Universidad de Oriente

Núcleo de Anzoátegui

Escuela de Ingeniería y Ciencias Aplicadas

Departamento de Computación y Sistemas

DESARROLLO DE UNA APLICACIÓN WEB PARA EL SEGUIMIENTO DE PAGOS, VENTAS Y CONTROL DE STOCK PARA UNA EMPRESA DE VENTA DE ROPA PERSONALIZADA

Realizado por:

Br. Luis Agreda C.I: 26.886.935

Trabajo de Grado Presentado Ante la Universidad de Oriente

Como Requisito Parcial para Optar al Título de:

INGENIERO EN COMPUTACIÓN

Barcelona, 2025.

Universidad de Oriente

Núcleo de Anzoátegui
Escuela de Ingeniería y Ciencias Aplicadas

Departamento de Computación y Sistemas

DESARROLLO DE UNA APLICACIÓN WEB PARA EL SEGUIMIENTO DE PAGOS, VENTAS Y CONTROL DE STOCK PARA UNA EMPRESA DE VENTA DE ROPA PERSONALIZADA

Realizado por:

Br. Luis Agreda C.I: 26.886.935

TESIS REVISADA POR ASESOR ACADÉMICO

_____

Ing. Aquiles Torrealba

Barcelona, 2025.

Universidad de Oriente
Núcleo de Anzoátegui

Escuela de Ingeniería y Ciencias Aplicadas

Departamento de Computación y Sistemas

DESARROLLO DE UNA APLICACIÓN WEB PARA EL SEGUIMIENTO DE PAGOS, VENTAS Y CONTROL DE STOCK PARA UNA EMPRESA DE VENTA DE ROPA PERSONALIZADA

TESIS APROBADA POR EL DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS DE LA ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

CALIFICACIÓN OBTENIDA

_____

Prof. Aquiles Torrealba

Tutor Académico

_____ _____

Ing. Manuel Carrasquero
Jurado Principal

Ing. Claudio Cortinez
Jurado Principal

Barcelona, 2025

RESOLUCIÓN

De Acuerdo al Artículo 44 del Reglamento de Trabajos de Grado:

"Los trabajos son propiedad exclusiva de la Universidad de Oriente, y sólo podrán ser utilizados para otros fines con el consentimiento expreso del Consejo de Núcleo respectivo, quien participará al Consejo de Universidades"

DEDICATORIA

A mi familia, por ser mi fortaleza, A mis amigos, por hacer de este camino uno inolvidable.

AGRADECIMIENTOS

A Dios y la Virgen del Valle, a mi familia, sobre todo a mi hermano por ser gran ejemplo, así como a los profesores y amigos que han dejado huella.

RESUMEN

El presente trabajo de investigación tuvo como objetivo principal Desarrollar una aplicación web para el control de las ventas, pagos y manejo del stock, de una empresa dedicada a la venta de productos de ropa personalizada, en Puerto la cruz, estado Anzoátegui, basándose en la metodología del Proceso Racional Unificado. Para el logro de los objetivos planteados se realizaron visitas a la empresa PACHUS. donde se obtuvo información con el dueño de la empresa sobre el problema de la situación actual por la que pasa la empresa en el momento de gestionar sus equipos, y verificar el pago de sus facturas. Se realizó una observación de como es el sistema actual que posee la empresa para el control de sus facturas descubriendo las fallas existentes en el sistema el cual era totalmente a mano sin ninguna automatización de procesos; se tomaron los puntos más críticos para crear una aplicación que soportara no solo la información, sino que realizara procesos automáticos en diversas áreas teniendo como resultado la creación del Sistema Administrador de Ventas, Inventario y Pagos(SAVIP). El desarrollo del software posee un marco de trabajo en NextJS y la aplicación de una base de datos en PostgreSQL, así como, un lenguaje de programación Javascript, lenguaje de hojas de estilo en Cascada(CSS) y un lenguaje de marcado HTML.

TABLA DE CONTENIDO

RESOLUCIÓN iv
DEDICATORIA v
AGRADECIMIENTOS vi
RESUMEN vii
TABLA DE CONTENIDO viii
CAPÍTULO I 12
EL PROBLEMA 12
1.1. Planteamiento del problema 12
1.2 Objetivos 15
1.2.1 Objetivos general 15
1.2.2 Objetivos específicos 15
CAPÍTULO II 16
MARCO TEÓRICO 16
2.1 Antecedentes 16
2.2 Bases Teóricas Referenciales 18
2.2.1 Aplicación Web 18
2.2.2 Tienda virtual 18
2.2.3 Control de ventas 18
2.2.4 Gestión de stock 19
2.2.5 Paypal 19
2.2.6 JSON 20

2.2.7 HTML 20
2.2.8 CSS 20
2.2.9 Node.js 21
2.2.10 JavaScript 21
2.2.11 Typescript 21
2.2.12 React 22
2.2.13 NextJS 22
2.2.14 Base de datos 22
2.2.15. Prisma 23
2.2.16 PostgreSQL 23
2.2.17 Lenguaje Unificado de Modelado (UML) 24
2.2.18 Modelo de Casos de Uso 24
2.2.19 Proceso Unificado de Software (R.U.P) 25
2.2.19.1 Ciclo de Vida del Proceso Unificado de Desarrollo de Software (R.U.P) 25
2.2.20 Programación basada en componentes 26
CAPÍTULO III 27
MARCO METODOLÓGICO 27
3.1 Tipo de investigación 27
3.2 Nivel de investigación 27
3.3 Técnicas a utilizar 27
3.3.1 Observación directa 28
3.3.2 Revisión documental 28

3.3.3 Entrevista no estructurada..... 28
3.3.4 Lenguaje de Modelado Unificado (UML) 29
CAPÍTULO IV 30
RESULTADOS 30
4.1. FASE DE INICIO 30
4.1.1 Flujo De trabajo De Requisitos..... 31
4.1.1.1 Descripción del Sistema Actual..... 31
4.1.1.2 Modelo de Dominio 32
4.1.1.3 Identificación del sistema propuesto 34
4.1.1.4 Requisitos funcionales del sistema 34
4.1.1.5 Requisitos no funcionales del sistema 35
4.1.1.6 Identificación de los riesgos 36
4.1.1.7 Modelo de casos de uso. 37
4.1.2 Flujo de Análisis 52
4.1.2.1. Modelo de Análisis 52
4.1.2.2. Paquete de Análisis del Sistema..... 52
4.1.2.3. Diagrama de Clases de Análisis 53
4.1.2.4. Diagrama de Colaboración..... 60
4.1.3 Flujo de trabajo de diseño..... 70
4.1.3.1. Arquitectura Candidata..... 70
4.1.3.2 Prototipo de Interfaz de Usuario..... 71
4.1.4 Evaluación de la fase de inicio 72
4.2 Fase de Elaboración..... 73

4.2.1. Flujo de Trabajo de Requisito 73
4.2.2 Flujo de trabajo de diseño 73
4.2.2.1. Identificación de las clases 74
4.2.3 Flujo de trabajo de diseño 77
4.2.3.1 Arquitectura del sistema 78
4.2.3.2 Diagrama de Componentes 79
4.2.4 Diagrama de Despliegue 80
4.2.5 Diseño de la Base de Datos 81
4.2.6 Flujo de trabajo de implementación 89
4.2.6.1 Interfaz Principal de SAVIP 90
4.2.7 Evaluación de la fase de elaboración 91
4.3 Fase de Construcción 92
4.3.1 Flujo de trabajo de implementación 92
4.3.2 Flujo de trabajo de Prueba 109
4.3.2.1 Pruebas de Unidad 110
4.3.3 Conclusión de la fase de construcción 112
CONCLUSIONES 114
RECOMENDACIONES 116
BIBLIOGRAFÍA 117
METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO: 121

CAPÍTULO I
EL PROBLEMA

1.1. Planteamiento del problema

En la actualidad, las empresas se encuentran en la búsqueda de nuevas alternativas que permitan mantenerse competitivas en el mercado, lo que incluye contar con un sistema eficiente para el manejo de las operaciones. La tecnología marca una pauta importante en el desarrollo organizacional y su uso es un factor significativo para las ventajas competitivas de cualquier empresa. Esto conlleva a la necesidad de buscar una alternativa rápida, efectiva y segura para el manejo masivo de información, minimizando los resultados ineficientes y sustituyendo los procesos manuales por procesos automatizados.

Después de la pandemia del COVID-19 y el incremento en el uso de tecnología digital, la sociedad actual busca resultados y soluciones inmediatas a sus necesidades sin tener que trasladarse a un lugar físico. En su lugar, se espera poder resolver problemas con todo lo que tenga al alcance de sus manos. Es por ello que los comercios y locales buscan explotar todo el potencial de las redes sociales y las aplicaciones web, como la propuesta por Mora (2010), aplicación que los usuarios pueden utilizar accediendo a un servidor web a través de Internet con el fin de promocionar y expandir sus servicios a una mayor cantidad de personas.

No obstante, muchas de estas empresas aún enfrentan desafíos significativos relacionados con la gestión de pagos, ventas y control de inventario. El uso de métodos tradicionales y manuales para estas tareas no

solo consume tiempo y recursos, sino que también está propenso a errores humanos que pueden resultar costosos.

Partiendo de esta necesidad se ha observado que una de estas empresas es Pachus, localizada en el Centro Comercial Puerto Plaza en Puerto La Cruz, edo. Anzoátegui. Dedicada a la comercialización de una amplia gama de ropa personalizada, ofreciendo tanto creaciones propias como servicios de personalización a clientes y empresas. El principal problema radica en que al ser visible en las redes sociales su cartera de clientes incrementó, conociendo clientes en otros estados y fuera del país. Al incrementar los clientes se incrementaron las solicitudes y al no contar con un sistema automatizado, la actual manera de trabajar es con una sola persona de contacto que tiene el único número al que los clientes piden información, un correo electrónico y una página en la red social Instagram, donde los clientes pueden ver sus productos, los registros se llevan a mano en la agenda personal del dueño, dado que no se tiene otra herramienta para organizar la información se generan problemas como “la disponibilidad de productos para ciertas fechas”.

El hecho de no tener un registro con datos actualizados y centralizado ha afectado el proceso de ventas y por error se ha llegado a unos productos que no estaban de acuerdo con las especificaciones del cliente, acarreando un problema tanto para la compañía como para el cliente que quedaba insatisfecho, aunado a la deficiente verificación de la información respecto a el stock de los productos dejando a los clientes en la espera de largos tiempos por una respuesta, del mismo modo existen problemas al manejar una sola pasarela de pago, por lo cual los clientes solicitan métodos de pagos alternativos a los bancos nacionales, esto demanda, la conversión monetaria y la relación de los pagos con los clientes manualmente.

En función de lograr dar una respuesta, este proyecto se enfocó en el desarrollo de una aplicación web, para el manejo de la información relacionada a las ventas y pagos de los productos, también servirá como catálogo para presentarle a los clientes diferentes productos ofrecidos por la empresa, así como la disponibilidad de las mismas, la integración de una nueva pasarela de pago y mejorar la accesibilidad de la información y la interacción de los usuarios con la empresa y viceversa.

Para llevar a cabo el proyecto se utilizó técnicas de recolección de información necesarias para conocer el funcionamiento actual en cuanto al registro, control y venta de los productos. Como metodología, se empleó el Proceso Unificado Racional (RUP), propuesto por Booch, Rumbaugh y Jacobson (2000), contemplando las fases de inicio, elaboración y construcción de la aplicación web. Además, se utilizó el Lenguaje de Modelado Unificado (UML), que permite realizar el diseño de la arquitectura del software, evaluando los requerimientos del sistema para que puedan ser analizados y modelados de manera efectiva

La implementación de esta aplicación web no solo responde a una necesidad actual del mercado, sino que también representó una oportunidad significativa para optimizar las operaciones, mejorar la satisfacción del cliente y fortalecer la competitividad en el sector. La integración de métodos de pago como PayPal facilitó las transacciones brindando a los clientes una opción segura y confiable para realizar sus compras en línea.

1.2 Objetivos

1.2.1 Objetivos general

Desarrollar una aplicación web para el seguimiento de pagos, venta y control de stock para la empresa de venta de ropa personalizada PACHUS.

1.2.2 Objetivos específicos

- Describir la situación actual del sistema objeto de estudio
- Identificar los requerimientos funcionales y no funcionales del sistema propuesto
- Diseñar la arquitectura de la aplicación mediante una plataforma web que satisfaga los requerimientos de los usuarios
- Codificar el software según los requerimientos establecidos.

CAPÍTULO II
MARCO TEÓRICO

2.1 Antecedentes

Trujillo (2023), implementó una aplicación bajo entorno web de ventas online para la empresa COMPUSERVICE ISAAC, esta empresa se dedica a la venta de productos de cómputo y telecomunicaciones, presta servicios de instalación de software, con el fin de automatizar los procesos manuales asociados al registro y control de consultas. Contó con el cumplimiento de las tres primeras fases del proceso unificado de desarrollo de software (RUP) y lenguajes de programación PHP y JavaScript, así como el lenguaje de estilo en cascada CSS usando el framework bootstrap y HTML 5 para el diseño de interfaces amigables y de fácil uso. A partir de su análisis, se tomó los casos de uso y problemas potenciales relacionados con la experiencia del usuario, lo que me servirá para enfocar mi proyecto en optimizar la interacción de los usuarios con la plataforma.

Ccoyllo (2022), desarrolló una aplicación web para la automatización del Proceso de Gestión de Pedidos en el Restaurante “Manu Barra Cevichera” en Perú, el objetivo del presente trabajo de investigación fue desarrollar un sistema web, que permita una mejor gestión a la hora de categorizar, listar platos y bebidas de la carta llevando un mejor control de las órdenes de pedidos realizadas en el restaurante, la administración de las órdenes de pedidos y el control de las órdenes de pedidos realizadas en el restaurante. Logró proporcionar solución a dichos problemas existentes, con el fin de ser manejada por la empresa, se realizó utilizando el lenguaje de programación PHP y base de datos relacional con MYSQL. De este trabajo se tomó el

enfoque para diseñar e implementar bases de datos relacionales y que puedan ser escalables.

Hernandez (2023), implementó un sistema de administración de ventas y control de inventarios para tiendas en línea, el caso de estudio: Virtual Store, ubicado en Quito. Ecuador, con el objetivo de diseñar una aplicación para la empresa que permite el registro, seguimiento y control de los productos, pedidos, y pagos. El sistema fue desarrollado, empezando con la fase de Análisis y Diseño, continuando con la construcción de la aplicación a través de las tecnologías de Node.js y Handlebars.js. y para el despliegue de la aplicación en la web mediante Azure App Services. Este análisis brinda una guía para ajustar el manejo y la conexión con los servicios para el despliegue.

Zambrano (2022), desarrolló un sistema web para el control de inventario de la empresa “imprenta & publicidad Cuyabeno”, ubicada en Riobamba, Ecuador. El trabajo cumplió con el trabajo de las tres primeras fases de la metodología RUP. Como resultado se obtuvo un sistema que permitió controlar el stock de los productos para siempre saber cuántas unidades hay en stock. Esta aplicación sigue el patrón de diseño Modelo-Vista-Controlador, permitiendo la gestión de usuarios con autenticación y roles definidos. Los datos se almacenan en bases de datos SQL Server. De este trabajo se tomó el modelo y permisos relacionados a los roles para gestión de controladores, sistema de autenticación.

2.2 Bases Teóricas Referenciales

2.2.1 Aplicación Web

Según Sergio Mora (2010)**:**

En la ingeniería de software se denomina aplicación web aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador, es decir, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.

2.2.2 Tienda virtual

Según Castaño (2016):

Una tienda virtual es aquel lugar en el que se venden productos en internet. De otra forma, esto se puede interpretar como una página web perteneciente a una empresa, en la que la empresa presenta su oferta de productos y permite a sus clientes no solamente visualizar el catálogo de productos de la empresa, sino también realizar una compra a través de esta página web.

2.2.3 Control de ventas

Según Camilo Clavijo (2023):

El control de ventas es un registro que asegura la productividad de la fuerza de ventas de una empresa, asociado a los productos y los vendedores. Este control permite que un negocio se mantenga alerta y coherente en sus acciones para mantenerse, así cómo vender más.

2.2.4 Gestión de stock

Según Morillo (2015)

La gestión del stock se trata de aquella capacidad de una organización de mantener controlada la cantidad de unidades que se tienen de forma física o virtual de cada producto. En ese sentido, un modo ideal de mantener la gestión del stock es implementar algún método que permita a la empresa que se tenga un número óptimo de stock de cada producto, manteniendo un equilibrio entre las entradas y salidas de estos.

2.2.5 Paypal

Según Ríos Rivera (2022)

Es un servicio de pago en línea, también llamado pasarela de pago o agregador de pago, que usa internet. Funciona como un puente, ejemplificando sería algo así: una persona envía dinero a otra o a una empresa sin tener sus datos como número de tarjeta, sólo el correo electrónico que utiliza en PayPal. El pago llega al perfil de quien está cobrando y los fondos se transfieren a una cuenta bancaria. También es posible pagar con PayPal y el sistema toma el dinero de la tarjeta de crédito o débito establecida por el dueño del perfil.

2.2.6 JSON

Según Ana Ferreria (2024):

JSON, acrónimo de JavaScript Object Notation, es un formato de texto sencillo para representar estructuras de datos organizadas en objetos y arrays. La estructura básica de JSON involucra pares de clave-valor, donde cada clave es un identificador único y el valor correspondiente puede ser un número, string, booleano, array u otro objeto JSON.

2.2.7 HTML

Según Julian Perez (2011):

El HTML se encarga de desarrollar una descripción sobre los contenidos que aparecen como textos y sobre su estructura. complementando dicho texto con diversos objetos (como fotografías, animaciones, videos, etc.). Es un lenguaje muy simple y general que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos

2.2.8 CSS

Segun Diego Santos (2023):

CSS son las siglas en inglés para «hojas de estilo en cascada» (Cascading Style Sheets). Básicamente, es un lenguaje que maneja el diseño y presentación de las páginas web, es decir, cómo lucen cuando un usuario las visita. Funciona junto con el lenguaje HTML que se encarga del contenido básico de los sitios.

2.2.9 Node.js

Según Cassingena (2022):

Es un entorno de ejecución de JavaScript del lado del servidor que permite construir aplicaciones web escalables y de alto rendimiento. Node.js utiliza un modelo de entrada/salida sin bloqueo y basado en eventos, lo que lo hace eficiente y adecuado para aplicaciones en tiempo real.

2.2.10 JavaScript

Según Guadalupe (2023):

JavaScript es un lenguaje de programación que los desarrolladores utilizan para hacer páginas web interactivas. Desde actualizar fuentes de redes sociales a mostrar animaciones y mapas interactivos, las funciones de JavaScript pueden mejorar la experiencia del usuario de un sitio web. Como lenguaje de scripting del lado del servidor, se trata de una de las principales tecnologías de la World Wide Web. Por ejemplo, al navegar por Internet, en cualquier momento en el que vea un carrusel de imágenes, un menú desplegable “click-to-show” (clic para mostrar), o cambien de manera dinámica los elementos de color en una página web, estará viendo los efectos de JavaScript.

2.2.11 Typescript

TypeScript es una extensión al lenguaje de programación JavaScript que se caracteriza por ampliar su sintaxis en el ámbito de los tipos.

Es un lenguaje transpilado a JavaScript, lo que se traduce y compila antes de ser usado en un navegador. Esto no significa que sea un reemplazo, sino que

lo complementa, permitiendo que utilicemos el nuevo código de TypeScript y el lenguaje que ya conocemos de JavaScript. (Negrete, 2017)

2.2.12 React

Según Gorka Lauciria (2019):

React es una librería Javascript focalizada en el desarrollo de interfaces de usuario. Creada por facebook, React contiene una colección de fragmentos de código JavaScript reutilizables utilizados para crear interfaces de usuario (UI) llamadas componentes. ReactJS no es un framework de JavaScript, debido a que solo es el responsable de renderizar los componentes de la capa de vista de una aplicación.

2.2.13 NextJS

Según Fuente oficiales de la API de Kinsta:

Next.js es un framework de React que te permite construir sitios web y aplicaciones web estáticas supercargadas, amigables con el SEO y extremadamente orientadas al usuario, utilizando el framework de React. Next.js es conocido por la mejor experiencia de los desarrolladores a la hora de construir aplicaciones listas para usar en producción con todas las características que necesitas.

2.2.14 Base de datos

Según Luis Rivas y Evelin Yegres(2017):

Una base de datos es una colección de datos relacionados. Por datos, quiere decir hechos conocidos que pueden registrarse y que tienen un

significado implícito. Una base de datos tiene las siguientes propiedades implícitas:

- Una base de datos representa algunos aspectos del mundo real, en ocasiones denominado mini mundo o universo del discurso. Los cambios en el mini mundo se reflejan en la base de datos
- Una base de datos es una colección de datos lógicamente coherentes, con algunos significados inherentes. Un conjunto aleatorio de datos no puede considerarse como una base de datos.
- Las bases de datos se diseñan, construyen y pueblan con datos para un propósito específico. Está destinada a un grupo de usuarios y tiene algunas aplicaciones preconcebidas de interés para dichos usuarios.

2.2.15. Prisma

Según González (2017):

Es una técnica de programación para lograr convertir datos procedentes de una base de datos en un lenguaje orientado a objetos. Es un ORM (ObjectRelationalMapping) un modelo de programación cuya misión es transformar las tablas de una base de datos de forma que las tareas básicas, que realizan los programadores, están simplificadas.

2.2.16 PostgreSQL

Según Lardirère (2012):

PostgreSQL es una herramienta dedicada al almacenamiento y tratamiento de datos. A su vez, es un sistema de bases de datos de código abierto, altamente estable, que proporciona soporte a diferentes funciones de SQL, como claves foráneas, sub-consultas, disparadores y diferentes tipos y

funciones definidas por el usuario. Además, aumenta el lenguaje SQL ofreciendo varias funciones que escalan y reservan meticulosamente las cargas de trabajo de datos. Se utiliza principalmente para almacenar datos para muchas aplicaciones móviles, web, geoespaciales y de análisis.

2.2.17 Lenguaje Unificado de Modelado (UML)

Según Booch, Jacobson y Rumbaugh (2000):

Se define UML como un lenguaje modelado visual que se usa para especificar, visualizar y construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, demonios de aplicación, y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar.

2.2.18 Modelo de Casos de Uso

El modelo de casos de uso describe un sistema en términos de sus distintas formas de utilización cada una de las cuales se conoce como un caso de uso. Cada caso de uso o flujo se compone de una secuencia de eventos iniciada por el usuario. Dado que los casos de usos describen el sistema a desarrollarse, los cambios en los requisitos significarán cambios en los casos de uso. Para comprender los casos de uso de un sistema primero es necesario saber quiénes son sus usuarios, para ello, se define el concepto de actor, que es el tipo de usuario que está involucrado en la utilización de un sistema y que además es una entidad externa al propio sistema. Juntos, el actor y el caso de

uso, representan los dos elementos básicos de este modelo. (Weitzenfeld 2005).

2.2.19 Proceso Unificado de Software (R.U.P)

Alonso y Martínez, (2005) definen al proceso unificado como el desarrollo de software que describe el conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema de software. Está dirigido por casos de uso, centrado en la arquitectura del sistema, y es iterativo e incremental.

2.2.19.1 Ciclo de Vida del Proceso Unificado de Desarrollo de Software (R.U.P)

Según Rumbaugh, Jacobson y Booch, (2000) Cada ciclo se desarrolla a lo largo del tiempo. Este tiempo a su vez consta de cuatro fases: inicio, elaboración, construcción y transición. A través de una secuencia de modelos, los implicados visualizan lo que está sucediendo en esas fases. Dentro de cada fase, los directores o desarrolladores pueden descomponer adicionalmente el trabajo en iteraciones con sus incrementos resultantes.

- **Fase de Inicio**: Rumbaugh, Jacobson y Booch, (2000) Establecen que durante esta fase se desarrolla una descripción del producto final a partir de una buena idea y se presenta el análisis de negocio para el producto**.**
- **Fase de Elaboración**: Rumbaugh, Jacobson y Booch, (2000) Mencionan que durante esta fase se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema.

- La relación entre la arquitectura del sistema y el propio sistema es primordial.
- **Fase de Construcción**: Rumbaugh, Jacobson y Booch, (2000) Establecen que, durante esta fase, la línea base de la arquitectura crece hasta convertirse en el sistema completo.
- **Fase de Transición**: Rumbaugh, Jacobson y Booch, (2000) Mencionan que esta fase cubre el periodo durante el cual el producto se convierte en la versión beta. En la versión beta un número reducido de usuarios con experiencia prueba el producto e informa de defectos y deficiencias. Los desarrolladores corrigen los problemas e incorporan algunas de las mejoras sugeridas en una versión general dirigida a la totalidad de la comunidad de usuarios.

2.2.20 Programación basada en componentes

La programación orientada a componentes (que también es llamada basada en componentes) es una rama de la ingeniería del software, con énfasis en la descomposición de sistemas ya conformados en componentes funcionales o lógicos con interfaces bien definidas usadas para la comunicación entre componentes. Se considera que el nivel de abstracción de los componentes es más alto que el de los objetos y por lo tanto no comparten un estado y se comunican intercambiando mensajes que contienen datos.

CAPÍTULO III
MARCO METODOLÓGICO

3.1 Tipo de investigación

De acuerdo a Arias (2012), los tipos de investigación se identifican: según el nivel, según el Diseño y según el Propósito. En lo que corresponde a los tipos de investigación del presente proyecto, estos se encuentran clasificados en descriptiva que se corresponde con el nivel de investigación y, en una investigación de campo de acuerdo al diseño de investigación.

3.2 Nivel de investigación

Según Arias (2006), el nivel de investigación para este proyecto será descriptivo. Esta Investigación abarca el diseño y desarrollo de una aplicación web para solventar la problemática que presenta este tipo de organización, para lograr el diseño primero será necesario ir directamente al sitio de estudio para poder describir las características y el comportamiento que presenta, así como el conjunto de actividades que se desenvuelven dentro de la misma con sus implicaciones, de esta manera se puede hacer un diseño apropiado para la organización. Es por ello que el nivel de investigación es el mencionado anteriormente.

3.3 Técnicas a utilizar

A fin de alcanzar los objetivos de este proyecto, es indispensable aplicar técnicas orientadas a obtener el máximo de información y datos exactos y cercanos a la realidad:

3.3.1 Observación directa

Arias (2006), describe la observación directa como uno de los métodos más utilizados, tanto por ser el más antiguo como por su eficiencia. Consiste en acudir al sitio de estudio para recoger datos y toda la información necesaria a través de su propia observación. Esta técnica se usó para observar actividades de comportamientos de los participantes y objetos físicos, para posteriormente anotar secuencias, detectar fallas, percibir anomalías y definir requerimientos.

3.3.2 Revisión documental

Está constituida por la revisión de material bibliográfico relacionado con el proyecto a desarrollar, utilizando apoyo de datos que pueden haber sido producto de mediciones hechas por otros, es decir, tesis, libros, folletos, internet, metodologías, leyes y normas, con el propósito de obtener una base teórica amplia. Con esto se obtuvo una base teórica ampliada para el proyecto. Los documentos o referencias secundarias provienen de libros, manuales, tesis.

3.3.3 Entrevista no estructurada

Se emplea al personal de la empresa y clientes en cuanto al manejo de almacenamiento y protección de datos; tratar de obtener un bosquejo de las ideas del cliente de cómo quiere que funcione el sistema con el fin de comprender los requisitos fundamentales para la construcción del proyecto.

3.3.4 Lenguaje de Modelado Unificado (UML)

Para el diseño de la aplicación es importante comprender los requisitos, sus limitaciones y su alcance para establecer una aplicación que satisfaga todas las necesidades del funcionamiento, para lograr todo lo anterior es indispensable el uso de los lenguajes UML y webml que cuentan con una serie de diagramas que facilitan la información necesaria para el diseño de la aplicación.

CAPÍTULO IV
RESULTADOS

Este capítulo abarca el desarrollo de todos los métodos previamente elegidos para la realización de este proyecto, desde su aplicación hasta los resultados obtenidos de los mismos. Estos resultados deben analizarse y presentarse de manera adecuada para abordar el problema planteado al inicio del proyecto.

4.1. FASE DE INICIO

Durante esta etapa de la fase de inicio es conocer y analizar las actividades y procesos llevados a cabo en el área de estudio, lo que permite identificar los requisitos funcionales del sistema mediante los diferentes diagramas y modelos implementados en el Proceso Unificado de Desarrollo de Software (RUP) y el Lenguaje Unificado de Modelado (UML) para la representación del sistema. Se comienza con la delimitación del ámbito y alcance del sistema como elemento primordial, para dar paso a una propuesta inicial de la arquitectura del sistema (primeras versiones de los modelos), así como, la identificación de riesgos, actores y requerimientos adicionales.

A continuación, en la figura 4.1 se presenta el desarrollo del Proceso Unificado de Software (RUP), resaltando la Fase de Inicio.

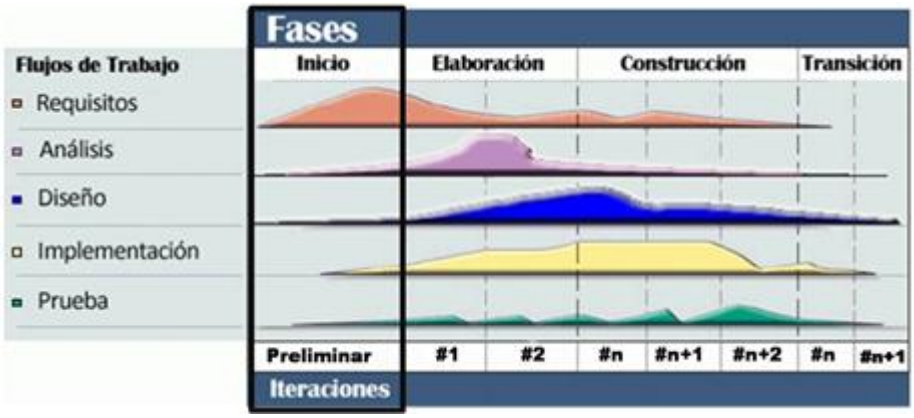


Figura 4.1 Fase de Inicio del RUP.

Fuente: Rumbaugh, Jacobson y Booch, (2000).

4.1.1 Flujo De trabajo De Requisitos

Los requisitos son capacidades y condiciones con las cuales debe cumplir el sistema. El trabajo de los requisitos es que se conozcan las características del sistema, los procesos que se deben cumplir y las necesidades a satisfacer, por lo que deben estudiarse los siguientes aspectos:

4.1.1.1 Descripción del Sistema Actual

PACHUS es una empresa dedicada a la venta de Ropa personalizada y, desde hace años con un equipo de trabajo de dueño encargado de todos los asuntos legales y contables de la compañía actualmente el funcionamiento de la empresa trabaja de la siguiente manera:

1- La empresa trabaja con pedidos previos, estos actualmente son comunicados al dueño por llamadas telefónicas o correos electrónicos y anotados en su agenda personal.

2- El dueño después de confirmar el pago del cliente asienta en su libreta la fecha de entrega para dicho cliente.

3- En caso de pedir un producto con falta de stock, el dueño hace un presupuesto (cantidad, tiempo y precio), se lo envía al cliente y espera la confirmación y el pago para comenzar los operativos de la producción de dicho producto.

5- El dueño tiene sus "clientes fijos" (empresas que regularmente lo contratan), pero él se ocupa de la tarea de hacer publicidad al negocio, calcular mensualmente la contabilidad, estar pendiente de cambios fiscales y legales (% de impuestos, ISLR, IVA, entre otros) y otros cálculos para la administración y toma de decisiones; Todo esto usando su libreta personal y las hojas de cálculo de Microsoft Excel.

6- Cuenta con una tienda, que proporciona una amplia gama de productos relacionados con franelas, sueters, gorras.

7- El dueño tiene que mostrar tanto productos propios como la cantidad de que el cliente requiera información respecto al modelo de los diseños.

4.1.1.2 Modelo de Dominio

Este modelo se encarga de capturar los objetos más importantes vinculados al contexto del sistema. Estos objetos representan los eventos que tienen lugar en el entorno donde se desenvuelven. La Figura 4.2 muestra el diagrama de modelo de dominio del sistema actual, lo que facilita comprender la interacción entre usuarios y los procesos desarrollados dentro de la empresa.

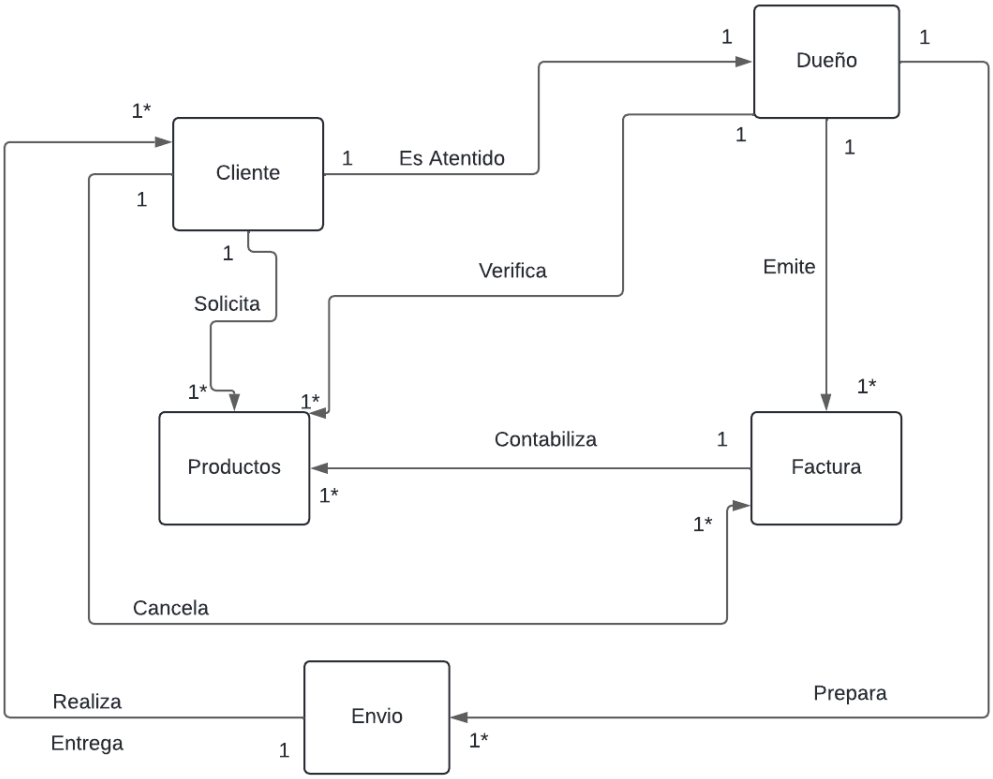


Figura 4.2 Modelo de Dominio.

Fuente: Elaboración Propia.

- **Explicación**

En el diagrama se observa un flujo central donde el Cliente realiza una Solicitud de productos, la cual es atendida por el equipo de ventas (el dueño). Este proceso desencadena una secuencia de verificación de stock (donde se identifican los cuellos de botella marcados con *), Contabilización manual de los productos y Preparación del pedido. Posteriormente, el Dueño interviene para emitir la Factura y finalmente se gestiona el Envío.

Las relaciones revelan dos problemas clave:

- Dependencia excesiva del Dueño: Todas las facturas y validaciones pasan por una sola persona, creando retrasos (ej: cliente Cancela por demoras).
- Procesos manuales: La Verificación y Contabilización se realizan sin integración con sistemas digitales, lo que explica los errores en inventario y pagos.

4.1.1.3 Identificación del sistema propuesto

SAVIP (Sistema administrativo para Ventas, Inventario y Pagos) es el nombre que se le asignó al sistema propuesto para la automatización de las actividades referentes a ventas, pagos y control de stock, además de otros cálculos que apoyen a “PACHUS” a la toma de decisiones.

4.1.1.4 Requisitos funcionales del sistema

Los requisitos funcionales del sistema son declaraciones de los servicios que debe proporcionar el sistema, la manera como este debe reaccionar a entradas particulares y de cómo desenvolverse en situaciones específicas. Estos requisitos se muestran a continuación.

- Debe validar la información ingresada por el usuario y mostrar el formato según el rol del mismo.
- Mostrar y gestionar los artículos correspondientes a la tienda; precio, cantidad, tallas, etc. Tanto para el administrador, como para los usuarios en general.

- Debe contar con filtros que permitan facilitar la búsqueda de información.
- El sistema debe generar ordenes con detalles de productos, cantidades y precios.
- El sistema debe permitir solo a los usuarios con privilegios de administrador cambiar la información relevante como: precios, disponibilidad, etc.

4.1.1.5 Requisitos no funcionales del sistema

Son los requisitos del usuario no relacionados directamente con la funcionalidad del sistema, es decir, no pueden ser observados sin embargo son de vital importancia para garantizar el mejor diseño del sistema a desarrollar. Estos requisitos tocan los aspectos de implementación, portabilidad, dependencia de plataformas y demás cualidades de mantenimientos y disponibilidad, mostrados a continuación:

- El sistema debe de ser web.
- El sistema debe contener los niveles mínimos de seguridad para proteger los datos que gestiona.
- Accesible: la base de datos debe estar ubicada en un servidor en el cual se puedan acceder los datos a través de la red de área local.
- Amigable: la aplicación debe ser fácil de comprender y manejar por el usuario, es por ello que el diseño de la interfaz de usuario debe poseer una estructura amigable.
- Eficiente: Debe agilizar los procesos cotidianos para lograr aumentar la productividad en el trabajo realizado y disminuir los tiempos de ocio ocasionados por el retardo en la respuesta del sistema actual.

- Extensible: el software debe permitir la incorporación de nuevas funcionalidades en su estructura. Debe estar apto para el ingreso de futuras ampliaciones y el soporte de mayor cantidad de datos a través del tiempo.
- Mantenible: el administrador del sistema puede modificar o actualizar cualquier aspecto del mismo cuando se requiera para así garantizar su mantenimiento
- Redes sociales: El sistema cuenta con un apartado en donde dará a conocer sus redes sociales.

4.1.1.6 Identificación de los riesgos

Para llevar a cabo esta fase es necesario detectar los riesgos que pueda sufrir el sistema y anticipar aquellos que podrían afectar la aplicación en el futuro lo que permitirá evitarlos o controlarlos en una etapa temprana del proyecto. A continuación, en la Tabla 4.1 se muestra una lista de los riesgos del sistema identificados hasta el momento, así como el plan de contingencia según su nivel de impacto

Riesgo	Impacto	Responsable	Contingencia
Incapacidad de utilizar el sistema por falta de conexión a la red	Crítico	Desarrollador de Software	Contar con un servidor local para el manejo y respaldo de la información
Fallas de acceso a la base de datos.	Crítico	Desarrollador de Software	Verificar que se establezca una conexión entre el servidor y los usuarios

Interfaces poco amigables al usuario	No Critico	Desarrollador de Software	Mejorar el diseño de las interfaces del sistema.
Ingreso erróneo de datos por parte de los usuarios	Crítico	Usuarios finales.	Agregar debajo de los campos de texto un ejemplo del formato correcto
Algún usuario no autorizado puede modificar algún registro del sistema	Crítico	Desarrollador de Software	Establecer niveles de acceso entre usuarios especificando el contenido que puedan trabajar

Tabla 4.1 Identificación de los Riesgos del Sistema

Fuente: Elaboración Propia.

4.1.1.7 Modelo de casos de uso.

El modelo de casos de uso describe los que hace el sistema para cada tipo de usuario, es decir, cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico. A continuación, en la Figura 4.3 se muestra el caso de uso general para el sistema, así como las funciones u operaciones principales expuestas en la Tabla 4.2.

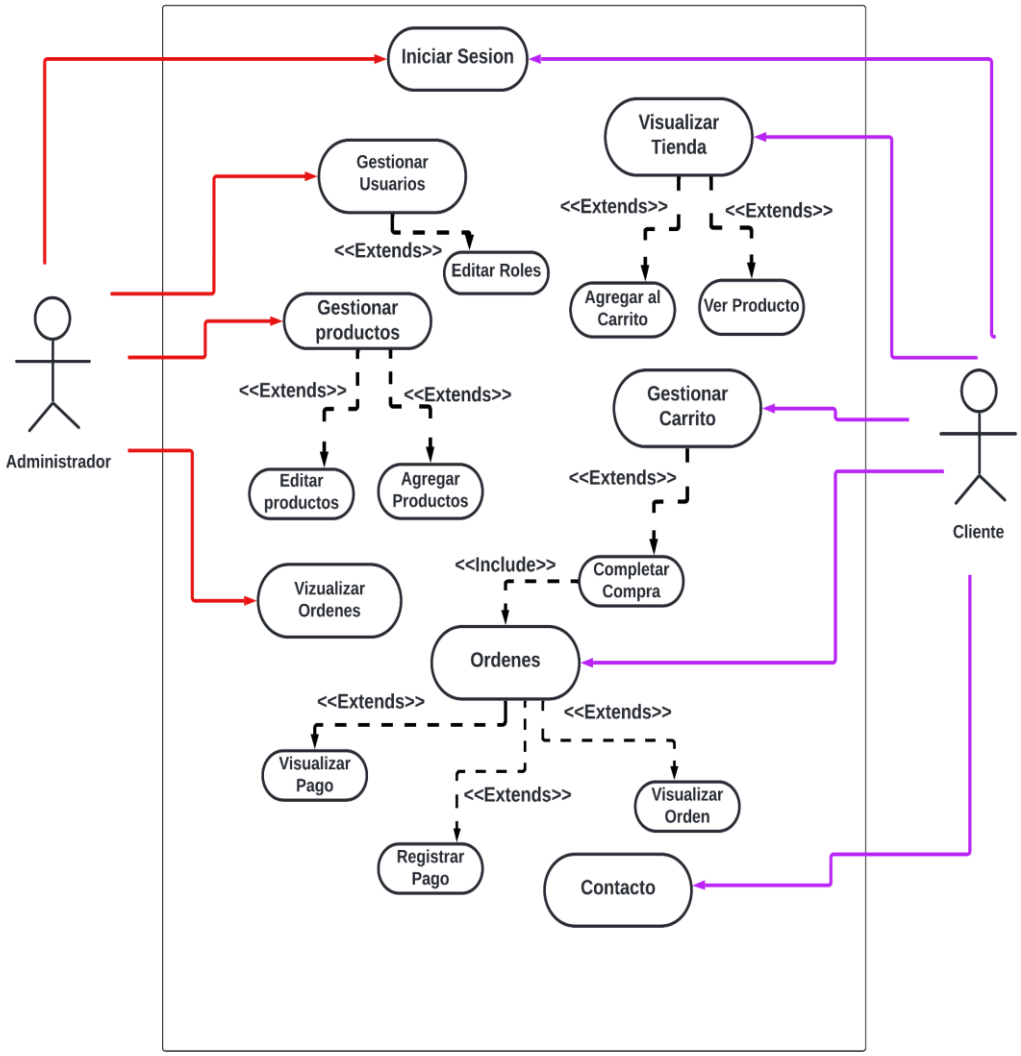


Figura 4.3 Diagrama de caso de uso general del sistema.

Fuente: Elaboración propia.

Explicación de los Casos de Usos

En los siguientes casos de uso se podrá observar cómo ciertos modelos de caso de uso utilizan la metodología C.R.U.D, esta consiste en una

agrupación de cuatro acciones que permiten gestionar la información del caso de uso, siendo éstas: Create (crear), Read (leer), Update (actualizar) y Delete (eliminar), según requiera el caso.

Caso de Uso	Descripción
Iniciar Sesión	Permite la entrada de los usuarios a la interfaz de login del Sistema.
Gestionar Usuarios	Permite visualizar y cambiar los Roles de los usuarios
Gestionar Productos	Permite agregar,modificar,eliminar los productos que ofrece la empresa a los clientes.
Visualizar Órdenes	permite filtrar cada orden hecha y visualizar actores y estatus dentro de ella
Órdenes	Permite visualizar y hacer seguimientos del estado.
Gestionar Carrito	Permite incrementar o decrementar productos del carrito, y generar una orden
Visualizar Tienda	Apartado donde puede visualizar todos los productos y la información relacionado a cada producto
Contacto	Permite entablar conversación con el Dueño de la tienda a través de mensajería SMTP

Tabla 4.2 Identificación de Casos de Uso

Fuente: Elaboración Propia.

Descripción del caso de uso: Iniciar Sesión

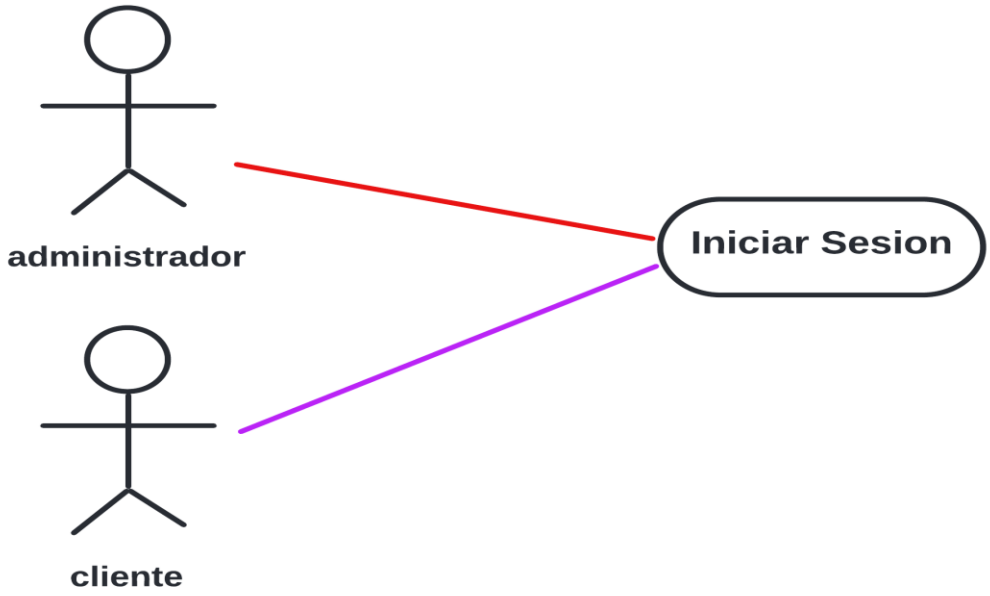


Figura 4.4 Diagrama del caso de uso Iniciar Sesión

Fuente: Elaboración Propia.

Descripción: El usuario desea iniciar sesión en el sistema.

Actor: Administrador, Cliente.

Precondición: Poseer una cuenta.

Flujo normal:

1. El actor del sistema inicia el programa.

2. El sistema despliega la ventana de acceso al sistema.

3. El actor introduce los datos correspondientes.

4. El sistema autoriza el acceso si es usuario o si está registrado y con las restricciones pertinentes según el tipo de usuario.

5. Fin del caso de uso.

Flujo alternativo: Si es la primera vez que el usuario desea iniciar sesión en el sistema, sucederá la secuencia del flujo normal antes mencionado,

sin embargo, será agregado al sistema como un usuario nuevo y con el rol predeterminado, es decir, cliente.

Flujo de eventos alternos:

1. En el evento 3 si el autor ingresa datos incorrectos, el sistema devolverá un mensaje de error.
2. El actor puede cancelar la operación o salir del caso en cualquiera de los eventos.

Descripción del caso de uso: Visualizar tienda.

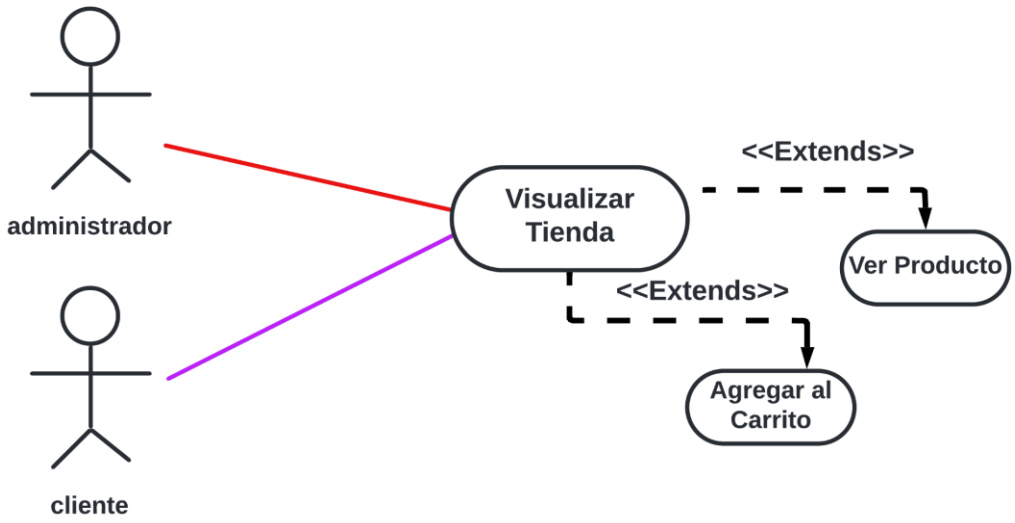


Figura 4.5 Diagrama del caso de uso Visualizar Tienda
Fuente: Elaboración Propia.

Descripción: El usuario puede visualizar todos los productos ofrecidos y seleccionar cuales desea agregar a su carrito de compras.

Actor: Administrador, cliente.

Precondición: Ingresar a la página

Flujo normal:

1. El sistema muestra todos los productos.

2. El usuario selecciona el botón “Agregar al carrito”.

3. El sistema agrega el producto al carrito del usuario.

Flujo alternativo:

1. El sistema muestra todos los productos.

2. El usuario selecciona “Ver producto”.

3. El sistema muestra una ventana con más información detallada acerca del producto seleccionado.

4. El usuario selecciona el botón “Agregar al carrito”.

5. El sistema agrega el producto al carrito del usuario.

Postcondición: La información del carrito del usuario se actualiza según el número de productos agregados.

Descripción del caso de uso: Gestionar carrito

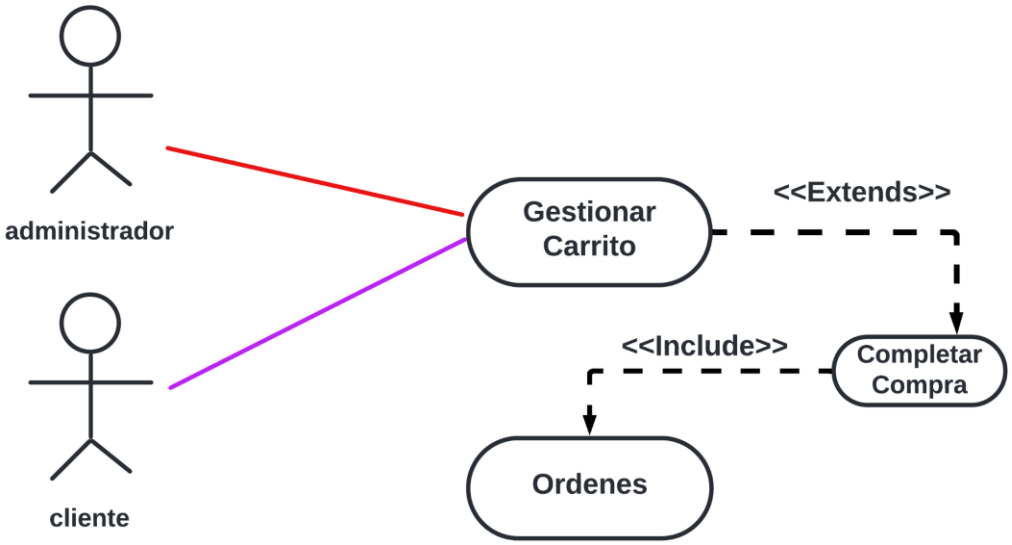


Figura 4.6 Diagrama del caso de uso Gestionar Carrito

Fuente: Elaboración Propia.

Descripción: Gestionar la cantidad de productos dentro del carrito

Actor: Administrador, cliente**.**

Precondición: el usuario debe de haber agregado anteriormente un producto de la tienda.

Flujo normal:

1.El sistema muestra todos los productos agregados por el usuario.

2.El sistema calcula el costo total de todos los productos del carrito y el equivalente en dólares**.**

Flujo alternativo:

1. El sistema mostrará una página vacía si el cliente no posee ningún producto en su carrito de compras.

2. El usuario desea aumentar la cantidad de un producto.

- El usuario selecciona la opción “+” en el producto que desea aumentar la cantidad.
- El sistema aumenta la cantidad del producto.
- Se actualiza el número de productos del carrito y se actualiza el monto total a pagar.

3. El usuario desea disminuir la cantidad de un producto.

- El usuario selecciona la opción “-” en el producto que desea reducir la cantidad.
- El sistema reduce la cantidad del producto.
- Si el usuario reduce la cantidad de un producto el cual cuenta solo con una existencia, el sistema eliminará el producto del carrito.
- Se actualiza el número de productos del carrito y se actualiza el monto total a pagar.

4. El usuario desea eliminar la totalidad de un producto.

- El usuario selecciona la opción “Eliminar producto”.
- El sistema borra el producto y sus existencias del carrito del usuario.
- Se actualiza el número de productos del carrito y se actualiza el monto total a pagar.

Postcondición: En este apartado se detalla cómo se completa una compra.

Descripción: Crea una orden de compra con los productos seleccionados por el usuario.

Actor: Dueño, cliente.

Precondición: Debe haber al menos un producto en el carrito del usuario.

Flujo normal:

1. El usuario selecciona los productos y cantidades que desea.

2. El usuario selecciona la opción “Completar compra”

Postcondición: El usuario es redirigido a la pestaña de órdenes.

Descripción del caso de uso: Gestionar Órdenes

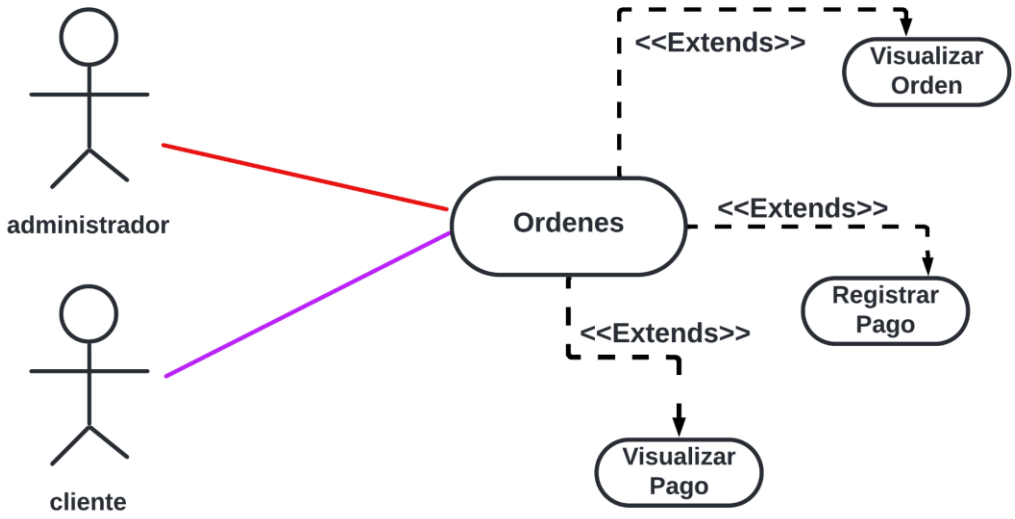


Figura 4.7 Diagrama del caso de uso Gestionar Órdenes

Fuente: Elaboración Propia.

Descripción: Permite el registro de los pagos y la visualización de las órdenes realizadas.

Actor: Administrador, cliente.

Precondición: Sesión iniciada dentro del sistema.

Flujo normal:

1. El usuario realizó una compra.

2. El sistema muestra todas las órdenes que ha realizado.

Flujo alternativo:

1. El usuario selecciona visualizar orden.

- El sistema muestra todas las órdenes realizadas.

- El usuario selecciona la opción “Visualizar” sobre la orden que desea ver.
- El sistema despliega toda la información correspondiente a la orden seleccionada.

2. El usuario selecciona visualizar métodos de pago.

- El usuario selecciona la opción “Métodos de pago”.
- El sistema muestra toda la información con respecto a datos bancarios.

3. El usuario selecciona registrar pago.

- El usuario selecciona la opción “Registrar pago” sobre la orden que desea completar.
- El usuario completa las casillas solicitadas pedidas por paypal.
- El sistema completa la orden de compra.

4. El usuario selecciona registrar pago.

- El usuario completó una orden de compra y desea visualizar sus órdenes de pago.
- El usuario selecciona la opción “Visualizar pagos”
- El usuario es redirigido a la pestaña de pagos.

5. El usuario no ha realizado ninguna orden.

- El sistema muestra una lista vacía de órdenes.

Postcondición: Si se cumple el flujo alternativo número cuatro (4), el usuario es redirigido al caso de uso “Visualizar pagos”.

Descripción del caso de uso: Gestionar usuarios

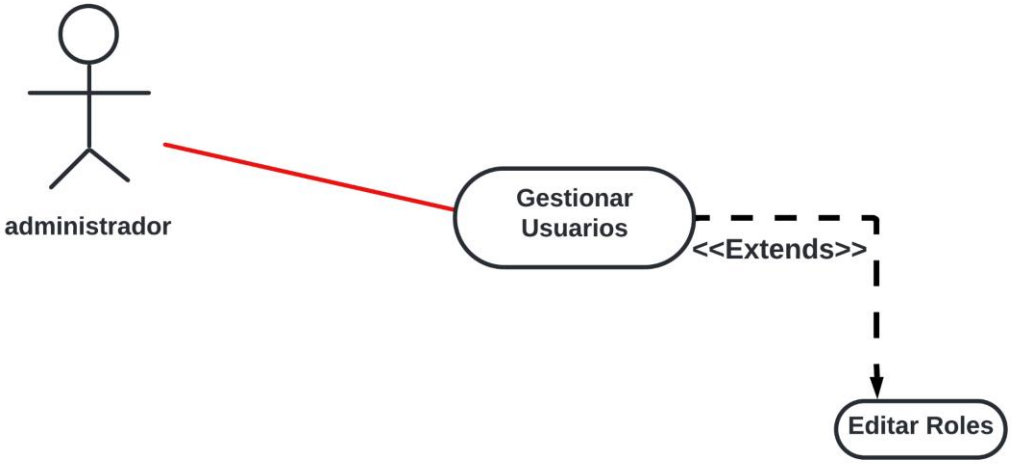


Figura 4.8 Diagrama del caso de uso Gestionar Usuarios

Fuente: Elaboración Propia.

Descripción: Permite gestionar la información de los usuarios del sistema.

Actor: Administrador.

Precondición: El usuario debe tener sesión iniciada en el sistema y poseer el rol de administrador y haber activado la interfaz de Gestionar Usuario.

Flujo normal:

1. El usuario activa la interfaz de gestionar usuario.

2. El sistema muestra un listado con los usuarios registrados de forma alfabética y las opciones a seleccionar.

3. El actor edita los datos y solicita guardar.

4. El sistema registra los datos.

5. Finaliza el caso de uso.

Flujo alternativo:

1. El administrador selecciona la opción de “Editar usuario”.

- El administrador modifica los datos que desee del usuario seleccionado.
- El sistema actualiza los datos del usuario.

Descripción del caso de uso: Gestionar Productos

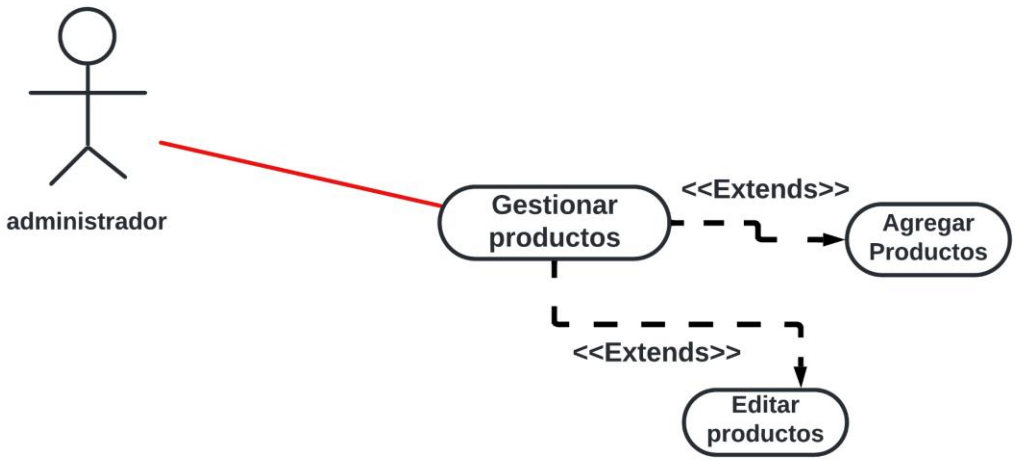


Figura 4.9 Diagrama del caso de uso Gestionar Productos

Fuente: Elaboración Propia.

Descripción: Permite gestionar la información de los productos del sistema.

Actor: Administrador.

Precondición: el usuario debe ingresar al sistema con sus credenciales, debe estar verificado y tener el nivel de administrador y haber activado la interfaz de gestionar equipos.

Flujo normal:

1. El usuario activa la interfaz de gestionar equipos.

2. El sistema muestra un listado con los equipos registrados y las opciones a seleccionar.

3. Si el usuario selecciona la opción agregar equipo el sistema despliega el formulario correspondiente.

4. El usuario ingresa los datos correspondientes y solicita guardarlos.

5. Los datos son validados por el sistema mostrando un mensaje de error si algún campo no está en el formato establecido o está en blanco.

6. El sistema registra los datos.

Flujo alternativo:

1. El administrador selecciona el botón de “Agregar producto”.

- El sistema muestra los campos requeridos para poder agregar un nuevo producto.
- El administrador rellena los campos.
- El administrador selecciona el botón “Guardar cambios”.
- El sistema crea un nuevo producto.
- El sistema actualiza la lista de productos.

2. El administrador selecciona la opción “Editar producto”**.**

- El administrador modifica los datos que desee del producto seleccionado.
- El administrador selecciona “Guardar cambios”.
- El sistema actualiza los datos del producto.

3. El administrador selecciona la opción “Borrar producto”**.**

- El administrador elimina del sistema el producto seleccionado.
- El administrador confirma con el botón “Borrar producto”.
- El sistema actualiza la lista de productos.

4. De no haber un producto existente, el sistema muestra una lista vacía de productos.

Descripción del caso de uso: Contacto

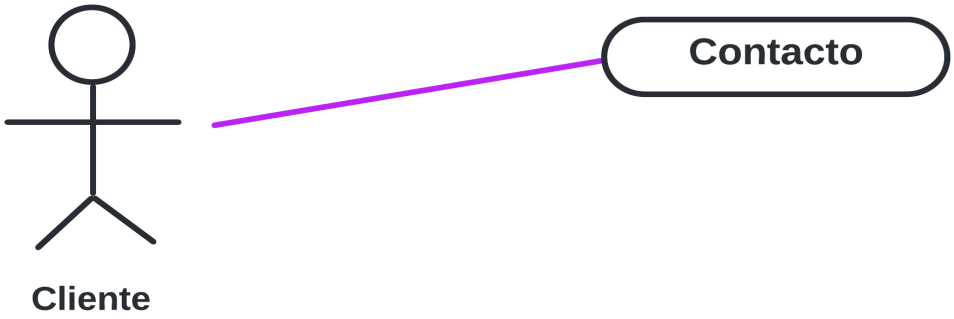


Figura 4.10 Diagrama del caso de uso Contacto

Fuente: Elaboración Propia.

Descripción: El usuario desea contactar con el dueño.

Actor: Cliente

Precondición: Estar en la sección de contacto dentro de la página

Flujo normal:

1. El actor del sistema inicia el programa.

2. El actor ingresa a la sección de contacto

2. El sistema despliega la ventana de acceso al sistema.

3. El actor introduce los datos correspondientes.

4. El sistema autoriza con las restricciones pertinentes según el tipo de datos llenados en los campos.

5. El sistema muestra de forma de una notificación de mensaje enviado.

Flujo alternativo:

1. El actor ingresa los datos en los respectivos campos.

2. El sistema lee los datos y muestra utilizando una notificación el error obtenido.

3. El sistema muestra marcando en rojo que campo se ha ingresado datos erróneos

Descripción del caso de uso: Visualizar Órdenes

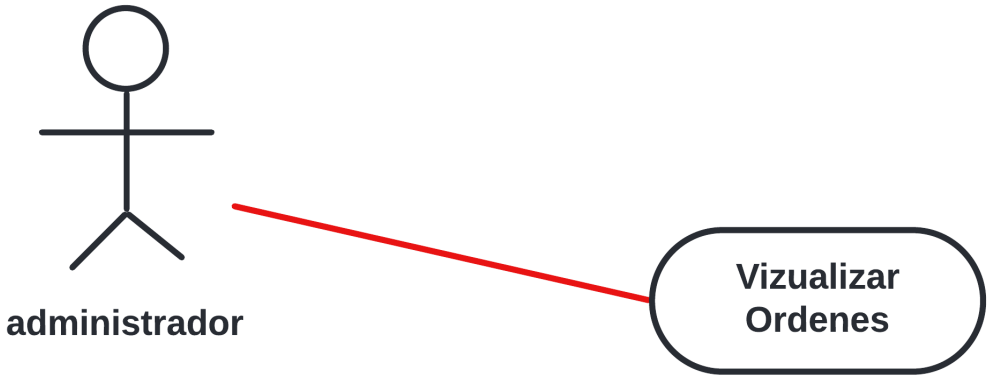


Figura 4.11 Diagrama del caso de uso Contacto

Fuente: Elaboración Propia.

Descripción: El usuario puede ver todos los órdenes hechos dentro del sistema.

Actor: administrador.

Precondición: El usuario debe tener sesión iniciada en el sistema y poseer el rol de administrador.

1. El sistema muestra una lista de todas las órdenes existentes y la información de cada uno de ellas.

Flujo alternativo:

1. El administrador selecciona el botón "ver orden”.

- El sistema muestra los campos relacionados sobre la orden.

4.1.2 Flujo de Análisis

El flujo de análisis se refiere al proceso de examinar y comprender cómo se mueve la información a través del sistema. Este análisis implica estudiar cómo los datos son recopilados, procesados, almacenados y distribuidos dentro de la estructura del sistema de información. En este apartado se identifica de manera detallada cómo la información se desplaza a través del sistema, ayudando a la eficiencia operativa y la efectividad de la gestión de la información.

4.1.2.1. Modelo de Análisis

Con el fin de establecer las conexiones de cada caso de uso desde la perspectiva interna del sistema, se generó el modelo de análisis. Este modelo está compuesto por clases y paquetes estereotipados, los cuales proporcionan estructura a la visión interna del sistema.

Una clase de análisis actúa como una representación abstracta de una o más clases y generalmente se clasifican de la siguiente manera: interfaz (representan las comunicaciones y conexiones entre el usuario y el sistema), controlador (generalmente se configuran para administrar una función específica del sistema) y entidad (representan información persistente rastreada por el sistema).

4.1.2.2. Paquete de Análisis del Sistema

El diagrama de paquetes de análisis muestra como un sistema está dividido en agrupaciones lógicas representando las dependencias existentes entre las mismas.

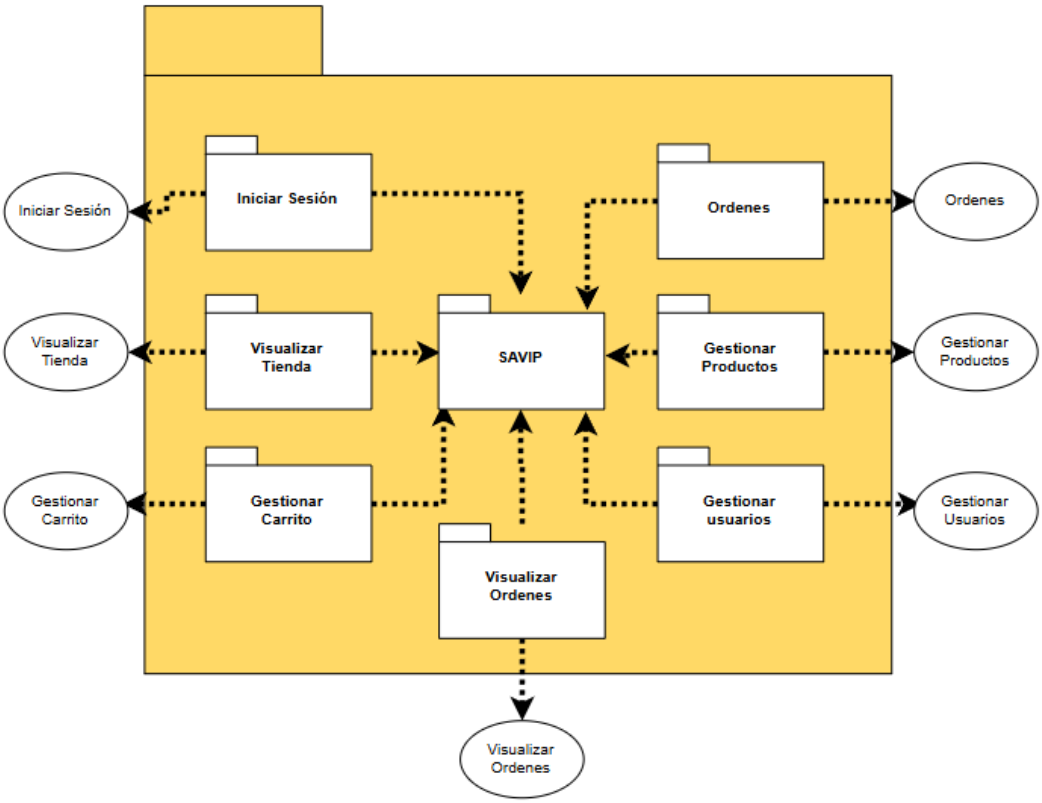


Figura 4.12 Diagrama de paquetes de análisis del sistema SAVIP.

Fuente Elaboración Propia

4.1.2.3. Diagrama de Clases de Análisis

Los diagramas de clases de análisis permiten a través del análisis de los casos de uso del sistema y la identificación de las clases de análisis, realizar el diseño de la estructura del sistema.

- Diagrama de clases de análisis para el caso de iniciar sesión

  El caso de uso iniciar sesión permite a los usuarios del mismo acceder a este mediante un correo único para cada actor y su contraseña, el gestor iniciar sesión es el encargado de verificar que los datos que el usuario introdujo se encuentren previamente

registrados en la base de datos y que estos sean correctos para brindar el acceso que corresponda. En la figura 4.13 se muestra el diagrama de clases de análisis descrito anteriormente.

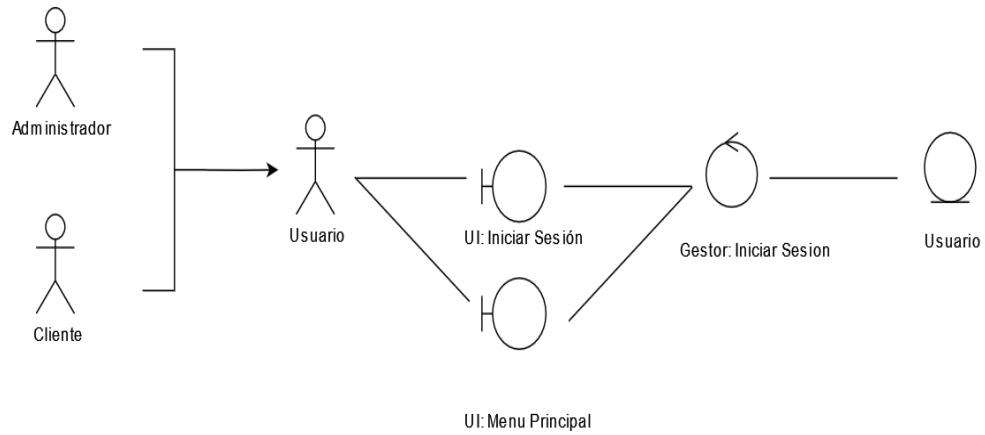


Figura 4.13 Diagrama de análisis del caso de uso Iniciar Sesión.

Fuente: Elaboración Propia.

- Diagrama de clases de análisis para el caso de uso Gestionar Usuarios

El caso de uso gestionar clientes permite al administrador mostrar y editar a los clientes dentro del sistema, el gestor gestionar usuarios es el encargado de mostrar un listado de los clientes almacenados en el sistema a través de la interfaz gestionar clientes, así como activar las interfaces editar y ser a su vez procesada por el gestor correspondiente para posteriormente enviar dicha información a la entidad respectiva. En la figura 4.14 se muestra el diagrama de clases de análisis descrito anteriormente.

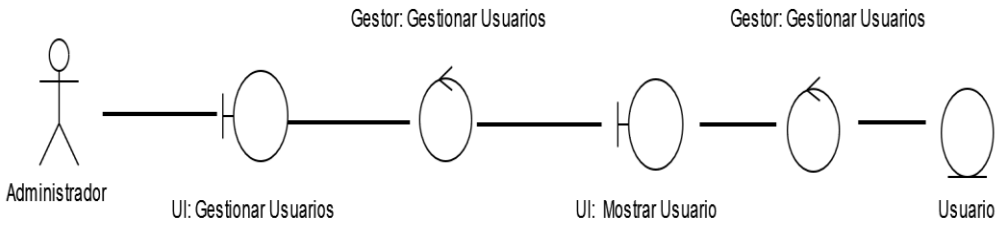


Figura 4.14 Diagrama de clases de análisis del caso de uso Gestionar Usuarios

Fuente Elaboración Propia.

- Diagrama de clases de análisis para el caso de uso Gestionar Producto

  El caso de uso gestionar producto permite al administrador agregar nuevos equipos al sistema y el gestor de nuevo equipo es el encargado de validar la información ingresada por el usuario para posteriormente ser enviada a la entidad respectiva. Así mismo el administrador podrá modificar y eliminar equipos ya ingresados en el sistema según sea el caso deseado. En la figura 4.15 se muestra el diagrama de clases de análisis descrito anteriormente.

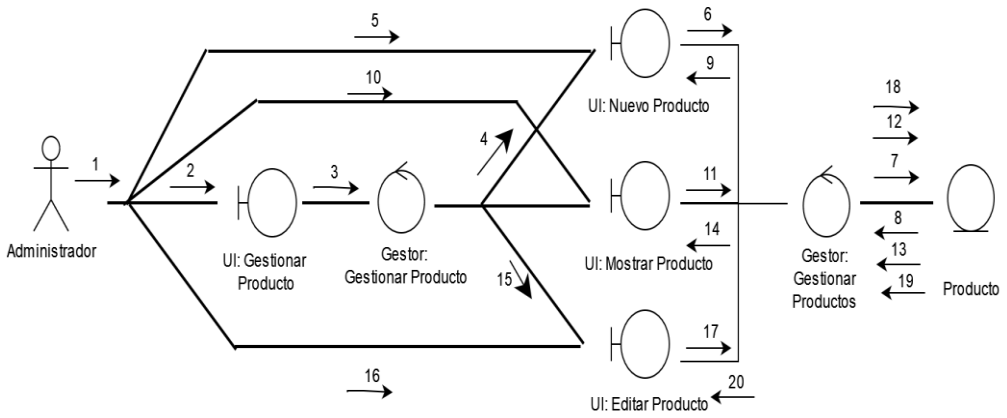


Figura 4.15 Diagrama de clases de análisis para el caso de uso gestionar producto.

Fuente Elaboración propia

- Diagrama de clases de análisis para el caso de uso Visualizar Tienda.

El caso de uso visualizar tienda permite a los usuarios visualizar los productos disponibles a través del llamado a la entidad “Tienda”, además de sus precios correspondientes. El gestor visualizar tienda es el encargado de almacenar los productos seleccionados por el usuario en las entidades “User” e “Productos”. En la figura 4.16 se muestra el diagrama de clases de análisis descrito anteriormente.

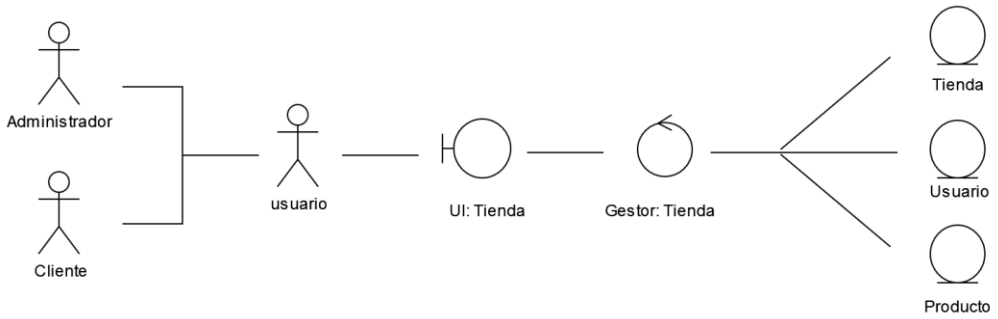


Figura 4.16 Diagrama de clases de análisis para el caso de uso Visualizar Tienda

Fuente: Elaboración Propia.

- Diagrama de clases de análisis para el caso de uso gestionar carrito.

El caso de uso gestionar carrito permite a los usuarios visualizar los productos que hayan añadido con anterioridad a su carrito, esto gracias al llamado a la entidad “Tienda”. Además de los precios correspondientes. El gestor gestionar carrito es el encargado de realizar las operaciones de añadir, restar, eliminar productos y almacenar el resultado final de los productos seleccionado por el usuario en las entidades “User” y “Productos”; por último, se encarga de activar la interfaz de órdenes. En la figura 4.17 se muestra el diagrama de clases de análisis descrito anteriormente.

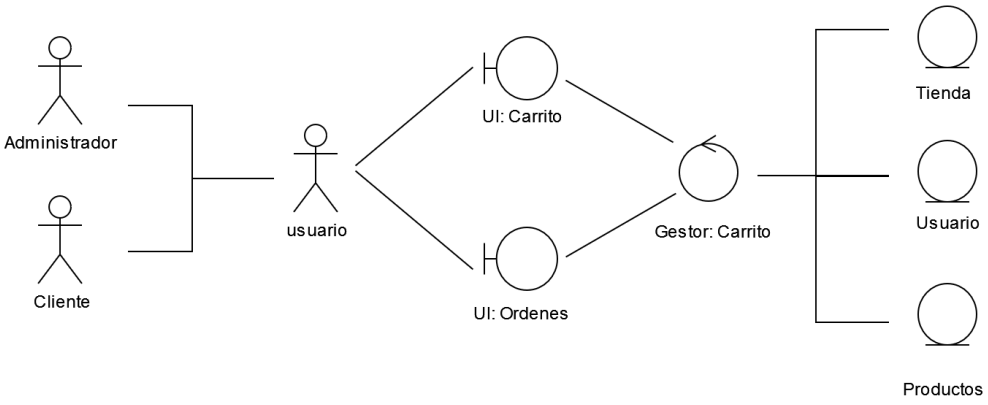


Figura 4.17 Diagrama de clases de análisis para el caso de uso Gestionar Carrito

Fuente: Elaboración Propia.

- Diagrama de clases de análisis para el caso de uso Órdenes.

  El caso de uso gestionar órdenes permite a los usuarios visualizar todas las órdenes que haya llevado a cabo en el sistema, esto gracias al llamado a las entidades “User” y “Factura”. El gestor de órdenes es el encargado de activar la interfaz seleccionada por el usuario. De haber activado la interfaz “Registrar pago” esta permite visualizar los campos que el usuario debe rellenar para llevar a cabo el pago de su orden. Será el gestor de registro el encargado de enviar y resguardar la información en las entidades “User”, “Registro”. En la figura 4.18 se muestra el diagrama de clases de análisis descrito anteriormente.

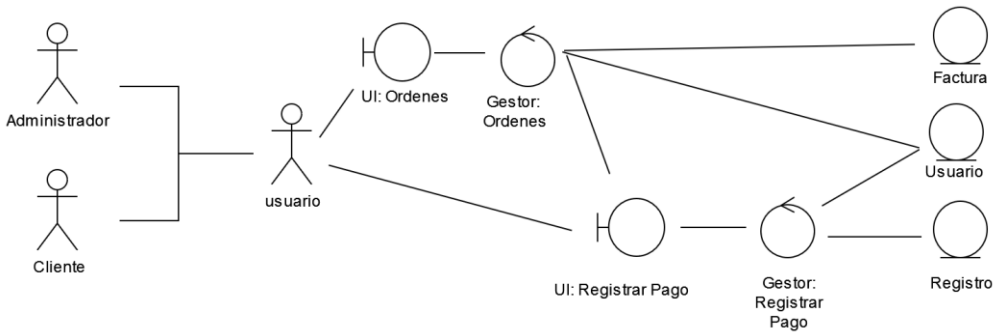


Figura 4.18 Diagrama de clases de análisis para el caso de uso Órdenes

Fuente: Elaboración Propia.

- Diagrama de clases de análisis para el caso de uso Visualizar Órdenes.

  El caso de uso visualizar pagos permite a los usuarios visualizar todas las órdenes de pago llevado a cabo en el sistema, esto gracias al llamado a las entidades “User”, “Factura” y “Orden”. El gestor pagos es el encargado de hacer llamado a las entidades “User”, “Factura” y “Orden” para mostrar el contenido y el estatus de cada pago seleccionado por el usuario, además el gestor órdenes, es el encargado de activar la interfaz órdenes si así lo desea el usuario. En la figura 4.19 se muestra el diagrama de clases de análisis descrito anteriormente.

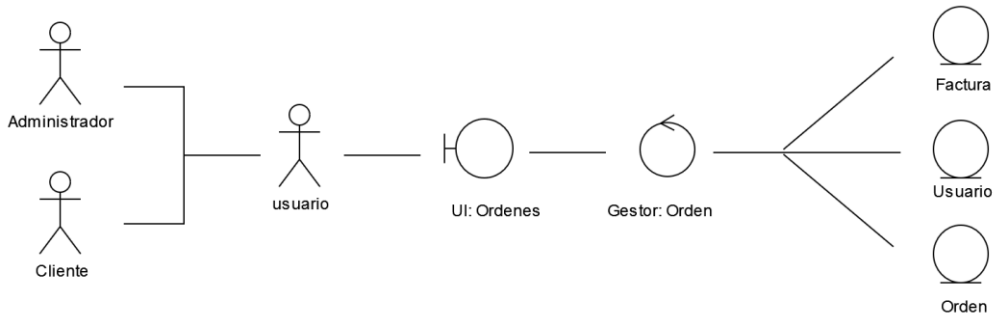


Figura 4.19 Diagrama de clases de análisis para el caso de uso Visualizar Pagos

Fuente: Elaboración Propia.

4.1.2.4. Diagrama de Colaboración

Los diagramas de colaboración muestran las interacciones entre los objetos y las asociaciones que existen entre las clases. Los diagramas de comunicación ofrecen una visión del conjunto de las relaciones entre los objetos a medida que se ejecutan en el software.

- Diagrama de colaboración para el caso de uso Iniciar Sesión.

A continuación, se muestra en la figura 4.20 el diagrama de colaboración para el caso de uso Iniciar sesión junto a su leyenda correspondiente.

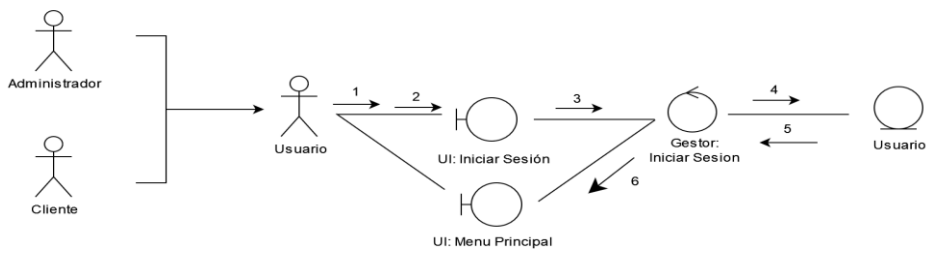


Figura 4.20 Diagrama de colaboración para el caso de uso Iniciar Sesión.

Fuente: Elaboración Propia.

Leyenda:

- Iniciar sesión

1. El actor inicia la interfaz IU iniciar sesión.

2. El actor introduce los datos necesarios para entrar al sistema.

3. Envía los datos al Gestor: iniciar sesión.

4. El gestor verifica la información en la entidad “Usuario”.

5. Devuelve el resultado de la entidad al gestor.

6. Muestra el resultado, permitiendo el acceso a la interfaz del menú principal del sistema o un mensaje de error.

- Diagrama de colaboración para caso de uso Gestionar Clientes

A continuación, se muestra en la figura 4.21 el diagrama de colaboración para el caso de uso gestionar clientes junto a su leyenda correspondiente.

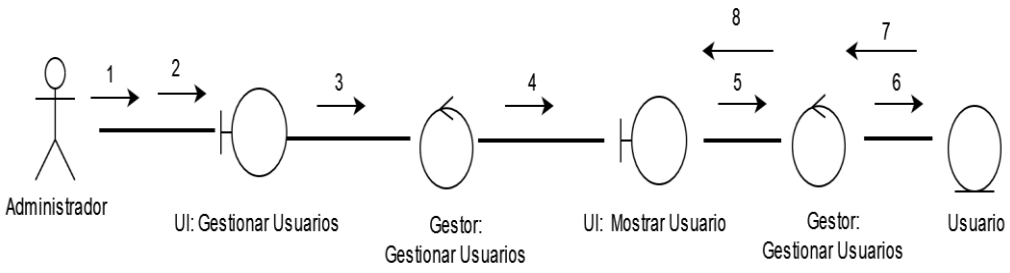


Figura 4.21 Diagrama de colaboración para el caso de uso Gestionar Clientes

Fuente: Elaboración Propia.

Leyenda:

1. El administrador inicia la interfaz “UI: Gestionar usuarios”.

2. En la interfaz selecciona la opción entre las ofrecidas en la interfaz.

3. Envía los datos al “Gestor: Gestionar usuarios”.

4. En la interfaz el administrador ingresa los datos modificados del usuario.

5. Envía los datos al “Gestor: Gestionar usuarios”.

6. El gestor verifica la información en la entidad “usuario”.

7. Devuelve el resultado de la entidad al gestor.

8. El sistema muestra un mensaje de operación exitosa o error según sea el caso.

- Diagrama de colaboración para caso de uso Gestionar Productos

A continuación, se muestra en la figura 4.22 el diagrama de colaboración para el caso de uso gestionar equipo junto a su leyenda correspondiente.

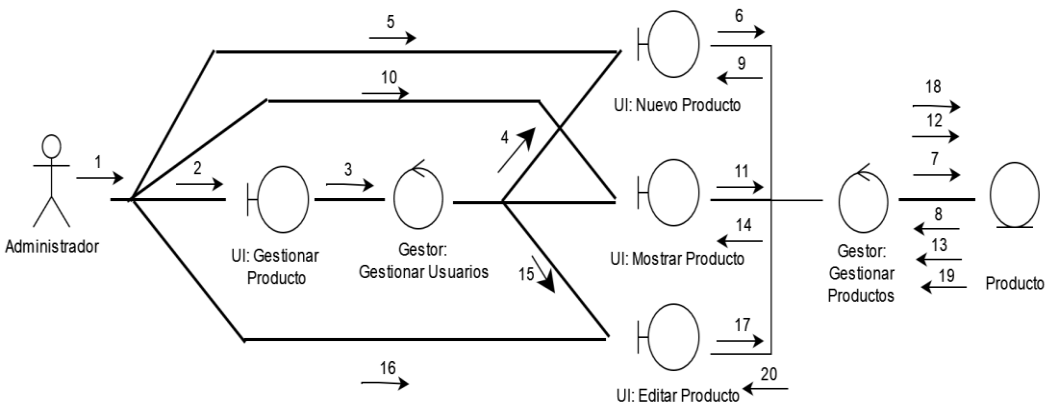


Figura 4.22 Diagrama de colaboración para el caso de uso Gestionar Productos

Fuente: Elaboración Propia.

Leyenda:

-Gestionar Productos

1. El actor inicia la interfaz “UI: Gestionar Producto”.
2. En la interfaz selecciona la opción entre las ofrecidas en la interfaz.
3. Envía los datos al “Gestor: Gestionar Usuarios”.
4. Activa la interfaz “UI: Nuevo Producto”.
5. En la interfaz el presidente ingresa los datos del nuevo equipo.
6. Envía los datos al “Gestor: Gestionar Productos”.
7. El gestor verifica la información en la entidad “Producto”.
8. Devuelve el resultado de la entidad al gestor.

9. El sistema muestra un mensaje de operación exitosa o error según sea el caso.

10. Activa la interfaz “UI: Mostrar Producto”.

11. Envía los datos al “Gestor: Gestionar Productos”.

12. El gestor verifica la información en la entidad “Equipo”.

13. Devuelve el resultado de la entidad al gestor.

14. El sistema muestra los datos del cliente de forma detallada.

15. Activa la interfaz “IU: Editar Producto”.

16. En la interfaz el presidente modifica los datos del cliente.

17. El gestor “Gestor: Gestionar Producto” valida los datos ingresados por el presidente.

18. Se establece comunicación con la entidad “Producto”.

19. Devuelve los datos de la entidad al gestor.

20. El sistema muestra un mensaje de operación exitosa o error según sea el caso.

- Diagrama de colaboración para el caso de uso Visualizar Tienda.

A continuación, se muestra en la figura 4.23 el diagrama de colaboración para el caso de uso Visualizar Tienda junto a su leyenda correspondiente.

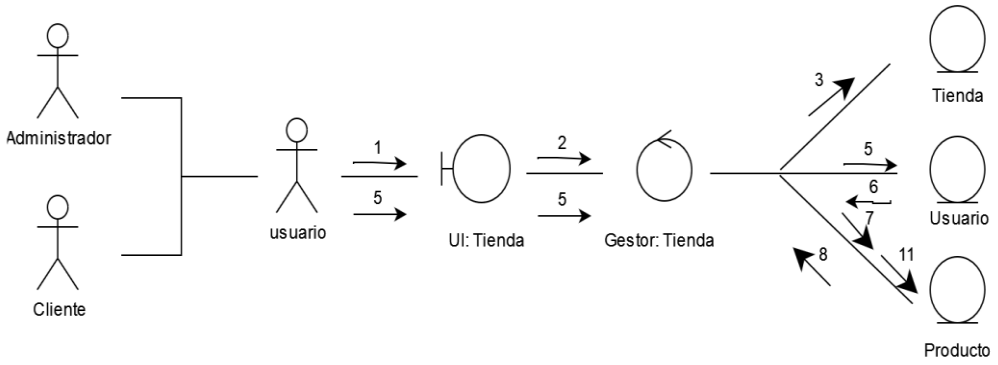


Figura 4.23 Diagrama de colaboración para el caso de uso Visualizar Tienda

Fuente: Elaboración Propia.

Leyenda:

Visualizar Tienda

1. El actor inicia la interfaz IU Tienda.
2. La interfaz solicita la información de los productos de la tienda al gestor: Tienda.
3. El gestor busca la información en la entidad “Tienda”.
4. Devuelve el resultado de la entidad al gestor.
5. El gestor solicita la información del actor a la entidad “User”.
6. Devuelve el resultado de la entidad al gestor.
7. El gestor solicita la información del carrito a la entidad “Producto”
8. Devuelve el resultado de la entidad al gestor.
9. El actor selecciona el producto que desea añadir a su carrito.
10. Envía los datos al “Gestor: Tienda”.

11.El gestor actualiza la información de la entidad “Producto”.

- Diagrama de colaboración para el caso de uso Gestionar Carrito.

A continuación, se muestra en la figura 4.24 el diagrama de colaboración para el caso de uso Gestionar Carrito junto a su leyenda correspondiente.

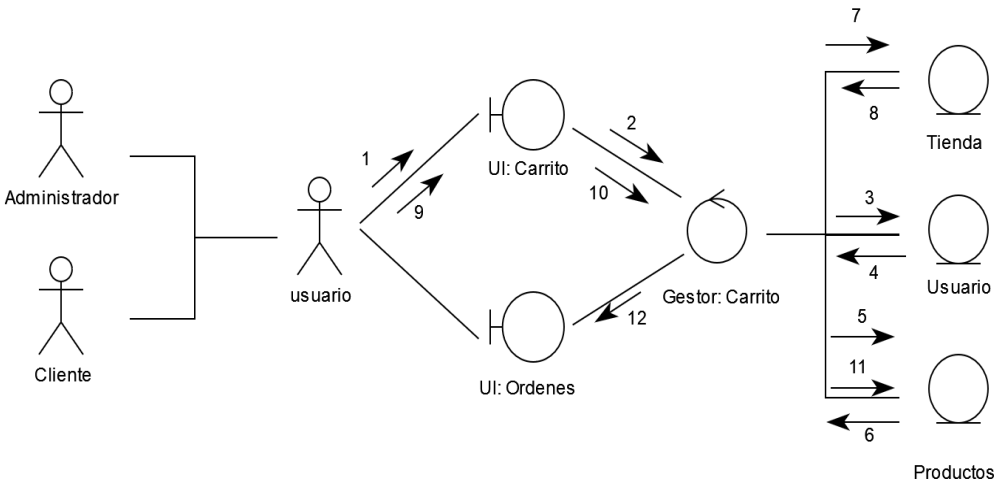


Figura 4.24 Diagrama de colaboración para el caso de uso Gestionar Carrito

Fuente: Elaboración Propia.

Leyenda:

Gestionar Carrito

1. El actor inicia la interfaz IU carrito.
2. La interfaz solicita la información del carrito del actor al “Gestor: Carrito”.

3. El gestor busca la información del actor en la entidad “User”.
4. Devuelve el resultado de la entidad al gestor.
5. El gestor solicita los productos que posee el actor en su carrito, realizando un llamado a la entidad “Producto”.
6. Devuelve el resultado de la entidad al gestor.
7. El gestor solicita la información de los productos asociados al carrito del actor, realizando un llamado a la entidad “Tienda”.
8. Devuelve el resultado de la entidad al gestor.
9. El actor gestiona los productos de su carrito.
10. Envía los datos al “Gestor: Carrito”.
11. El gestor actualiza la información de la entidad “Productos”.
12. Se activa la interfaz IU órdenes.

- Diagrama de colaboración para el caso de uso Gestionar Órdenes.

A continuación, se muestra en la figura 4.25 el diagrama de colaboración para el caso de uso Gestionar Órdenes junto a su leyenda correspondiente.

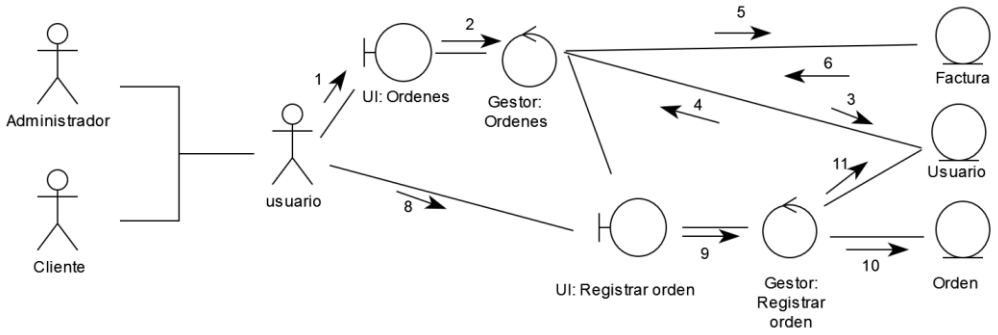


Figura 4.25 Diagrama de colaboración para el caso de uso Gestionar Órdenes.

Fuente: Elaboración Propia.

Leyenda:

Gestionar Órdenes

1. El actor inicia la interfaz IU órdenes.

2. La interfaz solicita la información de las órdenes del actor al “Gestor: Órdenes”.

3. El gestor solicita la información del actor en la entidad “User”.

4. Devuelve el resultado de la entidad.

5. El gestor solicita la información de las órdenes del actor en la entidad “Factura”.

6. Devuelve el resultado de la entidad al gestor.

7. Activa la interfaz “IU: registrar pago”.

8. El actor ingresa la información del pago que desea registrar.

9. Envía los datos introducidos al “Gestor: Registrar Pago”.
10. El gestor almacena la información en la entidad “Orden”.
11. El gestor almacena la información en la entidad “User”.

- Diagrama de colaboración para el caso de uso Visualizar Órdenes.

A continuación, se muestra en la figura 4.26 el diagrama de colaboración para el caso de uso Visualizar pagos junto a su leyenda correspondiente.

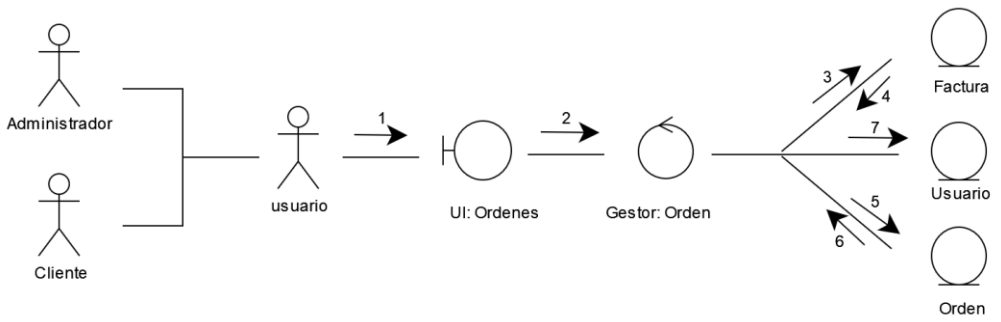


Figura 4.26 Diagrama de colaboración para el caso de uso Visualizar Pagos

Fuente: Elaboración Propia.

Leyenda:

Visualizar pagos

1. El actor inicia la interfaz “IU: pagos”.
2. La interfaz solicita la información de los pagos realizados por el actor al “Gestor: Órdenes”.
3. El gestor solicita la información del actor a la entidad “usuario”.

4. Devuelve el resultado de la entidad.
5. El gestor solicita la información de cada pago realizado por el actor y su estatus a la entidad “usuario”.
6. Devuelve el resultado de la entidad al gestor.
7. El gestor solicita la información del contenido de productos que posee cada pago del actor a la entidad “Factura”.
8. Devuelve el resultado de la entidad al gestor.

4.1.3 Flujo de trabajo de diseño

Luego de realizada la identificación y estudio de los requisitos funcionales, se procede al diseño de los mismos, tomando en cuenta requisitos adicionales para así desarrollar la arquitectura del sistema. La siguiente etapa arrojará el diseño de la arquitectura, base de datos e interfaz.

4.1.3.1. Arquitectura Candidata

Es la encargada de identificar y determinar los elementos y componentes tecnológicos necesarios para satisfacer los requisitos y objetivos del sistema. En la figura 4.27 se muestra la arquitectura candidata para el sistema SAVIP.

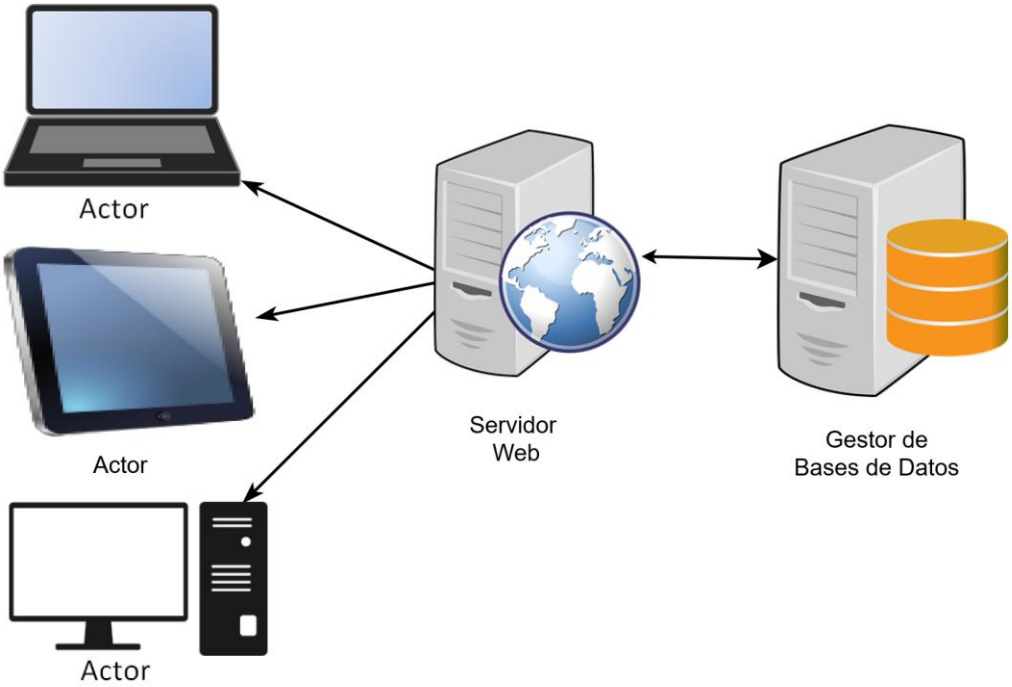


Figura 4.27 Arquitectura candidata para el sistema SAVIP.
Fuente Elaboración Propia

4.1.3.2 Prototipo de Interfaz de Usuario

En la Figura 4.28 se puede observar el prototipo de la interfaz Iniciar Sesión del Sistema SAVIP.

Figura 4.28 Interfaz Iniciar Sesión SAVIP.
Fuente Elaboración Propia

4.1.4 Evaluación de la fase de inicio

El desarrollo de la fase de inicio permitió la identificación del sistema mediante el análisis de las interacciones entre las partes del mismo, además de la diagramación del modelo de dominio del mismo, en el cual se plasmaron las clases más importantes del contexto del sistema, asimismo fue descrito el funcionamiento de la empresa en cuanto a la entrada, manejo, venta y facturación de productos con la intención de abordar todas las actividades fundamentales de la organización en sí y poder determinar los requisitos funcionales y no funcionales a los que estará sujeto el sistema, dando origen a los casos de usos a lo que será sometido el mismo; además de esto se consiguió identificar los posibles riesgos a los que podría estar expuesto.

La fase de inicio, fue la base fundamental del diseño del sistema SAVIP, ya que a través de la elaboración de los diagramas de clase de análisis y colaboración se obtuvo un bosquejo del aspecto funcional que debía tener el software a desarrollar para así cumplir con los requerimientos ya previamente establecidos.

4.2 Fase de Elaboración

Durante la siguiente etapa se establece la línea de la arquitectura del sistema iniciada en la fase anterior, se busca recopilar la información necesaria para determinar los requisitos que pudieron haber quedado pendientes, así como establecer una base sólida para la siguiente etapa de construcción, lo cual incluye además de la arquitectura y la base de datos.

4.2.1. Flujo de Trabajo de Requisito

El flujo de trabajo de requisitos es el proceso de identificar, documentar, revisar y gestionar los requisitos funcionales y no funcionales que deben ser satisfechos por el sistema. El objetivo es garantizar que los requisitos del sistema estén claramente definidos y correctamente implementados en el diseño y desarrollo del sistema de información.

4.2.2 Flujo de trabajo de diseño

El flujo de trabajo de requisitos es el proceso de identificar, documentar, revisar y gestionar los requisitos funcionales y no funcionales que deben ser satisfechos por el sistema. El objetivo es garantizar que los requisitos del sistema estén claramente definidos y correctamente implementados en el diseño y desarrollo del sistema de información.

4.2.2.1. Identificación de las clases

Una vez identificados y descritos los casos de uso del sistema SAVIP, se procedió a realizar las tablas y diagramas de análisis correspondiente a cada caso de uso.

Clases interfaz

La utilidad de las clases de interfaz es que estas permiten la comunicación del usuario con el sistema SAVIP, dichas clases se encuentran descritas a continuación en la tabla 4.3.

Clase	Descripción
UI: Iniciar Sesión	Permite a los actores acceder al sistema mediante su cuenta ingresando correo electrónico y contraseña.
UI: Menu principal	Ofrece a los usuarios las opciones disponibles según si inició o no, previamente en el sistema.
IU: Perfil	Permite a los actores modificar información personal, además muestra opciones adicionales según el rol de actor.
IU: Tienda	Permite visualizar el listado de todos los productos registrados en el sistema, además de sus características y la opción de añadir cualquier producto al carrito
IU: Órdenes	Permite visualizar el listado de todas las órdenes realizadas por un actor en general. Además da la opción de visualizar el contenido de cada una de las órdenes.

IU: Carrito	Permite visualizar el listado de todos los productos y/o plan de suscripción añadido por un actor en específico, además de permitir añadir, modificar o eliminarlos del carrito. Finalmente muestra el costo total de la orden.
IU: Gestionar usuarios	Permite visualizar al administrador un listado de todos los usuarios que una vez han iniciado sesión en el sistema, asimismo, ofrece opciones para editar el rol del actor.
IU: Gestionar productos	Permite visualizar al administrador un listado de todos los productos en el sistema, asimismo, ofrece las opciones de agregar, editar y eliminar productos del sistema.
IU: Mostrar Usuario	Permite visualizar los datos de un usuario en específico, mostrando toda la información almacenada sobre este en el sistema.
IU: Agregar Producto	Permite ingresar los datos de nuevos productos al sistema.
IU: Mostrar Producto	Permite visualizar los datos de un producto en específico, mostrando toda la información almacenada de dicho equipo en el sistema.
IU: Editar Equipo	Permite modificar los datos de los productos ya guardados en el sistema.

Tabla 4.3 Clases Interfaces

Fuente: Elaboración Propia

Clases de control

Estas clases de control son el establecimiento del manejo de las funciones del software en desarrollo. Estas clases se encuentran descritas en la tabla 4.4.

Clase	Descripción
Gestor: Iniciar Sesión	Recibe los datos ingresados por el usuario para el inicio de sesión (correo y contraseña), busca la información en la base de datos y concede o niega el acceso al sistema.
Gestor: Tienda	Realiza consultas a la entidad “Tienda” y regresa la lista de los productos disponibles. Además almacena la información de los productos seleccionados por el usuario en las entidades “User” e “productos”.
Gestor: Menú principal	Recibe el estatus de la sesión de los usuarios en el sistema, asimismo, consulta a la entidad “User” el rol del usuario y se encarga de realizar operaciones de redirección a diferentes interfaces.
Gestor: Perfil	Recibe la información del usuario con la sesión activa en función al rol que este posea en la entidad “User”, busca la información en la base de datos y da acceso a la información personal del mismo.
Gestor: Menú	Realiza consultas a la entidad “Tienda” y regresa la lista de los productos disponibles. Además almacena la información de los productos seleccionados por el usuario en las entidades “User” e “productos”.
Gestor: Órdenes	Recibe los datos suministrados por el usuario y almacena la información en la entidad “User”, “Registro” y “Factura”.
Gestor: Carrito	Realiza consultas a las entidades “productos”, “User” y “Tienda”, de esta manera regresa la lista de los productos elegidos por el usuario, también realiza la operaciones de añadir, restar y eliminar productos; por último, redirección hacia la interfaz de órdenes.
Gestor: Gestionar Productos	Realiza consultas a la entidad productos y regresa listada parte de la información almacenada en la misma, también realiza las operaciones de registrar, mostrar, modificar y eliminar equipos almacenados en el sistema.
Gestor: Gestionar Usuarios	Realiza consultas a la entidad usuario y regresa listada parte de la información almacenada en la misma, también realiza las operaciones para modificar los roles de los usuarios almacenados en el sistema.

Tabla 4.4 Clases de Control

Fuente: Elaboración Propia

Clases de entidad

Estas clases son el establecimiento de las bases para el diseño de la base de datos para el software y su desarrollo. Dichas clases serán explicadas en la tabla 4.5.

Clase	Descripción
Usuario	Constituye la información asociada a los usuarios del sistema, su rol dentro del mismo y los permisos que poseen los usuarios
Producto	Constituye la información asociada a los productos de la empresa con los cuales presta sus servicios.
Roles	Constituye la información asociada a los roles y permisos de los usuarios en el sistema.
Tienda	Constituye la información asociada a los productos y planes del sistema.
Factura	Constituye a la información asociada al número de orden de los pagos realizados por los usuarios del sistema.

Tabla 4.5 Clases Entidades

Fuente: Elaboración Propia

4.2.3 Flujo de trabajo de diseño

Durante el desarrollo de esta fase se procede a definir los componentes que van a formar parte de la base del sistema, es decir, la descripción de forma robusta que permita el desarrollo adecuado de los modelos y la integración de los mismos. A continuación, se mostrará el diagrama de diseño del sistema y su arquitectura, además modelado y diseño de base de datos.

4.2.3.1 Arquitectura del sistema

La arquitectura del sistema se establece de manera sólida y estable de forma tal que soporte los requisitos analizados previamente. Esta arquitectura se ve representada por media de un conjunto de capas expuestas en la Figura 4.29, detalladas a continuación:

- La capa específica de la aplicación comprende los paquetes: Iniciar Sesión, Gestionar Clientes, Gestionar productos, Gestionar Usuarios, Gestionar órdenes, Generar productos y Visualizar productos. Estos paquetes se interconectan entre sí y con la capa general de la aplicación.

- La capa general de la aplicación contempla el inicio de Sesión de La Aplicación SAVIP, la cual se conecta con la capa intermedia.

- La Capa intermedia, comprende el comportamiento lógico del sistema. En esta capa se cuenta con el Servidor de Node para el manejo estático y dinámico del entorno, El lenguaje JavaScript que ejecuta los scripts que contienen la funcionalidad del sistema, el Sistema manejador de Base de Datos PostgreSQL que contiene la base de datos del sistema y la funcionalidad para establecer conexión con ésta y el navegador web, por el cual corren JavaScript, (encargado de otorgar animaciones e interacciones al sistema), HTML (encargado de la estructura de la página) y CSS (encargado del aspecto visual del sistema).

- La capa de software del sistema contempla el sistema operativo en el cual se ejecuta el navegador web para correr la aplicación a través del protocolo HTTPS/IP.

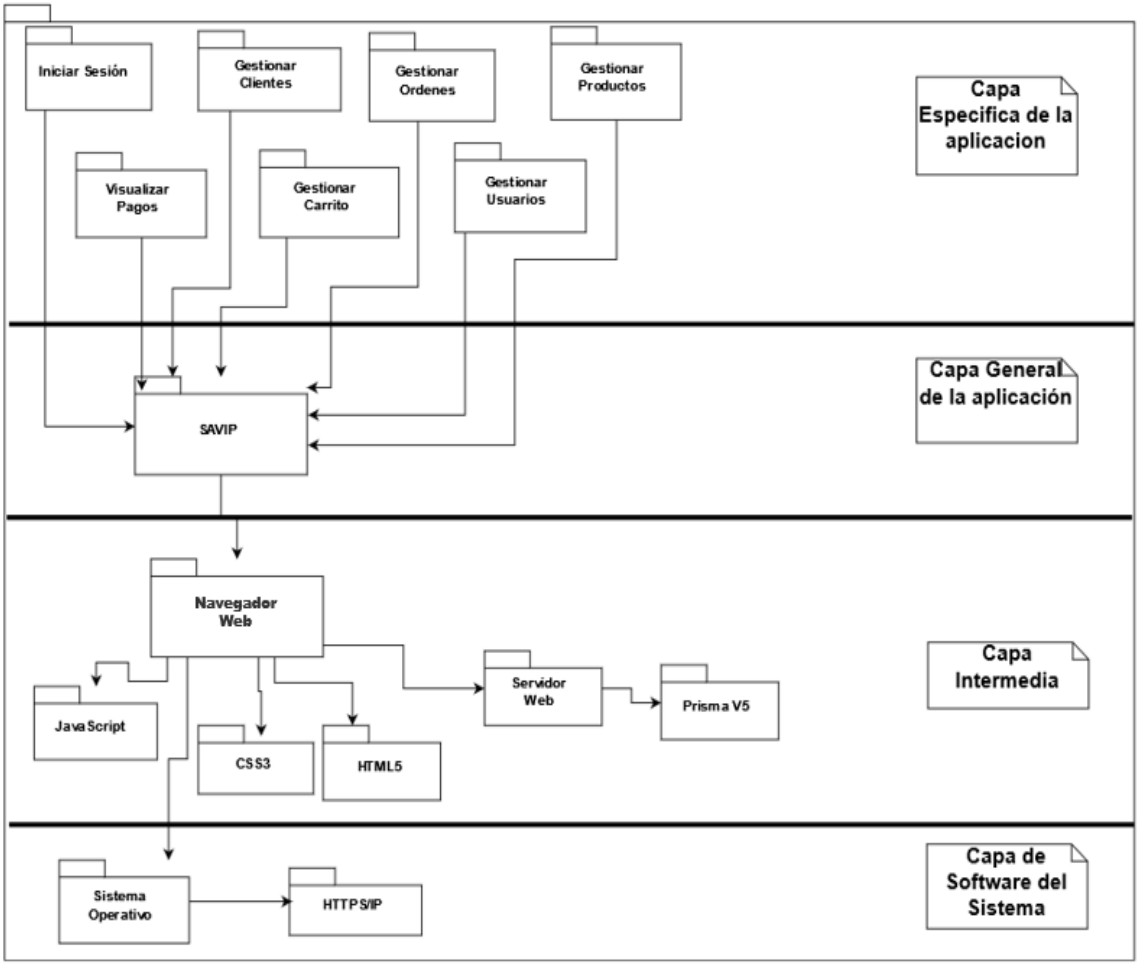


Figura 4.29 Diagrama de capas del software SAVIP.
Fuente Elaboración Propia

4.2.3.2 Diagrama de Componentes

El diagrama de componentes se utiliza para modelar la vista estática de un sistema, describiendo los elementos físicos y sus relaciones. Pueden ser simples archivos, paquetes, bibliotecas, entre otros. En la figura 4.30 se muestra el diagrama de componentes del sistema SAVIP, el cual evidencia la dependencia existente entre los componentes del mismo partiendo de la información obtenida en las fases anteriores.

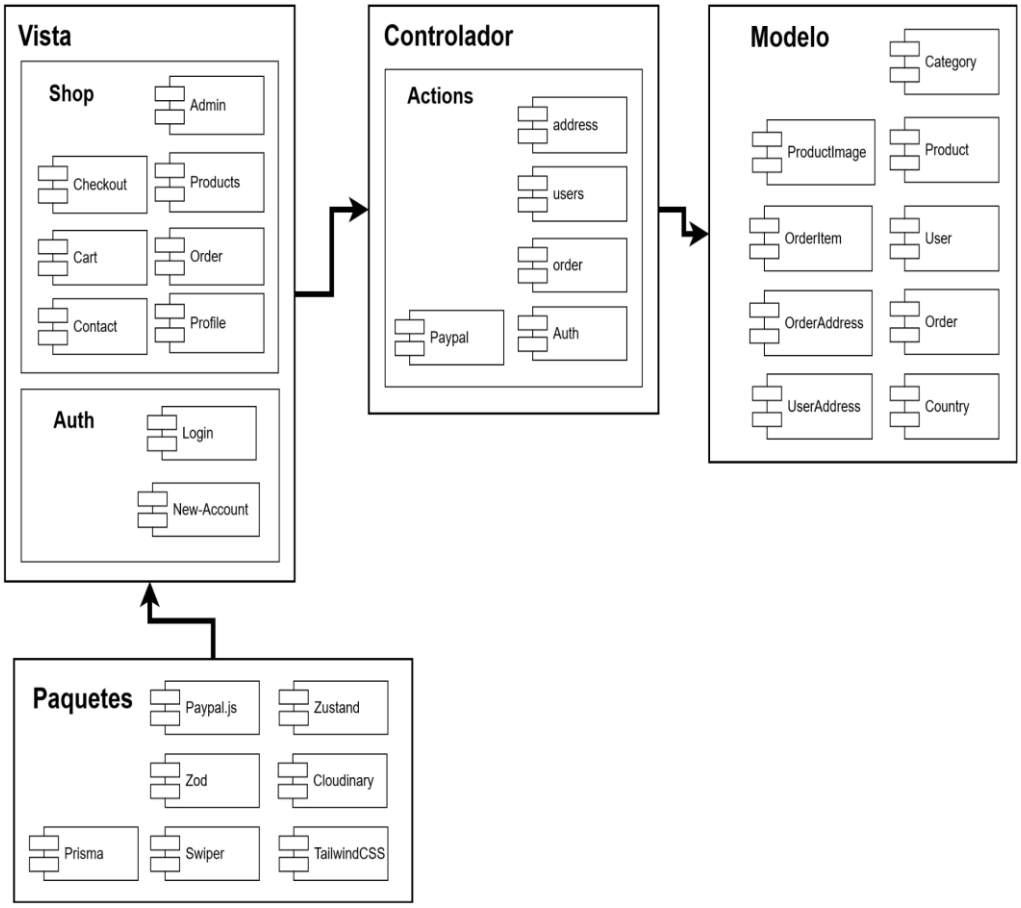


Figura 4.30 Diagrama de componentes del sistema SAVIP.
Fuente Elaboración propia.

4.2.4 Diagrama de Despliegue

El diagrama de despliegue permite detallar y comprender la infraestructura y las configuraciones de red donde se despliega el sistema. A continuación, en la figura 4.31 se observa el diagrama de despliegue para el sistema SAVIP.

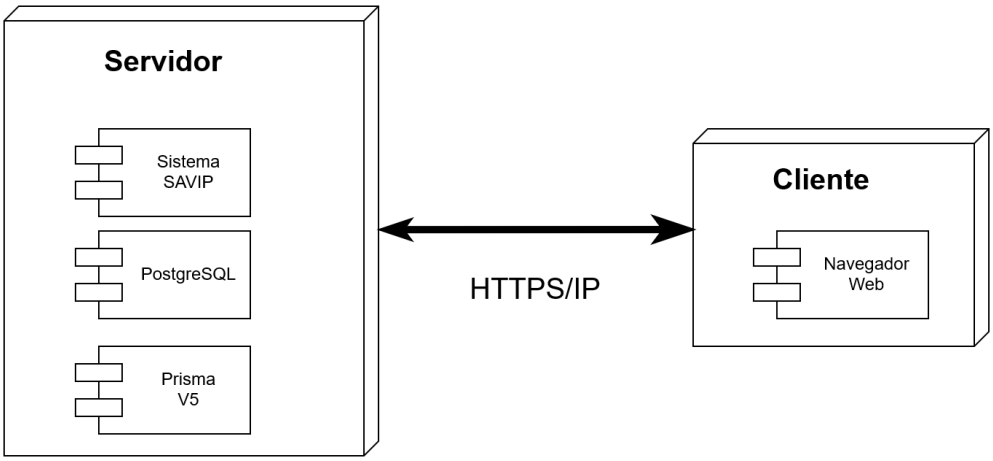


Figura 4.31 Diagrama de despliegue del sistema SAVIP.
Fuente Elaboración propia.

4.2.5 Diseño de la Base de Datos

En este diagrama representado en la figura 4.32 se pueden observar todas las entidades que forman parte del sistema SAVIP, dentro de las cuales se pueden mencionar a la entidad “Usuario”, “Categorías”, “User”,”ordenes”, “Productos”.

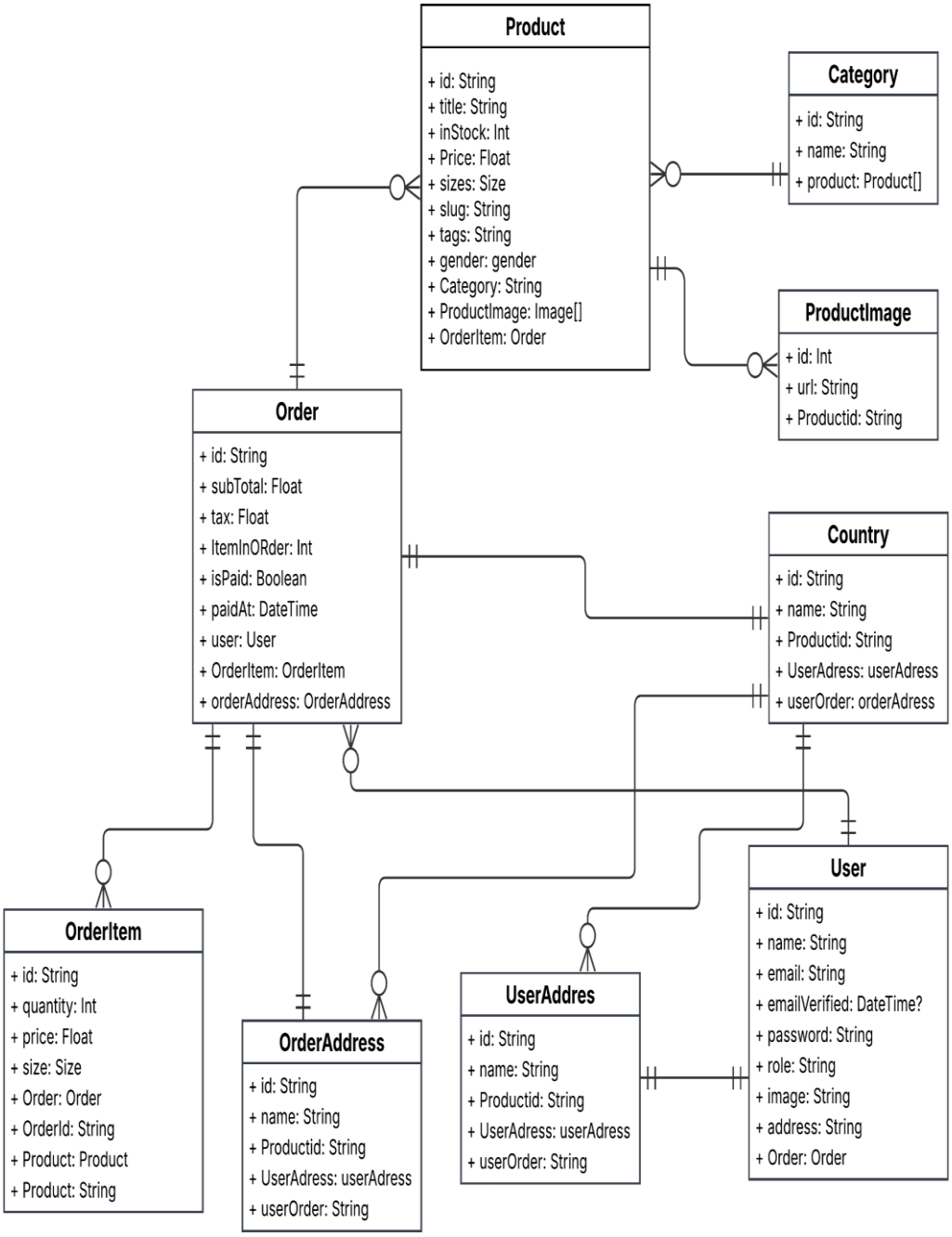


Figura 4.32 Diagrama relacional de base de datos.

Fuente Elaboración propia.

En la Tabla 4.6 se puede observar la información perteneciente a la entidad “User”, en la cual se almacenan los datos de los clientes con acceso al sistema

Campo	Tipo de Dato	Descripción
id	String	Identificador unico del cliente
name	String	Nombre del cliente.
email	String	Correo electrónico del cliente.
emailVerified	DateTime	Identificador para saber cuándo se verificó el email
password	String	Contraseña asociada para ingresar en la cuenta
role	Role	Rol que posee el cliente (Usuario, Administrador)
image	String	Imagen del cliente.
address	UserAddress	Atributo relacionado a la dirección. Indica en qué ciudad vive el cliente.
Order	Order[]	Arreglo de órdenes que ha creado el usuario

Tabla 4.6 Entidad User

Fuente: Elaboración Propia.

En la Tabla 4.7 se puede observar la información perteneciente a la entidad “Category”, en la cual se almacenan los datos de las categorías de los productos.

Campo	Tipo de Dato	Descripción
id	String	Identificador unico del cliente
name	String	nombre de categoría
Producto	Product[]	arreglo de productos asociado a una categoría

Tabla 4.7 Entidad Tienda

Fuente: Elaboración Propia.

En la Tabla 4.8 se puede observar la información perteneciente a la entidad “Product”, en la cual se almacenan los datos de las categorías de los productos.

Campo	Tipo de Dato	Descripción
id	String	Identificador único del producto
title	String	Nombre del producto
description	String	descripción del producto
inStock	Int	cantidad del producto en stock
price	Float	precio del producto
sizes	String	tallas del producto
slug	String	nombre del producto para el URL

tags	String[]	palabras clave asignada al producto
gender	String	Genero del producto (hombres,mujeres,unisex)
category	category	categoría del producto
ProductImage	ProductImage[]	Imágenes del producto
OrderItem	OrderItem[]	Orden relacionada al producto si se usó para un compra

Tabla 4.8 Entidad Product

Fuente: Elaboración Propia.

En la Tabla 4.9 se puede observar la información perteneciente a la entidad “ProductImage”, en la cual se almacenan las órdenes de compra de cada cliente.

Campo	Tipo de Dato	Descripción
id	Int	Identificador unico del cliente
url	string	Url donde se almacena las imágenes de los producto
product	Product	producto que está relacionado a este imagen

Tabla 4.9 Entidad ProductImage

Fuente: Elaboración Propia.

En la Tabla 4.10 se puede observar la información perteneciente a la entidad “Country”, en la cual se almacenan los productos que posea el cliente en su carrito de compra al confirmar el pago.

Campo	Tipo de Dato	Descripción
id	String	Identificador único del país
name	String	Nombre del país
userAddress	UserAddress[]	Relación a varios usuarios que son del país en concreto
OrderAddress	OrderAddress[]	Relación a varios órdenes que son del país concreto

Tabla 4.10 Entidad Country

Fuente: Elaboración Propia.

En la Tabla 4.11 se puede observar la información perteneciente a la entidad “UserAddress”, en la cual se almacenan las direcciones asociados a los clientes.

Campo	Tipo de Dato	Descripción
id	String	Identificador único de la dirección
firstName	String	Primer nombre del usuario de esta dirección
lastName	String	Apellido del usuario de esta dirección
address	String	dirección relacionada al usuario
address2	String	segunda dirección relacionada al usuario
postalCode	String	Código postal de de direccion del usuario
phone	String	Número telefónico del usuario
city	String	Ciudad de la dirección de su usuario
country	country	Relación con la tabla Country
user	user	Relación con la tabla de user

Tabla 4.11 Entidad UserAddress
Fuente: Elaboración Propia.

En la Tabla 4.12 se puede observar la información perteneciente a la entidad “Order”. En el cual se almacenan los datos relacionados con las órdenes

Campo	Tipo de Dato	Descripción
id	String	Identificador único de la orden
subTotal	Float	Total del precio sin sumar el impuesto
tax	Float	Tasa del impuesto
itemsInOrder	Int	cantidad de productos dentro de la orden
isPaid	Boolean	Booleano donde verifica si está pagado o no la orden
paidAt	DateTime	fecha almacenada cuando se ha pagado la orden creada
User	User	Relación al usuario que creó la orden
OrderItem	orderItem	arreglo
transaccionId	String	Identificador de la transaccion

Tabla 4.12 Entidad Order

Fuente: Elaboración Propia.

En la Tabla 4.13 se puede observar la información perteneciente a la entidad “OrderItem”. En el cual se almacenan los datos relacionados con los productos relacionados a una orden

Campo	Tipo de Dato	Descripción
id	String	Identificador único del producto dentro de la orden
quantity	Int	Cantidad del producto
price	Float	Precio del producto
size	String	Talla del producto
order	Order	Orden relacionada al producto
product	Product	Producto relacionado

Tabla 4.13 Entidad OrderItem

Fuente: Elaboración Propia.

En la Tabla 4.14 se puede observar la información perteneciente a la entidad “OrdernAddress”, en la cual se almacenan los datos relacionados a la dirección de la orden.

Campo	Tipo de Dato	Descripción
id	String	Identificador único de la dirección
firstName	String	Primer nombre del usuario de esta dirección
lastName	String	Apellido del usuario de esta dirección
address	String	dirección relacionada al usuario
address2	String	segunda dirección relacionada al usuario
postalCode	String	Código postal de de direccion del usuario
phone	String	Número telefónico del usuario
Order	Order	Relación con la tabla de ordenes
Country	Country	relación con la tabla de country

Tabla 4.14 Entidad OrdernAddress

Fuente: Elaboración Propia

4.2.6 Flujo de trabajo de implementación

Una vez llevado a cabo el diseño técnico del sistema, el flujo de trabajo de implementación se encarga de convertir el sistema propuesto en una solución operativa y funcional. Este flujo de trabajo implica la

escritura de código, la configuración de sistemas, la integración de componentes, pruebas y puesta en marcha del sistema, además se encarga de garantizar que el sistema se implemente de manera efectiva y sin problemas, de esta manera, cumpliendo con los requisitos del negocio y tecnologías establecidos.

4.2.6.1 Interfaz Principal de SAVIP

El diseño de las interfaces gráficas del sistema de información SAVIP fue realizado con el fin de proporcionar diseños consistentes, amigables e intuitivos. Cada una de las interfaces será descrita a continuación:

- **Interfaz de “Ingresar”**

Esta interfaz es la encargada de gestionar el perfil del usuario, permitiéndole identificar si es cliente o administrador. En la figura 4.33 se puede apreciar la interfaz de Ingresar.

Figura 4.33. Interfaz de “Perfil”

Fuente: Elaboración Propia

- **Interfaz de “Tienda”**

Esta interfaz es la encargada de mostrarle al usuario todos los productos del gimnasio que se encuentren a la venta, en ella se puede visualizar una pequeña imagen de los productos, su precio y una breve descripción de los mismos. A su vez, cada producto tiene la opción de “Agregar al carrito” (permite agregar el producto al carrito de compras) y la opción “Ver más” (redirige al usuario a una pestaña específica del producto donde se muestra la información del producto). En la figura 4.34 se muestra la interfaz Tienda.

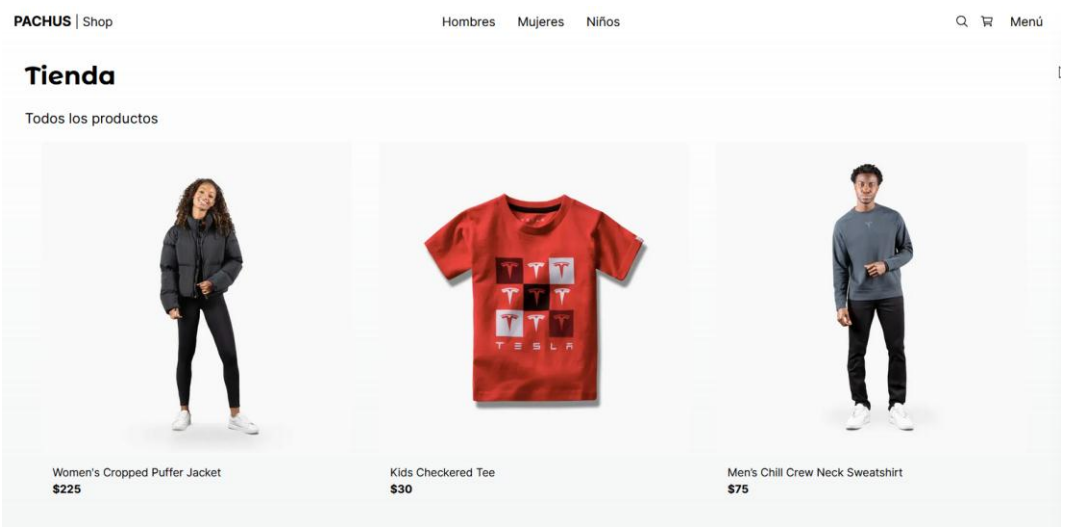


Figura 4.34 Interfaz de “Perfil”

Fuente: Elaboración Propia

4.2.7 Evaluación de la fase de elaboración

Durante el desarrollo de esta fase del proceso unificado de desarrollo de software se continuó con la diagramación necesaria mediante el diagrama de clases de diseño, el cual permite visualizar de manera panorámica toda la estructura del sistema incluyendo la base de datos pertinente a este. De igual forma se presenta la propuesta y diseño de una

arquitectura robusta, estable y candidata a ser implementada en las fases venideras. Se logró establecer una visión ampliada del sistema, la viabilidad de todos los requisitos levantados en la fase de inicio y el diseño del prototipo de la interfaz de acceso al sistema.

4.3 Fase de Construcción

La fase de construcción se centra en la codificación y creación de los diferentes módulos y componentes del sistema de información. Además, se realizan pruebas continuas para garantizar que el software cumple con los requisitos funcionales y no funcionales definidos previamente. La fase de construcción es crucial para asegurar que el sistema de información se desarrolle de manera efectiva y eficiente.

4.3.1 Flujo de trabajo de implementación

Se procede a implementar lo diseñado mediante el desarrollo de componentes, codificación del software, relación con la base de datos, integración de módulo y funcionalidad del sistema, estableciendo así la estructura del sistema en el código.

- Interfaz de “Producto”

En la figura 4.35 se observa la interfaz destinada a mostrar los productos de manera individual por el usuario, para sí decido el usuario agregarlo al carrito.

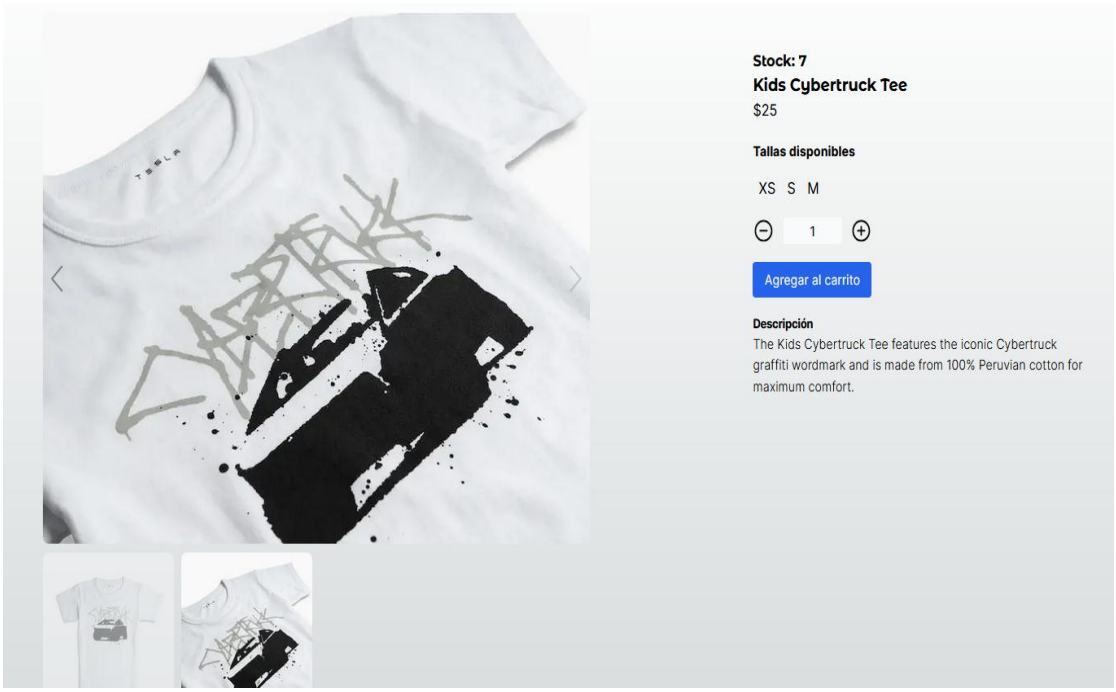
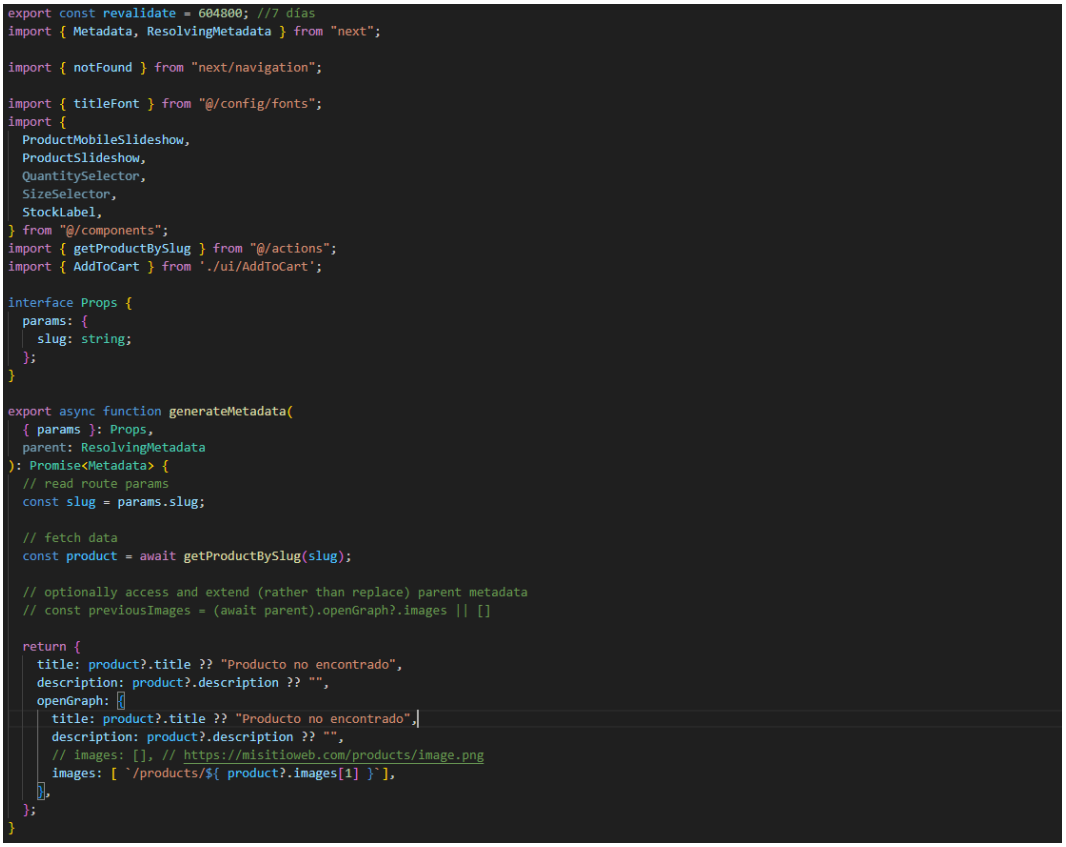


Figura 4.35 Interfaz de “Productos”

Fuente Elaboración Propia



```tsx
export const revalidate = 604800; //7 días
import { Metadata, ResolvingMetadata } from "next";

import { notFound } from "next/navigation";

import { titleFont } from "@/config/fonts";
import {
  ProductMobileSlideshow,
  ProductSlideshow,
  QuantitySelector,
  SizeSelector,
  StockLabel,
} from "@/components";
import { getProductBySlug } from "@/actions";
import { AddToCart } from './ui/AddToCart';

interface Props {
  params: {
    slug: string;
  };
}

export async function generateMetadata(
  { params }: Props,
  parent: ResolvingMetadata
): Promise<Metadata> {
  // read route params
  const slug = params.slug;

  // fetch data
  const product = await getProductBySlug(slug);

  // optionally access and extend (rather than replace) parent metadata
  // const previousImages = (await parent).openGraph?.images || []

  return {
    title: product?.title ?? "Producto no encontrado",
    description: product?.description ?? "",
    openGraph: {
      title: product?.title ?? "Producto no encontrado",
      description: product?.description ?? "",
      // images: [], // https://misitioweb.com/products/image.png
      images: [ `/products/${ product?.images[1] }`],
    },
  };
}
```

Figura 4.36 Código fuente de la Interfaz “Producto”

Fuente Elaboración Propia

```tsx
export default async function ProductBySlugPage({ params }: Props) {
  const { slug } = params;
  const product = await getProductBySlug(slug);

  if (!product) {
    notFound();
  }

  return (
    <div className="h-full mt-10 mb-10 mx-5 grid grid-cols-1 md:grid-cols-3 gap-3">
      {/* Slideshow */}
      <div className="col-span-1 md:col-span-2 ">
        {/* Mobile Slideshow */}
        <ProductMobileSlideshow
          title={product.title}
          images={product.images}
          className="block md:hidden"
        />

        {/* Desktop Slideshow */}
        <ProductSlideshow
          title={product.title}
          images={product.images}
          className="hidden md:block "
        />
      </div>

      {/* Detalles */}
      <div className="col-span-1 px-10 lg:mt-10">
        <StockLabel slug={product.slug} />

        <h1 className={` ${titleFont.className} antialiased font-bold text-xl`}>
          {product.title}
        </h1>

        <p className="text-lg mb-5">${product.price}</p>

        <AddToCart product={ product } />

        {/* Descripción */}
        <h3 className="font-bold text-sm">Descripción</h3>
        <p className="font-light">{product.description}</p>
      </div>
    </div>
  );
}
```

Figura 4.37 Código fuente de la Interfaz “Producto”

Fuente Elaboración Propia

- Interfaz de “Carrito”

En la figura 4.38, se observa la interfaz destinada a mostrar los productos de manera individual por el usuario, en el caso del usuario decida decida agregar al carrito.

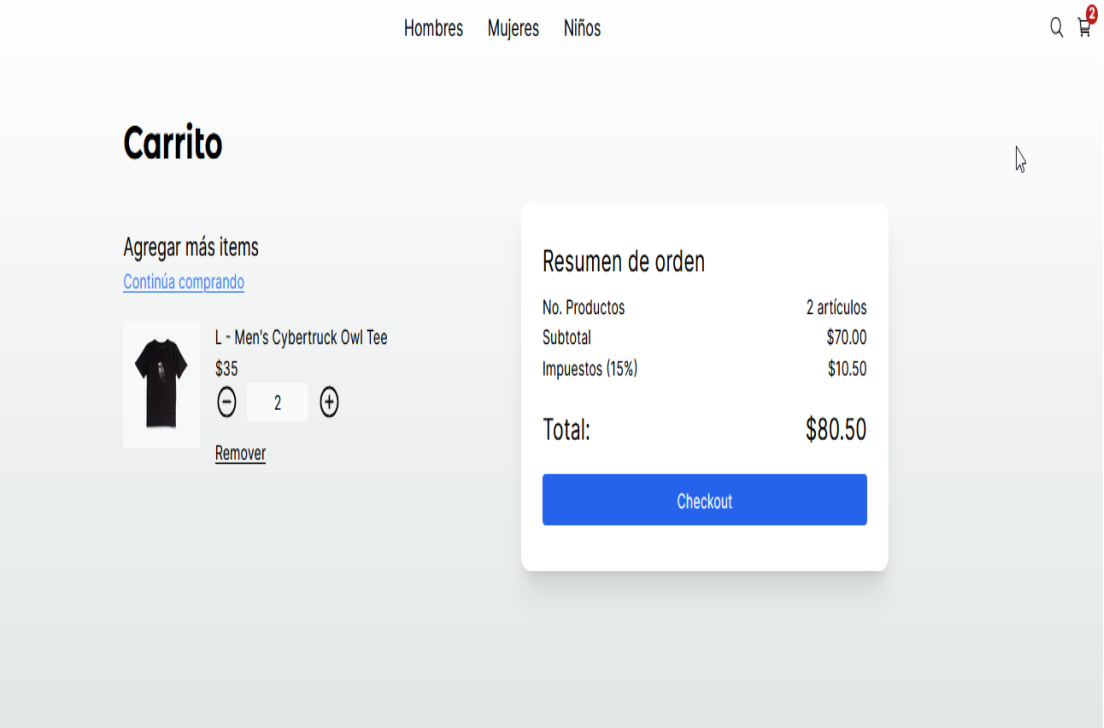
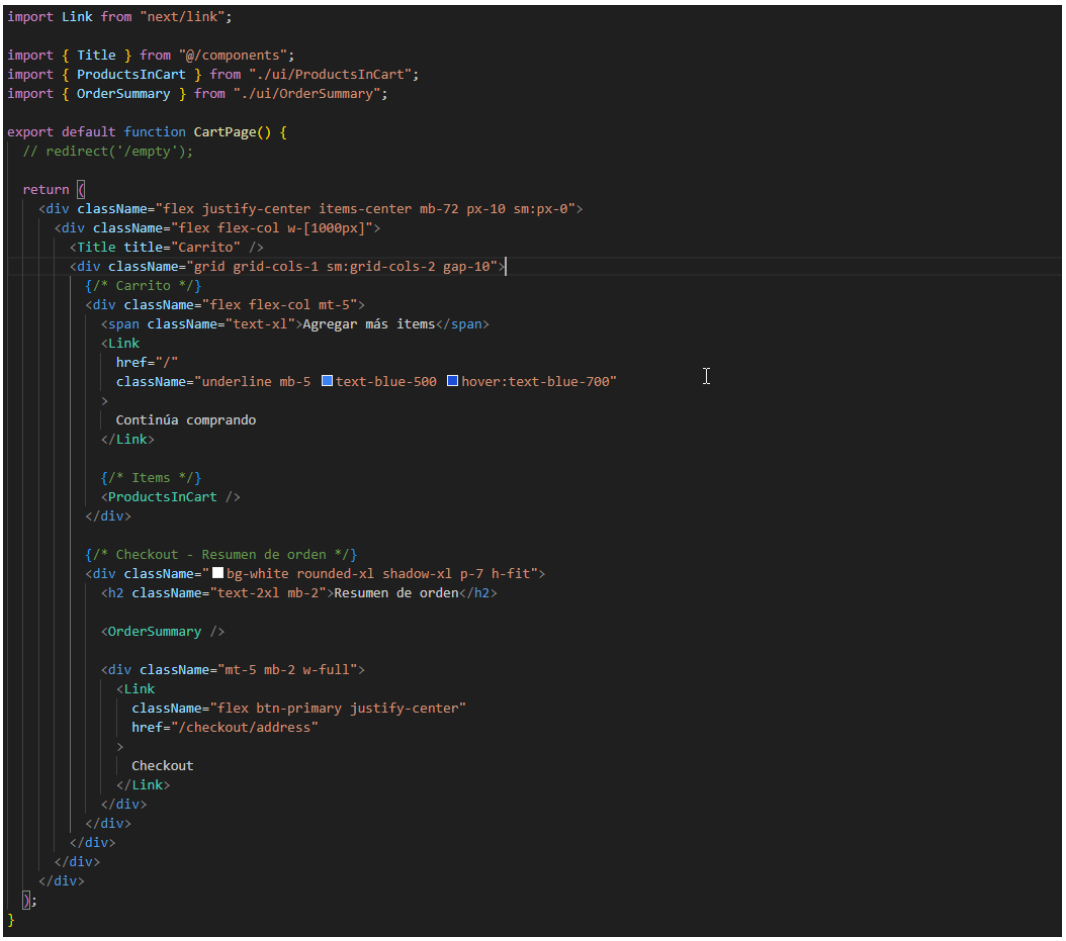


Figura 4.38 Interfaz de “Carrito”

Fuente Elaboración Propia



```
import Link from "next/link";

import { Title } from "@/components";
import { ProductsInCart } from "./ui/ProductsInCart";
import { OrderSummary } from "./ui/OrderSummary";

export default function CartPage() {
  // redirect('/empty');

  return (
    <div classNameName="flex justify-center items-center mb-72 px-10 sm:px-0">
      <div classNameName="flex flex-col w-[1000px]">
        <Title title="Carrito" />
        <div classNameName="grid grid-cols-1 sm:grid-cols-2 gap-10">
          {/* Carrito */}
          <div classNameName="flex flex-col mt-5">
            <span classNameName="text-xl">Agregar más items</span>
            <Link
              href="/"
              classNameName="underline mb-5 text-blue-500 hover:text-blue-700"
            >
              Continúa comprando
            </Link>

            {/* Items */}
            <ProductsInCart />
          </div>

          {/* Checkout - Resumen de orden */}
          <div classNameName="bg-white rounded-xl shadow-xl p-7 h-fit">
            <h2 classNameName="text-2xl mb-2">Resumen de orden</h2>

            <OrderSummary />

            <div classNameName="mt-5 mb-2 w-full">
              <Link
                classNameName="flex btn-primary justify-center"
                href="/checkout/address"
              >
                Checkout
              </Link>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
}
```

Figura 4.39 Código fuente de la Interfaz “Carrito”

Fuente Elaboración Propia

- Interfaz de “Checkout”

En la figura se observa la interfaz destinada a mostrar los campos del checkout para tomar en cuenta.

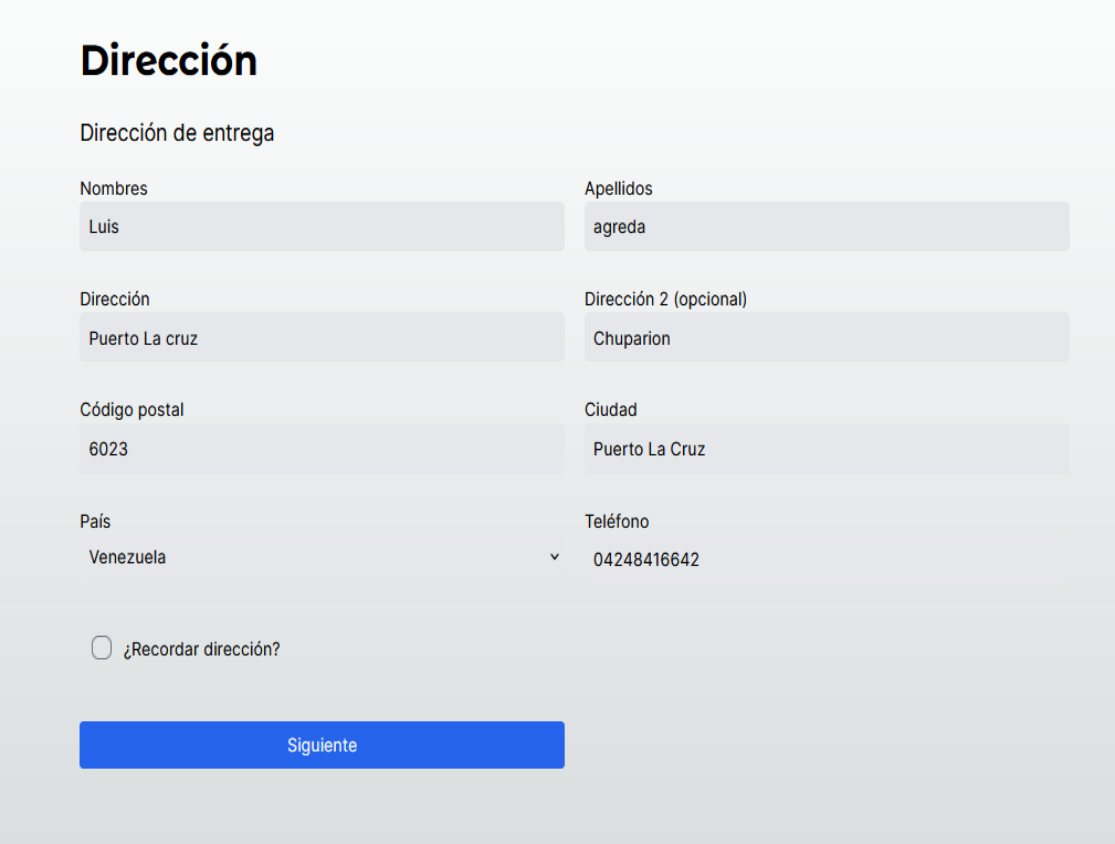


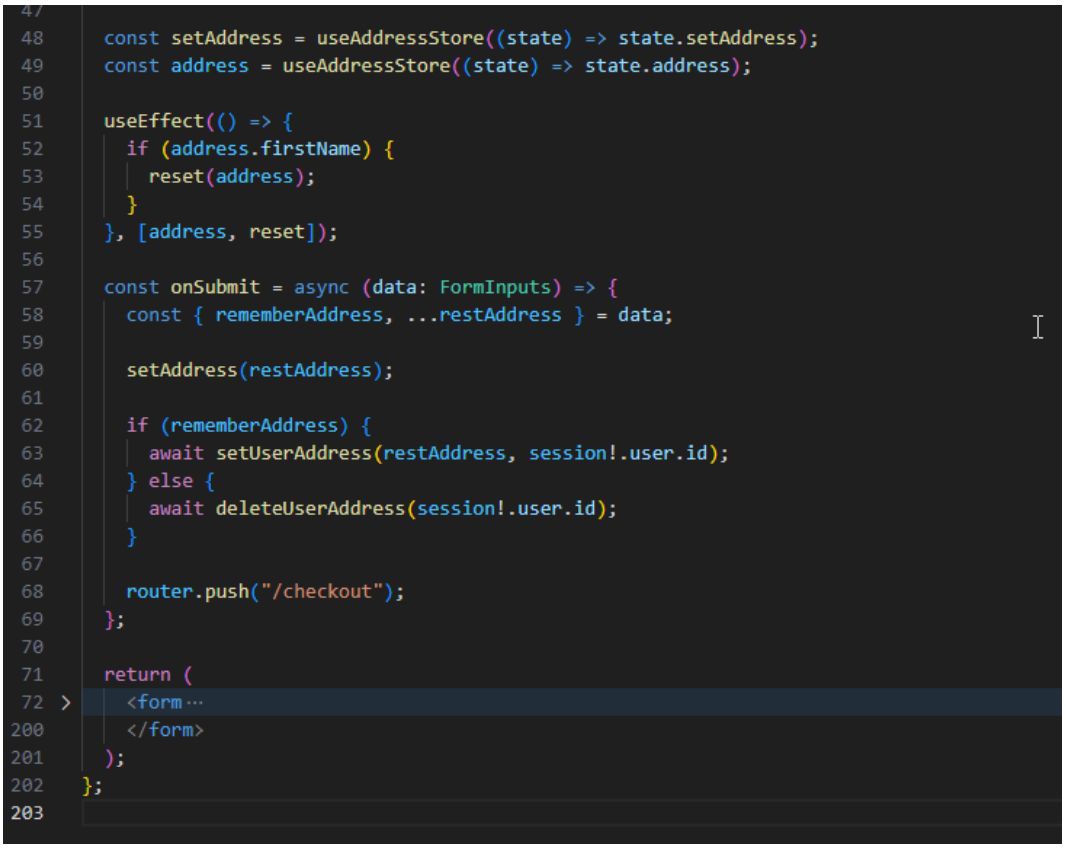
Figura 4.40 Interfaz de “Checkout”

Fuente Elaboración Propia

```
) > (shop) > checkout > address > ui > AddressForm.tsx > AddressForm
"use client";

import { useEffect } from "react";
import { useRouter } from "next/navigation";
import { useSession } from "next-auth/react";
import { useForm } from "react-hook-form";
import clsx from "clsx";

import type { Address, Country } from "@/interfaces";
import { useAddressStore } from "@/store";
import { deleteUserAddress, setUserAddress } from "@/actions";

type FormInputs = {
  firstName: string;
  lastName: string;
  address: string;
  address2?: string;
  postalCode: string;
  city: string;
  country: string;
  phone: string;
  rememberAddress: boolean;
};

interface Props {
  countries: Country[];
  userStoredAddress?: Partial<Address>;
}

export const AddressForm = ({ countries, userStoredAddress = {} }: Props) => {
  const router = useRouter();
  const {
    handleSubmit,
    register,
    formState: { isValid },
    reset,
  } = useForm<FormInputs>({
    defaultValues: {
      ...(userStoredAddress as any),
      rememberAddress: false,
    },
  });

  const { data: session } = useSession({
    required: true,
  });
```

Figura 4.41 Interfaz de “Checkout” seccion del Address

Fuente Elaboración Propia



```
47
48    const setAddress = useAddressStore((state) => state.setAddress);
49    const address = useAddressStore((state) => state.address);
50
51    useEffect(() => {
52      if (address.firstName) {
53        reset(address);
54      }
55    }, [address, reset]);
56
57    const onSubmit = async (data: FormInputs) => {
58      const { rememberAddress, ...restAddress } = data;
59
60      setAddress(restAddress);
61
62      if (rememberAddress) {
63        await setUserAddress(restAddress, session!.user.id);
64      } else {
65        await deleteUserAddress(session!.user.id);
66      }
67
68      router.push("/checkout");
69    };
70
71    return (
72  >   <form···
200       </form>
201     );
202   };
203
```

Figura 4.42 Interfaz de “Checkout” seccion del Address

Fuente Elaboración Propia

Segunda interfaz de continuación de checkout donde se puede apreciar una antesala de la factura a crearse si se procede con la compra.

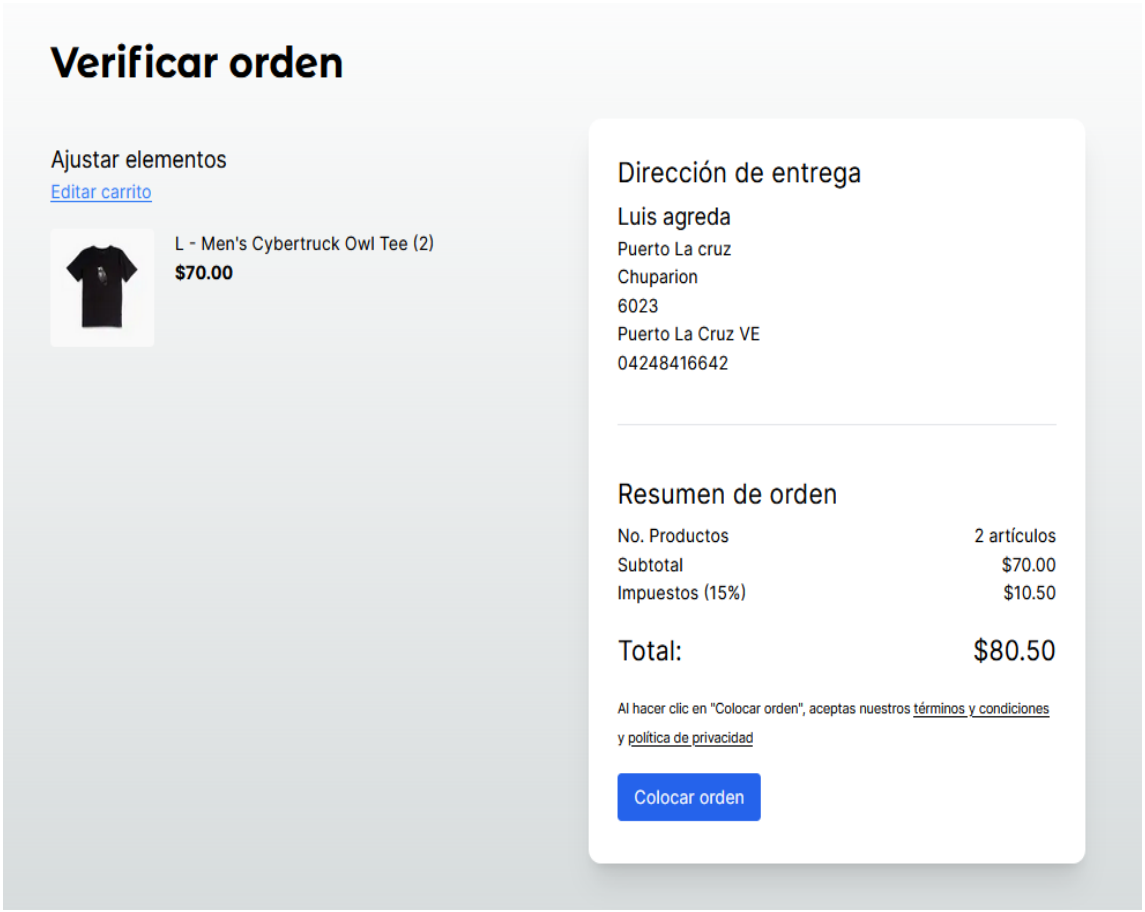


Figura 4.43 Interfaz de “Checkout”

Fuente Elaboración Propia

```tsx
import Link from "next/link";

import { Title } from "@/components";
import { ProductsInCart } from "./ui/ProductsInCart";
import { PlaceOrder } from "./ui/PlaceOrder";

export default function CheckoutPage() {
  return (
    <div classNameName="flex justify-center items-center mb-72 px-10 sm:px-0">
      <div classNameName="flex flex-col w-[1000px]">
        <Title title="Verificar orden" />

        <div classNameName="grid grid-cols-1 sm:grid-cols-2 gap-10">
          {/* Carrito */}
          <div classNameName="flex flex-col mt-5">
            <span classNameName="text-xl">Ajustar elementos</span>
            <Link href="/cart" classNameName="underline mb-5 text-blue-500 hover:text-blue-700">
              Editar carrito
            </Link>

            {/* Items */}
            <ProductsInCart />
          </div>

          {/* Checkout - Resumen de orden */}
          <PlaceOrder />
        </div>
      </div>
    </div>
  );
}
```

Figura 4.44 Código fuente de la Interfaz “Checkout”

Fuente Elaboración Propia

- Interfaz de “Órdenes” para administrador

En la figura se observa la interfaz destinada a mostrar las órdenes hechas dentro del sistema donde se puede ver el id, y el dueño quien creó la Orden.

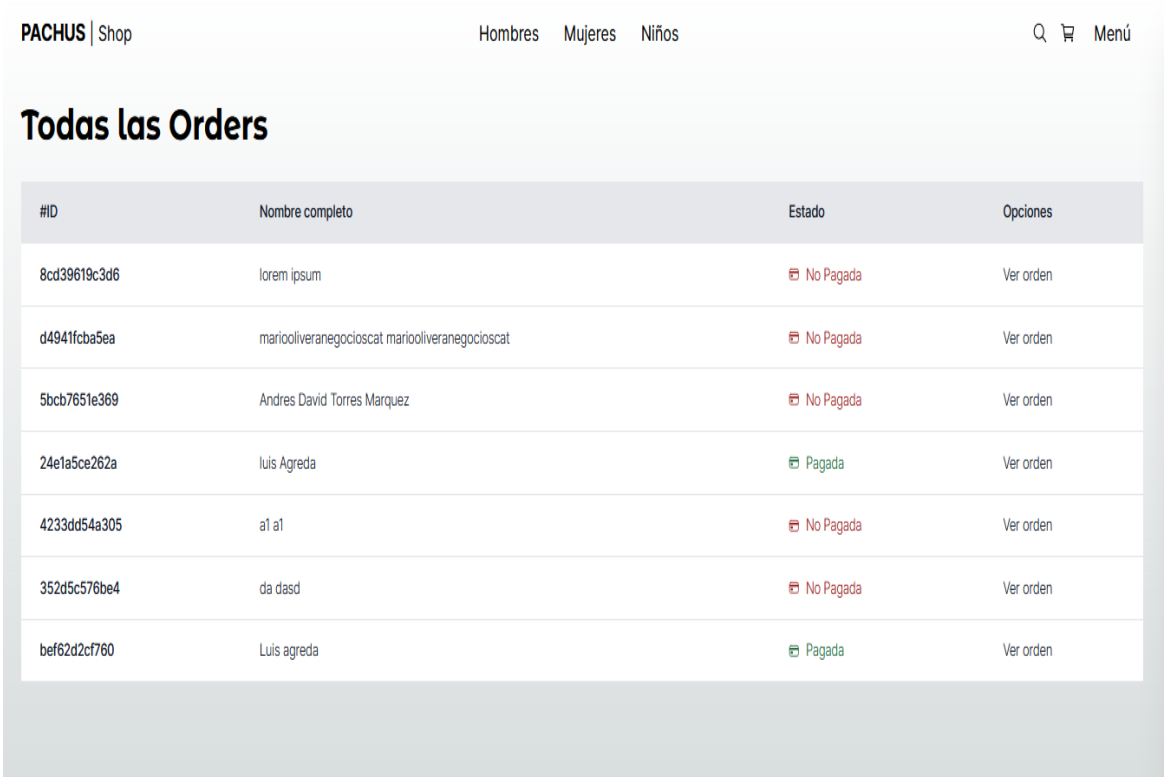
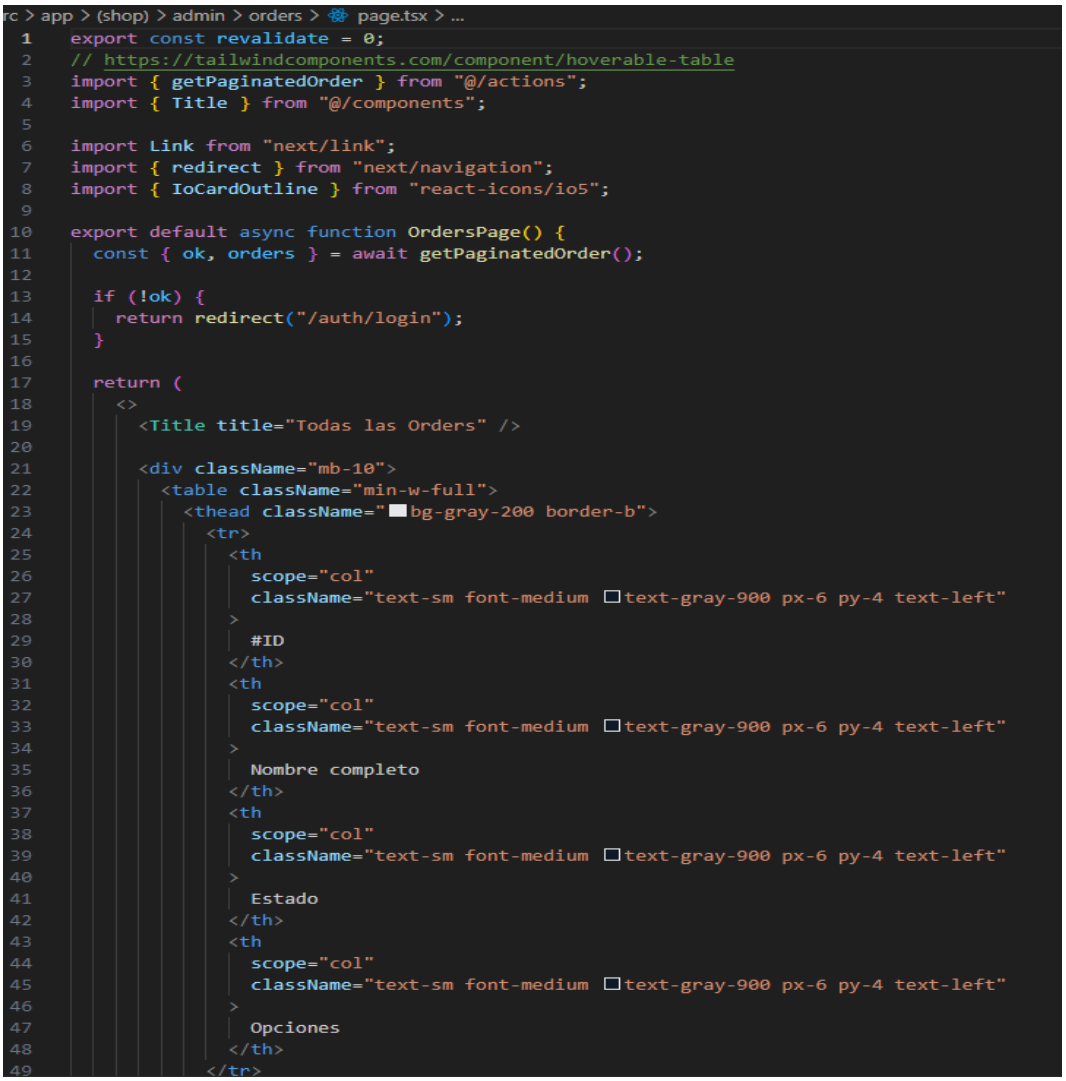


Figura 4.45 Interfaz de “Ordenes”

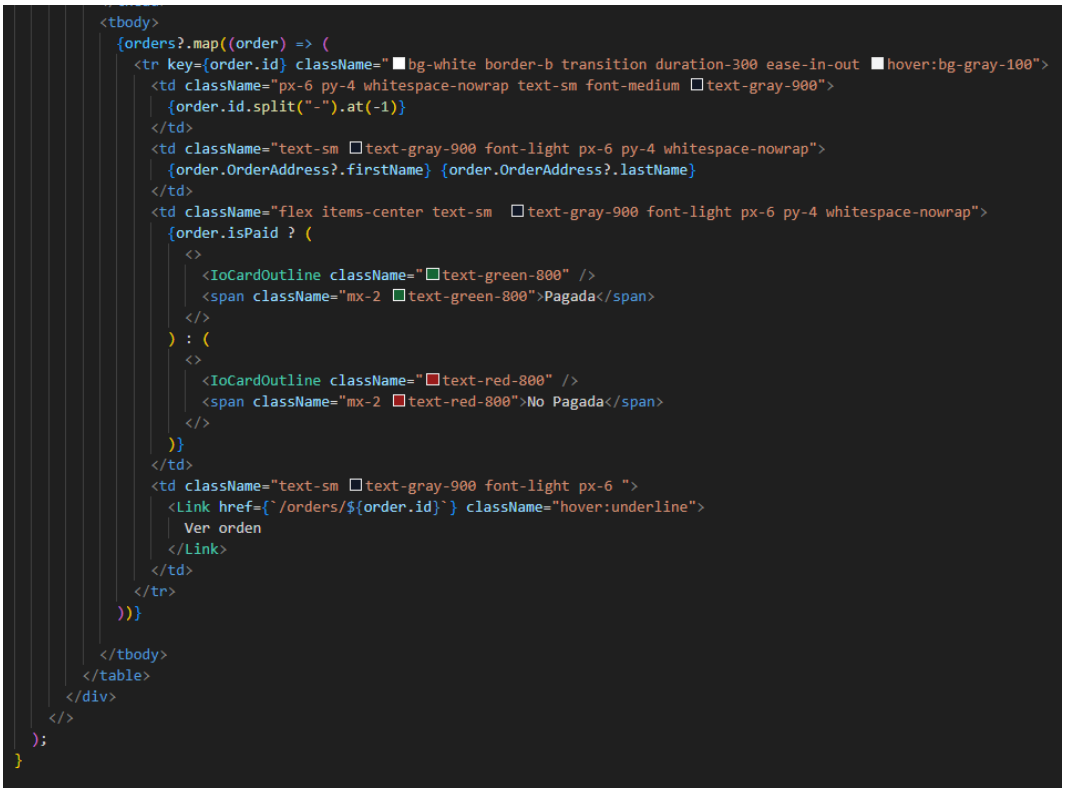
Fuente Elaboración Propia



```
rc > app > (shop) > admin > orders > page.tsx > ...
1   export const revalidate = 0;
2   // https://tailwindcomponents.com/component/hoverable-table
3   import { getPaginatedOrder } from "@/actions";
4   import { Title } from "@/components";
5
6   import Link from "next/link";
7   import { redirect } from "next/navigation";
8   import { IoCardOutline } from "react-icons/io5";
9
10  export default async function OrdersPage() {
11    const { ok, orders } = await getPaginatedOrder();
12
13    if (!ok) {
14      return redirect("/auth/login");
15    }
16
17    return (
18      <>
19        <Title title="Todas las Orders" />
20
21        <div className="mb-10">
22          <table className="min-w-full">
23            <thead className="bg-gray-200 border-b">
24              <tr>
25                <th
26                  scope="col"
27                  className="text-sm font-medium text-gray-900 px-6 py-4 text-left"
28                >
29                  #ID
30                </th>
31                <th
32                  scope="col"
33                  className="text-sm font-medium text-gray-900 px-6 py-4 text-left"
34                >
35                  Nombre completo
36                </th>
37                <th
38                  scope="col"
39                  className="text-sm font-medium text-gray-900 px-6 py-4 text-left"
40                >
41                  Estado
42                </th>
43                <th
44                  scope="col"
45                  className="text-sm font-medium text-gray-900 px-6 py-4 text-left"
46                >
47                  Opciones
48                </th>
49              </tr>
```

Figura 4.46 Código fuente de la Interfaz “Orders”

Fuente Elaboración Propia



```
        <tbody>
          {orders?.map((order) => (
            <tr key={order.id} className="bg-white border-b transition duration-300 ease-in-out hover:bg-gray-100">
              <td className="px-6 py-4 whitespace-nowrap text-sm font-medium text-gray-900">
                {order.id.split("-").at(-1)}
              </td>
              <td className="text-sm text-gray-900 font-light px-6 py-4 whitespace-nowrap">
                {order.OrderAddress?.firstName} {order.OrderAddress?.lastName}
              </td>
              <td className="flex items-center text-sm  text-gray-900 font-light px-6 py-4 whitespace-nowrap">
                {order.isPaid ? (
                  <>
                    <IoCardOutline className="text-green-800" />
                    <span className="mx-2 text-green-800">Pagada</span>
                  </>
                ) : (
                  <>
                    <IoCardOutline className="text-red-800" />
                    <span className="mx-2 text-red-800">No Pagada</span>
                  </>
                )}
              </td>
              <td className="text-sm text-gray-900 font-light px-6 ">
                <Link href={`/orders/${order.id}`} className="hover:underline">
                  Ver orden
                </Link>
              </td>
            </tr>
          ))}

        </tbody>
      </table>
    </div>
  </>
 );
}
```

Figura 4.47 Código fuente de la Interfaz “Orders”

Fuente Elaboración Propia

- Interfaz de “Productos” desde un rol de administrador

En la figura 4.48, se observa la interfaz destinada a mostrar los productos dentro del sistema donde se puede ver el listado donde se podrá modificar y agregar nuevos productos al sistema.

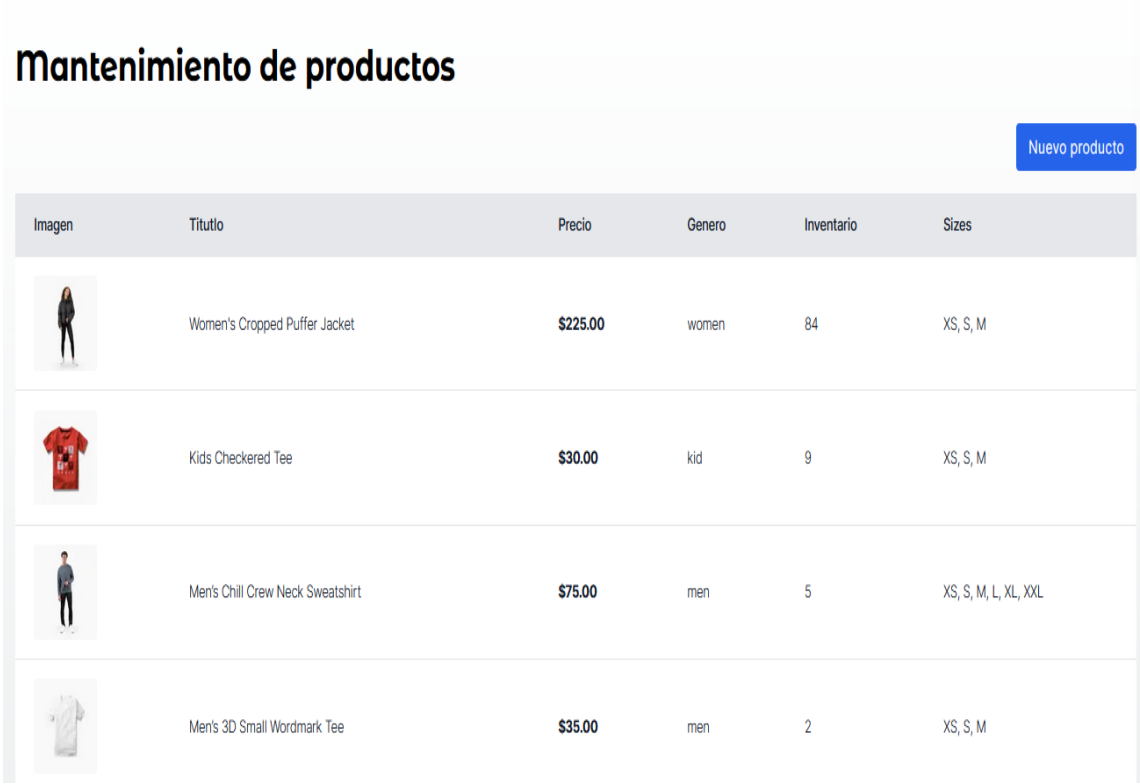


Figura 4.48 Interfaz de “Productos”

Fuente Elaboración Propia

src > app > (shop) > admin > products > page.tsx > ...

```tsx
export const revalidate = 0;
// https://tailwindcomponents.com/component/hoverable-table
import { getPaginatedOrder, getPaginatedProductsWithImages } from "@/actions";
import { Pagination, ProductImage, Title } from "@/components";

import Link from "next/link";
import { redirect } from "next/navigation";
import { IoCardOutline } from "react-icons/io5";
import Image from "next/image";
import { currencyFormat } from "@/utils";

interface Props {
  searchParams: {
    page?: string;
  };
}

export default async function ProductsAdminPage({ searchParams }: Props) {
  const page = searchParams.page ? parseInt(searchParams.page) : 1;

  const { products, currentPage, totalPages } =
    await getPaginatedProductsWithImages({ page });

  return (
    <>
      <Title title="Mantenimiento de productos" />

      <div className="flex justify-end mb-5">
        <Link href={"/admin/product/new"} className="btn-primary">
          Nuevo producto
        </Link>
      </div>

```

Figura 4.49 Código fuente de la Interfaz “Productos”

Fuente Elaboración Propia

```
<div className="mb-10">
  <table className="min-w-full">
    <thead className="bg-gray-200 border-b">
      <tr>
        <th
          scope="col"
          className="text-sm font-medium text-gray-900 px-6 py-4 text-left"
        >
          Imagen
        </th>
        <th
          scope="col"
          className="text-sm font-medium text-gray-900 px-6 py-4 text-left"
        >
          Titutlo
        </th>
        <th
          scope="col"
          className="text-sm font-medium text-gray-900 px-6 py-4 text-left"
        >
          Precio
        </th>
        <th
          scope="col"
          className="text-sm font-medium text-gray-900 px-6 py-4 text-left"
        >
          Genero
        </th>
        <th
          scope="col"
          className="text-sm font-medium text-gray-900 px-6 py-4 text-left"
        >
          Inventario
        </th>
        <th
          scope="col"
          className="text-sm font-medium text-gray-900 px-6 py-4 text-left"
        >
          Sizes
        </th>
      </tr>
    </thead>
```

Figura 4.50 Código fuente de la Interfaz “Productos”

Fuente Elaboración Propia

```
          <tbody>
            {products?.map((product) => (
              <tr
                key={product.id}
                className="bg-white border-b transition duration-300 ease-in-out hover:bg-gray-100"
              >
                <td className="px-6 py-4 whitespace-nowrap text-sm font-medium text-gray-900">
                  <Link href={`/product/${product.slug}`}>
                    <ProductImage
                      src={product.ProductImage[0]?.url}
                      alt={product.title}
                      width={80}
                      height={80}
                      className="object-cover w-20 h-20 rounded"
                    />
                  </Link>
                </td>
                <td className="text-sm text-gray-900 font-light px-6 py-4 whitespace-nowrap">
                  <Link
                    href={`/admin/product/${product.slug}`}
                    className="hover:underline"
                  >
                    {product.title}
                  </Link>
                </td>
                <td className="text-sm text-gray-900 font-bold px-6 ">
                  {currencyFormat(product.price)}
                </td>
                <td className="text-sm text-gray-900 font-light px-6 ">
                  {product.gender}
                </td>
                <td className="text-sm text-gray-900 font-light px-6 ">
                  {product.inStock}
                </td>
                <td className="text-sm text-gray-900 font-light px-6 ">
                  {product.sizes.join(", ")}
                </td>
              </tr>
            ))}
          </tbody>
        </table>
        <Pagination totalPages={totalPages} />
      </div>
    </>
  );
}
```

Figura 4.51 Código fuente de la Interfaz “Productos”

Fuente Elaboración Propia

4.3.2 Flujo de trabajo de Prueba

Una de las fases más importantes del ciclo de vida de Desarrollo de Software es el Flujo de Trabajo Pruebas, en ella se realizan evaluaciones para obtener la calidad de un producto, y/o mejorarlo, mediante la identificación de defectos y problemas; esto permitirá entregar al cliente un producto de software libre de defectos o de errores. Las pruebas como

técnicas de comprobación dinámica permitirán evaluar la calidad del producto, en este caso de la aplicación para mejorarla identificando problemas o defectos.

4.3.2.1 Pruebas de Unidad

Con el objetivo de validar cada componente desarrollado y revisar el flujo de datos, tanto entrada como salidas del sistema SAVIP, se planea realizar pruebas individuales que cubran las principales funciones del mismo al momento de ingresar datos y recibir las respuestas. Este proceso se detalla a través de las siguientes tablas explicativas.

- Prueba “Agregar Producto”

En la tabla 4.15 se puede observar en detalle las pruebas referentes al perfil.

Variable	Valor	Solicitud	Resultado
Nombre	**	X	No permite la entrada de caracteres especiales
	346	O	Registro exitoso
	Women's Puffer Jacket	O	Registro exitoso
Slug	**	X	No permite la entrada de caracteres especiales
	346	O	Registro exitoso
	Women's Puffer Jacket	O	Registro exitoso
Descripción	Women's Cropped Puffer Jacket	O	Registro exitoso

Precio	Doce	X	No se permite la entrada de caracteres alfabeticos
	12	O	Registro exitoso
Inventario	Ocho	X	No se permite la entrada de caracteres alfabeticos
	8	O	Registro exitoso

Tabla 4.15. Prueba “Editar plan”

Fuente: Elaboración Propia

- Prueba registrar usuario

En la tabla 4.16 se puede observar en detalle las pruebas referentes a los campos de registro usuario.

Variable	Valor	Solicitud	Resultado
Nombre Completo	Alan Brito2	O	Registro exitoso
	123235	x	No permite la entrada puros caracteres numéricos
	Aquiles	O	Registro exitoso
Correo Electrónico	123123	X	no permite la entrada que no cumpla con el formato de @gmail.com
	Admin@email.com	O	Registro exitoso
	Correo personal	X	no permite la entrada que no cumpla con el formato de @gmail.com
Contraseña	contraseña	O	Registro exitoso

Tabla 4.16. Prueba “Editar plan”

Fuente: Elaboración Propia

- Prueba “Dirección de entrega”

En la tabla 4.17 se puede observar en detalle las pruebas referentes a la dirección de entrega.

Variable	Valor	Solicitud	Resultado
Nombres	Monkey D. Luffy	O	Registro exitoso
Apellidos	Konosuba	O	Registro exitoso
Dirección	Av. Calle13	O	Registro exitoso
Dirección 2	Ennies Lobby	O	Registro exitoso
Código Postal	CS1234	O	Registro exitoso
	6023	O	Registro exitoso
Ciudad	Barcelona	O	Registro exitoso
	Calle Vazquez 2	O	Registro exitoso
Teléfono	04248416642	O	Registro exitoso
	0424UIC1234	X	No se permite la entrada de caracteres alfabeticos
	234252352623	X	No se permite más de 12 caracteres

Tabla 4.17 Prueba “Editar plan”

Fuente: Elaboración Propia

4.3.3 Conclusión de la fase de construcción

Durante la etapa de construcción del sistema SAVIP, se detallaron los elementos finales del software teniendo en cuenta el análisis y diseño previos. Se crearon los diseños de las interfaces visuales y se llevaron a

cabo pruebas individuales en los principales componentes que conforman el sistema. Esto permitió identificar y corregir los errores encontrados, resultando en una primera versión funcional del software lista para su uso.

CONCLUSIONES

- Se realizó un diagnóstico de la situación actual del sistema con la finalidad de poder tomar las decisiones convenientes y oportunas para el mejoramiento de las operaciones, de esta manera se generó información primordial para tratar la problemática planteada.
- Se ilustró en un diagrama del modelo de dominio, permitiendo de esta manera la estructura del sistema actual y la problemática a resolver mediante el sistema propuesto.
- Se concretó un análisis exhaustivo de los requisitos necesarios para llevar a cabo el primer modelo general del sistema. Se identificó a los actores que forman parte del mismo, junto a los distintos casos de uso, los cuales permitieron obtener una visión general de su funcionamiento.
- Se diseñó la estructura de la base de datos para el sistema propuesto y las interfaces del usuario, además de cumplir con un manejo intuitivo y amigable para el usuario, mejorando de esta manera la experiencia de los mismos.
- Se codificó el software teniendo en cuenta los requisitos funcionales y no funcionales identificados durante el análisis exhaustivo, a su vez, se culminó con éxito todos los procesos fundamentales del sistema propuesto.
- La elección de Next.js permitió implementar renderizado híbrido (SSR/SSG) para la interfaz de usuario, optimizando tanto el rendimiento como el posicionamiento SEO del catálogo digital de PACHUS.
- PostgreSQL demostró ser ideal para el modelo de negocio gracias a su robusto soporte para jerarquías de datos y relaciones complejas, permitiendo gestionar eficientemente la estructura de productos, categorías y variantes de tallas.

- Prisma se consolidó como el ORM perfecto, simplificando radicalmente la creación de APIs con validación automática, reduciendo los errores comunes en operaciones CRUD y acelerando el desarrollo de funciones críticas como la gestión de inventario en tiempo real.
- Se realizaron las pruebas correspondientes que verificaron la correcta lectura y funcionamiento de los datos introducidos por el usuario, por último, se evaluó la confiabilidad, la integridad, la confidencialidad y la disponibilidad de la información de manera segura del sistema, dando como resultado el correcto funcionamiento de la aplicación web.

RECOMENDACIONES

Desarrollar procedimientos que faciliten el cuidado de la base de datos del sistema, como realizar copias de seguridad de los datos y poder recuperarlos en caso de existir algún percance.

Añadir atributos a la entidad “Tienda” que permita llevar un inventario de manera automatizada de las tallas, colores, modelos de los productos pertenecientes a la tienda.

Añadir atributos a la entidad “productos” que permita llevar un inventario específico del stock según la talla.

Con el objetivo de mejorar los procesos dentro de la empresa se propone optimizar el sistema mediante el agregado de nuevas funcionalidades referentes al proceso de negocio, para así hacerlo cada vez más completo y funcional.

Traducir el contenido de la aplicación diferentes idiomas podrían ampliar su alcance y hacerlo accesible a una audiencia más diversa.

Efectuar la fase de transición del proceso racional unificado (RUP) para el desarrollo de la aplicación web.

BIBLIOGRAFÍA

Abata y Rosales (2022). Tesis. Recuperado a partir de http://repositorio.ug.edu.ec/handle/redug/65327

Aguirre Pomahuali. (2022). Desarrollar un sistema web para mejorar la eficiencia y eficacia del proceso de ventas de la empresa Ezzeta. Tesis para optar el título profesional de Ingeniero Informático, Escuela Profesional de Ingeniería Informática, Facultad de Ingeniería, Universidad Nacional de Ingeniería, Lima, Perú.

Arias (2012), Definiendo investigación y su clasificación por niveles según Arias. https://fredjmr.wordpress.com/2020/08/30/definiendo-investigacion-y-su-clasificacion-por-niveles-segun-arias/

Arias, F. (2012). Metodología de la investigación. Quinta edición. Editorial McGraw-Hill.

Berzosa (2022), ¿Qué es un sistema web? https://quees.com/sistema-web/.

Booch, G.; Jacobson, I. y Rumbaugh, J. (2000). El Lenguaje unificado de modelado. Manual de referencia. Madrid, España: Pearson Educación, S.A.

Ccoyllo, J. (2022). Sistema web para la automatización del proceso de gestión de pedidos en el restaurante Manu Barra Cevichera, Ayacucho 2021 (Tesis de grado). Universidad Ayacucho Federico Froebel, Ayacucho, Perú.

El Economista. (2022, June 25). Cómo funciona PayPal.Junio 18, 2024, from https://www.paypal.com/mx/webapps/mpp/what-is-paypal

Elmasri R. y Navathe B. (1997). Sistemas de bases de datos. Conceptos fundamentales. Addison-Wesley Iberoamericana. Traducción de Fundamentals of DatabaseSystems.

Gómez (2007), Fundamentos de bases de datos. https://slideplayer.es/slide/15155281/

Hernández (2022), ¿Qué es Nextjs y para qué sirve? https://www.dongee.com/tutoriales/next-js-que-es-y-para-que-sirve/

Hernández Moya, E. A., & Navas Silva, J. A. (2023). Desarrollo de un sistema de administración de ventas y control de inventarios para tiendas en línea. caso de estudio: invirtual store. Trabajo de titulación previo a la obtención del Título de Ingenieros en Ciencias de la Computación, Carrera de Computación, Universidad Politécnica Salesiana, Sede Quito, Ecuador.

Hernández, R., Collado, C., Baptista, M. (2007). Metodología de la investigación. Quinta edición. Editorial McGraw-Hill.

IT DO. (2023, 9 de marzo). ¿Qué es Node.js y para qué sirve?. https://www.lucushost.com/blog/que-es-node-js/

Inbound Cycle. (2023). JSON. Diccionario de Marketing Online. https://www.inboundcycle.com/diccionario-marketing-online/json

Laucirica& Durán (2019), ¿Qué es React?, ¿Por qué usar React)?. https://desarrolloweb.com/articulos/que-es-react-motivos-uso.html

López (2022). Sistema de información para la gestión administrativa del gimnasio HealthyLifestyle. [Tesis Pregrado, Universidad Inca Garcilaso de la Vega]

Martínez A. y Martínez R. (2000). Guía a RationalUnifedProcess [Tesis de doctorado, Universidad de Castilla la Mancha, España]. ResearchGate. https://www.researchgate.net/publication/268005509

Marqués, M. (2011). Base de datos. Castellón de la Plana, España: Publicaciones de la Universidad Jaume I.

Moreno y Delgado (2022), Diseño de un sistema de información para la gestión de gimnasios locales. [Tesis pregrado, Universidad Distrital Francisco José de Caldas]

Negrete (2017), TypeScript, ¿qué es y cómo se diferencia de JavaScript?. https://www.crehana.com/blog/transformacion-digital/que-es-typescript/

Pressman, R. (2001). Ingeniería del Software. Un enfoque práctico. España: McGraw-Hill.

Puleo (1985), Paradigmas de la información, Universidad de Los Andes (ula), Mérida, p. 205.

Sampieri (2003), Investigación de campo. https://investigaciondecampo.com/investigacion-sampieri/

Trujillo Nieto, B. V. (2023). Propuesta de implementación de un sistema web de ventas online para la empresa Compuservice Isaac - Huacho; 2022 (Tesis para optar el título profesional de Ingeniero de Sistemas). Escuela Profesional de Ingeniería de Sistemas, Facultad de Ciencias e Ingeniería, Universidad Nacional del Santa, Chimbote, Perú.

Troya Zambrano, L. F. (2023). Desarrollo de una aplicación web para el control de inventario de la empresa "Imprenta & Publicidad Cuyabeno " utilizando asp.net. Trabajo de Titulación para optar al título de Ingeniería en Tecnologías de la Información, Carrera de Ingeniería en Tecnologías de la Información, Facultad de Ingeniería, Universidad Nacional de Chimborazo, Riobamba.

Vargas & Barragán (2019), Sistema de Información para la gestión de Gimnasios. [Tesis pregrado, Universidad Distrital Francisco José de Caldas]

Yahuasi (2020), Sistema de información web de control administrativo. [Tesis pregrado, Universidad Pública De El Alto]

Zavala, Introducción a Node.js. https://sg.com.mx/revista/32/introduccion-nodejs

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

TÍTULO	DESARROLLO DE UNA APLICACIÓN WEB PARA EL SEGUIMIENTO DE PAGOS, VENTAS Y CONTROL DE STOCK PARA UNA EMPRESA DE VENTA DE ROPA PERSONALIZADA
SUBTÍTULO	

AUTOR(ES):

APELLIDOS Y NOMBRES	**CVLAC / E_MAIL**	
Agreda O Luis A	**CVLAC:**	26.886.935
	E_MAIL	luis.agreda98@gmail.com

PALÁBRAS O FRASES CLAVES:

E-commerce
Paypal
Gestion de Stock
Sistema Administrativo
desarrollo software
software libre
RUP
UML

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ÁREA	SUBÁREA
Escuela de Ingeniería y Ciencias Aplicadas	Ingeniería en Computación

RESUMEN (ABSTRACT):

El presente trabajo de investigación tuvo como objetivo principal Desarrollar una aplicación web para el control de las ventas, pagos y manejo del stock, de una empresa dedicada a la venta de productos de ropa personalizada, en Puerto la cruz, estado Anzoátegui, basándose en la metodología del Proceso Racional Unificado. Para el logro de los objetivos planteados se realizaron visitas a la empresa PACHUS. donde se obtuvo información con el dueño de la empresa sobre el problema de la situación actual por la que pasa la empresa en el momento de gestionar sus equipos, y verificar el pago de sus facturas. Se realizó una observación de como es el sistema actual que posee la empresa para el control de sus facturas descubriendo las fallas existentes en el sistema el cual era totalmente a mano sin ninguna automatización de procesos; se tomaron los puntos más críticos para crear una aplicación que soportara no solo la información, sino que realizara procesos automáticos en diversas áreas teniendo como resultado la creación del Sistema Administrador de Ventas, Inventario y Pagos(SAVIP). El desarrollo del software posee un marco de trabajo en NextJS y la aplicación de una base de datos en PostgreSQL, así como, un lenguaje de programación Javascript, lenguaje de hojas de estilo en Cascada(CSS) y un lenguaje de marcado HTML.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO

CONTRIBUIDORES

APELLIDOS Y NOMBRES	**ROL / Código CVLAC / E_MAIL**	
Aquiles Torrealba	**ROL**	**CA** | **AS** | **TU X** | **JU**
	CVLAC:	7.385.840
	E_MAIL	
Manuel Carrasquero	**ROL**	**CA** | **AS** | **TU** | **JU X**
	CVLAC:	7.374.987
	E_MAIL	manuelscm@gmail.com
Claudio Cortínez	**ROL**	**CA** | **AS** | **TU** | **JU X**
	CVLAC:	12.115.334
	E_MAIL	prof.claudio.cotinez@gmail.com

FECHA DE DISCUSIÓN Y APROBACIÓN:

2025	05	08
AÑO	**MES**	**DIA**

LENGUAJE: SPA.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO

ARCHIVO (S):

NOMBRE DE ARCHIVO	TIPO MIME
NAZTTG_AOLA2025	Application/msword

ALCANCE

ESPACIAL: inespacial

TEMPORAL: intemporal

TÍTULO O GRADO ASOCIADO CON EL TRABAJO:

Ingeniero en Computación

NIVEL ASOCIADO CON EL TRABAJO:

Pregrado

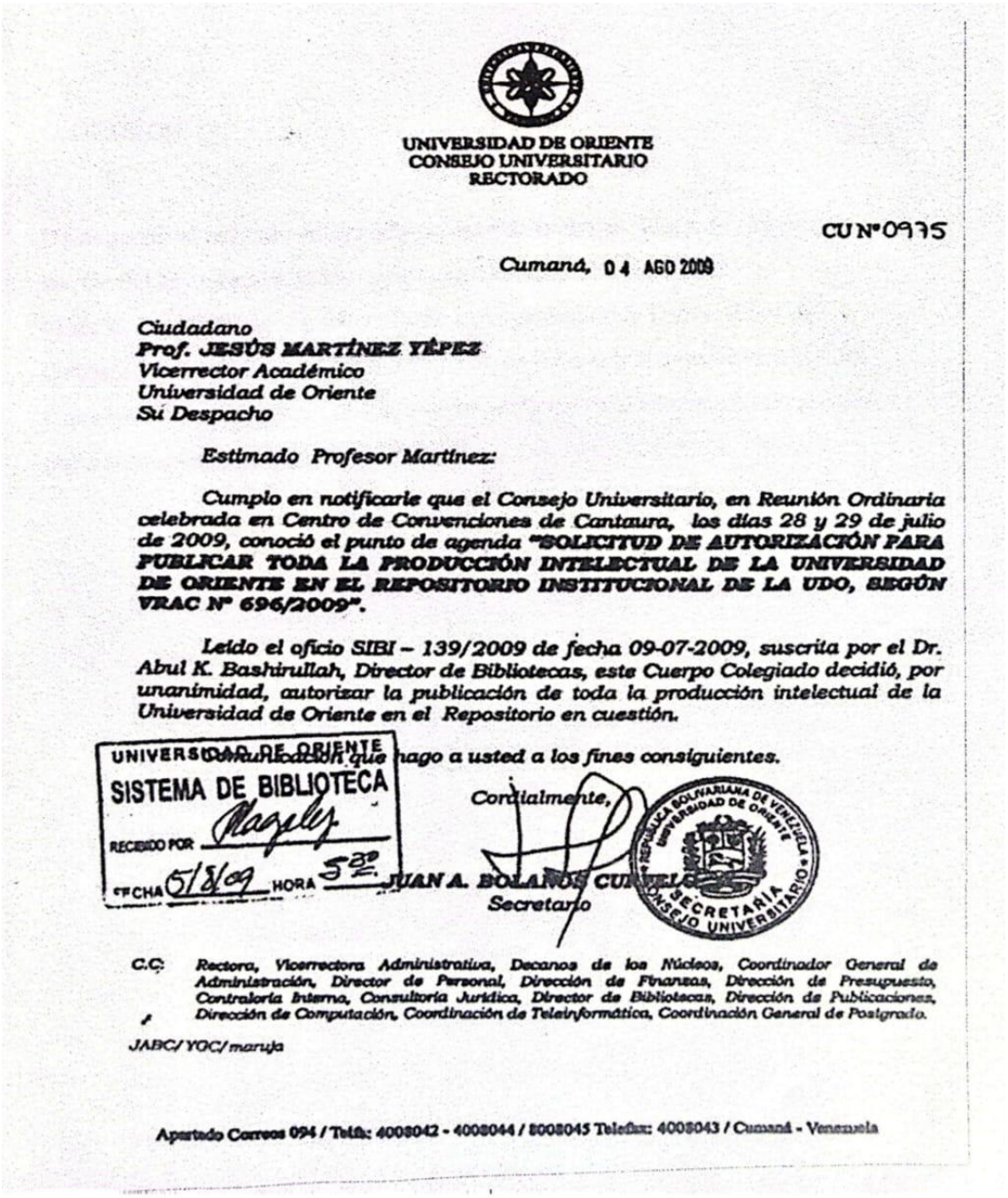
ÁREA DE ESTUDIO:

Departamento de Computación y Sistemas

INSTITUCIÓN:

Universidad de Oriente/Núcleo de Anzoátegui.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO



UNIVERSIDAD DE ORIENTE
CONSEJO UNIVERSITARIO
RECTORADO

CU Nº0975

Cumaná, 04 AGO 2009

Ciudadano
Prof. JESÚS MARTÍNEZ YÉPEZ
Vicerrector Académico
Universidad de Oriente
Su Despacho

Estimado Profesor Martínez:

Cumplo en notificarle que el Consejo Universitario, en Reunión Ordinaria celebrada en Centro de Convenciones de Cantaura, los días 28 y 29 de julio de 2009, conoció el punto de agenda **"SOLICITUD DE AUTORIZACIÓN PARA PUBLICAR TODA LA PRODUCCIÓN INTELECTUAL DE LA UNIVERSIDAD DE ORIENTE EN EL REPOSITORIO INSTITUCIONAL DE LA UDO, SEGÚN VRAC Nº 696/2009"**.

Leído el oficio SIBI – 139/2009 de fecha 09-07-2009, suscrita por el Dr. Abul K. Bashirullah, Director de Bibliotecas, este Cuerpo Colegiado decidió, por unanimidad, autorizar la publicación de toda la producción intelectual de la Universidad de Oriente en el Repositorio en cuestión.

Comunicación que hago a usted a los fines consiguientes.

UNIVERSIDAD DE ORIENTE
SISTEMA DE BIBLIOTECA
RECIBIDO POR
FECHA 5/8/09 HORA 5:20

Cordialmente,

JUAN A. BOLAÑOS CURIEL
Secretario

C.C: Rectora, Vicerrectora Administrativa, Decanos de los Núcleos, Coordinador General de Administración, Director de Personal, Dirección de Finanzas, Dirección de Presupuesto, Contraloría Interna, Consultoría Jurídica, Director de Bibliotecas, Dirección de Publicaciones, Dirección de Computación, Coordinación de Teleinformática, Coordinación General de Postgrado.

JABC/ YOC/ maruja

Apartado Correos 094 / Telfs: 4008042 - 4008044 / 8008045 Telefax: 4008043 / Cumaná - Venezuela

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO

Derechos

De acuerdo al artículo 41 del reglamento de trabajos de grado (Vigente a partir del II Semestre 2009, según comunicación CU-034-2009)

“Los Trabajos de grado son exclusiva propiedad de la Universidad de Oriente y solo podrán ser utilizadas a otros fines con el consentimiento del consejo de núcleo respectivo, quien lo participara al Consejo Universitario, para su autorización”

Luis Agreda

26.886.935

AUTOR

Aquiles Torrealba	**Manuel Carrasquero**	**Claudio Cortínez**
7.385.840	7.374.987	12.115.334
TUTOR	**JURADO**	**JURADO**

Prof. Claudio Cortínez

CORDINADOR(A)

COMISION DE TRABAJO DE GRADO