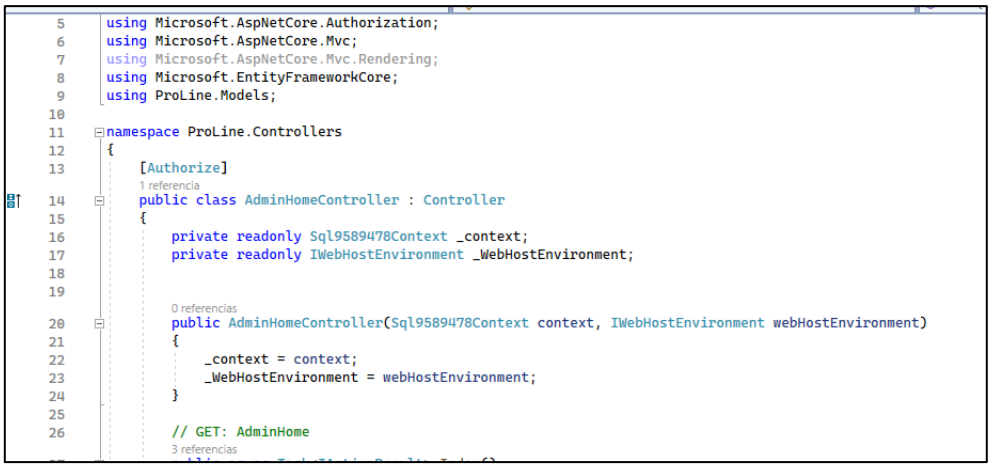
```
15          public LoginController(IUsuarioService usuServ)
16          {
17              _UsuServ = usuServ;
18          }
19
            0 referencias
20          public IActionResult Index()
21          {
22              return View();
23          }
24
25          [HttpPost]
            0 referencias
26          public async Task<IActionResult> Index(Usuario _usuario)
27          {
28              var usu = _UsuServ.ValidarUsuario(_usuario.Correo, _usuario.Contrasena);
29
30              if (usu != null)
31              {
32                  var claims = new List<Claim>
33                  {
34                      new Claim(ClaimTypes.Name, usu.Nombre),
35                      new Claim(ClaimTypes.Email, usu.Correo)
36                  };
37
38                  var claimsIdentity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);
39
40                  await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(claimsIdentity));
41
42                  return RedirectToAction("Index", "AdminHome");
43              }
44              else
45              {
46                  return View();
47              }
48          }
```

El método Authorize se usa en cada controlador de las vistas de administrador, y se encarga de verificar que la cookie de autorización exista. Si existe, entonces permite cargar las vistas que ese controlador controla. En cambio, si la cookie no existe, entonces niega el acceso a las vistas que controla y carga la vista de Login.

A continuación, la figura 42 muestra un fragmento del código de un controlador de algunas vistas de administrador, AdminHomeController, donde se observa el método Authorize. Además, en el mismo archivo del controlador antes mencionado, la figura 43 muestra el uso de otro método para reforzar la seguridad, ValidateAntiForgeryToken, que verifica el origen de la solicitud HttpPost que recibe y valida que provenga de una de las vistas de la aplicación para cargar la vista, y no que provenga de otro medio, caso en el cual mostrará una vista de error.

Figura 42.

Fragmento de AdminHomeController.cs con el método Authorize.



```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using ProLine.Models;

namespace ProLine.Controllers
{
    [Authorize]
    1 referencia
    public class AdminHomeController : Controller
    {
        private readonly Sql9589478Context _context;
        private readonly IWebHostEnvironment _WebHostEnvironment;


        0 referencias
        public AdminHomeController(Sql9589478Context context, IWebHostEnvironment webHostEnvironment)
        {
            _context = context;
            _WebHostEnvironment = webHostEnvironment;
        }

        // GET: AdminHome
        3 referencias
```

Figura 43.

Fragmento de AdminHomeController.cs con el método ValidateAntiForgeryToken.

```
55
56          // POST: AdminHome/Create
57          // To protect from overposting attacks, enable the specific properties you want to bind to.
58          // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
59          [HttpPost]
60          [ValidateAntiForgeryToken]
            0 referencias
61          public async Task<IActionResult> Create([Bind("IdProducto,Nombre,Marca,Descripcion,Precio,Categoria,CantidadDisponible")] Producto producto)
62          {
63              if (ModelState.IsValid)
64              {
65                  var files= HttpContext.Request.Form.Files;
66                  if (files.Count > 0)
67                  {
68                      string webRootPath = _WebHostEnvironment.WebRootPath;
69                      var upload = webRootPath + WC.IMAGEPATH;
70
71                      string fileName = Guid.NewGuid().ToString();
72                      string extension = Path.GetExtension(files[0].FileName);
73                      using (var fileStream = new FileStream(Path.Combine(upload, fileName + extension), FileMode.Create))
74                      {
```

5.1.3 Estilos.

En este proyecto se hizo uso de Bootstrap, que se vale de CSS para determinar clases que luego son aplicados a los diferentes elementos de las páginas web. Los parámetros que estas clases afectan son, entre otros: los colores, fuentes y tamaños de los textos, el espaciado entre contenedores, el color y forma de los botones, la separación de los elementos en una lista, el color del fondo de la página, etc.

Esto facilita mantener un aspecto visual homogéneo en todas las vistas de la aplicación, y también hace más sencillo la aplicación del diseño orientado hacia la experiencia de usuario en todas las vistas, como se describe en 2.3.3 (Diseño de interfaces de usuario).

A continuación, la figura 44 muestra un fragmento del código dentro de una de las clases de Bootstrap:

Figura 44.

Fragmento de hoja de estilos.

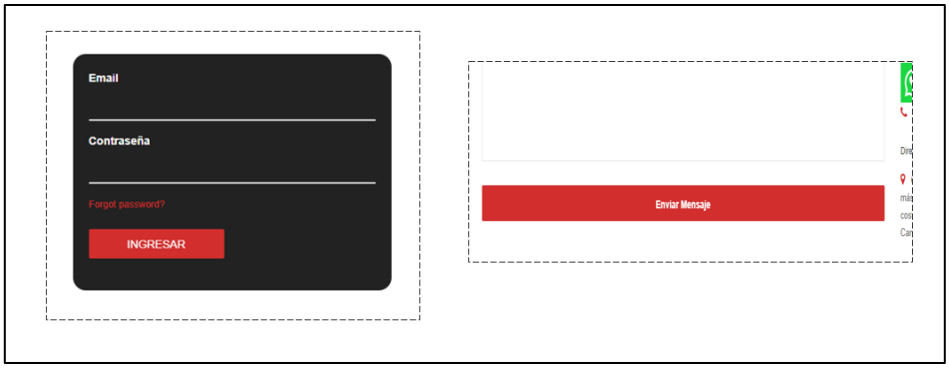
```
le.css   site.css
1   html {
2       font-size: 14px;
3   }
4
5   @media (min-width: 768px) {
6       html {
7           font-size: 16px;
8       }
9   }
10
11  html {
12      position: relative;
13      min-height: 100vh;
14  }
15
16
17  .login-form {
18          display: flex;
19          width: 100%;
20          height: 100vh;
21          align-items: center;
22          justify-content: center;
23  }
24  #page-title {
25          margin-top: 100px;
26  }
27
28  .marginTop {
29          margin-top: 100px;
30  }
```

A continuación, se muestran algunos de los elementos en diferentes vistas cuyas características se definen en las hojas de estilo, y que por tanto son homogéneos.

En la figura 45 a continuación, se aprecian dos botones en vistas diferentes. Uno en la vista de Login y otro en la vista de Nosotros. Se aprecia que ambos son del mismo color, y que el texto tiene las mismas características. También se observa que contrastan con el resto del contenido, lo cual los hace resaltar.

Figura 45

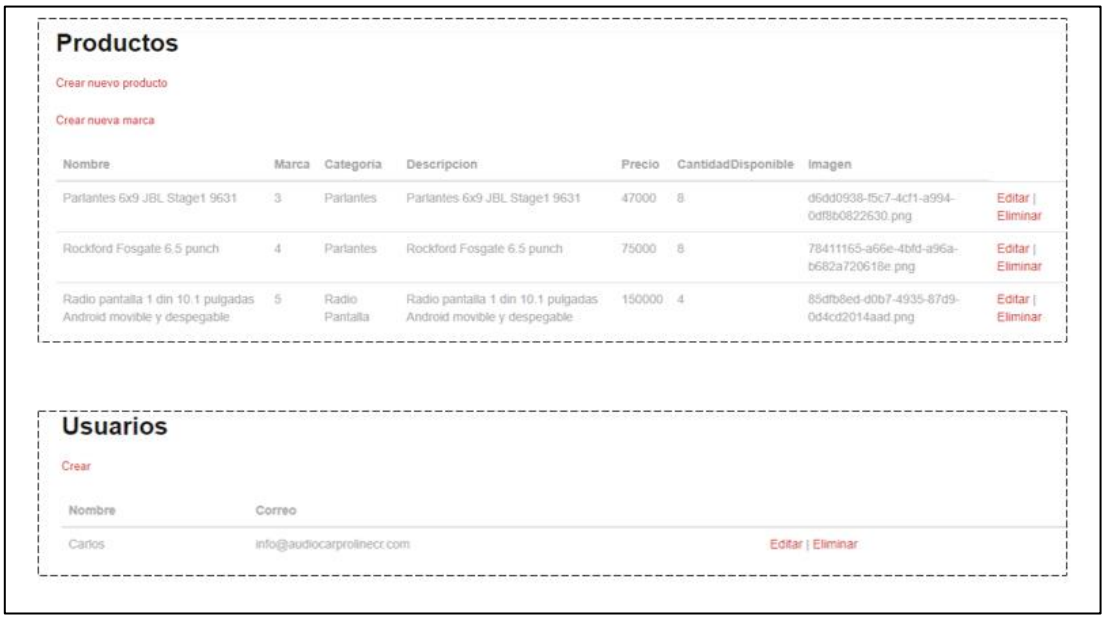
Botones en la vista de Login y en la vista de Nosotros.



En la siguiente figura, la figura 46, se muestran dos listas en vistas diferentes. Una es la lista de productos guardados en la vista de AdminHome, y la otra es la lista de usuarios administradores en la vista AdminUsuarios. Se observa que las tablas tienen el mismo estilo en sus líneas horizontales, sin líneas verticales, con líneas de color gris y los encabezados de las columnas con la letra en negrita. También los enlaces para realizar acciones sobre los elementos de la lista son iguales, en color rojo contrastando con el resto.

Figura 46.

Listas en vistas AdminHome y AdminUsuarios.



Otro de los elementos importantes para dar estilo a todas las vistas es el uso de un layout. En este proyecto se usaron dos layout diferentes: uno para las vistas de administrador y otro para las vistas de los clientes. La diferencia es necesaria, ya que en estos layouts se tiene un encabezado con enlaces a las diferentes vistas, y de acuerdo con el rol (Administrador o cliente) se tendrá acceso a las vistas que les conciernen. Además del encabezado, en el layout se encuentra un pie de página, con información acerca de la empresa.

A continuación, se muestran cuatro ilustraciones: La primera (Figura 47) muestra un fragmento del código del layout para los clientes, la segunda (Figura 48) muestra cómo se ve el encabezado de las vistas para el cliente (Donde se aprecian los enlaces a las vistas que les conciernen), la tercera (Figura 49) muestra cómo se ve el encabezado de las vistas para los administradores (Donde se aprecian los enlaces a las vistas de los administradores) y la cuarta (Figura 50) muestra cómo se ve el pie de página (Presente en todas las vistas).

Figura 47.

Fragmento del código del layout para clientes.

```
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
        </button>
        <a class="logo" href="~/Home">
            <img src="~/images/logo/logo-light.png" alt="Audiocar Proline">
        </a>
    </div>

    <!-- Collect the nav links, forms, and other content for toggling -->
    <div class="collapse navbar-collapse pull-right" id="header-navbar-collapse-1">
        <ul class="nav navbar-nav navbar-left">

            <li class="nav-item">
                @if (IdPed != null)
                {
                    <a asp-area="" asp-controller="Home" asp-action="Seguir">Inicio</a>
                }
                else
                {
                    <a asp-area="" asp-controller="Home" asp-action="Index">Inicio</a>
                }
            </li>
            <li class="nav-item">
                <a asp-area="" asp-controller="Nosotros" asp-action="Index">Nosotros</a>
            </li>
            <li class="nav-item">
                <a asp-area="" asp-controller="Carrito" asp-action="Edit">Carrito de compras</a>
            </li>
            <li class="nav-item">
                <a asp-area="" asp-controller="Login" asp-action="Index">Log In</a>
            </li>
        </ul>
        <!-- li end -->
    </div>
```

Figura 48.

Encabezado de las vistas para el cliente.

Figura 49.

Encabezado de las vistas para los administradores.

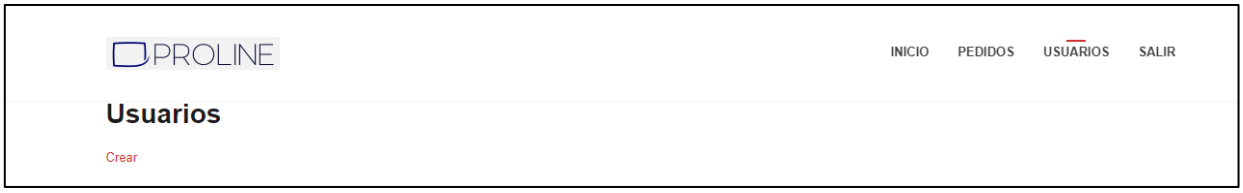
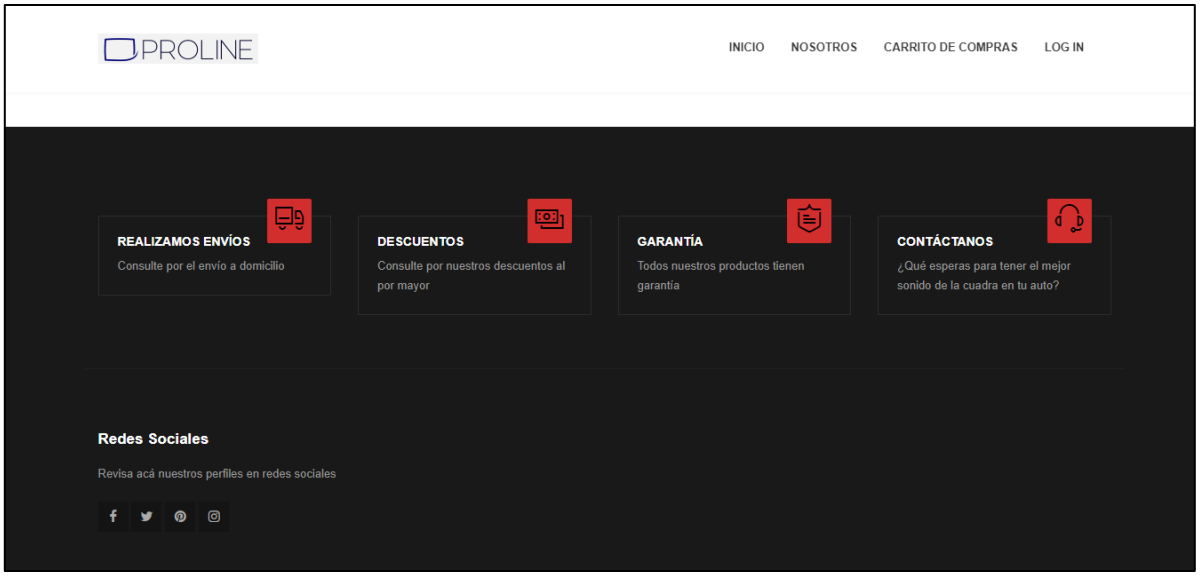


Figura 50.

Pie de página.



Siguiendo los principios descritos en 2.3.3 (Diseño de interfaces de usuario), el logotipo, los enlaces de navegación, y el pie de página, son visibles en todas las páginas del sitio.

5.1.4 Módulos

A continuación, se describen los módulos realizados, de acuerdo con lo que se propuso en la fase de diseño (véase 4.3).

5.1.4.1 Módulo 1: Inicio de sesión

Los métodos utilizados en este módulo se describen con detalle en 5.1.2.2 (Validación del usuario administrador), donde también se ilustra la vista de los usuarios de la aplicación.

5.1.4.2 Módulo 2: Mantenimiento de catálogo

Este módulo consiste en los controladores AdminHomeController y sus vistas asociadas: Index, Create, Edit y Delete; y AdminMarcas y sus vistas asociadas: Index, Create, Edit y Delete. Después de pasar por la autenticación, el usuario administrador tiene la vista Index de AdminHome como su página de inicio. Desde allí puede ver una lista con todos los productos que se muestran en el catálogo, con opciones de crear un producto nuevo, editar los productos

ya existentes o eliminar un producto. Cuando selecciona alguna de esas tres opciones, se carga su vista respectiva desde donde puede realizar la acción solicitada.

A continuación, las figuras 51, 52, 53 y 54 muestran parcialmente las vistas Index, Create, Edit y Delete de AdminHome, respectivamente, con los elementos antes descritos. Se puede apreciar también la aplicación de la regla de los tercios, mencionada en 2.3.3 (Diseño de interfaces de usuario):

Figura 51.

Vista Index de AdminHome.

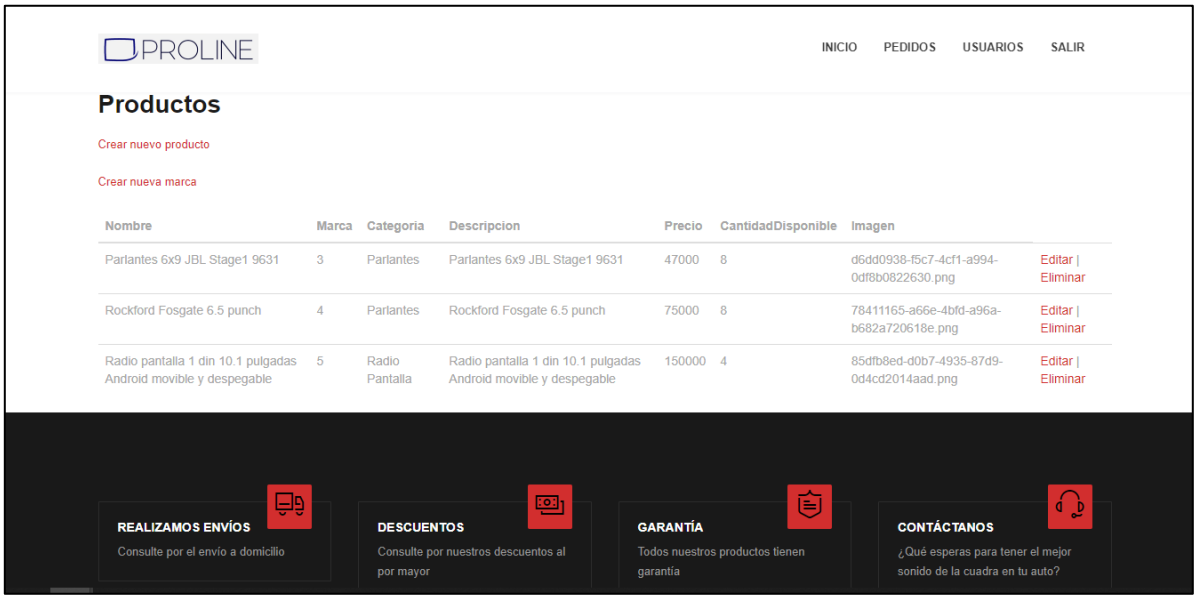


Figura 52.

Vista Create de AdminHome.

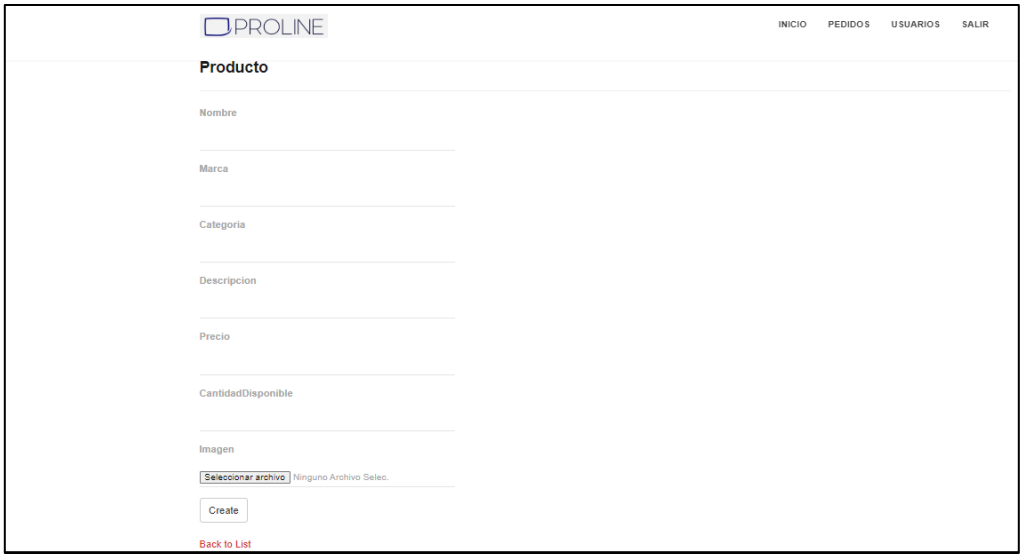


Figura 53.

Vista de Edit de AdminHome, con información de un producto.

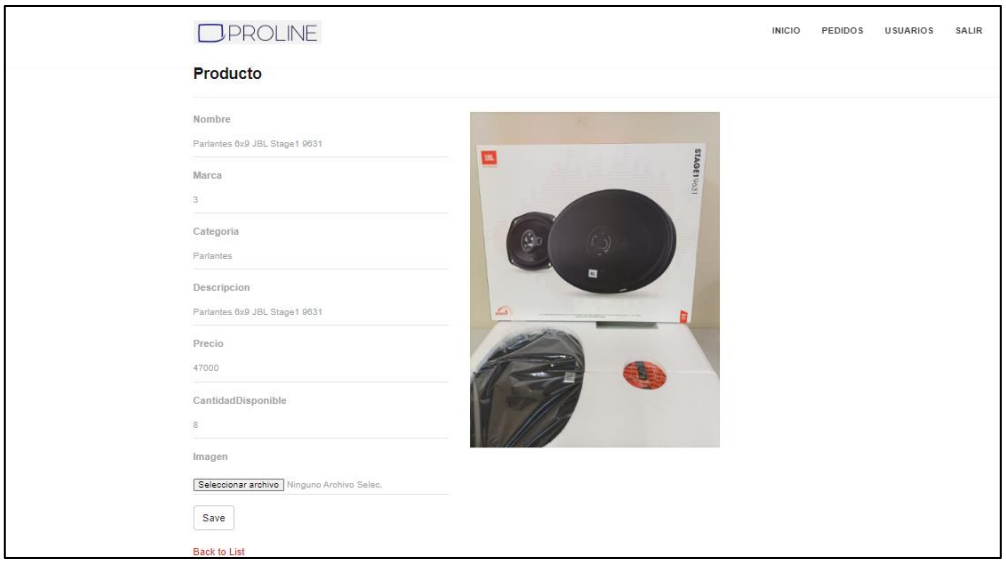


Figura 54.

Vista de Delete de AdminHome.

También en la vista Index de AdminHome hay un enlace para crear nuevas marcas. Éste lo llevará a la vista Index de AdminMarcas, igual a la vista anterior, pero en lugar de ver y administrar productos, lo hace con las marcas.

Además, en la parte inferior de esta vista se encuentra el enlace al manual de usuario, para que pueda ser consultado en cualquier momento y desde cualquier lugar.

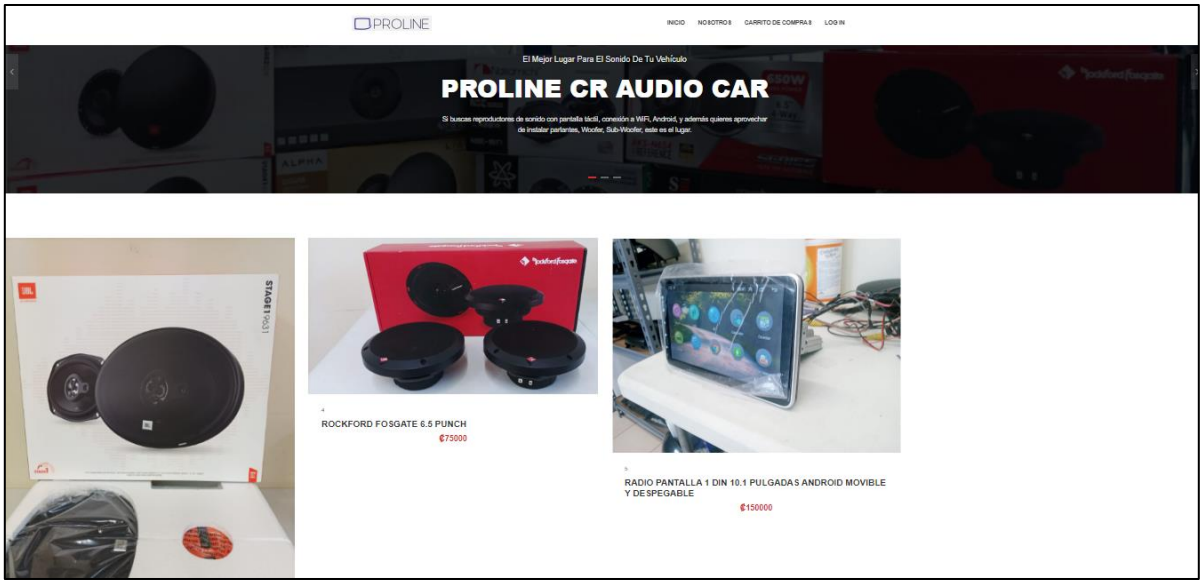
5.1.4.3 Módulo 3: Catálogo

Este módulo se compone por el controlador HomeController y sus vistas correspondientes. La vista Index de Home es la que carga como página de inicio para la solución. En ella se observa un carrusel con tres imágenes y mensajes. Debajo del carrusel, se muestra una lista de todos los productos, y si el usuario desliza el mouse sobre una imagen de un producto, podrá observar el botón para agregar el producto al carrito de compras. Si lo hace, se crea un registro para la tabla Pedido, y otro registro para la tabla PedidoXProducto.

La figura 55 (A continuación) muestra cómo se ve la vista Index de Home. La imagen se ajustó para que muestre el carrusel, y los primeros elementos de la lista de productos:

Figura 55.

Vista Index de Home.



Los elementos visuales se acomodan de manera simétrica en el eje vertical, dentro de una cuadrícula, de acuerdo a lo mencionado en 2.3.3 (Diseño de interfaces de usuario). Sin embargo, las imágenes son de diferentes tamaños en cuanto el eje horizontal, proporcionando una asimetría organizada en cada fila de tres elementos, teniendo como límite en el eje horizontal a la imagen de mayor tamaño.

La lista de los productos se muestra gracias a que la vista recibe un modelo de datos cuando el controlador la carga. Este modelo de datos es una lista que recibe todos los registros contenidos en la tabla de productos. Al usar un foreach para recorrer esta lista, se pueden recuperar los valores contenidos en cada atributo dentro de cada registro, y mostrarse en los diferentes elementos de la vista. En este caso, se muestran la imagen, la marca, el nombre y el precio de cada producto. A continuación, la figura 56 muestra el fragmento de código de la vista Index del controlador Home donde ocurre lo descrito:

Figura 56.

Fragmento del código de la vista Index del controlador Home.

```
<div class="col-sm-12 col-md-6 col-lg-3 product">
<div class="product-img">
    @if(item.Imagen != null)
    {
    <img src="@WC.IMAGEPATH@Html.DisplayFor(m=>item.Imagen )" alt="Product" />

    }else{
    <img src="~/images/Sin_imagen_disponible.jpg" alt="Product" />

    }
    <div class="product-hover">
        <div class="product-action">
            <form action="@Url.Action("LlenarCarrito", "Carrito", new {id = item.IdProducto} )">
            <button class="btn btn-primary" type="submit">Meter al carrito</button>
            </form>
            @*<a class="btn btn-primary" href="#">Ver detalles</a>*@
        </div>
    </div>
    <!-- .product-overlay end -->
</div>
<!-- .product-img end -->
<div class="product-bio">
    <div class="prodcut-cat">
        <a href="#">@Html.DisplayFor(m=>item.Marca)</a>
    </div>
    <!-- .product-cat end -->
    <div class="prodcut-title">
        <h3>
    <a href="#">@Html.DisplayFor(m=>item.Nombre)</a>
        </h3>
    </div>
    <!-- .product-title end -->
    <div class="product-price">
<span class="symbole">₡</span><span>@Html.DisplayFor(m=>item.Precio)</span>
    </div>
```

Si hace clic en otro producto para agregarlo al carrito de compras no se genera otro registro en la tabla Pedido, sino únicamente en la tabla PedidoXProducto. Esto se logra a través de dos métodos que se encuentran en el controlador CarritoController.

El primero, llamado LlenarCarrito, se encarga de crear un pedido con todos los campos en blanco menos el campo de ID que al ser autoincrementable se crea automáticamente sumando 1 al valor del ID del último registro en la tabla. Luego, recupera el número del ID y lo coloca en una variable de tipo TempData para poder conservar esa ID y utilizarla luego. Además, crea un registro en la tabla ProductoXPedido con la ID del pedido recién creado, y el ID del producto al que se le hizo clic. Luego, carga otra vista, igual que la vista Index, pero con la diferencia que ésta segunda vista, llamada Seguir, invoca al segundo método, LlenarCarrito2.

A partir de este punto, siempre que se cargue alguna de las vistas del controlador Home, se valida si la variable TempData que almacena el ID del pedido se encuentra vacía (Lo que solo

ocurre cuando se carga la página por primera vez) o si tiene algún valor. Si ocurre lo segundo, entonces carga siempre la vista Seguir del controlador Home.

El método LlenarCarrito2 no crea un registro en la tabla Producto, sino que hace uso del ID almacenado en la variable de tipo TempData establecida en el método anterior, y así a través de un if crea otro registro en ProductoXPedido en caso de que se seleccione un producto por primera vez, y en caso de que se seleccione un producto por segunda o más veces, aumenta el valor del campo Cantidad cada vez que se seleccione dicho producto a través de un update. Esto es posible gracias a que la tabla ProductoXPedido usa una llave compuesta por las llaves foráneas ID_Pedido y ID_Producto, lo que permite relacionar una cantidad indefinida de productos a un mismo pedido, ya que cada registro se hace único al cambiar el valor de ID_Producto.

A continuación, las figuras 57 y 58 muestran los métodos LlenarCarrito y LlenarCarrito2, respectivamente.

Figura 57.

Método LlenarCarrito.

```
0 referencias
public ActionResult LlenarCarrito(int idped, Pedido pedido, Productoxpedido pXpedido, int id)
{
    _context.Pedidos.Add(pedido);
    _context.SaveChanges();

    idPedido = pedido.IdPedido;
    TempData["idPedidoEntreVistas"] = idPedido;

    idped = idPedido;
    pXpedido.IdPedido = idped;
    pXpedido.IdProducto = id;
    pXpedido.Cantidad = 1;

    _context.Productoxpedidos.Add(pXpedido);

    _context.SaveChanges();

    return RedirectToAction("Seguir", "Home");
}
```

Figura 58.

Método LlenarCarrito2.

```
0 referencias
public ActionResult LlenarCarrito2(int idped, Productoxpedido pXpedido, int id)
{
    idped = (int)TempData["idPedidoEntreVistas"];
    pXpedido.IdPedido = idped;
    pXpedido.IdProducto = id;

    var listapxped = _context.Productoxpedidos.ToList();
    int cantProdXPed = listapxped.Where(p => p.IdPedido == idped && p.IdProducto == id)
                        .Select(p => p.Cantidad)
                        .FirstOrDefault();
    cantProdXPed++;
    pXpedido.Cantidad = cantProdXPed;

    if (pXpedido.Cantidad == 1)
    {
        _context.Productoxpedidos.Add(pXpedido);
        _context.SaveChanges();

    }
    else if (pXpedido.Cantidad > 1)
    {
        _context.ChangeTracker.Clear();
        _context.Productoxpedidos.Update(pXpedido);
        _context.SaveChanges();
    }

    return RedirectToAction("seguir", "Home");
}
```

5.1.4.4 Módulo 4: Carro de compras

Este módulo se compone de varias vistas, ya que es aquí donde el cliente ingresa la mayor parte de la información necesaria para finalizar el pedido. Cuando el usuario hace clic en la opción “Carrito de Compras” en la barra superior derecha es dirigido a la vista Edit del controlador Carrito, donde podrá visualizar los pedidos que ha realizado.

Esta lista se llena de la misma manera que se llena la lista en el menú de inicio, al recorrer una tabla del modelo de datos. En este caso, se recorre la tabla ProductoXPedido y se lleva a cabo un select para todos los registros cuyo valor en ID_Pedido coincida con el valor almacenado en la variable de tipo TempData durante la ejecución del método LlenarCarrito. El usuario puede presionar el botón Continuar para pasar a la siguiente vista.

A continuación, las figuras 59 y 60 muestran el código encargado de llenar la lista donde se ven todos los productos relacionados con el pedido y su visualización en la vista correspondiente, respectivamente.

Figura 59.

Fragmento de código del controlador Carrito.

```
<h2>Verificar pedido</h2>

<hr />
<div class="row">
    <div class="col-md-8">

        @foreach (var item in Model)
            {
                @if (item.IdPedido == IdPed)
            {
                    <p>Producto: @item.IdProducto</p>
                    <p>Cantidad: @item.Cantidad</p>

                }
            }
    </div>
</div>
<div>
        <form action="@Url.Action("Create", "Clientes")">
            <button class="btn btn-primary" type="submit">Continuar</button>
        </form>
    </div>
```

Figura 60.

Fragmento la vista Edit del controlador Carrito.

El botón “Continuar” lleva al usuario a la vista Create del controlador Clientes. Allí encuentra un formulario para escribir sus datos personales. Al presionar el botón “Continuar” se capturan todos los valores ingresados en el formulario y se almacenan en un registro de la tabla Cliente en la base de datos y al mismo tiempo se crea una variable de tipo TempData para almacenar el

ID del cliente, que será utilizado más adelante. Justo después se carga la vista Edit del controlador Pagos.

A continuación, las figuras 61, 62 y 63 muestran: el código dentro del controlador Clientes encargado de dichas funciones; el código de la vista Create del mismo controlador; y la visualización de la vista.

Figura 61.

Fragmento de código del controlador Clientes.

```
// POST: Clientes/Create
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
0 referencias
public async Task<IActionResult> Create([Bind("IdCliente,Nombre,Telefono,TipoDeIdentidad,Correo")] Cliente cliente)
{
    if (ModelState.IsValid)
    {
        _context.Add(cliente);
        await _context.SaveChangesAsync();
        TempData["idClienteEntreVistas"] = cliente.IdCliente;
        return RedirectToAction("Edit","Pago");
    }
    return View(cliente);
}
```

Figura 62.

Fragmento de la vista Create del controlador Cliente.

```
<h4>Cliente</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="IdCliente" class="control-label">Número de identidad</label>
                <input asp-for="IdCliente" class="form-control" />
                <span asp-validation-for="IdCliente" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Nombre" class="control-label">Nombre</label>
                <input asp-for="Nombre" class="form-control" />
                <span asp-validation-for="Nombre" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Telefono" class="control-label">Teléfono</label>
                <input asp-for="Telefono" class="form-control" />
                <span asp-validation-for="Telefono" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="TipoDeIdentidad" class="control-label">Tipo de identidad: Persona Física, Jurídica, DIMEX...</label>
                <input asp-for="TipoDeIdentidad" class="form-control" />
                <span asp-validation-for="TipoDeIdentidad" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Correo" class="control-label">Correo</label>
                <input asp-for="Correo" class="form-control" />
                <span asp-validation-for="Correo" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Continuar" class="btn btn-primary" />
            </div>
```

Figura 63.

Visualización de la vista Create del controlador Cliente.

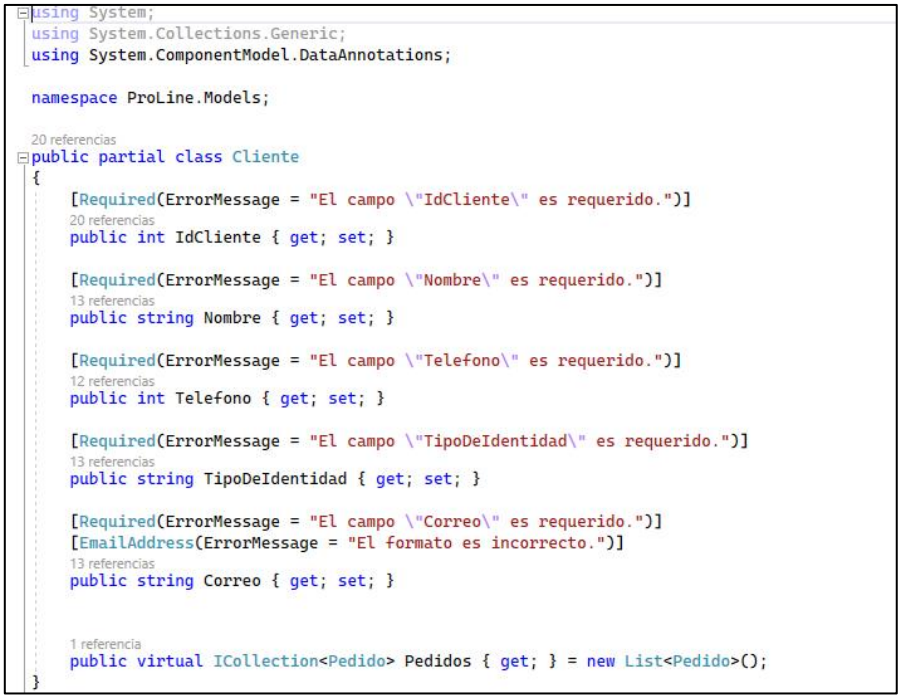
Los formularios que los usuarios deben llenar cuentan con una validación de los datos, para evitar que los datos obligatorios se envíen vacíos, y también asegurar un formato correcto en algunos campos como la dirección de correo electrónico, o la cantidad de dígitos en un número de teléfono. Los formularios en todas las vistas cuentan con esta validación, pero se tomará como ejemplo el formulario de esta vista de creación de clientes.

La validación es posible al agregar el método Required a los atributos de las clases dentro de la carpeta Modelo, y en el caso de formatos específicos, se usa otro método también. Por ejemplo, para validar que el string ingresado tiene el formato de un correo electrónico, se utiliza el método EmaiAddress. Estos métodos para la autenticación reciben como parámetro un string que muestra como mensaje de error cuando el campo está vacío (En el caso de Required) o cuando el dato ingresado no tiene el formato requerido (el caso de EmaiAddress).

A continuación, la figura 64 muestra el código de la clase Cliente.cs, que se encuentra en la carpeta Modelo, donde se puede observar el uso de lo mencionado en el párrafo anterior.

Figura 64.

Validación de campos desde la clase Cliente.cs en carpeta Modelo.



```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace ProLine.Models;

20 referencias
public partial class Cliente
{
    [Required(ErrorMessage = "El campo \"IdCliente\" es requerido.")]
    20 referencias
    public int IdCliente { get; set; }

    [Required(ErrorMessage = "El campo \"Nombre\" es requerido.")]
    13 referencias
    public string Nombre { get; set; }

    [Required(ErrorMessage = "El campo \"Telefono\" es requerido.")]
    12 referencias
    public int Telefono { get; set; }

    [Required(ErrorMessage = "El campo \"TipoDeIdentidad\" es requerido.")]
    13 referencias
    public string TipoDeIdentidad { get; set; }

    [Required(ErrorMessage = "El campo \"Correo\" es requerido.")]
    [EmailAddress(ErrorMessage = "El formato es incorrecto.")]
    13 referencias
    public string Correo { get; set; }


    1 referencia
    public virtual ICollection<Pedido> Pedidos { get; } = new List<Pedido>();
}
```

La vista Edit del controlador Pagos es la última parada del cliente antes de finalizar el pedido. Aquí encuentra información de cómo realizar el pago por medio de transferencia bancaria, y un formulario donde debe llenar los datos del pedido que se encuentran en blanco. Utilizando las variables de tipo TempData que almacenaron el ID del pedido y el ID del cliente, se completan

los campos respectivos. El usuario completa los demás campos (Excepto la fecha, que se guardó automáticamente cuando se creó el pedido en la vista Index del controlador Home). Al presionar el botón “Finalizar” se guardan los campos por medio de un update al registro del pedido en la tabla Pedido de la base de datos.

A continuación, las figuras 65, 66 y 67 muestran: el código dentro del controlador Pagos encargado de dichas funciones; el código de la vista Edit del mismo controlador; y la visualización de la vista.

Figura 65.

Fragmento de código del controlador Pagos.

```
// POST: PagoController/Edit/5
[HttpPost]
[ValidateAntiForgeryToken]
0 referencias
public ActionResult Edit(Pedido pedido, string FormaDePago, int NumeroDeTransferencia, string Direccion, string Comentario)
{

    if (ModelState.IsValid)
    {*/
    pedido.IdPedido = (int)TempData["idPedidoEntreVistas"];
    pedido.Cliente = (int)TempData["idClienteEntreVistas"];
    pedido.FormaDePago = FormaDePago;
    pedido.NumeroDeTransferencia = NumeroDeTransferencia;
    pedido.Direccion = Direccion;
    pedido.UsuarioAsignado = 1;
    pedido.Comentario = Comentario;

    _context.ChangeTracker.Clear();
        _context.Update(pedido);
        _context.SaveChanges();
        return RedirectToAction("Index","Home");
}
```

Figura 66.

Fragmento del código de la vista Edit del controlador Pagos.

```
</div>
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="IdPedido" />
            <div class="form-group">
                <input type="hidden" asp-for="Cliente" />
            </div>
            <div class="form-group">
                <label asp-for="FormaDePago" class="control-label"></label>
                <input asp-for="FormaDePago" class="form-control" />
                <span asp-validation-for="FormaDePago" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="NumeroDeTransferencia" class="control-label"></label>
                <input asp-for="NumeroDeTransferencia" class="form-control" />
                <span asp-validation-for="NumeroDeTransferencia" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="hidden" asp-for="Fecha" />
            </div>
            <div class="form-group">
                <label asp-for="Direccion" class="control-label"></label>
                <input asp-for="Direccion" class="form-control" />
                <span asp-validation-for="Direccion" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="hidden" asp-for="UsuarioAsignado" />
            </div>
            <div class="form-group">
                <label asp-for="Comentario" class="control-label"></label>
                <input asp-for="Comentario" class="form-control" />
                <span asp-validation-for="Comentario" class="text-danger"></span>
            </div>
            <div class="form-group">
```

Figura 67.

Visualización de la vista Edit del controlador Pagos.

5.1.4.5 Módulo 5: Información de contacto

En este módulo se encuentra el controlador Nosotros, y su vista Index. La vista muestra información de la empresa, incluyendo un mapa de Google que se agrega a la vista a través de

un contenedor de tipo iframe que lleva como parámetro el enlace de Google Maps con un pin puesto en la ubicación física del local de la empresa. Esto se muestra en la siguiente ilustración, figura 68, tras la cual la figura 69 muestra el código en la vista:

Figura 68.

Mapa de Google en vista Nosotros.

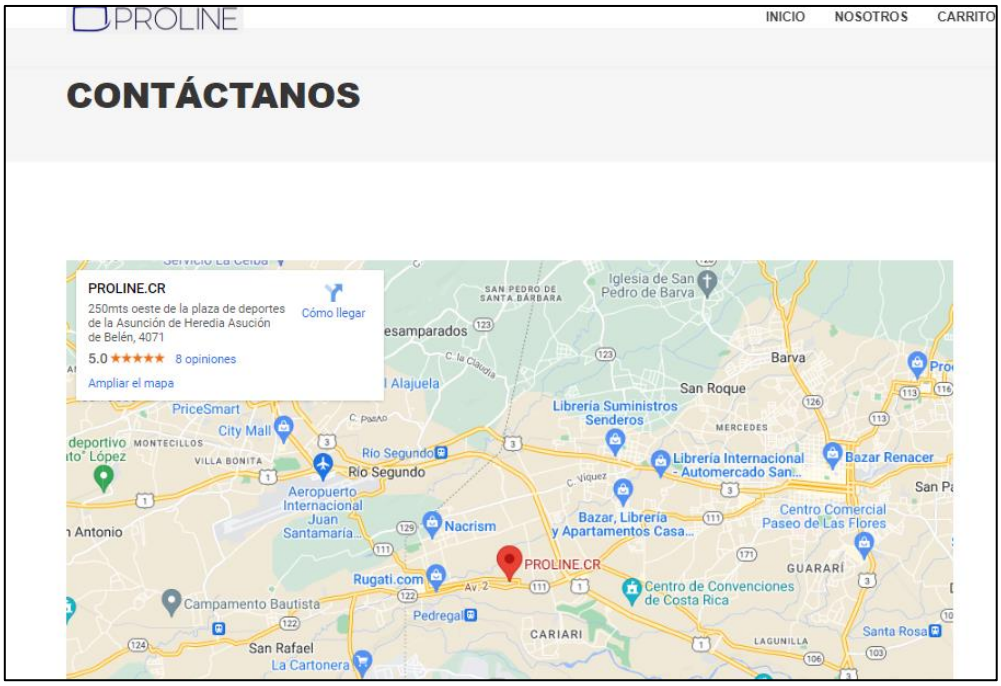


Figura 69.

Código de la vista Index del controlador Nosotros para el mapa de Google.

```
<!-- #page-title end -->
<!-- Google Maps
============================================= -->
<section class="google-maps pb-0">
    <div class="container">
        <div class="row">
            <div class="col-sm-12">
                <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d62867.95424287281!2d-84.2014288541831!3d9.996432967758048!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13
            </div>
        </div>
    </div>
</section>
<!-- .google-maps end -->
<!-- Contact #1
============================================= -->
<section class="contact">
    <div class="container">
        <div class="row">
            <div class="col-xs-12 col-sm-12 col-md-8 widget-contact pl-0 pr-0 p-none-xs p-none-sm">
                <form id="contact-form" method="post">
```

También hay un formulario para que el usuario pueda solicitar información a la empresa. En este formulario ingresa su nombre, su dirección de correo, y un mensaje. Al presionar el botón

Enviar, se envía un correo a la dirección info@audiocarprolinecr.com, que es el correo corporativo que maneja la empresa. El formulario ocupa dos tercios del espacio horizontal de la página, y el tercio restante es ocupado por la información de contacto, de conformidad con la regla de los tercios explicada en 2.3.3 (Diseño de interfaces de usuario).

A continuación, las figuras 70, 71 y 72 muestran la vista, el código de la vista y el código del controlador relacionados esta función, respectivamente:

Figura 70.

Fragmento de vista Nosotros: Envío de correo,

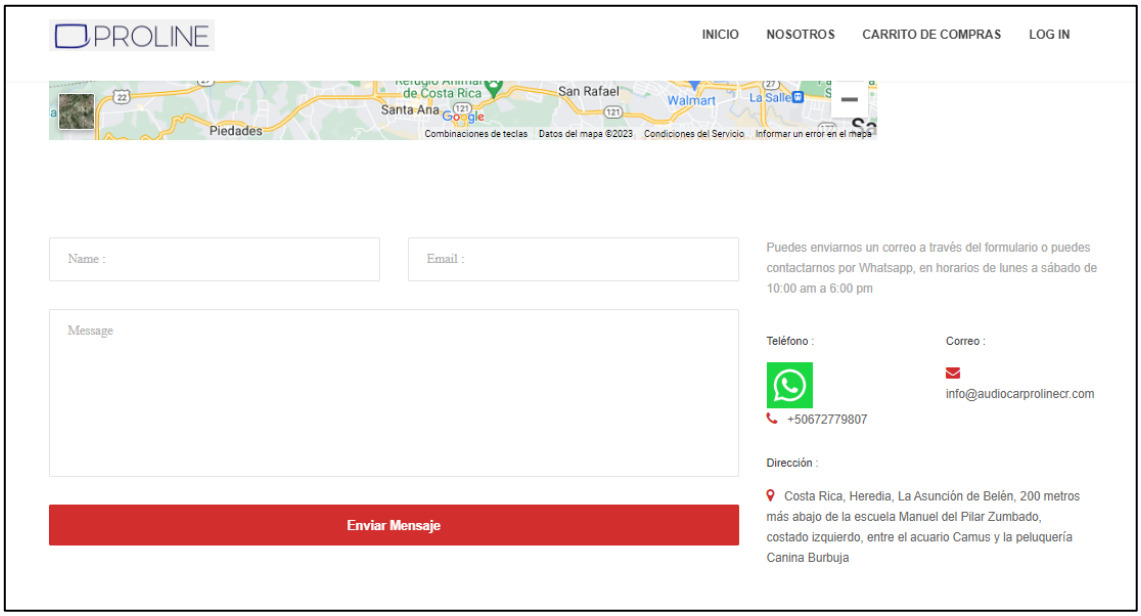


Figura 71.

Fragmento del código en la vista Nosotros para el envío de correo.

```
<section class="contact">
    <div class="container">
        <div class="row">
            <div class="col-xs-12 col-sm-12 col-md-8 widget-contact pl-0 pr-0 p-none-xs p-none-sm">
                <form id="contact-form" method="post">
                    <div class="col-md-6">
                        <input type="text" class="form-control mb-30" name="customerName" id="name" placeholder="Name :" required />
                    </div>
                    <div class="col-md-6">
                        <input type="email" class="form-control mb-30" name="customerEmail" id="email" placeholder="Email :" required />
                    </div>
                    <div class="col-md-12">
                        <textarea class="form-control mb-30" name="customerMessage" id="customerMessage" rows="4" placeholder="Message" required></textarea>
                    </div>
                    <div class="col-md-12">
                        <button type="submit" id="submit-message" class="btn btn-primary btn-block">Enviar mensaje</button>
                    </div>
                    <div class="col-xs-12 col-sm-12 col-md-12 mt-xs">
                        <!--Alert Message-->
                        <div id="contact-result">
                        </div>
                    </div>
                </form>
            </div>
```

Figura 72.

Código en el controlador de Nosotros para el envío de correo.

```
[HttpPost]
0 referencias
public IActionResult Index(String customerName, String customerEmail, String customerMessage)
{

    var message = new MimeMessage();
    message.From.Add(new MailboxAddress(customerName, "audiocarprolinecr.com"));
    message.To.Add(new MailboxAddress("ProLine", "audiocarprolinecr.com"));
    message.Subject = customerName + " " + customerEmail;
    message.Body = new TextPart("html")
    {
        Text = @customerMessage
    };
    using (var client = new MailKit.Net.Smtp.SmtpClient())
    {
        client.Connect("a2plcpnl0895.prod.iad2.secureserver.net", 587, true);

        // Note: only needed if the SMTP server requires authentication
        client.Authenticate("info@audiocarprolinecr.com", "ElReyBab");

        client.Send(message);
        client.Disconnect(true);
    }

    return View();

}
```

En la última imagen se puede apreciar que el envío de correo ocurre gracias a que el contenido del formulario se inserta en una variable de tipo message, y que luego se crea otra variable, de tipo SmtpClient, cuyos métodos reciben por parámetro la variable de tipo message con el cuerpo del correo, el host del servidor de correos electrónicos y las credenciales del usuario en ese host. Estos objetos y métodos son del namespace NetCore.MailKit, instalado como un paquete NuGet.

5.1.4.6 Módulo 6: Pedidos

Este módulo cuenta con el controlador AdminPedidos, y sus vistas asociadas: Index, Edit y Delete. La lógica y las vistas de este módulo son iguales a la del módulo de mantenimiento de catálogo, excepto que muestra al usuario administrador una lista de todos los pedidos realizados por los clientes en vez de mostrar los productos. No tiene la opción de crear pedidos, porque los usuarios administradores no necesitan esta opción, pero si necesitaran crear un pedido pueden ir al módulo correspondiente para los usuarios clientes.

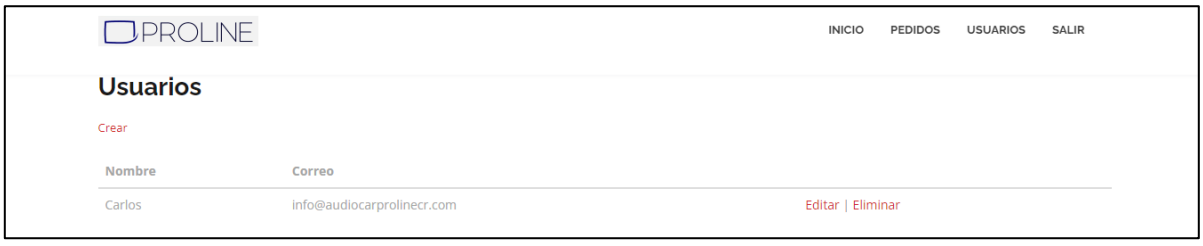
5.1.4.7 Módulo 7: Mantenimiento de usuario

El módulo de mantenimiento de usuario tiene le controlador AdminUsuarios, y sus vistas asociadas: Index, Create, Edit y Delete. La lógica y las vistas de este módulo, al igual que el anterior, son iguales a las del módulo de mantenimiento de catálogo, pero mostrando los usuarios que tienen acceso a los módulos de administración.

La figura 73 muestra la visualización de la vista Index del controlador AdminUsuarios:

Figura 73.

Vista Index del controlador AdminUsuarios.



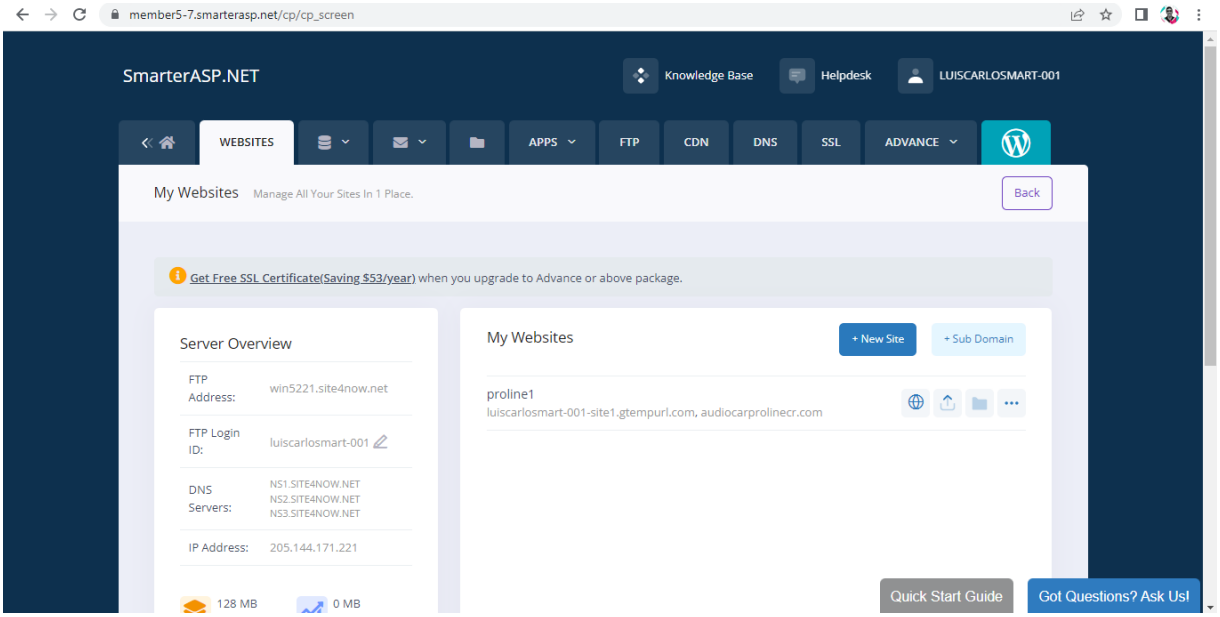
5.1.5 Despliegue en la web

El paso final para que la aplicación pueda ser utilizada es la publicación de ésta en un servidor web. Como se mencionó en 3.5.3, se usó el servicio de hosting de SmarterAsp.net. Éste permite un despliegue sencillo de proyectos de ASP.Net Core.

A continuación, la figura 74 muestra la interfaz del usuario en la página de inicio de SmarterAsp.net cuando se ha iniciado sesión, donde se puede observar el nombre de dominio como un sitio web publicado por medio de sus servicios.

Figura 74.

Panel de smarterasp.net.



La configuración que debe hacerse en Visual Studio 2022 para realizar el despliegue debe llevar los parámetros que se indican en el servidor web. Además se debe descargar un archivo de configuración que se aplica al proyecto a través de una herramienta de asistente. A continuación, la figura 75 muestra dicha configuración, y la figura 76 muestra el lugar donde se escriben dichos parámetros.

Figura 75.

Información de configuración en smarterasp.net.

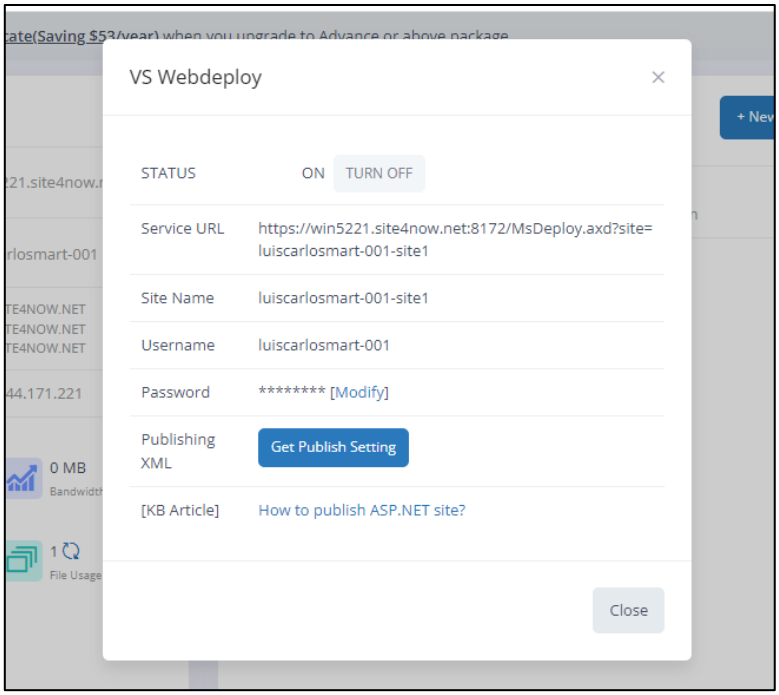
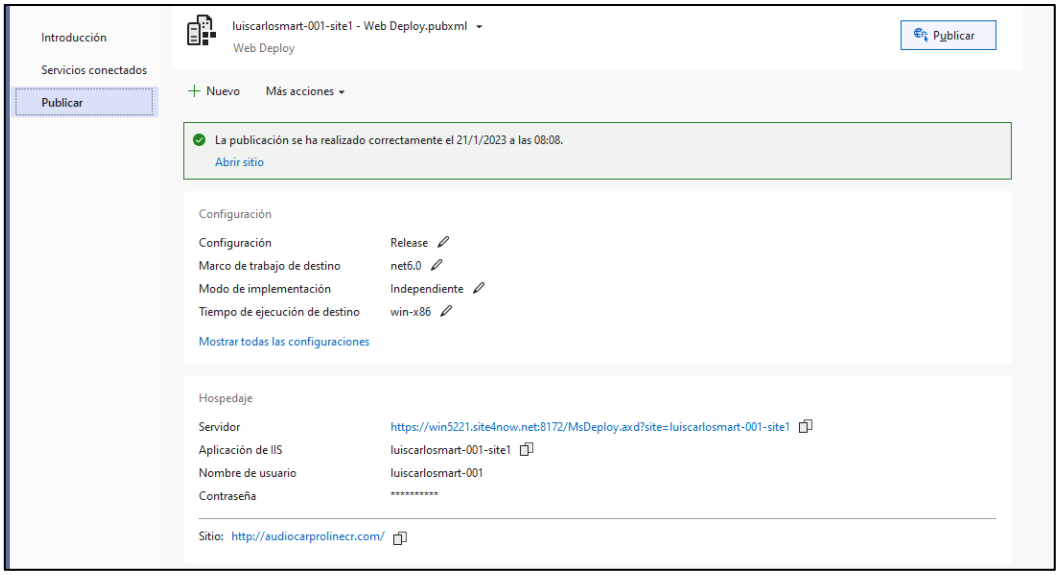


Figura 76.

Configuración para web deployment en Visual Studio.

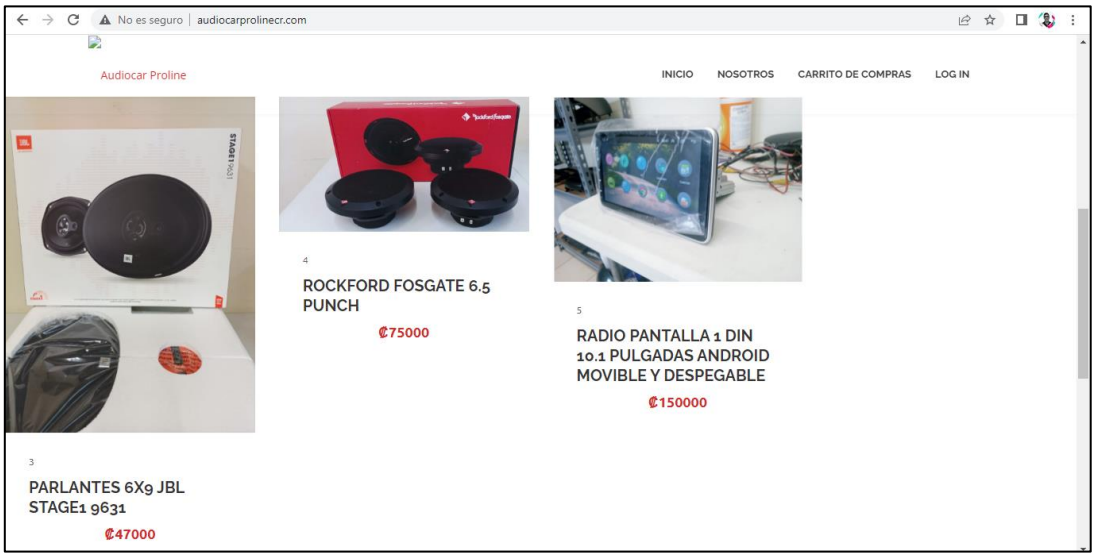


Cuando se ha configurado todo con los parámetros correctamente y se ejecuta la opción de publicar, Visual Studio sube todos los archivos al servidor web y crea otros archivos necesarios

para la ejecución de la aplicación en el servidor web. La figura 77 muestra la visualización de la aplicación ya publicada, desde un navegador a través de internet.

Figura 77.

Aplicación publicada en la web.



5.2 PRUEBAS

Esta sección reseña el plan de testing y las pruebas realizadas. Contiene además observaciones que se desprenden de lo observado durante las pruebas.

Dentro de los objetivos del proyecto, el cuarto objetivo se desarrolla en esta sección: Determinar y ejecutar las pruebas de la aplicación web junto a los usuarios que lo utilizarán, para verificar y validar el correcto funcionamiento de la aplicación web, así como el cumplimiento de los requerimientos.

La fase que se lleva a cabo en esta sección es la de pruebas del sistema y manual de usuario, y la variable observada es: Pruebas y validación de la aplicación web.

5.2.1 Plan de testing

El plan de testing consistió en la realización de cuatro tipos de pruebas: Pruebas unitarias, pruebas de integración, pruebas de validación y pruebas del sistema. Esto de conformidad con lo propuesto en 3.5.3. Cada prueba se realizó haciendo uso de la plantilla mostrada en 2.6 (Técnicas para probar y validar software). Los encabezados a continuación describen las diferentes pruebas realizadas.

5.2.2 Pruebas unitarias

Se revisó el código fuente en tiempo de ejecución para comprobar que todas las funciones trabajan correctamente. Cada módulo fue probado para verificar su funcionalidad y el manejo adecuado de excepciones. Cuando se encontraron errores, éstos fueron corregidos y la prueba fue realizada nuevamente después de la corrección.

A continuación, la tabla 15 muestra el uso de una plantilla de test. En este caso, se realizaron pruebas al módulo de inicio de sesión.

Tabla 15.

Testing de Módulo 1: Inicio de sesión.

Nombre: Pruebas unitarias del módulo inicio de sesión.				
Referencia	Acción	Resultado esperado	Resultado obtenido	Estado
Caso test U.1.1	Campos vacíos			
Escenario 1	No llenar ningún campo y presionar “enviar”.	Mensaje de error indicado cada campo vacío que se debe llenar.	Se permanece en la misma página y se muestra mensaje de error.	Ok

Escenario 2	Dejar solo un campo en blanco y presionar enviar.	Mensaje de error indica que dicho campo está en blanco y se debe llenar.	Se permanece en la misma página y se muestra mensaje de error.	Ok
Caso test U.1.2	Lo introducido no cumple con las condiciones			
Escenario 1	Ingresar un dato en el campo de correo electrónico que no siga el formato adecuado.	Mensaje de error que pide corregir el campo del correo electrónico y usar un formato válido.	Se permanece en la misma página y se muestra mensaje de error.	Ok
Caso test U.1.3	Se introducen credenciales incorrectas			
Escenario 1	Ingresar datos que no existen en la base de datos.	Mensaje de error indicando que los datos son incorrectos.	Se permanece en la misma página y se muestra mensaje de error.	Ok
Caso test U.1.4	Se introducen credenciales correctas			
Escenario 1	Ingresar datos que existen en la base de datos.	Carga la página de AdminHome.	Carga la página de AdminHome.	Ok
Realizado por:	Luiscarlos Martínez		Resultado:	Exitosa
Observaciones:	Se puede cambiar el nombre a Admin Log In. Se debe agregar la funcionalidad de recuperación de contraseña.			

5.2.3 Pruebas de integración

Durante la fase de diseño (Véase 4.3) se realizaron los diagramas de diseño basado en eventos (Véase 4.3.1). A través de las pruebas de integración se verificó que los eventos de cada módulo

desencadenaran las acciones correctas, especialmente las que son relacionadas a la conexión con la base de datos y las operaciones relacionadas con el manejo de la información (Lectura, agregar, quitar y modificar campos) incluyendo el paso entre vistas de información que no se guarda en la base de datos.

A continuación, la tabla 16 muestra el uso de una plantilla de test en la aplicación de una prueba de integración:

Tabla 16.

Prueba de integración de módulo 7: mantenimiento de usuarios.

Nombre: Pruebas de integración del módulo de mantenimiento de usuarios.				
Referencia	Acción	Resultado esperado	Resultado obtenido	Estado
Caso test I.7.1	Crear un usuario nuevo			
Escenario 1	Se llena el formulario y se presiona el botón Crear.	En la vista de index de usuarios debe aparecer el usuario nuevo en la lista de usuarios.	El usuario se creó correctamente y aparece en la lista de usuarios.	Ok
Escenario 2	Se intenta iniciar sesión con el usuario nuevo.	Carga la vista de inicio para administradores con acceso a las vistas correspondientes.	La vista y el layout correspondientes cargan correctamente.	Ok
Caso test I.7.2	Editar un usuario			
Escenario 1	Se intenta cambiar el nombre de usuario, el correo y la contraseña de un usuario.	Los cambios son visibles en la vista de index de usuarios (excepto la contraseña, que no se muestra).	Los cambios se efectuaron correctamente y son visibles en la vista de index de usuarios.	Ok

Escenario 2	Se intenta iniciar sesión con las nuevas credenciales	El usuario puede iniciar sesión con las nuevas credenciales.	Se inicia sesión exitosamente con las nuevas credenciales.	Ok
Caso test I.7.3	Eliminar un usuario			
Escenario 1	Se intenta eliminar un usuario.	El usuario deja de aparecer en la lista de la vista index de usuarios.	Se elimina correctamente un usuario y deja de aparecer en la lista de la vista index de usuarios.	Ok
Escenario 2	Se intenta iniciar sesión con un usuario eliminado.	El usuario intenta iniciar sesión con las credenciales de un usuario eliminado y el intento debe fallar.	Falló el intento de inicio de sesión.	Ok
Realizado por:	Luiscarlos Martínez		Resultado:	Exitosa
Observaciones:	Ninguna.			

5.2.4 Pruebas de validación

Durante la fase inicial del proyecto se llevó a cabo la recopilación de requerimientos (Véase 4.2.1) y a partir de ella se realizó el análisis de estos, dando como resultado los escenarios de caso de uso (Véase 4.2.2). Las pruebas de validación se llevaron a cabo junto con los usuarios finales de la aplicación web, verificando los documentos de escenarios de caso de uso. También se validó que se siguieran prácticas recomendadas para aplicaciones de E-Commerce, de acuerdo con lo descrito en 2.4.

Como resultado de estas pruebas, los usuarios finales se dieron por satisfechos con respecto a sus requerimientos iniciales. Se registraron sus observaciones para futuras mejoras. Las observaciones más importantes para ellos son:

- Se desea que se muestre el nombre de la marca y no el ID de la marca en las vistas donde se muestran productos.
- Se desea simplificar el proceso para eliminar pedidos, ya que actualmente hay que eliminar primero los registros de Productos por Pedido antes de eliminar un pedido (Debido a la integridad referencial en la base de datos).
- Se desea simplificar el proceso de creación de productos, para poder crear una marca y un producto en la misma vista.

A continuación, la tabla 17 muestra una plantilla de plan de testing usada para realizar una prueba de validación.

Tabla 17.

Prueba de validación del escenario de caso de uso 2: Mantenimiento de catálogo.

Nombre: Pruebas de validación del escenario de caso de uso 2: Mantenimiento de catálogo.				
Referencia	Acción	Resultado esperado	Resultado obtenido	Estado
Caso test V.2.1	Acceso único para administradores			
Escenario 1	El usuario intenta acceder a la vista de administración del catálogo (AdminHome) sin iniciar sesión a través del enlace directamente.	Se redirecciona a la vista de iniciar sesión.	Se redirecciona al usuario a la vista de iniciar sesión, lo que el usuario final da por aceptable.	Ok
Caso test V.2.2	El usuario quiere agregar productos al catálogo			

Escenario 1	El usuario intenta crear un producto nuevo.	El usuario crea un producto y éste es visible en la vista de inicio de los usuarios clientes, con todas las características a mostrar.	El producto se crea correctamente y son visibles en el catálogo. El usuario final lo da por aceptable, con algunas observaciones.	Ok
Caso test V.2.3	El usuario quiere eliminar productos al catálogo			
Escenario 1	El usuario intenta eliminar un producto.	El producto se elimina y no aparece más en el catálogo.	Se elimina correctamente un producto y deja de aparecer en el catálogo. El usuario lo da por aceptable.	Ok
Caso test V.2.4	El usuario quiere editar un producto del catálogo			
Escenario 1	El usuario intenta editar los campos de un producto.	El usuario intenta editar los campos de un producto y estos cambios deben reflejarse en el catálogo.	Los cambios se reflejan en el catálogo. El usuario lo da por aceptable.	Ok
Realizado por:	Luiscarlos Martínez		Resultado:	Exitosa
Observaciones:	El usuario quisiera que desde la vista de agregar un nuevo producto se pueda agregar también una nueva marca para no tener que ir de una vista a otra. También, el usuario quisiera que más adelante se puedan incluir varias fotografías a cada producto.			

5.2.5 Pruebas del sistema

Una vez realizadas las pruebas anteriores y de haber conseguido aprobar la validación por parte del usuario final, se procedió a darle uso a la aplicación. Inicialmente está en fase de pruebas

del sistema, que son las últimas pruebas realizadas. Se instruyó a los usuarios para que hagan uso de todas las funciones del sistema, y se mantuvo la comunicación en la etapa inicial de la implementación de la solución en la jornada laboral de la empresa.

Para esta etapa de pruebas no se utilizó la plantilla de test, sino que se creó un documento para registrar las incidencias, observaciones y peticiones de los usuarios. Este documento, llamado "Seguimiento para la aplicación web ProLine", se encuentra en los apéndices (Apéndice 2) y no se describe en detalle su uso o sus posibles beneficios, aunque este tema se toca en el apartado de las recomendaciones.

5.3 MANUAL DE USUARIO

Esta sección contiene una reseña del manual de usuario que se entregó al usuario final. Éste puede encontrarse en la sección de apéndices (Apéndice 1).

Dentro de los objetivos del proyecto, el quinto objetivo se desarrolla en esta sección: Redactar el manual de usuario final llevando a cabo la documentación enfocada al uso de la aplicación web para que los colaboradores de ProLine CR se capaciten en el uso y mantenimiento de esta.

La fase que se lleva a cabo en esta sección es la de pruebas del sistema y manual de usuario, y la variable observada es: Manual de usuario con instrucciones de uso para los usuarios finales.

El manual de usuario fue realizado de acuerdo a lo mencionado en 3.5.5. Está guardado en una carpeta de Google Drive, y fue compartido con los usuarios finales. Contiene las siguientes secciones:

- Portada.
- Índice.
- Introducción.
- Contenido.

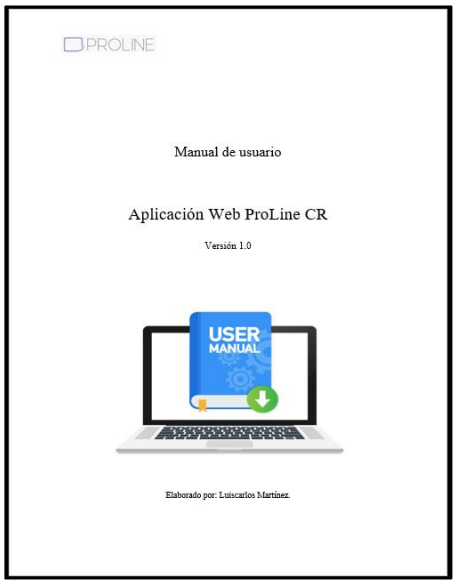
Se da a continuación una breve reseña de cada parte del manual.

Portada

Es la primera página del documento, y contiene el nombre de la aplicación web así como su versión más actual y el nombre del autor del manual. A continuación, la figura 78 muestra la imagen de la portada del manual de usuario:

Figura 78.

Portada de manual de usuario.

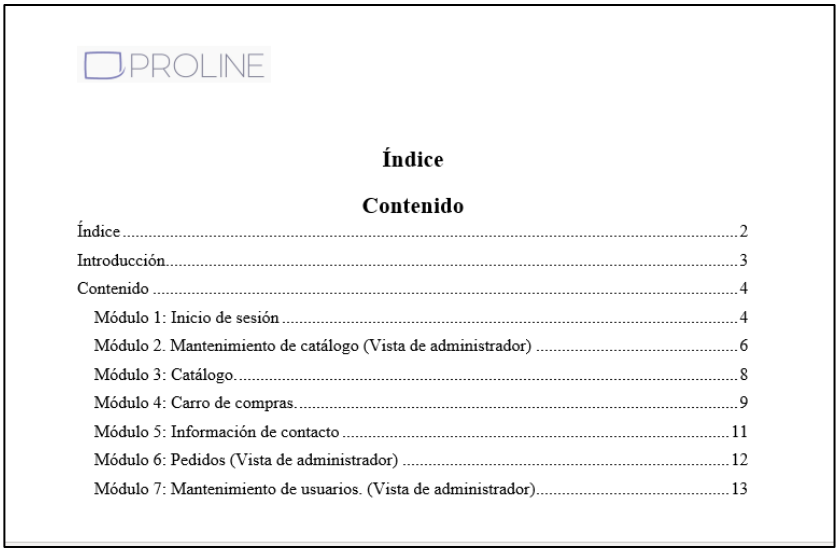


Índice

El índice del documento se encuentra en la segunda página. Muestra las páginas donde se encuentra cada sección, y las páginas donde se encuentra la explicación de cada módulo. Al hacer clic en cualquier título, se dirige al usuario a la página donde se encuentra el contenido. La figura a continuación, figura 79, muestra el índice.

Figura 79.

Índice del manual de usuario.



PROLINE

Índice

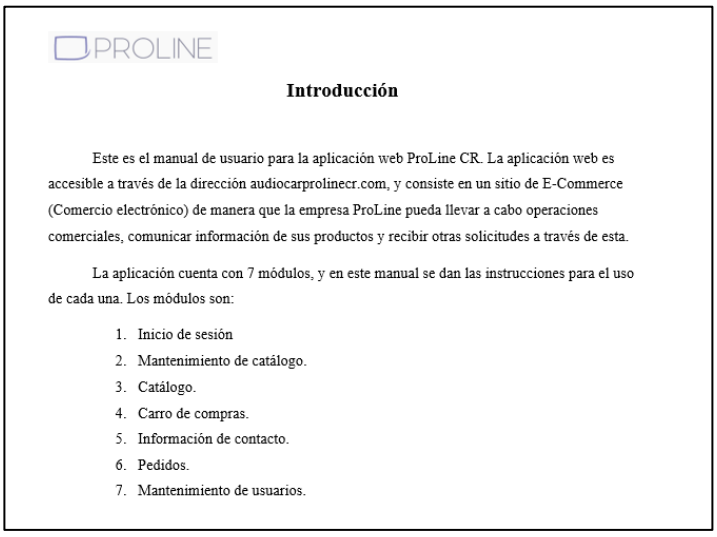
Contenido

Introducción

La introducción es la tercera página del documento. Se da una breve explicación de la aplicación web y se reseña de lo que el manual contiene. La figura 80 muestra la introducción del manual de usuario.

Figura 80.

Introducción del manual de usuario.



PROLINE

Introducción

Este es el manual de usuario para la aplicación web ProLine CR. La aplicación web es accesible a través de la dirección audiocarprolinecr.com, y consiste en un sitio de E-Commerce (Comercio electrónico) de manera que la empresa ProLine pueda llevar a cabo operaciones comerciales, comunicar información de sus productos y recibir otras solicitudes a través de esta.

La aplicación cuenta con 7 módulos, y en este manual se dan las instrucciones para el uso de cada una. Los módulos son:

1. Inicio de sesión
2. Mantenimiento de catálogo.
3. Catálogo.
4. Carro de compras.
5. Información de contacto.
6. Pedidos.
7. Mantenimiento de usuarios.

Contenido

El contenido del manual se compone de información y gráficas explicativas para indicar al usuario cómo hacer uso de la aplicación. Las figuras 81 y 82 (a continuación) ofrecen una muestra del contenido del manual.

Figura 81.

Fragmento del manual de usuario: Módulo 2.


PROLINE

Módulo 2. Mantenimiento de catálogo (Vista de administrador)

La página de inicio para administradores es el lugar donde se observa una lista de todos los productos publicados, con las opciones de crear un producto nuevo, editar los productos ya existentes o eliminar un producto. Cuando selecciona alguna de esas tres opciones, se carga su vista respectiva desde donde puede realizar la acción solicitada.

Es importante mencionar que la marca de un producto debe ser agregada antes de agregar el producto, para que pueda ser escogida entre las opciones al momento de crear un producto. Para ello, debe hacer clic en Crear Nueva Marca. Cargará entonces la vista con el formulario para crear una nueva marca donde únicamente se debe escribir el nombre de la nueva marca:

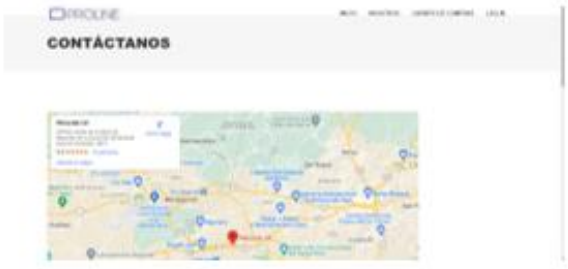
Figura 82.

Fragmento del manual de usuario: Módulo 5.

PROLINE

Módulo 5: Información de contacto

Para llegar a esta vista se debe hacer click arriba a la derecha en donde dice "NOSOTROS". En esta vista se muestra un mapa de Google con la ubicación del local, la información de contacto de la empresa: Correo, teléfono y dirección del local; y también hay dos opciones para que el usuario se comunique directamente con un administrador: el botón para abrir un chat de Whatsapp, y el formulario para enviar un mensaje por correo electrónico a la cuenta info@audiocarprolinecr.com.



CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES

Este capítulo presenta las conclusiones y recomendaciones del proyecto, llevándose a cabo un análisis de los resultados obtenidos durante el proceso de investigación.

6.1 CONCLUSIONES

A continuación, se enumeran los objetivos del proyecto y las conclusiones que se desprenden de cada uno, empezando por el objetivo general y siguiendo por los objetivos específicos.

6.1.1 Conclusiones del objetivo General

Entregar una propuesta de software a la empresa ProLine para implementar una aplicación web que se ajuste a sus requerimientos mejorando su comunicación con clientes y potenciales clientes.

El proceso de elaboración de la propuesta para la implementación de dicha aplicación web arrojó las siguientes conclusiones:

- A través del diagnóstico de la situación actual, en su apartado 4.1 (Descripción de la situación actual) queda clara la necesidad de una aplicación web para mejorar el flujo de trabajo y ampliar los canales de comunicación entre la empresa y clientes y potenciales clientes.
- Al finalizar la recolección y análisis de los datos en 4.2, se determina la viabilidad de implementar una aplicación web de acuerdo con la herramienta de casos de uso.
- Terminadas las pruebas de validación, descritas en 5.2.4, se concluye que la implementación de la aplicación web, propuesta como solución a la problemática, cumple con todos los requerimientos planteados de acuerdo con las observaciones y comentarios de los usuarios finales.

6.1.2 Conclusiones de los objetivos específicos

Determinar los requerimientos de una aplicación web para la empresa ProLine CR con la finalidad de definir las características de la aplicación a implementar.

Este objetivo se basó en la variable: Requerimientos de la empresa ProLine para la aplicación web. Como instrumentos, se usaron: la plantilla de historia de usuario y los diagramas de caso de uso. Como resultado, se tienen las siguientes conclusiones:

- Al llevar a cabo las entrevistas reseñadas en el punto 4.2.1, se logró determinar, eficazmente, los aspectos específicos que la empresa necesita para la implementación de una aplicación web.
- A través de los diagramas de caso de uso elaborados en el punto 4.2.3 efectivamente se lograron definir las características principales de la aplicación web, sirviendo esto de guía para el resto del proceso.

Elaborar el diseño lógico y físico del software y de la base de datos, usando el resultado del análisis de los requisitos recopilados para llevar a cabo una implementación exitosa que satisfaga todos los requisitos planteados.

La variable en la que se basó este objetivo es: Diseño lógico y físico del software y la base de datos. Los instrumentos usados fueron: Diagramas de diseño basado en eventos y diagramas de modelo de base de datos. Las conclusiones que se desprenden son:

- Se constata que el proceso de desarrollo se ve facilitado al contar con diagramas de diseño, como los elaborados en el punto 4.3.1, ya que, al estar sumergido en el desarrollo del código, éstos resultaron útiles para retomar el rumbo una vez que surgieron dudas o dificultades técnicas.

- Se reconfirmó la utilidad de un diagrama de modelo de bases de datos, ya que el realizado en el punto 4.3.2 sirvió como un recordatorio de la manera en que se relacionan los datos, proporcionando una guía clara durante el proceso de desarrollo.

Implementar una aplicación web con base en el diseño elaborado, según los requisitos recopilados, para permitir una mejor comunicación con los clientes y potenciales clientes de ProLine CR.

La variable en que se basó este objetivo es: Aplicación web para mejorar la comunicación con clientes y potenciales clientes de la empresa. El instrumento usado fue: Aplicación web hospedada en SmarterASP y accesible usando internet. Se obtienen las siguientes conclusiones:

- Se concluye que el lenguaje C#, el framework ASP.Net Core y el modelo de arquitectura MVC reseñados en los puntos 2.5.1, 2.5.2 y 2.5.3, respectivamente, y utilizados en la fase 5.1 facilitan el desarrollo de aplicaciones web al proporcionar un patrón claro para mantener el orden y la estructura del código desarrollado, además de permitir llevar a cabo todas las funciones necesarias para el cumplimiento de los requerimientos señalados en el punto 4.2.1.
- Se demuestra que es posible mantener una comunicación asíncrona y efectiva por medio de las funciones que la aplicación web ofrece en los módulos de catálogo, carrito de compras e información de contacto, descritos en los puntos 5.1.4.3, 5.1.4.4 y 5.1.4.5.
- Se logró determinar que la aplicación web propuesta cubre las necesidades descritas en el punto 4.2.1 y que las mismas responden a la realidad actual de la empresa; sin embargo, en la medida en que los requerimientos sean mayores se necesitará un software mucho más amplio y robusto.

Determinar y ejecutar las pruebas de la aplicación web junto a los usuarios que lo utilizarán, para verificar y validar el correcto funcionamiento de la aplicación, así como el cumplimiento de los requerimientos.

Este objetivo se basó en la variable: Pruebas y validación de la aplicación web. Como instrumentos se usó la plantilla de plan de test. Las conclusiones obtenidas son las siguientes:

- Se verifica que la plantilla de test, explicada en el punto 2.6 y utilizada en el punto 5.2, se puede ajustar a las necesidades de los diferentes tipos de pruebas a realizar, ya que permiten una planificación efectiva y un registro asertivo de las pruebas y sus resultados.
- Se evidencia la importancia de realizar las pruebas indicadas en el punto 5.2 antes de la implementación de una solución de software para corregir errores que entorpecen o incluso a veces imposibilitan el buen uso de la solución.
- Se confirma que la participación de los usuarios finales en el proceso de validación reseñado en el punto 5.2.4 es fundamental para lograr su satisfacción con la solución y la consecución de los objetivos propuestos.

Redactar el manual de usuario final enfocado al uso de la aplicación web para que los colaboradores de ProLine CR se capaciten en el uso y mantenimiento de dicha aplicación.

La variable sobre la cual se basó este objetivo es: Manual de usuario con instrucciones de uso para los usuarios finales. El instrumento usado es el documento del manual de usuario final. Tras lo cual se concluye:

- Se determina que el manual de usuario presentado en el punto 5.3 es un recurso valioso para acelerar la curva de aprendizaje de los nuevos usuarios y representa una fuente fundamental de consulta con información disponible en todo momento, para evitar la dependencia de un especialista para resolver dudas.

- Se demuestra que la práctica de almacenar el manual de usuario en una fuente de fácil acceso y edición, de acuerdo con lo señalado en el punto 3.5.5 y realizado en el punto 5.1, es de utilidad para mantenerlo actualizado a medida que se introducen cambios y mejoras en la aplicación web; facilitados en especial por la función de “Comentarios” que posee Google Docs.

6.2 RECOMENDACIONES

Esta sección contiene las recomendaciones dadas a partir de la implementación de la aplicación web para comercio electrónico de la empresa ProLine, las cuales se declaran a continuación:

- Se recomienda llevar a cabo la implementación propuesta en este proyecto, debido a que cumple con los objetivos determinados al inicio.
- Se insta a hacer uso del documento de seguimiento para la aplicación web entregado al final del proyecto, para continuar realizando mejoras y adaptar la solución a futuras tendencias para mantener la innovación y la competitividad de la empresa.
- Se recomienda extender invitaciones a los clientes para que revisen la aplicación web y la utilicen, de manera que puedan enviar sus impresiones y recomendaciones, a través del formulario de contacto, para así tener una fuente importante de retroalimentación con la finalidad de medir el impacto que la herramienta tiene y obtener también ideas para mejoras futuras.
- Se aconseja considerar la implementación de un inicio de sesión para usuarios clientes, con un nuevo módulo que les permita gestionar un histórico de pedidos, especialmente para clientes que compran al mayor, como un medio de fidelizarlos y facilitar la gestión de los datos producidos por el uso de la aplicación web.

- Se propone la adición de un módulo blog que permita compartir más información, que resulte atractiva y actualizada, como una manera de fomentar el consumo de los productos y servicios que la empresa ofrece, así como de posibilitar el establecimiento de una comunidad capaz de compartir experiencias positivas y conocimiento entre sí.
- Se aconseja informarse acerca de las técnicas de mercadeo por medio de internet para explotar aún más el recurso que la aplicación web representa.

REFERENCIAS BIBLIOGRÁFICAS

Alonso Amo, F., Martínez Normand, L. y Segovia Pérez, J. (2005). *Introducción a la Ingeniería del Software: modelos de desarrollo de programas*. Delta Publicaciones. Disponible en: https://elibro.net.uh.remotexs.xyz/es/ereader/bibliouh/170188?page=92. Consultado en: 07 de agosto de 2022

Anand, R. V., & Dinakaran, M. (2016). Popular agile methods in software development: Review and analysis. *International Journal of Applied Engineering Research*, 11(5), 3433-3437. Recuperado el 20 de marzo de 2023, de https://ijsta.com/papers/IJSTAV2N4Y16/IJSTA-V2N4R32Y16.pdf.

Aramburu Marcos y Gómez Juan Carlos. (2021). *¿Quién dijo que las tiendas físicas están en extinción?*. Think with Google. Recuperado el 12 de enero de 2022, de www.thinkwithgoogle.com/intl/es-419/insights/recorrido-del-consumidor/tiendas-omnicanal/ .

Beaird, J. (2007). *The principles of beautiful web design*. Sitepoint. Recuperado el 21 de marzo de 2023, de https://www.home.uni-osnabrueck.de/elsner/Skripte/Material/HTML/Artikel/The%20Principles%20of%20Beautiful%20Web%20Design.pdf.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... Thomas, D. (2001). *Manifiesto por el Desarrollo Ágil de Software*. Agile Alliance. Recuperado el 20 de marzo de 2023, de https://agilemanifesto.org/iso/es/manifesto.html.

Blum, Bruce I. (1996). *Beyond Programming, To a New Era of Design.* Oxford University Press, New York. Disponible en https://books.google.co.cr/books?hl=es&lr=&id=qYHmCwAAQBAJ&oi=fnd&pg=PR

13&dq=blum+1996+beyond+programming&ots=pyXyQMrB9t&sig=si1iENMm2721EWdSCbtDubXzhiE#v=onepage&q&f=false. Recuperado el 6 de agosto de 2022.

Canossa Montes De Oca, Héctor Andrés. (2019). *Planes de negocios: el comercio electrónico y la gestión de empresas en Costa Rica.* Revista Nacional de Administración de la UNED. Recuperado el 15 de diciembre de 2021, de https://revistas.uned.ac.cr/index.php/rna/article/view/2738/3558 .

Cass, Stephen. *Top Programming Languages.* (2022). Blog de la IEEE Spectrum. Recuperado el 29 de octubre de 2022, de https://spectrum.ieee.org/top-programming-languages-2022.

Cockburn, Alistair. (2001). *Writting Effective Use Cases.* Addison-Wesley. Recuperado el 20 de agosto de 2022, de https://ebooks7-24.com:443/?il=16414.

Cordero Pérez, C. (2021) *Comercio electrónico inició este año con vitalidad en Costa Rica y podría acelerar el ritmo.* Diario El Financiero. Recuperado el 10 de diciembre de 2021, de: https://www.elfinancierocr.com/tecnologia/comercio-electronico-inicio-este-2021-con/WQPEMZ6NPFEERHRJL4M4XOCSGQ/story/ .

Da Silva, Douglas. (2021). *¿Qué es omnicanal? Entiende el concepto.* Blog de ZenDesk. Recuperado el 6 de enero de 2022, de https://www.zendesk.com.mx/blog/omnicanal-que-es/ .

Eisenstein, E. (1994*). La revolución de la imprenta en la edad moderna europea (Vol. 162)*. Ediciones AKAL. Recuperado el 12 de febrero de 2023, de https://books.google.co.cr/books?hl=es&lr=&id=RFOH0qiHv_AC&oi=fnd&pg=PA7&dq=impacto+de+la+imprenta+en+la+comunicacion&ots=15KhuSFWXo&sig=gbt_7hh

Z1EKZkVXFliDQZjd7Fqs&redir_esc=y#v=onepage&q=impacto%20de%20la%20imprenta%20en%20la%20comunicacion&f=false .

Forbes Staff. (2021). *Costa Rica registró un crecimiento del 48% en las ventas en línea.* Forbes Centroamérica. Recuperado el 21 de diciembre de 2021, de https://forbescentroamerica.com/2021/06/28/costa-rica-registro-un-crecimiento-del-48-en-las-ventas-en-linea/ .

Hilard, Vincent. (2020). Gestión de un proyecto web - *Planificación, dirección y buenas prácticas (2da edición).* Biblioteca virtual Eni. Recuperado el 6 de agosto de 2022, de https://www.eni-training.com.uh.remotexs.xyz/portal/client/mediabook/home.

Larman, Craig. (2004). *Applying UML and Patterns (3ra edición)*. Pearson. Recuperado el 20 de agosto de 2022, de https://www.craiglarman.com/wiki/downloads/applying_uml/larman-ch6-applying-evolutionary-use-cases.pdf.

Laudon, K. C., Guercio Traver, C. (2014). *E-commerce 2013: negocios, tecnología, sociedad.* Pearson Educación. https://www-ebooks7-24-com-uh.knimbus.com:443/?il=3298

Marqués-Andrés, M. (2011). *Bases de datos*. Universitat Jaume I. Recuperado el 14 de septiembre de 2022, de http://repositori.uji.es/xmlui/bitstream/handle/10234/24183/s18.pdf?sequence=6&isAllowed=y.

Maxim, Bruce R. y Pressman, Roger S. (2021). *Ingeniería de software. Un enfoque práctico (9na edición).* McGraw Hill. Recuperado el 6 de agosto de 2022, de https://www-ebooks7-24-com-uh.knimbus.com:443/?il=16414&pg=1

Microsoft. (2023) Recuperado el 3 de enero de 2023, de https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp.

Microsoft. (2023) Recuperado el 3 de enero de 2023, de https://learn.microsoft.com/es-es/dotnet/core/introduction.

MongoDB. (Junio, 2018). Top 5 Considerations When Evaluating NoSQL Databases. Recuperado el 22 de marzo de 2023, de https://www.mongodb.com/es/collateral/top-5-considerations-when-evaluating-nosql-databases.

Muñoz Razo, C. (2015). *Cómo elaborar y asesorar una investigación de tesis.* Pearson Educación. https://www-ebooks7-24-com-uh.knimbus.com:443/?il=4107

Oficina de Integridad de la Investigación (Sin fecha). *Conceptos básicos de investigación*. Recuperado el 17 de octubre de 2022, de: http://ori.hhs.gov/education/products/sdsu/espanol/variables.htm

Personal del Centro de Investigación Observatorio del Desarrollo de la Universidad de Costa Rica (2022). *Primera Encuesta de Comercio Electrónico en Costa Rica*. Recuperado el 17 de septiembre de 2022, de: https://www.consumo.go.cr/estudios/observatorio%20de%20comercio%20electronico/CP-Encuesta_comercio_electronico_CR_2022.pdf

Pimienta Prieto, J. H., Estrada Coronado, R. M., de la Orden Hoz, A. (2018). *Metodología de la investigación: competencias + aprendizaje + vida*. Pearson Educación. https://www-ebooks7-24-com-uh.knimbus.com:443/?il=7587

Pressman, Roger. (2011) *Software Engineering: A Practitioner's Approach.* McGraw-Hill. Recuperado el 29 de octubre de 2022, de https://www.mlsu.ac.in/econtents/16_EBOOK-7th_ed_software_engineering_a_practitioners_approach_by_roger_s._pressman_.pdf.

Richards, M. (2015) *Software Architecture Patterns. Understanding Common Architecture Patterns and When to Use them.* O'Reilly. Recuperado el 3 de enero de 2023, de https://isip.piconepress.com/courses/temple/ece_1111/resources/articles/20211201_software_architecture_patterns.pdf.

Silberschatz, A., Sudarshan, S., Korth, H. F. (2014). *Fundamentos de bases de datos*. McGraw-Hill. Recuperado el 14 de septiembre de 2022, de https://www-ebooks7-24-com-uh.knimbus.com:443/?il=6907.

Silveira, D. (2021). *What is the Unity Principle of Design?*. Adobe. Recuperado el 21 de marzo de 2023, de https://xd.adobe.com/ideas/process/ui-design/principles-design-unity/#:~:text=Unity%20is%20the%20principle%20of,form%20an%20aesthetically%20pleasing%20design.

Sommerville, Ian. (2011). *Software Engineering (9na edición).* Addison-Wesley. Recuperado el 22 de agosto de 2022, de https://drive.google.com/file/d/1P8iFS36b43qJBVRaNbi6DseKgSXW7zMU/view?usp=drivesdk.

Stack Overflow. (2022) Recuperado el 30 de octubre de 2022, de https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks.

Universidad Nacional Autónoma de México (2017). Recuperado el 29 de octubre de 2022, de https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/1023/mod_resource/content/1/contenido/index.html#contenido.

GLOSARIO

Blog: Es un sitio web principalmente de carácter personal o empresarial, cuya finalidad es publicar cualquier cosa que se desee compartir.

Body: Es el contenido en un documento HTML. Solo puede haber un contenedor de tipo body en un documento.

Credenciales: Suele referirse a la combinación de un usuario y contraseña, que cuando se dan juntos y de forma correcta, conceden el acceso a áreas restringidas de una aplicación.

Div: Es un contenedor usado en documentos HTML. Su contenido puede ser prácticamente de cualquier tipo, por lo que es de tipo genérico y no describe lo que contiene.

Foreach: En programación, es un bloque de código en bucle que permite recorrer estructuras compuestas de varios elementos.

If: es una sentencia condicional, que evalúa una expresión y dependiendo de si se cumple o no una condición definida, ejecuta un conjunto de instrucciones u otro.

Método: En programación, es un bloque de código que contiene una serie de instrucciones. Un programa hace que se ejecuten las instrucciones al llamar al método y especificando los argumentos de método necesarios.

Namespace: La palabra clave namespace se usa en C# para declarar un ámbito que contiene un conjunto de objetos relacionados, entre los cuales pueden encontrarse: clases, métodos, estructuras, interfaces, etc.

String: Secuencia de caracteres usado para representar el texto.

CAPÍTULO VII: APÉNDICES Y ANEXOS

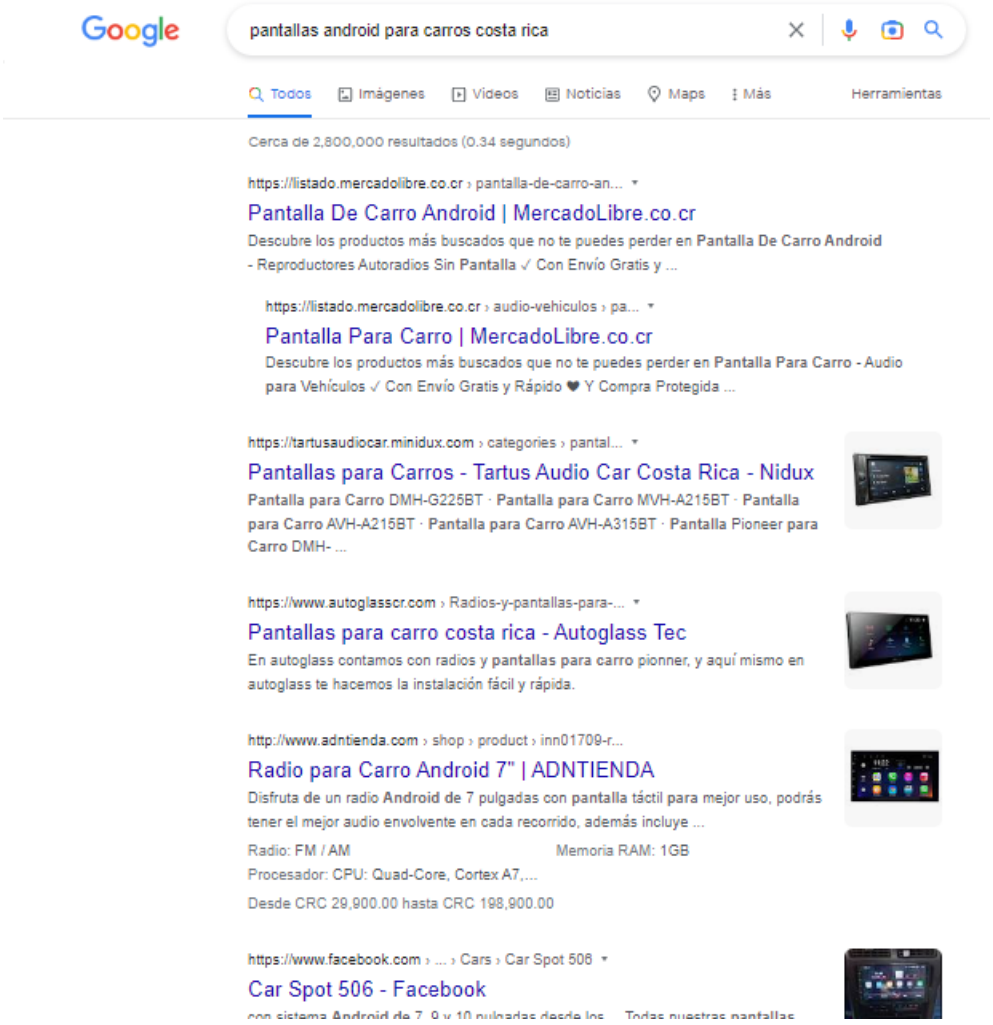
APÉNDICES

La presente sección muestra los apéndices al documento. Todos los apéndices son de propia autoría, caso contrario se catalogarían como anexos.

Para facilitar su inclusión y comprensión, se coloca cada apéndice en páginas separadas.

APÉNDICE 1. RESULTADOS DE BÚSQUEDAS DE PÁGINAS WEB SIMILARES.

Capturas de pantalla de una búsqueda en Google con las palabras: “Pantallas Android para carros Costa Rica”.



Google pantallas android para carros costa rica

https://www.facebook.com › ... › Cars › Car Spot 506

Car Spot 506 - Facebook

con sistema **Android** de 7, 9 y 10 pulgadas desde los ... Todas nuestras **pantallas Android** llevan el ... de semana para envíos por correos de **Costa Rica** y

★★★★½ Calificación: 4.5 · 15 votos

https://www.instagram.com › ...

Radios Android/Carplay Costa Rica (@dc90audiocar ...

SOLO SE ATIENDE CON PREVIA CITA Instalación y venta de Radios **Android**/Carplay 6 meses de garantía!! Kia's profile picture. Kia. Hyundai's profile picture.

https://www.sony.co.cr › car-receivers-players

Receptores y reproductores | Audio para auto - Sony Costa Rica

Radio para auto con **pantalla** táctil de 16,3 cm (6,4 pulg.) de DVD con BLUETOOTH®, **Android** Auto™ y CarPlay™. XAV-AX200. Agregar a favoritos.

https://www.tienda.repuestosfm.com › radios › radios-p...

-Radios para carro | Tienda Virtual de RepuestosFM

-Radios para carro. Mostrando 1–15 de 21 resultados. Ordenar por popularidad, Ordenar por ... **PANTALLA** HD SMARTGLASS **ANDROID** 10. $515,23 Añadir al carrito ...

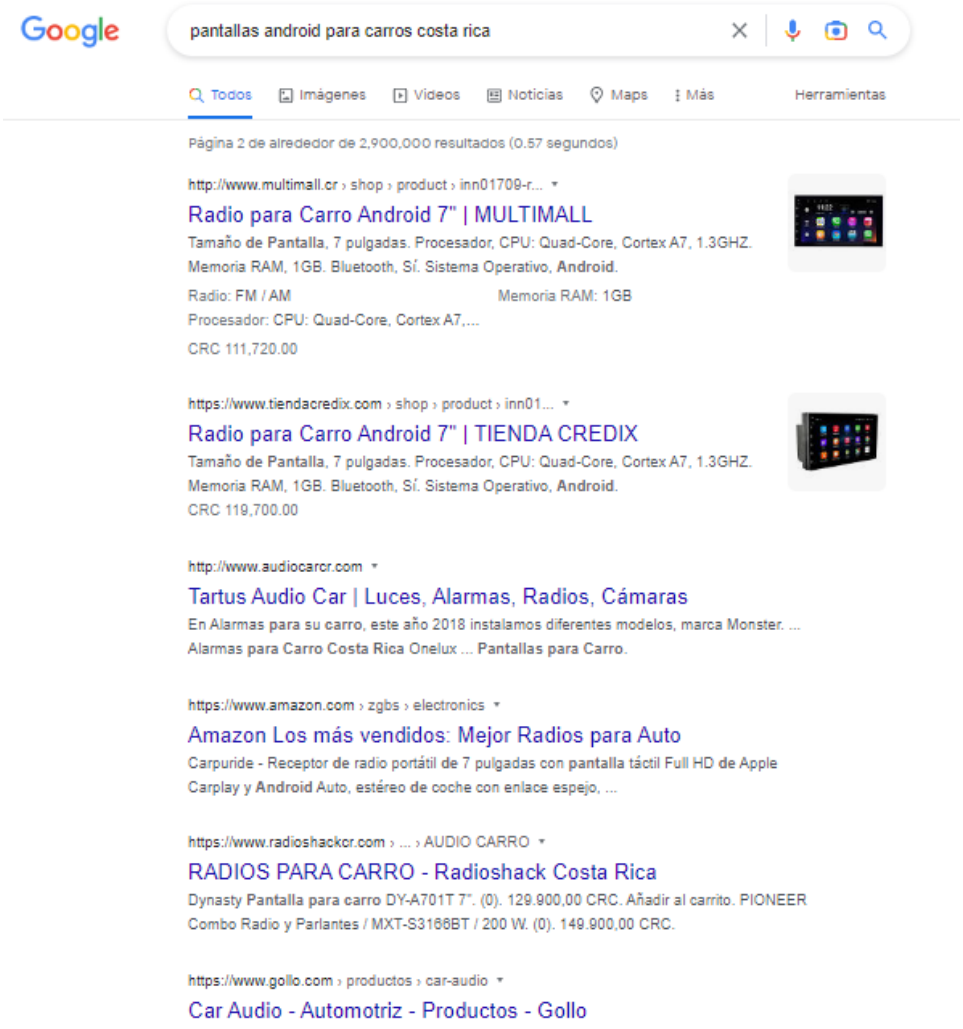
https://www.tiendamonge.com › pantalla-para-carro-dy...

Pantalla Para Carro Dynasty DY-710BAi | Monge Costa Rica

MirrorLink- Compatibilidad Espejo de iPhone y **Android**. Interface de control de volante. Soporte para tarjeta MicroSD de hasta 32GB. Ecualizador digital. Entrada ...

Entrada de audio: 3.5mm
Compatibilidad: Espejo Android y iPho...
Número de canales: 4

CRC 79,900.00 · Disponible



Se puede apreciar que la mayoría de los resultados señalan a:

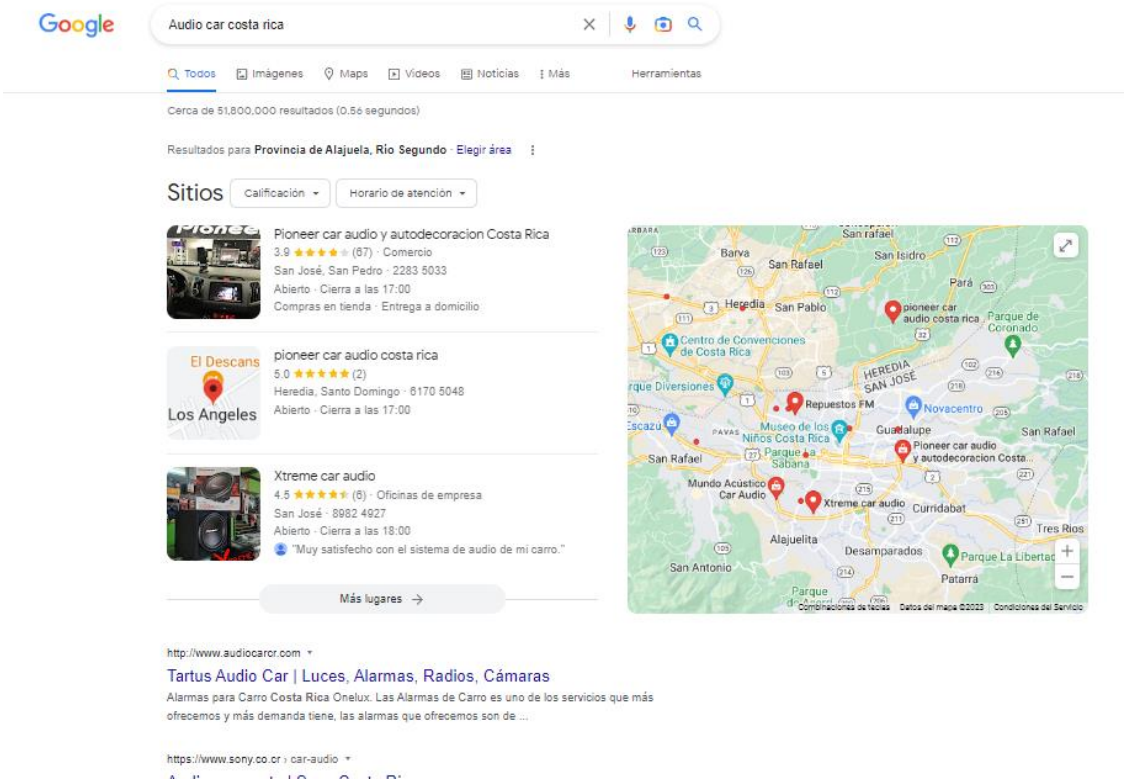
- Páginas de redes sociales (Ejemplo: Facebook).
- Páginas de comercio en línea (Ejemplo: mercadolibre.com).
- Páginas de comercios generales, no especializados.

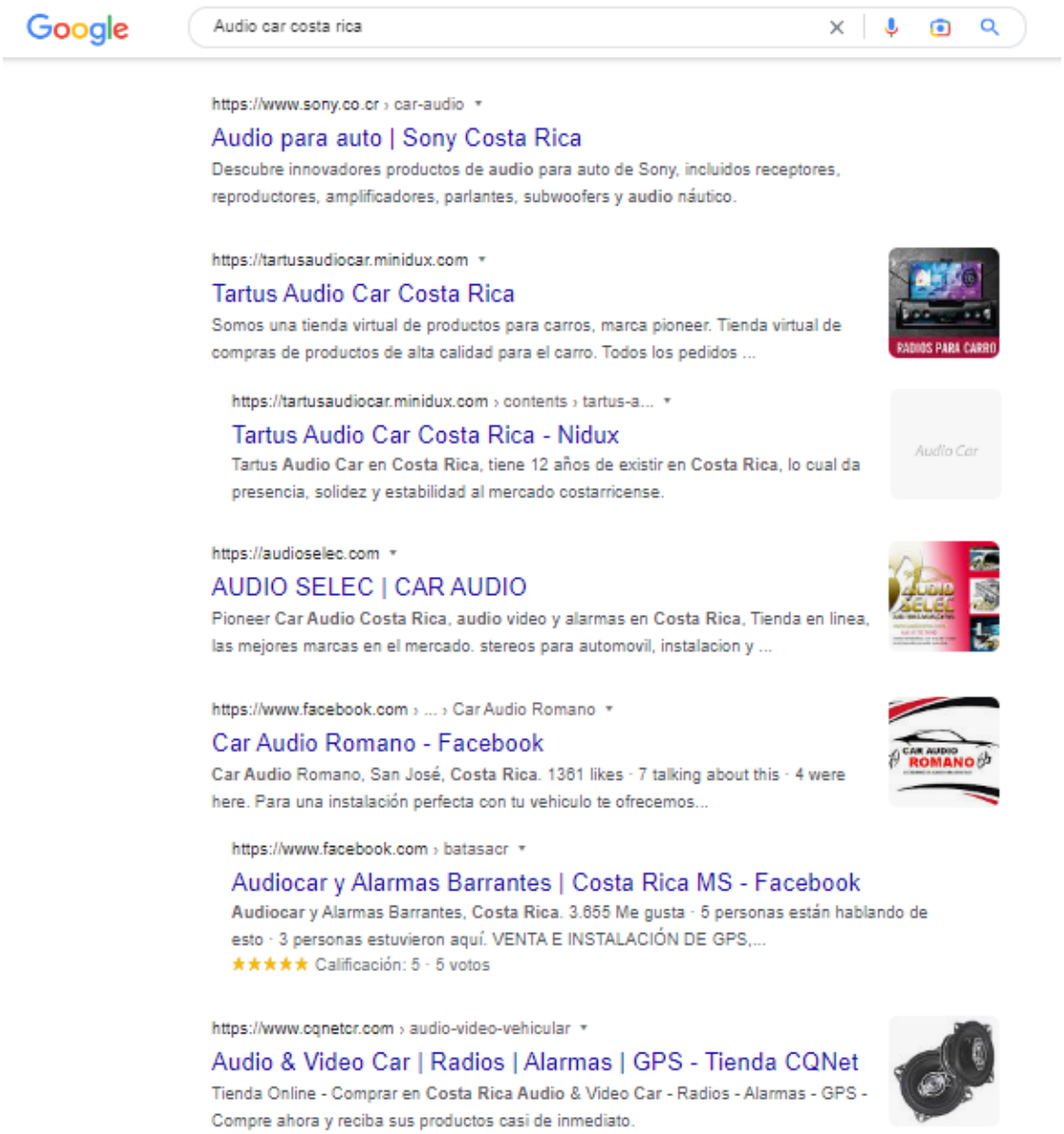
Son pocas las instancias de empresas especializadas y dedicadas al rubro de sistemas audiovisuales para vehículos automotores que aparecen en las primeras dos páginas de resultados, siendo la excepción:

- Tartus Audio Car.
- Autoglass CR.

- Audio Car CR.

Resultados de una búsqueda en Google para las palabras “Audio Car Costa Rica”:



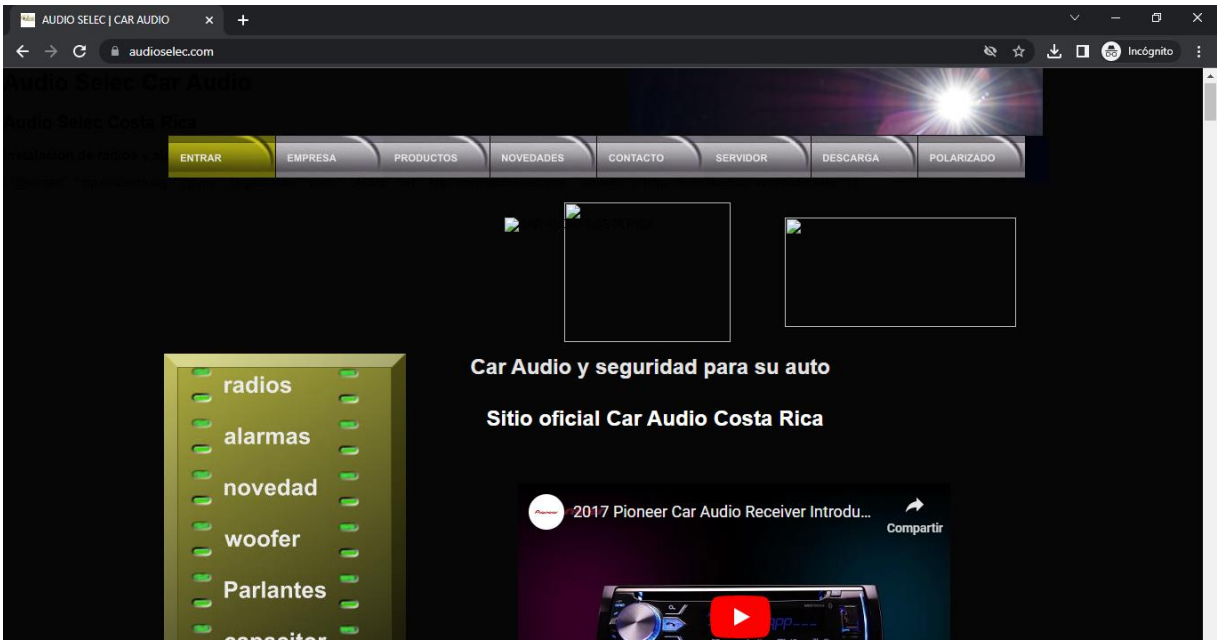


De igual manera, los resultados obtenidos de la búsqueda para las palabras “Audio Car Costa Rica” dan como resultado en su mayoría:

- Páginas de redes sociales (Ejemplo: Facebook).
- Páginas de comercio en línea (Ejemplo: mercadolibre.com).
- Páginas de comercios generales, no especializados.

También se puede observar un mapa con 18 ubicaciones resaltadas para comercios que ofrecen los mismos productos de ProLine CR en todo el Valle Central de Costa Rica. De estas 18, la mayoría no tienen página web, otras utilizan su perfil en redes sociales cómo su página web, y las que sí tienen página web (Solo dos) no son empresas especializadas en el rubro, y de hecho, una de ellas es de una marca de productos por lo que no se dedica a la comercialización directa hacia el consumidor final (Pioneer).

Captura de pantalla de una página de los primeros resultados en la búsqueda en Google para las palabras “Audio Car Costa Rica”:



Se puede apreciar que esta página web no funciona de forma adecuada porque no muestra algunas de sus imágenes. Al mismo tiempo, no sigue las pautas de diseño recomendadas por autores reconocidos en la materia y reseñadas en este trabajo en los puntos 2.3.3 y 2.4. lo cual desmejora la experiencia del usuario y la hace menos atractiva. Además, su información no está actualizada, como se aprecia en la fecha del video que tienen en su página de inicio.

APÉNDICE 2. ENTREVISTA A LOS COLABORADORES DE PROLINE CR.

A continuación, se enumeran las preguntas realizadas, seguidas de sus respectivas respuestas.

1. ¿Qué necesidad debe resolverse por medio de una aplicación web?

Respuesta: Tener un catálogo en línea y que los usuarios puedan realizar pedidos de manera remota.

2. ¿Qué tipo de acceso necesitan que tenga la aplicación web?

Respuesta: Necesitamos poder entrar desde una computadora y también desde un teléfono en cualquier lugar por medio de internet.

3. ¿Qué necesitan mostrar en el catálogo?

Respuesta: Toda la variedad de marcas y modelos de los productos que vendemos por medio de imágenes, características y precios y que solo pueda ser editado por los colaboradores de la empresa. Cada producto tendrá su vista con:

- ID del producto.
- Nombre del producto.
- Precio.
- Imágenes (al menos 5).
- Descripción.

4. ¿Qué interacción quieren tener con los clientes por medio de la aplicación web?

Respuesta: Que puedan comprar y también obtener los datos de contacto, pero no queremos que puedan escribirnos directamente a nuestro teléfono desde la plataforma, sino desde WhatsApp.

5. ¿Necesitan tener un trato diferenciado a los clientes?

Respuesta: No, no necesitamos que los clientes que compran al mayor vean la información de manera diferente a los clientes que compran al detal.

6. ¿Cómo quieren que sea el proceso de compra en la aplicación?

Respuesta: Que puedan colocar el producto en una canasta de compra y que puedan pagar por medio de transferencia y solicitar el envío o pagar en persona cuando pasen a recoger el producto. La orden de compra debe tener los siguientes datos de quien lo compra:

- Nombre y apellido.
- Cédula.
- Número de teléfono.
- Correo electrónico.
- Dirección.

Además, debe llevar los siguientes datos:

- Fecha.
- Producto.
- Cantidad.
- Precio.
- Impuesto.
- Total por producto.
- Monto total por pagar.
- Forma de pago.
- Número de transferencia (Si pagó por transferencia).

También queremos que los precios estén visibles para todas las personas que entren en la aplicación web, pero no las cantidades disponibles, sino que estén visibles solo para nosotros cuando entremos a administrar los productos.

APÉNDICE 3. MANUAL DE USUARIO.

PROLINE

Manual de usuario

Aplicación Web ProLine CR

Versión 1.0

Elaborado por: Luiscarlos Martínez.

PROLINE

ÍNDICE

CONTENIDO

INTRODUCCIÓN

Este es el manual de usuario para la aplicación web ProLine CR. La aplicación web es accesible a través de la dirección audiocarprolinecr.com, y consiste en un sitio de E-Commerce (Comercio electrónico) de manera que la empresa ProLine pueda llevar a cabo operaciones comerciales, comunicar información de sus productos y recibir otras solicitudes a través de esta.

La aplicación cuenta con 7 módulos, y en este manual se dan las instrucciones para el uso de cada una. Los módulos son:

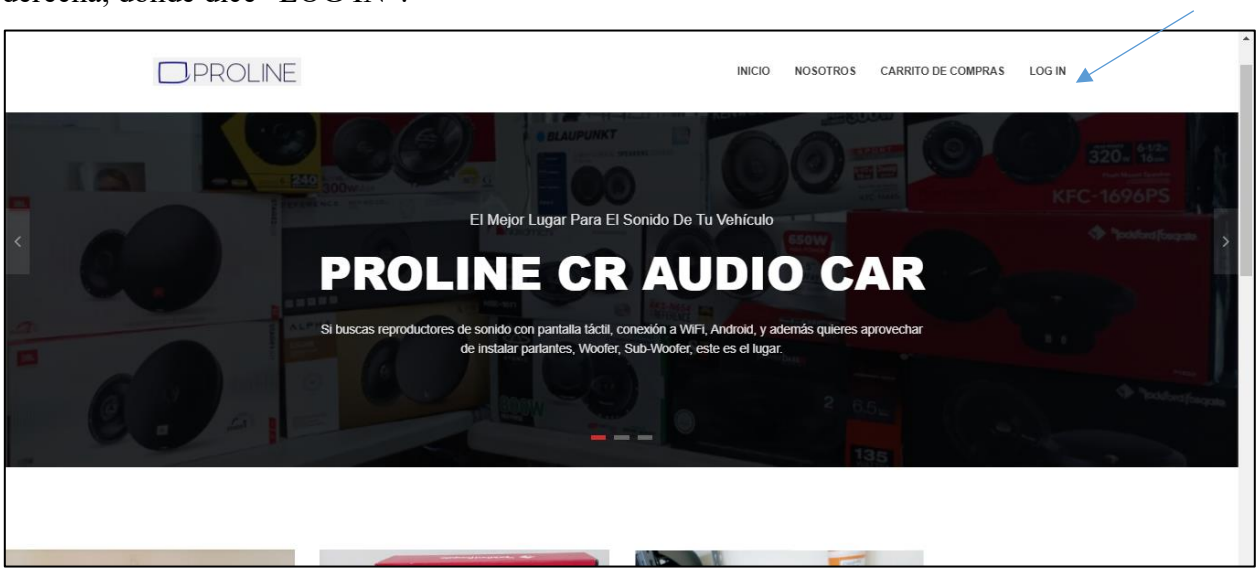
1. Inicio de sesión
2. Mantenimiento de catálogo.
3. Catálogo.
4. Carro de compras.
5. Información de contacto.
6. Pedidos.
7. Mantenimiento de usuarios.

PROLINE

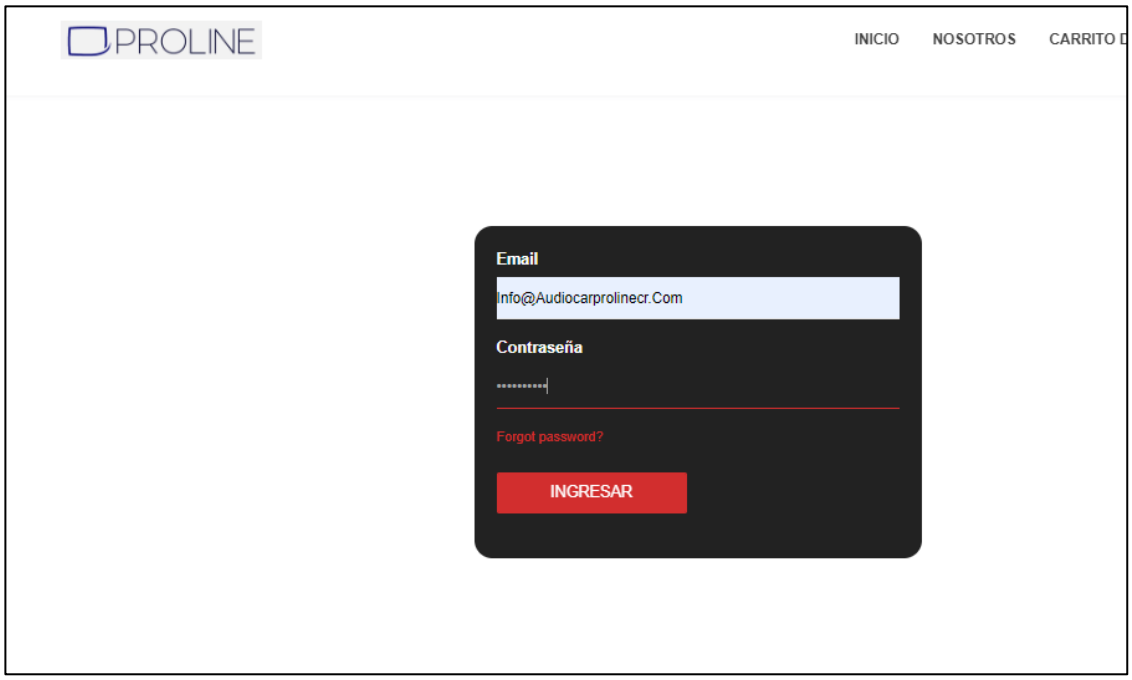
CONTENIDO

MÓDULO 1: INICIO DE SESIÓN

Para entrar a este módulo es necesario hacer clic en el enlace de la barra de arriba a la derecha, donde dice “LOG IN”.



Cuando carga la página de inicio de sesión, se deben escribir el usuario (correo) y la contraseña, y presionar el botón “INGRESAR”.

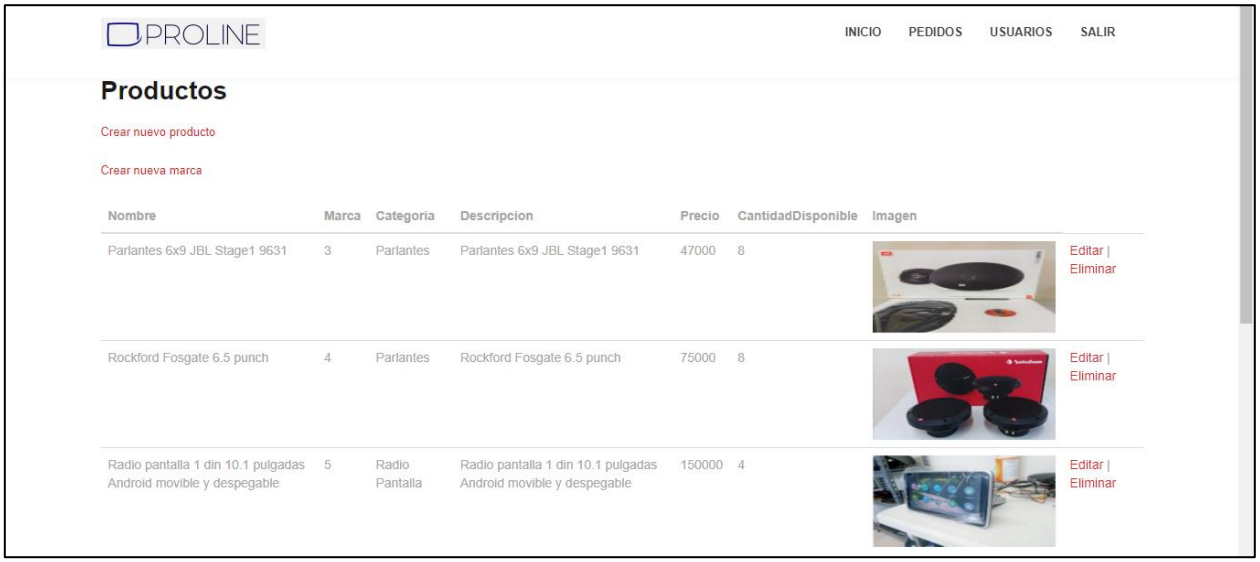


Si el usuario y la contraseña son correctos, cargará la página de inicio para administradores y las opciones en la barra de arriba a la derecha se cambian para llevar al usuario a las vistas para administradores. En cualquier momento que se desee cerrar la sesión, se puede presionar la opción “SALIR” de arriba a la derecha, tras lo cual se dirige al usuario a la vista de inicio y la barra de opciones de arriba a la derecha cambian de nuevo a las opciones para ir a las vistas de usuarios clientes.

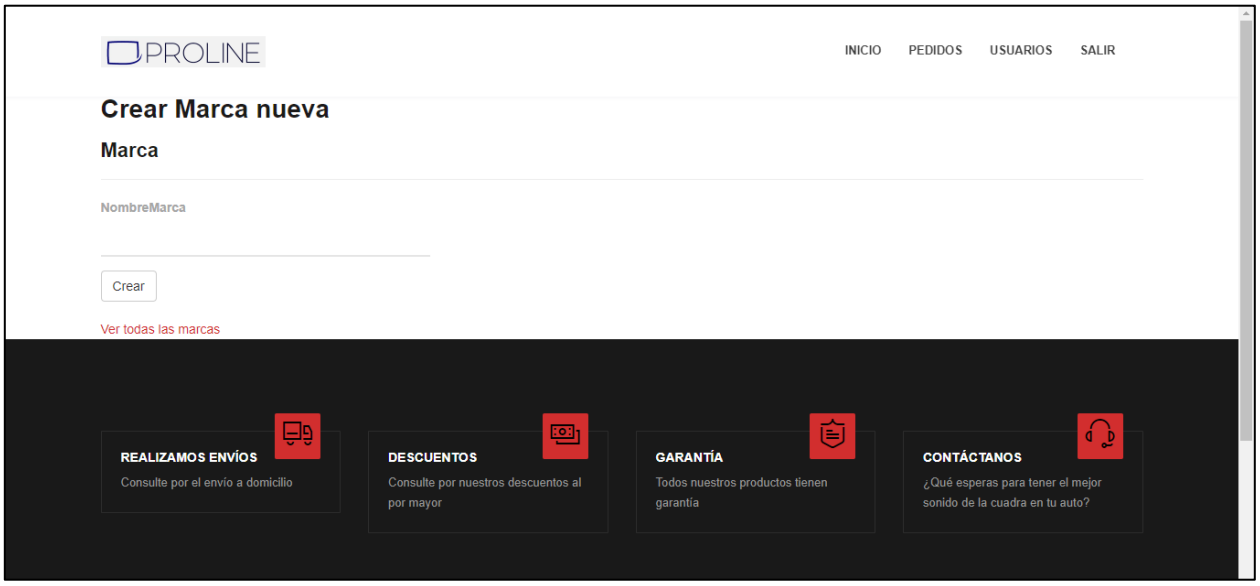
PROLINE

MÓDULO 2. MANTENIMIENTO DE CATÁLOGO (VISTA DE ADMINISTRADOR)

La página de inicio para administradores es el lugar donde se observa una lista de todos los productos publicados, con las opciones de crear un producto nuevo, editar los productos ya existentes o eliminar un producto. Cuando selecciona alguna de esas tres opciones, se carga su vista respectiva desde donde puede realizar la acción solicitada.



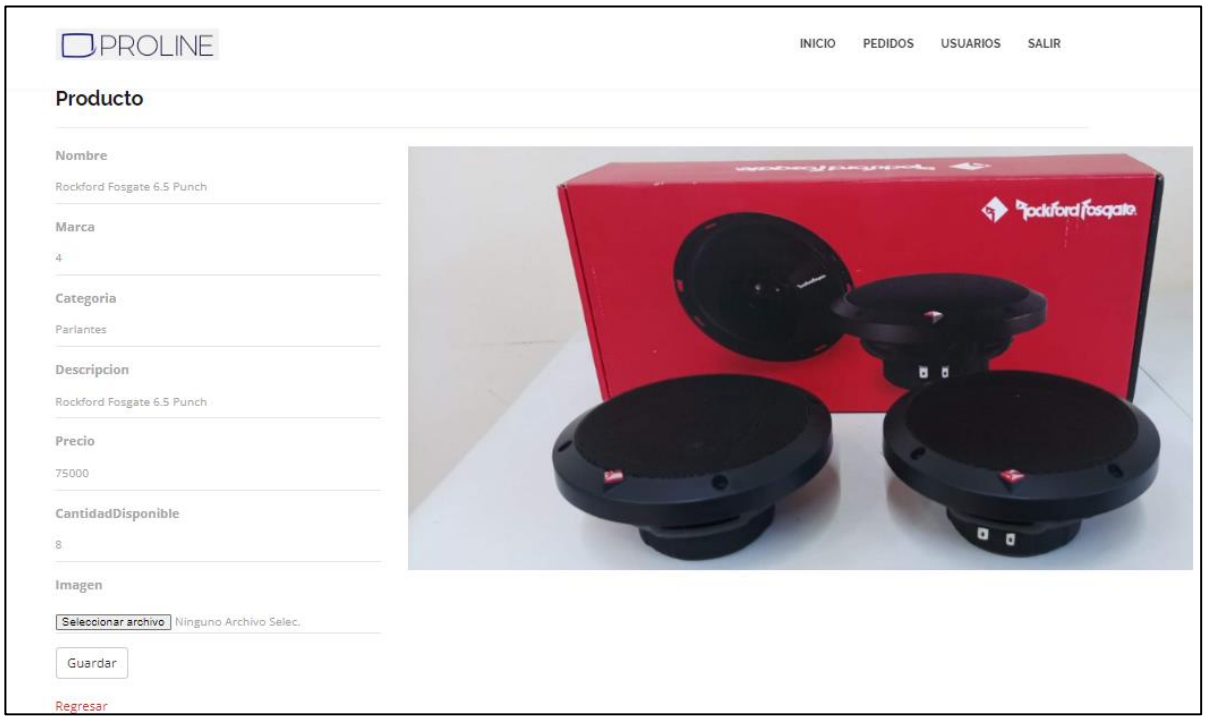
Es importante mencionar que la marca de un producto debe ser agregada antes de agregar el producto, para que pueda ser escogida entre las opciones al momento de crear un producto. Para ello, debe hacer clic en Crear Nueva Marca. Cargará entonces la vista con el formulario para crear una nueva marca donde únicamente se debe escribir el nombre de la nueva marca:



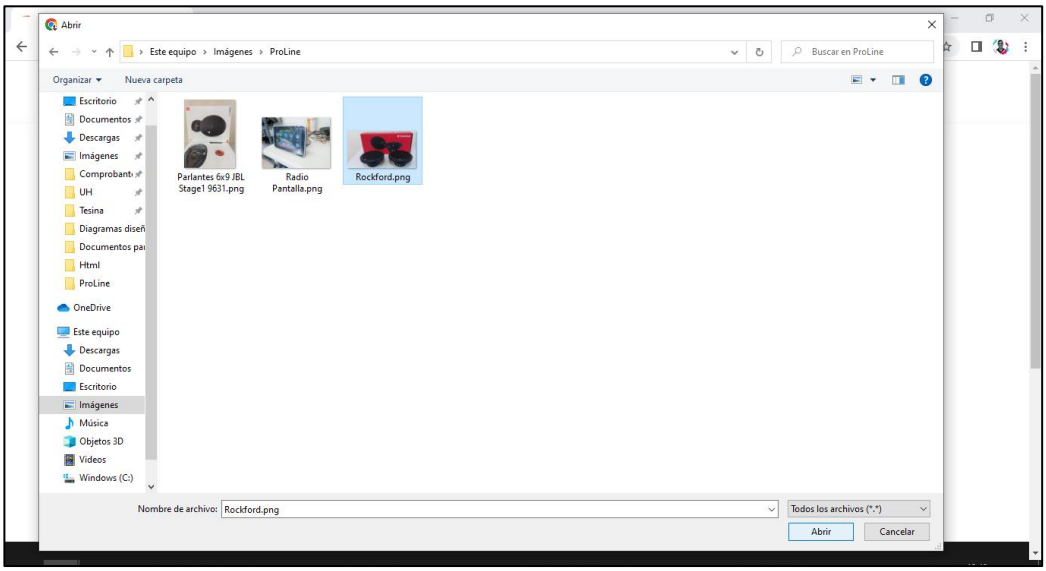
Si se desea ver una lista con todas las marcas, puede hacer clic en el botón “Ver todas las marcas”.

En cualquier momento que se desee volver a la vista principal de administradores se puede hacer clic en el botón de arriba a la derecha “INICIO”. Para cambiar la cantidad disponible para un producto debe hacer clic en el botón de Editar, al lado derecho de la imagen de un producto, para que cargue la vista de edición de dicho producto y allí pueda editarlo.

En esa vista también se podrán cambiar todas las otras características del producto:



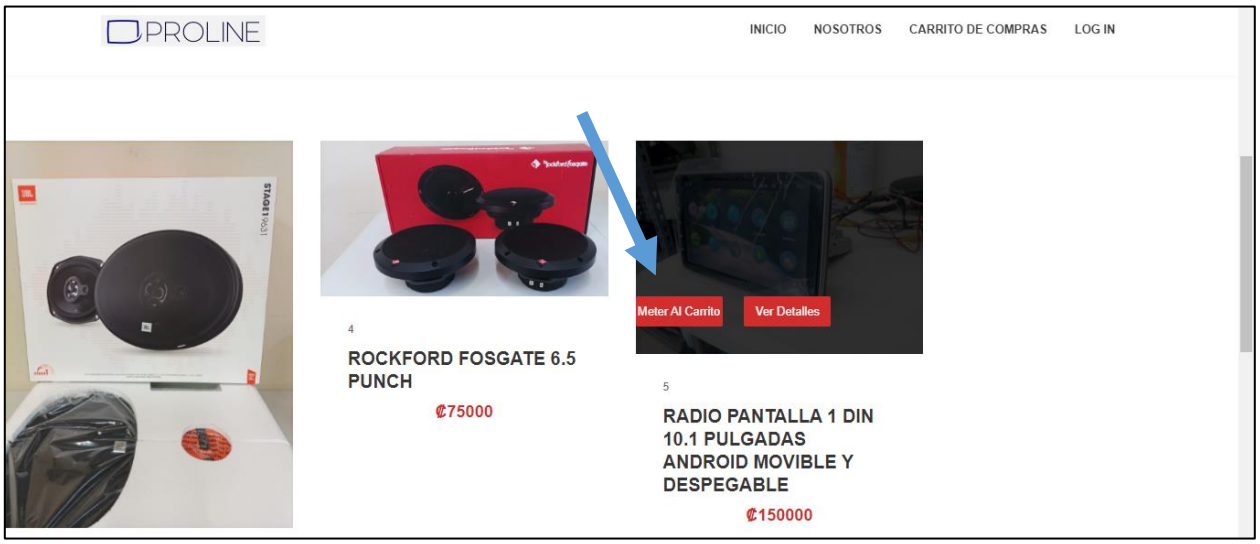
Para guardar la imagen del producto (O cambiarla en el caso de editar un producto existente) se debe presionar el botón Seleccionar archivo. En ese momento abrirá la ventana de exploración de archivos del dispositivo que se está usando, y allí se debe seleccionar la fotografía.



PROLINE

MÓDULO 3: CATÁLOGO.

Es la vista inicial de la aplicación web, y si el usuario se encuentra en otra vista podrá regresar a esta al hacer clic arriba a la derecha en “INICIO”, o bien haciendo clic en el logo de ProLine arriba a la izquierda. Acá los usuarios podrán ver todos los productos del catálogo con información relevante. Para hacer una compra, solo deben desplazar su mouse sobre la imagen de un producto, y hacer clic en el botón “Meter Al Carrito”.



Al hacerlo, se creará un pedido, y cada producto se agregará a ese pedido. El cliente puede agregar un producto más de una vez. Cuando desee, puede hacer clic en el botón de “Carrito de compras” para ver los artículos que ha ingresado al carrito.

PROLINE

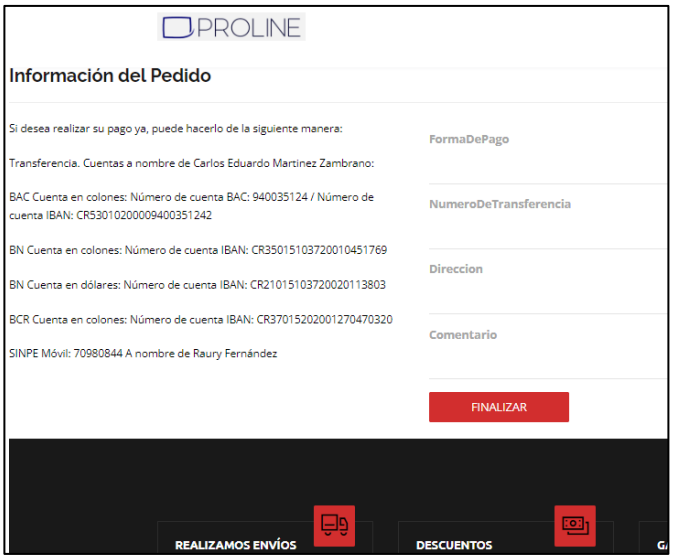
MÓDULO 4: CARRO DE COMPRAS.

Al hacer clic en Carrito de compras, podrá verificar los productos que se han ingresado al carrito.

Cuando haya verificado el pedido, hará clic en Continuar. Entonces aparecerá el formulario para completar la información del cliente. Debe llenar todos los campos, y en el campo de número de teléfono debe ingresar 8 dígitos, así como escribir un correo con el formato correcto. Si no hace algo de esto, entonces recibe un mensaje de alerta indicando lo que debe corregir antes de poder continuar.

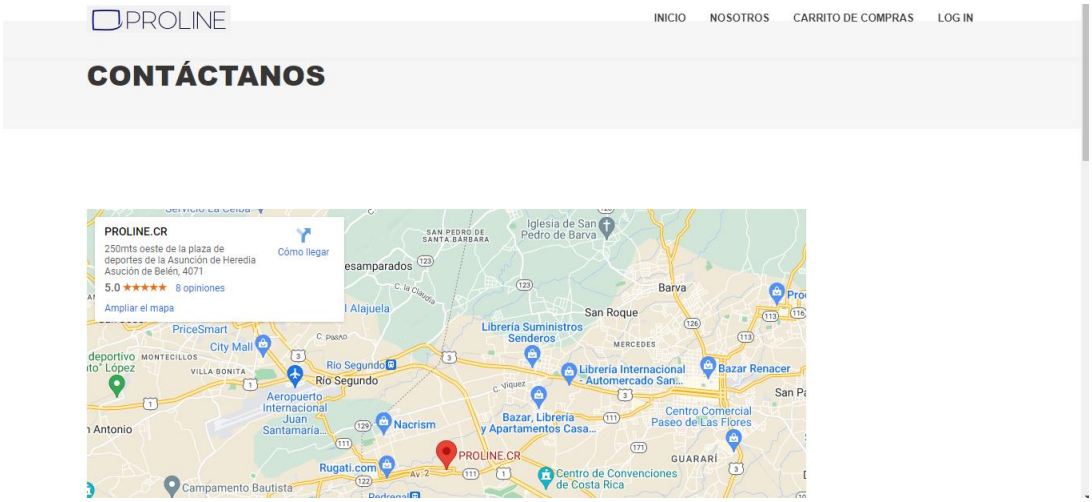
PROLINE

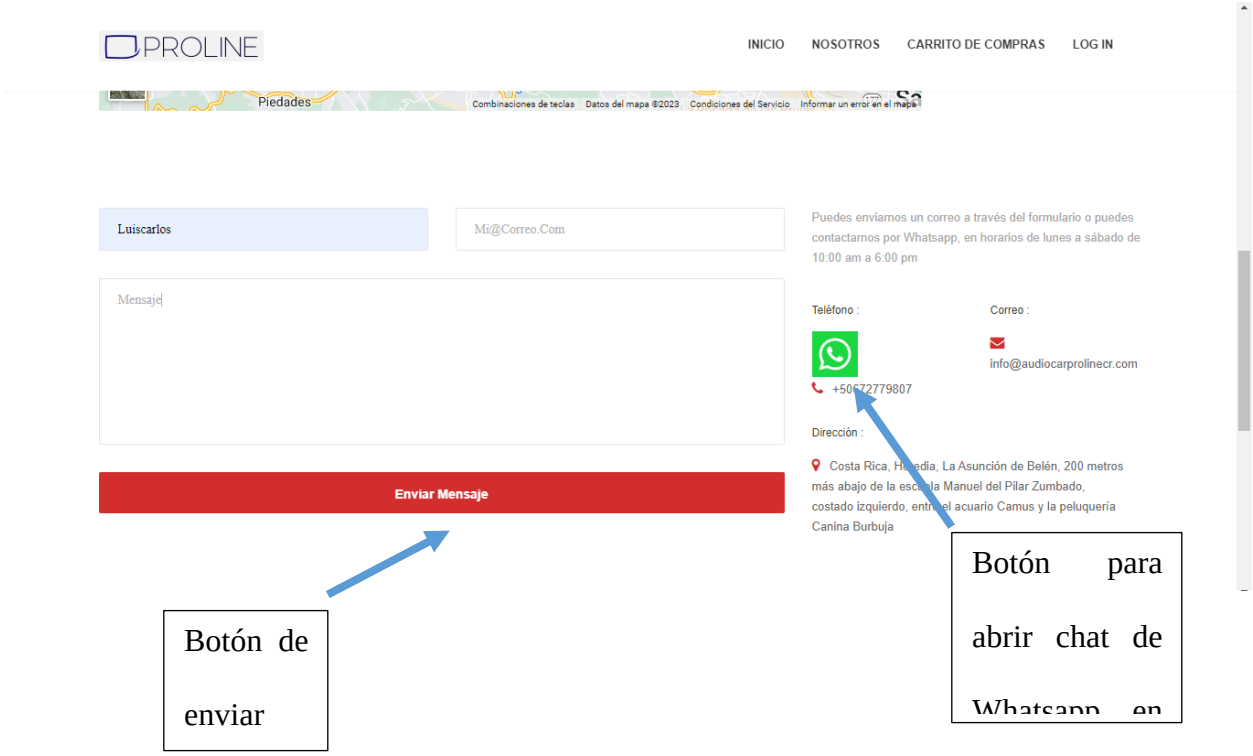
Al continuar, la siguiente ventana será para ingresar la información final del pedido. Se muestra la información para realizar transferencias. El cliente puede hacerla e informar esto llenando la información correspondiente en el formulario. Al hacer clic en Finalizar, habrá finalizado el pedido con éxito.



MÓDULO 5: INFORMACIÓN DE CONTACTO

Para llegar a esta vista se debe hacer clic arriba a la derecha en donde dice “NOSOTROS”. En esta vista se muestra un mapa de Google con la ubicación del local, la información de contacto de la empresa: Correo, teléfono y dirección del local; y también hay dos opciones para que el usuario se comunique directamente con un administrador: el botón para abrir un chat de WhatsApp, y el formulario para enviar un mensaje por correo electrónico a la cuenta info@audiocarprolinecr.com.






PROLINE

MÓDULO 6: PEDIDOS (VISTA DE ADMINISTRADOR)

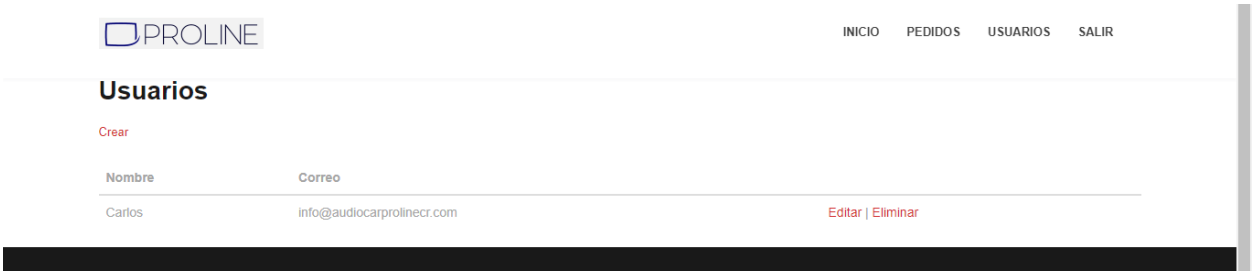
Esta vista es de acceso exclusivo para los usuarios administradores. Para llegar acá debe haberse iniciado sesión previamente, tras lo cual aparece la opción arriba a la derecha que dice “PEDIDOS”.

En esta vista se puede observar una lista con todos los pedidos que se han realizado, así como las opciones para editarlos o eliminarlos. Funciona de la misma manera que la lista de productos.

MÓDULO 7: MANTENIMIENTO DE USUARIOS. (VISTA DE ADMINISTRADOR)

Este módulo se compone de la vista para visualizar todos los usuarios administradores, y es sólo para los administradores. Para entrar a ella se hace clic en la opción de la barra superior, hacia la derecha, que dice “USUARIOS”.

Inicialmente muestra una tabla con todos usuarios administradores y las opciones para: agregar nuevos usuarios, editar los usuarios existentes y eliminar usuarios.



Al igual que en las vistas de productos, marcas y pedidos, se abrirá una ventana cuando se desee crear, editar o eliminar un usuario, donde se podrá realizar la acción deseada.

PROLINE

¿QUÉ HACER EN CASO DE PROBLEMAS?

Como todo software, la aplicación web ProLine no está libre de problemas. De encontrarse con alguna dificultad, se deben seguir los siguientes pasos:

- Intentar realizar la acción desde otro navegador. Por ejemplo, si normalmente se usa Google Chrome, intentarlo en Mozilla Firefox.
- Intentar realizar la acción desde otro dispositivo. Por ejemplo, si se está usando un dispositivo móvil, intentarlo desde una computadora.
- De persistir el problema, se debe contactar al desarrollador por los canales indicados.
- Se debe llenar el documento "Seguimiento para la aplicación web ProLine". Para ir a dicho documento, puede hacer clic acá: https://docs.google.com/document/d/1zarVOOIi2ZyLC6w6vGoxtztxaCgwtuX_/edit?usp=sharing&ouid=113420537718241334757&rtpof=true&sd=true

Más allá de usar dicho documento para registrar las incidencias que se presenten, se le invita a escribir sugerencias para mejoras en el apartado de observaciones, y escribir en el apartado de peticiones las solicitudes formales para realizar cambios en la aplicación.

APÉNDICE 4. SEGUIMIENTO PARA LA APLICACIÓN WEB PROLINE.

SEGUIMIENTO PARA LA APLICACIÓN WEB PROLINE 1.0

Este es un documento compartido, al cual tienen acceso los administradores de ProLine CR, y el desarrollador de la aplicación web ProLine 1.0.

Se llevará registro de las incidencias encontradas al hacer uso de la aplicación. También se registrarán las solicitudes para cambios y mejoras, así como cualquier otra observación relevante.

INCIDENCIAS

Fecha	Usuario que detectó la incidencia	Módulo afectado	Descripción

OBSERVACIONES

Usuario	Módulo	Observación

PETICIONES PARA MEJORAS

Usuario	Petición	Nivel de urgencia

APÉNDICE 5. CARTA DE ACEPTACIÓN DE PARTE DE LA EMPRESA.

PROLINE

Carta de aceptación de proyecto

ProLine CR

Belén, Heredia, 10 de febrero de 2023.

Estimado Luiscarlos Martínez, a través de la presente, ProLine CR tiene el agrado de comunicarle la aceptación del proyecto: "Propuesta de implementación de aplicación web para comercio electrónico de la empresa ProLine".

Además, le expresamos la intención de la empresa en llevar a cabo esta implementación lo antes posible, y también el deseo de continuar colaborando con usted para la introducción de nuevos módulos dentro de la aplicación web, no incluidas en los alcances de este primer proyecto, con el fin de aumentar y mejorar sus funcionalidades.

Atentamente,

Carlos Eduardo Martínez Zambrano

Gerente General de ProLine CR.

Tel. 7277 9807

ANEXOS

A continuación, se incluyen los anexos. Éstos consisten en documentación adicional al proyecto, que sirve para ampliar temas relacionados, y que son de autoría de terceros.

ANEXO 1. MANIFIESTO POR EL DESARROLLO ÁGIL DE SOFTWARE.

A continuación se muestran, en formato de imagen, las dos partes del Manifiesto por el Desarrollo Ágil de Software, tal como se puede observar en su sitio oficial en español, https://agilemanifesto.org/iso/es/principles.html.


Principios del Manifiesto Ágil
Seguimos estos principios:
Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
El software funcionando es la medida principal de progreso.
Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

ANEXO 2. LA GUÍA DE SCRUM.

A continuación se adjunta todo el texto de la Guía de Scrum, que se encuentra disponible de manera gratuita y en español, en la página web oficial de Scrum: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-Latin-South-American.pdf.

Ken Schwaber & Jeff Sutherland

La Guía de Scrum

La Guía Definitiva de Scrum: Las Reglas del Juego

Noviembre de 2020

PROPÓSITO DE LA GUÍA SCRUM

Desarrollamos Scrum a principios de la década de 1990. Escribimos la primera versión de la Guía Scrum en 2010 para ayudar a las personas de todo el mundo a comprender Scrum. Hemos

desarrollado la Guía desde entonces a través de pequeñas actualizaciones funcionales. Juntos, la respaldamos.

La Guía de Scrum contiene la definición de Scrum. Cada elemento del marco de trabajo tiene un propósito específico que es esencial para el valor general y los resultados obtenidos con Scrum. Cambiar el diseño o las ideas esenciales de Scrum, omitir elementos o no seguir las reglas de Scrum, oculta los problemas y limita los beneficios de Scrum, e incluso potencialmente lo vuelve inútil.

Seguimos el uso creciente de Scrum dentro de un mundo complejo en constante crecimiento. Nos sentimos honrados de ver que Scrum está siendo adoptado en muchos dominios que tienen un trabajo esencialmente complejo, más allá del desarrollo de productos de software donde Scrum tiene sus raíces. A medida que se extiende el uso de Scrum, los desarrolladores, investigadores, analistas, científicos y otros especialistas hacen el trabajo. Usamos la palabra "desarrolladores" en Scrum no para excluir, sino para simplificar. Si obtiene valor de Scrum, considérese incluido.

A medida que se utiliza Scrum, se pueden encontrar, aplicar y diseñar patrones, procesos y enfoques que se ajusten al marco de trabajo Scrum como se describe en este documento. Su descripción va más allá del propósito de la Guía Scrum porque son sensibles al contexto y difieren ampliamente entre los usos de Scrum. Tales tácticas para usar dentro del marco de trabajo Scrum varían ampliamente y se describen en otra parte.

Ken Schwaber y Jeff Sutherland, Noviembre de 2020

DEFINICIÓN DE SCRUM

Scrum es un marco de trabajo liviano que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptativas para problemas complejos.

En pocas palabras, Scrum requiere un *Scrum Master* para fomentar un entorno donde:

1. Un *Product Owner* ordena el trabajo de un problema complejo en un *Product Backlog*.
2. El *Scrum Team* convierte una selección del trabajo en un *Increment* de valor durante un *Sprint*.
3. El *Scrum Team* y sus interesados inspeccionan los resultados y se adaptan para el próximo *Sprint*.
4. *Repita*

Scrum es simple. Pruébelo como está y determine si su filosofía, teoría y estructura ayudan a lograr objetivos y crear valor. El marco de trabajo Scrum es incompleto de manera intencional, solo define las partes necesarias para implementar la teoría de Scrum. Scrum se basa en la inteligencia colectiva de las personas que lo utilizan. En lugar de proporcionar a las personas instrucciones detalladas, las reglas de Scrum guían sus relaciones e interacciones.

En este marco de trabajo pueden emplearse varios procesos, técnicas y métodos. Scrum envuelve las prácticas existentes o las hace innecesarias. Scrum hace visible la eficacia relativa de las técnicas actuales de gestión, entorno y trabajo, de modo que se puedan realizar mejoras.

TEORÍA DE SCRUM

Scrum se basa en el empirismo y el pensamiento *Lean*. El empirismo afirma que el conocimiento proviene de la experiencia y de la toma de decisiones con base en lo observado. El pensamiento *Lean* reduce el desperdicio y se enfoca en lo esencial.

Scrum emplea un enfoque iterativo e *Incremental* para optimizar la previsibilidad y controlar el riesgo. Scrum involucra a grupos de personas que colectivamente tienen todas las habilidades y experiencia para hacer el trabajo y compartir o adquirir dichas habilidades según sea necesario.

Scrum combina cuatro eventos formales para inspección y adaptación dentro de un evento contenedor, el *Sprint*. Estos eventos funcionan porque implementan los pilares empíricos de Scrum de transparencia, inspección y adaptación.

TRANSPARENCIA

El proceso y el trabajo emergentes deben ser visibles tanto para quienes realizan el trabajo como para quienes lo reciben. Con Scrum, las decisiones importantes se basan en el estado percibido de sus tres artefactos formales. Los artefactos que tienen poca transparencia pueden llevar a decisiones que disminuyan el valor y aumenten el riesgo.

La transparencia permite la inspección. La inspección sin transparencia es engañosa y derrochadora.

INSPECCIÓN

Los artefactos de Scrum y el progreso hacia los objetivos acordados deben inspeccionarse con frecuencia y con diligencia para detectar variaciones o problemas potencialmente indeseables. Para ayudar con la inspección, Scrum proporciona cadencia en forma de sus cinco eventos.

La inspección permite la adaptación. La inspección sin adaptación se considera inútil. Los eventos Scrum están diseñados para provocar cambios.

ADAPTACIÓN

Si algún aspecto de un proceso se desvía fuera de los límites aceptables o si el producto resultante es inaceptable, el proceso que se aplica o los materiales que se producen deben ajustarse. El ajuste debe realizarse lo antes posible para minimizar una mayor desviación.

La adaptación se vuelve más difícil cuando las personas involucradas no están empoderadas ni se autogestionan. Se espera que un *Scrum Team* se adapte en el momento en que aprenda algo nuevo a través de la inspección.

VALORES DE SCRUM

El uso exitoso de Scrum depende de que las personas se vuelvan más competentes en vivir cinco valores:

Compromiso, Foco, Franqueza, Respeto y Coraje

El *Scrum Team* se compromete a lograr sus objetivos y a apoyarse mutuamente. Su foco principal está en el trabajo del *Sprint* para lograr el mejor progreso posible hacia estos objetivos. El *Scrum Team* y sus interesados son francos sobre el trabajo y los desafíos. Los miembros del *Scrum Team* se respetan entre sí para ser personas capaces e independientes, y son respetados como tales por las personas con las que trabajan. Los miembros del *Scrum Team* tienen el coraje de hacer lo correcto, para trabajar en problemas difíciles.

Estos valores dan dirección al *Scrum Team* con respecto a su trabajo, acciones y comportamiento. Las decisiones que se tomen, los pasos que se den y la forma en que se use

Scrum deben reforzar estos valores, no disminuirlos ni socavarlos. Los miembros del *Scrum Team* aprenden y exploran los valores mientras trabajan con los eventos y artefactos Scrum. Cuando el *Scrum Team* y las personas con las que trabajan incorporan estos valores, los pilares empíricos de Scrum de transparencia, inspección y adaptación cobran vida y generan confianza.

SCRUM TEAM

La unidad fundamental de Scrum es un pequeño equipo de personas, un *Scrum Team*. El *Scrum Team* consta de un *Scrum Master*, un *Product Owner* y *Developers*. Dentro de un *Scrum Team*, no hay subequipos ni jerarquías. Es una unidad cohesionada de profesionales enfocados en un objetivo a la vez, el Objetivo del Producto.

Los *Scrum Teams* son multifuncionales, lo que significa que los miembros tienen todas las habilidades necesarias para crear valor en cada *Sprint*. También se autogestionan, lo que significa que deciden internamente quién hace qué, cuándo y cómo.

El *Scrum Team* es lo suficientemente pequeño como para seguir siendo ágil y lo suficientemente grande como para completar un trabajo significativo dentro de un *Sprint*, generalmente 10 personas o menos. En general, hemos descubierto que los equipos más pequeños se comunican mejor y son más productivos. Si los *Scrum Teams* se vuelven demasiado grandes, deberían considerar reorganizarse en múltiples *Scrum Teams* cohesivos, cada uno enfocado en el mismo producto. Por lo tanto, deben compartir el mismo Objetivo del Producto, el *Product Backlog* y el *Product Owner*.

El *Scrum Team* es responsable de todas las actividades relacionadas con el producto, desde la colaboración de los interesados, la verificación, el mantenimiento, la operación, la

experimentación, la investigación y el desarrollo, y cualquier otra cosa que pueda ser necesaria. Están estructurados y empoderados por la organización para gestionar su propio trabajo. Trabajar en *Sprints* a un ritmo sostenible mejora el enfoque y la consistencia del *Scrum Team*.

Todo el *Scrum Team* es responsable de crear un *Increment* valioso y útil en cada *Sprint*. Scrum define tres responsabilidades específicas dentro del *Scrum Team*: los *Developers*, el *Product Owner* y el *Scrum Master*.

DEVELOPERS

Las personas del *Scrum Team* que se comprometen a crear cualquier aspecto de un *Increment* utilizable en cada *Sprint* son *Developers*.

Las habilidades específicas que necesitan los *Developers* suelen ser amplias y variarán según el ámbito de trabajo. Sin embargo, los *Developers* siempre son responsables de:

- Crear un plan para el *Sprint*, el *Sprint Backlog*;
- Inculcar calidad al adherirse a una Definición de Terminado;
- Adaptar su plan cada día hacia el Objetivo del *Sprint*; y,
- Responsabilizarse mutuamente como profesionales.

PRODUCT OWNER

El *Product Owner* es responsable de maximizar el valor del producto resultante del trabajo del *Scrum Team*. La forma en que esto se hace puede variar ampliamente entre organizaciones, *Scrum Teams* e individuos.

El *Product Owner* también es responsable de la gestión efectiva del *Product Backlog*, lo que incluye:

- Desarrollar y comunicar explícitamente el Objetivo del Producto;
- Crear y comunicar claramente los elementos del *Product Backlog*;
- Ordenar los elementos del *Product Backlog*; y,
- Asegurarse de que el *Product Backlog* sea transparente, visible y se entienda.

El *Product Owner* puede realizar el trabajo anterior o puede delegar la responsabilidad en otros. Independientemente de ello, el *Product Owner* sigue siendo el responsable de que el trabajo se realice.

Para que los *Product Owner*s tengan éxito, toda la organización debe respetar sus decisiones. Estas decisiones son visibles en el contenido y el orden del *Product Backlog*, y a través del *Increment* inspeccionable en la *Sprint Review*.

El *Product Owner* es una persona, no un comité. El *Product Owner* puede representar las necesidades de muchos interesados en el *Product Backlog*. Aquellos que quieran cambiar el *Product Backlog* pueden hacerlo intentando convencer al *Product Owner*.

SCRUM MASTER

El *Scrum Master* es responsable de establecer Scrum como se define en la Guía de Scrum. Lo hace ayudando a todos a comprender la teoría y la práctica de Scrum, tanto dentro del *Scrum Team* como de la organización.

El *Scrum Master* es responsable de lograr la efectividad del *Scrum Team*. Lo hace apoyando al *Scrum*

Team en la mejora de sus prácticas, dentro del marco de trabajo de Scrum.

Los *Scrum Masters* son verdaderos líderes que sirven al *Scrum Team* y a la organización en general.

El *Scrum Master* sirve al *Scrum Team* de varias maneras, que incluyen:

- Guiar a los miembros del equipo en ser autogestionados y multifuncionales;
- Ayudar al *Scrum Team* a enfocarse en crear *Increments* de alto valor que cumplan con la
  Definición de Terminado;
- Procurar la eliminación de impedimentos para el progreso del *Scrum Team*; y,
- Asegurarse de que todos los eventos de Scrum se lleven a cabo y sean positivos, productivos y se mantengan dentro de los límites de tiempo recomendados en esta Guía.

El *Scrum Master* sirve al *Product Owner* de varias maneras, que incluyen:

- Ayudar a encontrar técnicas para una definición efectiva de Objetivos del Producto y la gestión del *Product Backlog*;
- Ayudar al *Scrum Team* a comprender la necesidad de tener elementos del *Product Backlog* claros y concisos;
- Ayudar a establecer una planificación empírica de productos para un entorno complejo; y,
- Facilitar la colaboración de los interesados según se solicite o necesite.

El *Scrum Master* sirve a la organización de varias maneras, que incluyen:

- Liderar, capacitar y guiar a la organización en su adopción de Scrum;
- Planificar y asesorar implementaciones de Scrum dentro de la organización;
- Ayudar a los empleados y los interesados a comprender y aplicar un enfoque empírico para el trabajo complejo; y,
- Eliminar las barreras entre los interesados y los *Scrum Teams*.

EVENTOS DE SCRUM

El *Sprint* es un contenedor para todos los demás eventos. Cada evento en Scrum es una oportunidad formal para inspeccionar y adaptar los artefactos Scrum. Estos eventos están

diseñados específicamente para habilitar la transparencia requerida. No operar cualquier evento según lo prescrito resulta en la pérdida de oportunidades para inspeccionar y adaptarse. Los eventos se utilizan en Scrum para crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum.

Lo óptimo es que todos los eventos se celebren al mismo tiempo y en el mismo lugar para reducir la complejidad.

EL SPRINT

Los *Sprints* son el corazón de Scrum, donde las ideas se convierten en valor.

Son eventos de duración fija de un mes o menos para crear consistencia. Un nuevo *Sprint* comienza inmediatamente después de la conclusión del *Sprint* anterior.

Todo el trabajo necesario para lograr el Objetivo del Producto, incluido la *Sprint Planning*, *Daily Scrum*s, *Sprint Review* y *Sprint Retrospective*, ocurre dentro de los *Sprints*.

Durante el *Sprint*:

- No se realizan cambios que pongan en peligro el Objetivo del *Sprint*; ● La calidad no disminuye;
- El *Product Backlog* se refina según sea necesario; y,
- El alcance se puede aclarar y renegociar con el *Product Owner* a medida que se aprende más.

Los *Sprints* permiten la previsibilidad al garantizar la inspección y adaptación del progreso hacia un Objetivo del Producto al menos cada mes calendario. Cuando el horizonte de un *Sprint* es demasiado largo, el Objetivo del *Sprint* puede volverse inválido, la complejidad puede crecer y el riesgo puede aumentar. Se pueden emplear *Sprints* más cortos para generar más ciclos de

aprendizaje y limitar el riesgo de costo y esfuerzo a un período de tiempo menor. Cada *Sprint* puede considerarse un proyecto corto.

Existen varias prácticas para pronosticar el progreso, como el trabajo pendiente (*burn-downs*), trabajo completado (*burn-ups*) o flujos acumulativos (*cumulative flows*). Si bien han demostrado su utilidad, no reemplazan la importancia del empirismo. En entornos complejos, se desconoce lo que sucederá. Solo lo que ya ha sucedido se puede utilizar para la toma de decisiones con miras al futuro.

Un *Sprint* podría cancelarse si el Objetivo del *Sprint* se vuelve obsoleto. Solo el *Product Owner* tiene la autoridad para cancelar el *Sprint*.

SPRINT PLANNING

La *Sprint Planning* inicia el *Sprint* al establecer el trabajo que se realizará para el *Sprint*. El *Scrum Team* crea este plan resultante mediante trabajo colaborativo.

El *Product Owner* se asegura de que los asistentes estén preparados para discutir los elementos más importantes del *Product Backlog* y cómo se relacionan con el Objetivo del Producto. El *Scrum Team* también puede invitar a otras personas a asistir a la *Sprint Planning* para brindar asesoramiento.

La *Sprint Planning* aborda los siguientes temas:

Tema uno: ¿Por qué es valioso este *Sprint*?
El *Product Owner* propone cómo el producto podría *Increment*ar su valor y utilidad en el *Sprint* actual. Luego, todo el *Scrum Team* colabora para definir un Objetivo del *Sprint* que comunica

por qué el *Sprint* es valioso para los interesados. El Objetivo del *Sprint* debe completarse antes de que termine la *Sprint Planning*.

Tema dos: ¿Qué se puede hacer en este *Sprint*?
A través de una conversación con el *Product Owner*, los *Developers* seleccionan elementos del *Product Backlog* para incluirlos en el *Sprint* actual. El *Scrum Team* puede refinar estos elementos durante este proceso, lo que aumenta la comprensión y la confianza.

Seleccionar cuánto se puede completar dentro de un *Sprint* puede ser un desafío. Sin embargo, cuanto más sepan los *Developers* sobre su desempeño pasado, su capacidad actual y su Definición de Terminado, más confiados estarán en sus pronósticos para el *Sprint*.

Tema tres: ¿Cómo se realizará el trabajo elegido?
Para cada elemento del *Product Backlog* seleccionado, los *Developers* planifican el trabajo necesario para crear un *Increment* que cumpla con la Definición de Terminado. A menudo, esto se hace descomponiendo los elementos del *Product Backlog* en elementos de trabajo más pequeños de un día o menos. La forma de hacerlo queda a criterio exclusivo de los *Developers*. Nadie más les dice cómo convertir los elementos del *Product Backlog* en *Increments* de valor.

El Objetivo del *Sprint*, los elementos del *Product Backlog* seleccionados para el *Sprint*, más el plan para entregarlos se denominan juntos *Sprint Backlog*.

La *Sprint Planning* tiene un límite de tiempo de máximo ocho horas para un *Sprint* de un mes. Para *Sprints* más cortos, el evento suele ser de menor duración.

DAILY SCRUM

El propósito de la *Daily Scrum* es inspeccionar el progreso hacia el Objetivo del *Sprint* y adaptar el *Sprint*

Backlog según sea necesario, ajustando el trabajo planificado entrante.

La *Daily Scrum* es un evento de 15 minutos para los *Developers* del *Scrum Team*. Para reducir la complejidad, se lleva a cabo a la misma hora y en el mismo lugar todos los días hábiles del *Sprint*. Si el *Product Owner* o *Scrum Master* están trabajando activamente en elementos del *Sprint Backlog*, participan como *Developers*.

Los *Developers* pueden seleccionar la estructura y las técnicas que deseen, siempre que su *Daily Scrum* se centre en el progreso hacia el Objetivo del *Sprint* y produzca un plan viable para el siguiente día de trabajo. Esto crea enfoque y mejora la autogestión.

Las *Daily Scrum*s mejoran la comunicación, identifican impedimentos, promueven la toma rápida de decisiones y, en consecuencia, eliminan la necesidad de otras reuniones.

La *Daily Scrum* no es el único momento en el que los *Developers* pueden ajustar su plan. A menudo se reúnen durante el día para discusiones más detalladas sobre cómo adaptar o volver a planificar el resto del trabajo del *Sprint*.

SPRINT REVIEW

El propósito de la *Sprint Review* es inspeccionar el resultado del *Sprint* y determinar futuras adaptaciones. El *Scrum Team* presenta los resultados de su trabajo a los interesados clave y se discute el progreso hacia el Objetivo del Producto.

Durante el evento, el *Scrum Team* y los interesados revisan lo que se logró en el *Sprint* y lo que ha cambiado en su entorno. Con base en esta información, los asistentes colaboran sobre qué hacer a continuación. El *Product Backlog* también se puede ajustar para satisfacer nuevas oportunidades. La

Sprint Review es una sesión de trabajo y el *Scrum Team* debe evitar limitarla a una presentación.

La *Sprint Review* es el penúltimo evento del *Sprint* y tiene un límite de tiempo de máximo cuatro horas para un *Sprint* de un mes. Para *Sprints* más cortos, el evento suele ser de menor duración.

SPRINT RETROSPECTIVE

El propósito de la *Sprint Retrospective* es planificar formas de aumentar la calidad y la efectividad.

El *Scrum Team* inspecciona cómo fue el último *Sprint* con respecto a las personas, las interacciones, los procesos, las herramientas y su Definición de Terminado. Los elementos inspeccionados suelen variar según el ámbito del trabajo. Se identifican los supuestos que los llevaron por mal camino y se exploran sus orígenes. El *Scrum Team* analiza qué salió bien durante el *Sprint*, qué problemas encontró y cómo se resolvieron (o no) esos problemas.

El *Scrum Team* identifica los cambios más útiles para mejorar su efectividad. Las mejoras más impactantes se abordan lo antes posible. Incluso se pueden agregar al *Sprint Backlog* para el próximo *Sprint*.

La *Sprint Retrospective* concluye el *Sprint*. Tiene un tiempo limitado a máximo tres horas para un *Sprint* de un mes. Para *Sprints* más cortos, el evento suele ser de menor duración.

ARTEFACTOS DE SCRUM

Los artefactos de Scrum representan trabajo o valor. Están diseñados para maximizar la transparencia de la información clave. Por lo tanto, todas las personas que los inspeccionan tienen la misma base de adaptación.

Cada artefacto contiene un compromiso para garantizar que proporcione información que mejore la transparencia y el enfoque frente al cual se pueda medir el progreso:

- Para el *Product Backlog*, es el Objetivo del Producto.
- Para el *Sprint Backlog*, es el Objetivo del *Sprint*.
- Para el *Increment* es la Definición de Terminado.

Estos compromisos existen para reforzar el empirismo y los valores de Scrum para el *Scrum Team* y sus interesados.

PRODUCT BACKLOG

El *Product Backlog* es una lista emergente y ordenada de lo que se necesita para mejorar el producto. Es la única fuente del trabajo realizado por el *Scrum Team*.

Los elementos del *Product Backlog* que el *Scrum Team* puede dar por Terminados dentro de un *Sprint* se consideran preparados para ser seleccionados en un evento de *Sprint Planning*. Suelen adquirir este grado de transparencia tras las actividades de refinamiento. El refinamiento del *Product Backlog* es el acto de dividir y definir aún más los elementos del *Product Backlog* en elementos más pequeños y precisos. Esta es una actividad continua para agregar detalles, como una descripción, orden y tamaño. Los atributos suelen variar según el ámbito del trabajo.

Los *Developers* que realizarán el trabajo son responsables del dimensionamiento. El *Product Owner* puede influir en los *Developers* ayudándolos a entender y seleccionar sus mejores alternativas.

Compromiso: Objetivo del Producto

El Objetivo del Producto describe un estado futuro del producto que puede servir como un objetivo para que el *Scrum Team* planifique. El Objetivo del Producto está en el *Product Backlog*. El resto del *Product*

Backlog emerge para definir "qué" cumplirá con el Objetivo del Producto.

> *Un producto es un vehículo para entregar valor. Tiene un límite claro, personas interesadas conocidas, usuarios o clientes bien definidos. Un producto puede ser un servicio, un producto físico o algo más abstracto.*

El Objetivo del Producto es el objetivo a largo plazo del *Scrum Team*. Ellos deben cumplir (o abandonar) un objetivo antes de asumir el siguiente.

SPRINT BACKLOG

El *Sprint Backlog* se compone del Objetivo del *Sprint* (por qué), el conjunto de elementos del *Product*

Backlog seleccionados para el *Sprint* (qué), así como un plan de acción para entregar el *Increment* (cómo).

El *Sprint Backlog* es un plan realizado por y para los *Developers*. Es una imagen muy visible y en tiempo real del trabajo que los *Developers* planean realizar durante el *Sprint* para lograr el Objetivo del *Sprint*. En consecuencia, el *Sprint Backlog* se actualiza a lo largo del *Sprint* a medida

que se aprende más. Debe tener suficientes detalles para que puedan inspeccionar su progreso en la *Daily Scrum*.

Compromiso: Objetivo del *Sprint*

El Objetivo del *Sprint* es el único propósito del *Sprint*. Si bien el Objetivo del *Sprint* es un compromiso de los *Developers*, proporciona flexibilidad en términos del trabajo exacto necesario para lograrlo. El Objetivo del *Sprint* también crea coherencia y enfoque, lo que alienta al *Scrum Team* a trabajar en conjunto en lugar de en iniciativas separadas.

El Objetivo del *Sprint* se crea durante el evento *Sprint Planning* y se agrega al *Sprint Backlog*. Mientras los *Developers* trabajan durante el *Sprint*, tienen en mente el Objetivo del *Sprint*. Si el trabajo resulta ser diferente de lo que esperaban, colaboran con el *Product Owner* para negociar el alcance del *Sprint Backlog* dentro del *Sprint* sin afectar el Objetivo del *Sprint*.

INCREMENT

Un *Increment* es un peldaño concreto hacia el Objetivo del Producto. Cada *Increment* se suma a todos los *Increments* anteriores y se verifica minuciosamente, lo que garantiza que todos los *Increments* funcionen juntos. Para proporcionar valor, el *Increment* debe ser utilizable.

Se pueden crear múltiples *Increments* dentro de un *Sprint*. La suma de los *Increments* se presenta en la *Sprint Review* apoyando así el empirismo. Sin embargo, se puede entregar un *Increment* a los interesados antes del final del *Sprint*. La *Sprint Review* nunca debe considerarse una puerta para liberar valor.

El trabajo no puede considerarse parte de un *Increment* a menos que cumpla con la Definición de Terminado.

Compromiso: Definición de Terminado

La Definición de Terminado es una descripción formal del estado del *Increment* cuando cumple con las medidas de calidad requeridas para el producto.

En el momento en que un elemento del *Product Backlog* cumple con la Definición de Terminado, nace un *Increment*.

La Definición de Terminado crea transparencia al brindar a todos un entendimiento compartido de qué trabajo se completó como parte del *Increment*. Si un elemento del *Product Backlog* no cumple con la Definición de Terminado, no se puede publicar ni presentar en la *Sprint Review*. En su lugar, vuelve al *Product Backlog* para su consideración futura.

Si la Definición de Terminado para un *Increment* es parte de los estándares de la organización, todos los *Scrum Team*s deben seguirla como mínimo. Si no es un estándar organizacional, el *Scrum Team* debe crear una Definición de Terminado apropiada para el producto.

Los *Developers* deben adherirse a la Definición de Terminado. Si hay varios *Scrum Team*s trabajando juntos en un producto, deben definir y cumplir mutuamente la misma Definición de Terminado.

NOTA FINAL

Scrum es gratuito y se ofrece en esta Guía. El marco de trabajo Scrum, como se describe aquí, es inmutable. Si bien es posible implementar solo partes de Scrum, el resultado no es Scrum. Scrum existe solo en su totalidad y funciona bien como un contenedor para otras técnicas, metodologías y prácticas.

AGRADECIMIENTOS

Personas

De los miles de personas que han contribuido a Scrum, debemos destacar a las que fueron fundamentales al principio: Jeff Sutherland trabajó con Jeff McKenna y John Scumniotales, y Ken Schwaber trabajó con Mike Smith y Chris Martin, y todos ellos trabajaron juntos. Muchos otros contribuyeron en los años siguientes y sin su ayuda Scrum no estaría refinado como lo está hoy.

Historia de la Guía de Scrum

Ken Schwaber y Jeff Sutherland co-presentaron Scrum por primera vez en la Conferencia OOPSLA en 1995. Básicamente, documentó el aprendizaje que Ken y Jeff adquirieron en los años anteriores y publicó la primera definición formal de Scrum.

La Guía de Scrum documenta Scrum como se ha desarrollado, evolucionado y sostenido durante más de 30 años por Jeff Sutherland y Ken Schwaber. Otras fuentes proporcionan patrones, procesos y enfoques que complementan el marco de trabajo Scrum. Estos pueden aumentar la productividad, el valor, la creatividad y la satisfacción con los resultados.

La historia completa de Scrum se describe en otros lugares. Para honrar los primeros sitios donde se probó y comprobó, reconocemos a Individual Inc., Newspage, Fidelity Investments e IDX (ahora GE Medical).

Traducción

Esta guía ha sido traducida de la versión original en inglés proporcionada por Ken Schwaber y Jeff

Sutherland. Las personas que han contribuido en la traducción son: Marcelo López, Marcelo García,

Jorge Abad, Fabian Schwartz y Lucho Salazar.

Información de contacto:
Nombre: Lucho Salazar

Correo electrónico: lucho.salazar@gmail.com

Sitio Web: http://www.gazafatonarioit.com

LinkedIn: http://www.linkedin.com/in/luchosalazar

CAMBIOS DE LA GUÍA SCRUM 2017 A LA GUÍA SCRUM 2020

Aún menos prescriptiva

A lo largo de los años, la Guía Scrum comenzó a ser un poco más prescriptiva. La versión 2020 tenía como objetivo que Scrum volviera a ser un marco de trabajo mínimamente suficiente al eliminar o suavizar el lenguaje prescriptivo. Por ejemplo, eliminó las preguntas de la *Daily Scrum*, suavizó el lenguaje sobre los atributos de los PBI, suavizó el lenguaje sobre los elementos retro en el *Sprint Backlog*, acortó la sección de cancelación de *Sprint* y más.

Un equipo, enfocado en un producto

El objetivo era eliminar el concepto de un equipo separado dentro de un equipo que ha llevado a un comportamiento de "proxy" o de "nosotros y ellos" entre el PO y el Equipo de Desarrollo. Ahora solo hay un *Scrum Team* enfocado en el mismo objetivo, con tres diferentes conjuntos de responsabilidades: PO,

SM y *Developers*.

Introducción del Objetivo del Producto

La Guía Scrum 2020 introduce el concepto de Objetivo del Producto para proporcionar enfoque al *Scrum*

Team hacia un objetivo valioso más grande. Cada *Sprint* debería acercar el producto al Objetivo del Producto general.

Un hogar para el Objetivo del *Sprint*, la Definición de Terminado y el Objetivo del Producto

Las Guías Scrum anteriores describían el Objetivo del *Sprint* y la Definición de Terminado sin realmente darles una identidad. No eran del todo artefactos, pero estaban algo unidos a los artefactos. Con la incorporación del Objetivo del Producto, la versión 2020 proporciona más claridad al respecto. Cada uno de los tres artefactos ahora contiene "compromisos" con ellos. Para el *Product Backlog* es el Objetivo del Producto, el *Sprint Backlog* tiene el Objetivo del *Sprint* y el *Increment* tiene la Definición de Terminado

(ahora sin las comillas). Existen para aportar transparencia y enfocarse en el progreso de cada artefacto.

Autogestión sobre autoorganización

Las Guías Scrum anteriores se referían a los Equipos de Desarrollo como autoorganizados, eligiendo quién y cómo hacer el trabajo. Con un enfoque más en el *Scrum Team*, la versión 2020 enfatiza un *Scrum Team* autogestionado, eligiendo quién, cómo y en qué trabajar.

Tres temas de la *Sprint Planning*

Además de los temas de la *Sprint Planning* de "Qué" y "Cómo", la Guía de Scrum 2020 pone énfasis en un tercer tema, "Por qué", en referencia al Objetivo del *Sprint*.

Simplificación general del lenguaje para una audiencia más amplia

La Guía Scrum 2020 ha hecho hincapié en eliminar declaraciones redundantes y complejas, así como en eliminar cualquier inferencia restante al trabajo de TI (por ejemplo, pruebas, sistema, diseño, requisito, etc.). La Guía Scrum ahora tiene menos de 13 páginas.