

**Universidad de Las Américas**  
Facultad de Ingenierías y Ciencias Agropecuarias  
*Ingeniería de Software*

**PROYECTO FINAL VALIDACIÓN Y VERIFICACIÓN DE SOFTWARE.**

**DATOS DEL LOS ALUMNOS:**

Enrique Merizalde, Jossue Játiva y Juan Aristizábal.

## Contenido

<b>1. Identificación del Problema</b>	<b>3</b>
1.1. Introducción	3
1.2. Descripción General:	3
1.3. Requisitos Específicos:	4
<b>2. Selección de Metodologías Integrales</b>	<b>5</b>
2.1. Especificación de Requisitos	5
2.2. Historias de Usuario	19
2.3. Descripción Funcionalidad Time to Market	22
2.4. Descripciones Técnicas de Prueba	24
<b>3. Propuesta Técnica:</b>	<b>27</b>
3.1. Casos de Uso	27
3.2. Diseño de Clases del Sistema	46
3.3. Implementación Sistema Informático	47
3.4. Casos de Prueba Técnica 1 – Prueba de Caja Blanca (Cobertura de Caminos)	51
3.5. Casos de Prueba Técnica 2 - Caja Negra Cobertura en Equivalencia	63
3.6. Casos de Prueba Técnica 3 - Caja Negra Cobertura de Valores Límites	69
3.7. Casos de Prueba Técnica 4 – Casos de Prueba	71
3.8. Resumen de Pruebas Ejecutadas y Recomendaciones	75
3.9. Link Video	77
3.10. Link GitHub	77
<b>4. Referencias:</b>	<b>78</b>

## 1. Identificación del Problema

### 1.1. Introducción

El proyecto *ChauchiSoft* busca desarrollar una plataforma que facilite la gestión y organización de productos para comerciantes informales. Como tal su objetivo principal es crear una herramienta que permita organizar, y administrar productos para su comercialización por parte de comerciante informales, esto permite llevar a cabo un registro de las ventas realizadas y facilitar la búsqueda de productos para ofrecer a sus clientes.

Entre los requisitos generales del proyecto antes mencionados tenemos los siguientes:

- **Gestión de Productos:** Registro, edición, eliminación de productos con detalles como nombre, descripción, categoría, precio y monto a ganar
- **Inventario en Tiempo Real:** Visualización y búsqueda de productos disponibles en función de las necesidades del cliente
- **Registro de Transacciones:** Administración de ventas con datos clave como fecha, productos vendidos, precio de venta y ganancias
- **Historial de Ventas:** Un módulo que permite a los comerciantes revisar sus transacciones o ventas.

### 1.2. Descripción General:

El objetivo del sistema *ChauchiSoft* es ayudar a profesionalizar el comercio informal para los comerciantes informales, mejorando la eficiencia en la venta de productos y así ofreciendo una solución ágil y moderna para el comercio electrónico a pequeña escala. Esto se logrará optimizar la experiencia tanto para vendedores como para clientes mediante herramientas tecnológicas fáciles de utilizar. El alcance de este sistema permite hacer referencia a distintos módulos del sistema:

- **Módulo de Gestión de Productos:**
  - Registro de productos que permita a los comerciantes añadir, editar y eliminar sus productos con detalles como nombre, descripción, categoría, precio y monto a ganar.
  - Clasificación de productos en categorías personalizadas para facilitar la organización y búsqueda.

- **Módulo de Inventario:**
  - Control de inventario en tiempo real que permita a los comerciantes visualizar los productos disponibles que puede ofrecer a sus clientes
  - Búsqueda de productos en el inventario según las necesidades del cliente.
- **Administración de Transacciones y Registro de Ventas:**
  - Registro de las transacciones realizadas, con campos para fecha de venta, producto(s), precio de venta, ganancia y datos del cliente.
  - Historial de ventas que permita a los comerciantes revisar transacciones previas, filtrar por fecha y producto para gestionar sus ventas.

### 1.3. Requisitos Específicos:

Los requisitos específicos del proyecto son los siguientes, y se dividen en requisitos funcionales y no funcionales.

Los requisitos funcionales son:

- **Gestión de Productos:** Registro, edición y eliminación de productos, clasificándolos en categorías personalizadas.
- **Inventario en Tiempo Real:** Visualización de los productos disponibles para los clientes
- **Registro de Transacciones:** Es un registro que incluye datos importantes como los productos vendidos, precio y datos del cliente
- **Historial de Ventas:** Permite filtrar por distintos parámetros para poder tener una mejor gestión y control de las ventas realizadas.

Entre los requisitos no funcionales se tienen los siguientes:

- Interfaz intuitiva para comerciantes con conocimientos tecnológicos básicos.
- Almacenamiento local de datos con medidas de seguridad.
- Optimización para condiciones de trabajo con recursos limitados.

Por otro lado, también se tienen supuestos restricciones y riesgos del proyecto. Entre los supuestos tenemos:

- Los comerciantes tendrán acceso a una computadora.
- Los precios establecidos serán finales y no se modificarán arbitrariamente.

Entre las restricciones del proyecto están:

- El equipo de desarrollo consta únicamente de tres personas.
- Cortes de electricidad de hasta 12 horas diarias limitan el tiempo efectivo de desarrollo.
- No se cuenta con productos listos para pruebas reales en las primeras fases del proyecto.

Finalmente, entre los riesgos del proyecto están:

- Dificultad de adopción tecnológica por parte de los comerciantes.
- Riesgo de pérdida de datos por almacenamiento local sin respaldo en la nube.
- Baja oferta o demanda de productos en la plataforma.

## 2. Selección de Metodologías Integrales.

### 2.1. Especificación de Requisitos

En esta sección se describirá la SRS (Especificación de Requerimientos de Software) del sistema *Chauchisoft*. Esta es una plataforma de gestión de productos diseñada para comerciantes informales. Se busca obtener una visión general de los objetivos, alcance y estructura del sistema antes mencionado al igual que los actores involucrados, terminología técnica y referencias utilizadas en el mismo.

### Introducción al Proyecto

#### a) Objetivo:

Se desea desarrollar una plataforma tecnológica que permita a los comerciantes informales gestionar su inventario, registrar transacciones y mejorar la visibilidad de sus productos hacia los clientes. *Chauchisoft* busca profesionalizar el comercio informal, mejorando la experiencia tanto para compradores, pero especialmente para vendedores informales de productos, por lo que se puede afirmar que este proyecto está dirigido principalmente hacia comerciantes informales con acceso a una computadora y conocimientos básicos del uso de tecnología.

#### b) Alcance:

El producto llamado *ChauchiSoft* tiene distintas funcionalidades que serán enlistadas a continuación:

- **Módulo de Gestión de Productos:** Este módulo permite el registro, edición y eliminación de productos con detalles como nombre, descripción, categoría,

precio y ganancias. Además, este módulo permite clasificación de productos en categorías personalizables.

- **Módulo de Inventario:** Este módulo permite el control de inventario además de la búsqueda eficiente de productos disponibles.
- **Módulo de Administración de Transacciones y Registro de Ventas:** Registro de transacciones con detalles como, productos, precio y datos del cliente

c) **Actores:** Existen 3 actores quienes contribuyeron en el desarrollo del sistema:

<b>Nombre</b>	Juan Aristizábal
<b>Rol</b>	Desarrollador & QA
<b>Categoría Profesional</b>	Ingeniero en Software
<b>Responsabilidades</b>	Desarrollo, implementación y testing módulos.
<b>Información de Contacto</b>	<a href="mailto:juan.aristizabal@udla.edu.ec">juan.aristizabal@udla.edu.ec</a>
<b>Aprobación</b>	Aprobado

<b>Nombre</b>	Enrique Merizalde
<b>Rol</b>	Desarrollador & QA
<b>Categoría Profesional</b>	Ingeniero en Software
<b>Responsabilidades</b>	Desarrollo, implementación y testing módulos.
<b>Información de Contacto</b>	<a href="mailto:enrique.merizalde@udla.edu.ec">enrique.merizalde@udla.edu.ec</a>
<b>Aprobación</b>	Aprobado

<b>Nombre</b>	Jossue Játiva
<b>Rol</b>	Desarrollador & QA
<b>Categoría Profesional</b>	Ingeniero en Software
<b>Responsabilidades</b>	Desarrollo, implementación y testing módulos principales.
<b>Información de Contacto</b>	<a href="mailto:jossue.jativa@udla.edu.ec">jossue.jativa@udla.edu.ec</a>
<b>Aprobación</b>	Aprobado

d) Acrónimos y Abreviaturas

- **ChauchiSoft:** Nombre de la plataforma.
- **CRUD:** Acrónimo para *Create, Read, Update, Delete*, que define las operaciones básicas de bases de datos.
- **Transacción:** Operación de venta registrada en el sistema.

e) Referencias:

Referencia	Título	Ruta	Fecha	Autor
1	Perfil de Proyecto Validación Software	<a href="#">Perfil de Proyecto Validación Software</a>	12 de noviembre de 2024	Enrique Merizalde, Juan Aristizábal y Jossue Játiva

Descripción General

a) Perspectiva del Producto

El producto ChauchiSoft es una plataforma independiente diseñada específicamente para comerciales informales. Esta no forma parte ni depende de otro sistema mayor. Este sistema tiene como objetivo principal satisfacer las necesidades de los comerciantes al facilitar la organización, búsqueda eficiente de productos y registro de transacciones para mejorar la experiencia tanto para los vendedores como para los compradores. El sistema se ejecutará localmente, permitiendo a los comerciantes operar incluso en condiciones de conectividad limitada o inexistente

b) Funcionalidad:

*Chauchisoft* incluye las siguientes funcionalidades principales:

- **Administración de Productos:** Registro, visualización, edición y eliminación de productos con sus atributos, además de organización de los productos en categorías personalizables.

- **Administración de Clientes:** Registro, edición y eliminación de clientes. Además, gestión de necesidades específicas de los clientes mediante descripciones.
- **Cálculo de Ganancias:** Cálculo automático de ganancias en base al precio de venta de los productos y porcentaje de ganancia.
- **Gestión de Inventario:** Actualización automática del inventario tras una venta y capacidad de búsqueda rápida para verificar la disponibilidad de productos.
- **Registro de Ventas:** Registro de transacciones incluyendo fechas, clientes, productos vendidos y el precio total. Además, historial de ventas con filtros por fecha, cliente o categoría.

c) **Características de los Usuarios**

Tipo de usuario	Comerciante informal
Formación	Secundaria completa; algunos con formación técnica o superior.
Habilidades	<ul style="list-style-type: none"> <li>- Conocimientos básicos en el manejo de computadores.</li> <li>- Capacidad para aprender rápidamente a usar herramientas digitales.</li> <li>- Experiencia mínima en ventas y gestión de productos.</li> </ul>
Actividades	<ul style="list-style-type: none"> <li>- Registrar productos en el sistema.</li> <li>- Actualizar el inventario según las ventas realizadas.</li> <li>- Consultar historial de transacciones par análisis de ventas y ganancias.</li> </ul>

d) **Restricciones**

- **Hardware:** El sistema debe ser ejecutable en computadoras con recursos limitados (CPU básica, 4 GB de RAM, 500 MB de almacenamiento libre).
- **Sistema Operativo:** Compatible con Windows (versiones recientes).



- **Metodología de Desarrollo:** Uso una metodología ágil para garantizar un desarrollo flexible y abierto a cambio, durante todas las etapas de software
- **Idiomas:** El sistema será desarrollado en español para adaptarse al mercado objetivo.
- **Seguridad:** Asegurar que los datos del comerciante se almacenen localmente con encriptación básica.

**e) Suposiciones y Dependencias**

- **Disponibilidad del Hardware:** Se asume que los comerciantes cuentan con una computadora funcional y básica.
- **Conectividad Limitada:** Aunque el sistema no depende de internet, se asume que los comerciantes tienen acceso ocasional a la red para actualizaciones futuras.
- **Experiencia del Usuario:** Se supone que los comerciantes pueden manejar herramientas simples con entrenamiento inicial.
- **Experiencia del Usuario en Ventas:** Se supone que los comerciantes que utilicen la plataforma ya tendrán cierta experiencia previa en ventas informales de productos lo que les permitirá comprender el sistema rápidamente.
- **Mantenimiento:** La evolución del sistema dependerá del soporte técnico proporcionado por los desarrolladores.

**f) Evolución Previsible del Sistema**

A futuro, se espera incorporar las siguientes mejoras:

- Soporte para múltiples idiomas, ampliando su uso en regiones con diferentes lenguas.
- Integración con sistemas en la nube para respaldos automáticos y sincronización de datos.
- Módulo de reportes avanzados, permitiendo análisis gráficos de ventas y tendencias.
- Aplicación móvil complementaria para que los comerciantes gestionen su negocio desde un dispositivo portátil.

#### g) Productos Esperados

- **Software Ejecutable:** Una versión estable del sistema *ChauchiSoft*.
- **Manual del Usuario:** Documento explicativo que detalle el uso del sistema, desde la instalación hasta las funcionalidades.
- **Código Fuente Documentado:** Para permitir futuras modificaciones o mejoras.
- **Plan de Capacitación:** Materiales y talleres para instruir a los comerciantes en el uso del sistema.

#### h) Aspectos Generales de la Metodología

El proyecto se desarrollará utilizando una metodología ágil que divide el trabajo en *sprints*, donde inicialmente se tendrá la recolección de información y documentación. Luego se tiene la fase de análisis y diseño de las soluciones, el desarrollo del sistema, las pruebas del sistema y finalmente la implementación.

#### i) Información a Disposición

Se ha realizado una pequeña investigación sobre temas relacionados con la generación de ingresos extra, venta de vehículos y comercio electrónico en Ecuador, para poder conocer como *ChauchiSoft* se podría integrar a este mercado.

- **Venta de Vehículos Usados en Quito:** Según una publicación de “El Comercio”, realizada el 22 de mayo de 2024, el mercado de vehículos usados en Ecuador ha experimentado un crecimiento significativo en la demanda y oferta debido a diversos factores económicos y del sector automotor. Entre las principales razones están la eliminación de restricciones para la importación de vehículos nuevos, la baja progresiva de aranceles para autos europeos, y el encarecimiento de autos nuevos debido a la depreciación del dólar frente a monedas como el yen. Esto ha hecho que los autos usados sean más atractivos para los consumidores. En ferias como las de Guamaní (Quito), se observa un aumento en los precios de los vehículos usados, con incrementos de hasta 10%. Sin embargo, los autos usados siguen siendo una opción popular debido a su menor depreciación anual en comparación con otros países, su funcionalidad, y el respaldo de marcas confiables como Chevrolet, Hyundai y Kia. Por cada auto

nuevo vendido, se comercializan dos o tres usados, según datos de portales especializados como Patiotuerca.

- Según un artículo publicado por “El Comercio” el 27 de diciembre de 2022, se aborda cómo generar ingresos adicionales trabajando en línea, destacando varias opciones viables en 2024. Entre estas, se incluyen:
  - **Plataformas de Encuestas y Tareas Simples:** Aplicaciones como Poll Play permiten a los usuarios ganar dinero respondiendo preguntas o realizando tareas pequeñas.
  - **Freelancing:** Sitios como Freelancer.com demandan habilidades en programación, diseño gráfico (Illustrator, InDesign), edición de video/audio, y marketing digital (SEM), así como conocimiento de idiomas como inglés o árabe.
  - **Minería de Datos y Diseño 3D:** Son sectores en crecimiento donde se buscan expertos en análisis de información y modelado 3D.

Aunque estas oportunidades pueden generar ingresos extras, los montos suelen ser bajos y requieren constancia. Además, cada plataforma tiene sus reglas específicas. Estas actividades son ideales para complementar ingresos principales

- Según un artículo publicado por “El Comercio”, el 4 de marzo de 2023, se describe cómo los negocios en Quito se han adaptado a una combinación de entornos virtuales y presenciales, un enfoque impulsado por la necesidad de mantener operaciones durante y después de la pandemia. Entre las estrategias destacadas están la adopción de comercio electrónico, el uso de redes sociales, y plataformas como WhatsApp para ventas, además de pagos digitales para garantizar seguridad y conveniencia. La transformación digital aún enfrenta desafíos, como la falta de madurez del mercado en segmentos pequeños. Sin embargo, iniciativas de capacitación, como las ofrecidas por *ConQuito*, están ayudando a miles de emprendedores a adaptarse a esta nueva realidad.

## Requisitos Específicos

### RF 1: Gestión de Inventario de Productos

- **RF 1.1:** El sistema deberá permitir el registro de productos con campos obligatorios como por ejemplo un nombre, una descripción, categoría, estado (nuevo/usado), numero propietario, precio y ganancias
- **RF 1.2:** Los usuarios podrán buscar productos por múltiples criterios como por ejemplo su estado, rango de precios, categoría y nombre
- **RF 1.3:** Se podrán cargar imágenes de los productos para su visualización en el sistema
- **RF 1.4:** El inventario debe permitir editar o eliminar registros existentes.

### RF 2: Gestión de Ventas

- **RF 2.1:** El sistema permitirá registrar las ventas realizadas vinculadas a clientes y vehículos seleccionados.
- **RF 2.2:** Se deberán generar automáticamente logs de las ventas realizadas con información del cliente, y de la ganancia que se tuvo en esa venta.
- **RF 2.3:** Se deberá tener una fecha y se deberá poder organizar las ventas por fecha.

### RF 3: Administración de Clientes:

- **RF 2.2:** Se deberán generar automáticamente logs de las ventas realizadas con información del cliente, y de la ganancia que se tuvo en esa venta.
- **RF 2.3:** Se deberá tener una fecha y se deberá poder organizar las ventas por fecha.

### RF 4: Generación de Logs:

- **RF 4.1:** El sistema deberá generar reportes de las ventas organizadas y filtradas por fechas. Los reportes deberán ser almacenados y deben poder exportarse o imprimirse.

## a) Requisitos Comunes de las Interfaces

### Entradas al Sistema

Productos	<ul style="list-style-type: none"><li>• Campos como: nombre, descripción, categoría, estado (nuevo/usado), descripción del propietario, precio y ganancias</li><li>• Ingreso de criterios de búsqueda como categoría, nombre, estado, descripción y precio</li></ul>
Clientes	<ul style="list-style-type: none"><li>• Campos como: nombre, contacto, dirección y descripción de lo que busca</li></ul>
Ventas	<ul style="list-style-type: none"><li>• Datos del cliente, datos del producto y ganancia obtenida</li></ul>
Reportes	<ul style="list-style-type: none"><li>• Parámetros para generación: rango de fechas, categorías, nombres, estado, rango de precios</li></ul>

### Salida del Sistema

- Lista de productos filtrada con todos sus detalles
- Listado de ventas realizadas según los parámetros ingresados
- Tablas de información de ventas que pueden ser descargadas o impresas.

#### ❖ Interfaces de Usuario

- El sistema debe ofrecer una interfaz gráfica amigable y responsive basada en las interfaces de OpenXava
- Se tendrá un menú lateral con las opciones del sistema además de todas las opciones de CRUD dentro de cada sección
- Además, se tendrá utilizar la gama de tonos azules o grises de OpenXava.
- La interfaz debe incluir un panel de búsqueda, panel de detalles de productos y clientes, y panel para registrar ventas y generar los logs de esas.

#### ❖ Interfaces de Hardware

- El sistema debe ser compatible para impresión de tablas que se vean en pantalla
- El sistema debe poder responder con facilidad y eficiencia a la búsqueda de productos

- El sistema de permitir subir archivos multimedia para poder subir las fotos de los productos
- ❖ **Interfaces de Software**
  - El sistema debe integrarse y/o brindar información relevante para sistemas de software como Excel para llevar cuentas.
  - Módulo log de ventas
  - Se podrán visualizar logs de los pagos realizados por las personas.
- ❖ **Interfaces de Comunicación**
  - La comunicación se realizará utilizando protocolos como HTTPS para garantizar la seguridad de los datos
  - El sistema correrá de manera local en las computadoras comunicando el front y el backend utilizando Java y OpenXava.

**b) Casos de Uso**

<b>Caso de Uso 1</b>	<i>Login</i>
<b>Actores</b>	<b>Usuario (Administrador)</b>
<b>Descripción</b>	<i>El administrador puede ingresar al sistema con sus credenciales de administrador para poder acceder a todas las demás funcionalidades.</i>

<b>Caso de Uso 2</b>	<i>Gestión de Clientes</i>
<b>Actores</b>	<i>Usuario (Administrador), Cliente</i>
<b>Descripción</b>	<i>El administrador puede gestionar la creación, edición, visualización y eliminación de clientes. Este caso de uso permite registrar a los clientes que desean adquirir un producto o servicio</i>

<b>Caso de Uso 3</b>	<i>Gestión de productos</i>
<b>Actores</b>	<i>Administrador: Usuario con permisos para gestionar productos</i>
<b>Descripción</b>	<i>Este caso de uso permite al Administrador realizar acciones de gestión de productos, como agregar, editar, eliminar y visualizar productos en el sistema.</i>

<b>Caso de Uso 4</b>	<i>Gestión de reporte de venta</i>
<b>Actores</b>	<i>Administrador: usuario con permisos para generar reportes de ventas</i>
<b>Descripción</b>	<i>Este caso de uso permite al Administrador generar un reporte detallado de las ventas realizadas, incluyendo información sobre los productos vendidos, cantidades, precios, cliente y el total de la venta. El reporte también puede incluir filtros como fecha, cliente y productos.</i>

<b>Caso de Uso 5</b>	<i>Gestión de facturación</i>
<b>Actores</b>	<i>Administrador: Usuario con permisos para generar facturas.</i>
<b>Descripción</b>	<i>El administrador puede generar facturas de ventas. El sistema permite almacenar la factura en el software.</i>

<b>Caso de Uso 6</b>	<i>Gestión de reporte de inventario</i>
<b>Actores</b>	<i>Administrador: usuario con permisos para gestionar inventario</i>
<b>Descripción</b>	<i>El administrador puede gestionar el inventario, actualizando la cantidad de productos disponibles y verificando el stock en tiempo real.</i>

<b>Caso de Uso 7</b>	<i>Agregar inventario</i>
<b>Actores</b>	<i>Administrador: Usuario con permisos para agregar productos al inventario</i>
<b>Descripción</b>	<i>El administrador puede agregar nuevas existencias de productos al inventario, manteniendo el stock actualizado en tiempo real.</i>

<b>Caso de Uso 8</b>	<i>Gestión de ganancia</i>
<b>Actores</b>	<i>Administrador: Usuario con permisos para calcular ganancias</i>
<b>Descripción</b>	<i>El administrador puede calcular la ganancia de los productos esperada ingresando el precio base del producto y el porcentaje de ganancia deseado. El sistema realiza el cálculo automáticamente.</i>

<b>Caso de Uso 9</b>	<i>Gestión de Categorías</i>
<b>Actores</b>	<i>Administrador: Usuario con permisos para gestionar categorías</i>
<b>Descripción</b>	<i>El administrador puede crear categorías a las cuales los productos pertenecerán con el fin de agruparlos en cierto orden tanto para organización interna como para filtrar al momento de intentar realizar una venta</i>

### c) Requisitos No Funcionales

#### ❖ Requisitos de rendimiento

- El sistema debe manejar hasta 50 usuarios concurrentes sin degradación del rendimiento.



- La respuesta para las búsquedas en el inventario debe ser menor a 2 segundos para un inventario de 10,000 productos.

❖ **Seguridad**

- Todos los datos deben ser cifrados mediante técnicas como AES-256.
- Los usuarios deben autenticarse con credenciales únicas al sistema
- El sistema debe registrar todas las actividades en un log de auditoría.

❖ **Fiabilidad**

- El sistema deberá operar con un tiempo medio entre fallos (MTBF) de al menos 10,000 horas

❖ **Disponibilidad**

- El sistema deberá garantizar una disponibilidad del 99.9% mensualmente, tomando en cuenta que este es un sistema que se ejecuta a demanda, es decir cada que el usuario lo necesita debería estar disponible.

❖ **Mantenibilidad**

- Las tareas de mantenimiento del sistema, como actualizaciones de versiones

❖ **Portabilidad**

- El sistema debe ser compatible con bases de datos MySQL.
- Debe ser fácilmente desplegable en sistemas operativos Windows, Linux.

**d) Otros Requisitos**

❖ **Requisitos Legales**

- Cumplir con la normativa de facturación electrónica del país.
- Garantizar la protección de datos personales según las leyes locales

❖ **Requisitos Culturales**

- El sistema deberá permitir localización de idioma para mostrar la interfaz en español o inglés según las preferencias del usuario.

**Estándares de Conformidad**

**a) Etapa 1: Documentación Inicial y Requisitos**

❖ **Perfil de Proyecto:**

- **Objetivo:** Definición clara de los propósitos y metas del sistema.
- **Alcance:** Identificación de los módulos y funcionalidades principales.

- **Justificación:** Explicación de la relevancia del proyecto para los comerciantes informales.
- **Supuestos y Riesgos:** Contextualización de las condiciones de uso y los posibles desafíos.
- **Restricciones:** Límites técnicos, de recursos o tiempo que puedan impactar el desarrollo.
- **Cronograma de Trabajo:** Planificación detallada de las etapas de desarrollo.

❖ **Especificación de Requisitos (SRS)**

- Requisitos funcionales y no funcionales
- Casos de uso iniciales
- Restricciones técnicas y de diseño
- El SRS debe ser exhaustivo, relevante y alineado con las necesidades de los usuarios

❖ **Modelos de Caso de Uso**

- Desarrollo de diagramas de casos de uso que representen de manera visual las interacciones entre los usuarios y el sistema.
- Incluyen escenarios clave como: gestión de inventario, registro de ventas y búsqueda de productos.
- Los casos de uso deben ser claros y detallados para facilitar su implementación y pruebas posteriores.

**b) Etapa 2: Diseño y Arquitectura**

❖ **Diagramas de Arquitectura C4**

- Presentación de un diagrama basado en el modelo C4 (Contexto, Contenedor, Componente y Código).
- **Contexto:** Descripción del sistema y sus interacciones con usuarios y sistemas externos.
- **Contenedores:** Componentes principales como *frontend*, *backend* y base de datos.
- **Componentes:** Detalles de cada módulo dentro del sistema.
- Este diagrama servirá como guía para los desarrolladores en la implementación de la solución

❖ **Diagrama de Clases**

- Creación de diagramas que definan las clases base del sistema, sus atributos, métodos y relaciones.
- Los diagramas deben ser extensibles y contemplar futuras integraciones o modificaciones.

❖ **Prototipos e Interfaz**

- Desarrollo de prototipos visuales para validar con los usuarios: Menús, paneles de búsqueda y registros.
- Opciones de CRUD (Crear, Leer, Actualizar y Eliminar) en las secciones principales.

**c) Etapa 3: Implementación y Pruebas**

❖ **Sprints de Desarrollo**

- Uso de *sprints* cortos (aproximadamente 1 semana) para implementar los casos de uso definidos.
- **Cada sprint debe incluir:**
  - **Planificación:** Selección de funcionalidades a implementar.
  - **Desarrollo:** Codificación siguiendo estándares de calidad y seguridad.
  - **Pruebas:** Validación funcional y corrección de errores antes de avanzar al siguiente sprint.

❖ **Revisión Continua**

- Revisión regular del código y la documentación generada.
- Realización de pruebas integradas para validar la interacción entre los diferentes módulos.

❖ **Entrega Parcial y Feedback**

- Entregas incrementales al final de cada sprint para recolectar retroalimentación de los usuarios finales.
- Ajustes en función de las observaciones para garantizar que el producto final sea útil y relevante.

**2.2. Historias de Usuario**

Las historias de usuario son una herramienta fundamental para definir las funcionalidades que debe tener el sistema *ChauchiSoft* desde la perspectiva de los usuarios. En esta sección, se presentan los casos específicos que reflejan las necesidades y expectativas de los comerciantes informales al interactuar con la plataforma. Cada historia de usuario está estructurada para describir el

contexto, las acciones y los resultados esperados, con el fin de guiar el desarrollo y la implementación de las funcionalidades más relevantes. A continuación, se detallan las historias de usuario que serán implementadas en el sistema, las cuales corresponden a diferentes aspectos clave de la gestión de productos, ventas e inventario.

<b>Como</b>	Comerciante
<b>Quiero</b>	Poder registrar los productos que poseo
<b>Para</b>	Poder tener mejor organización y posibilidades de venta

<b>Como</b>	Comerciante
<b>Quiero</b>	Poder editar los productos que tengo y cambiar su precio o detalles
<b>Para</b>	Poder tener información actualizada de los productos que se desea vender.

<b>Como</b>	Comerciante
<b>Quiero</b>	Poder eliminar los productos manualmente si ya no están disponibles
<b>Para</b>	No ofrecer productos que no poseo

<b>Como</b>	Comerciante
<b>Quiero</b>	Que los productos vendidos se eliminen de forma automática
<b>Para</b>	Garantizar que el inventario esté actualizado

<b>Como</b>	Comerciante
<b>Quiero</b>	Visualizar los productos registrados
<b>Para</b>	Tener control sobre mi inventario

<b>Como</b>	Comerciante
<b>Quiero</b>	Organizar mis productos en categorías personalizables
<b>Para</b>	Encontrar rápidamente productos según las necesidades de mis clientes.

<b>Como</b>	Comerciante
<b>Quiero</b>	Imprimir el detalle de mi inventario
<b>Para</b>	Tener un respaldo físico

<b>Como</b>	Comerciante
<b>Quiero</b>	Agregar fotos de mis productos al registrarlos o editarlos
<b>Para</b>	Tener imágenes que mostrar a mis clientes

<b>Como</b>	Comerciante
<b>Quiero</b>	Poder registrar mis clientes con sus necesidades
<b>Para</b>	Poder relacionar rápidamente productos con sus necesidades

<b>Como</b>	Comerciante
<b>Quiero</b>	Poder editar mis clientes
<b>Para</b>	Poder tener información actualizada sobre su contacto y necesidades.

<b>Como</b>	Comerciante
<b>Quiero</b>	Poder eliminar clientes
<b>Para</b>	No llenar el sistema de clientes que ya no desean mis productos

<b>Como</b>	Comerciante
<b>Quiero</b>	Visualizar mis clientes
<b>Para</b>	Tener información sobre todos ellos

<b>Como</b>	Comerciante
<b>Quiero</b>	Que el sistema calcule automáticamente la posible ganancia a obtener por cada producto
<b>Para</b>	Conocer cuánto dinero puedo obtener tras la venta de un producto

<b>Como</b>	Comerciante
<b>Quiero</b>	Calcular automáticamente la posible ganancia a obtener por cada producto
<b>Para</b>	Poder saber cuánto dinero puedo obtener tras la venta de un producto

<b>Como</b>	Comerciante
<b>Quiero</b>	Registrar cada venta con detalles como fecha, cliente, producto vendido y precio total
<b>Para</b>	Para llevar un historial organizado de mis transacciones.

<b>Como</b>	Comerciante
<b>Quiero</b>	Buscar productos en mi inventario utilizando distintos criterios
<b>Para</b>	Encontrar rápidamente lo que necesito para mis clientes

<b>Como</b>	Comerciante
<b>Quiero</b>	Poder utilizar el sistema sin necesidad de conexión a internet
<b>Para</b>	Seguir operando incluso durante interrupciones de conectividad o luz

### 2.3. Descripción Funcionalidad Time to Market

La herramienta Time to Market utilizada para la creación e implementación del sistema ChauchiSoft es OpenXava

#### a) ¿Qué es OpenXava?

OpenXava es un framework de código abierto para el desarrollo rápido de aplicaciones empresariales Java. Este marco de trabajo permite a los desarrolladores crear aplicaciones web de forma eficiente y con menos código que otros frameworks tradicionales. Su enfoque principal es facilitar la creación de aplicaciones orientadas a la gestión de datos, como sistemas de información empresariales, CRMs y ERPs. OpenXava es ideal para aplicaciones que requieren formularios extensos y una gestión detallada de datos estructurados.

#### b) Historia de OpenXava

OpenXava fue creado por Javier Paniza en 2005 como una alternativa a los frameworks más complejos que existían en ese momento para desarrollar aplicaciones empresariales. La idea principal surgía de la necesidad de tener un sistema que combinara productividad, simplicidad y flexibilidad. El framework se inspiró en técnicas de desarrollo como las de Visual Age para Smalltalk, pero aplicadas al ecosistema Java. Desde entonces, ha evolucionado gracias a una comunidad activa de desarrolladores y empresas que lo utilizan para proyectos reales en distintos sectores.

**c) ¿Para qué se Utiliza?**

OpenXava se utiliza principalmente para desarrollar aplicaciones empresariales que necesitan una interfaz de usuario web sofisticada y funcional. Los desarrolladores lo emplean para:

- **Sistemas de Gestión Empresarial:** Creación de ERPs, CRMs y sistemas de gestión de inventarios.
- **Aplicaciones Financieras:** Sistemas de facturación, control de gastos y administración de presupuestos.
- **Aplicaciones Educativas:** Gestión de alumnos, cursos y matrículas en instituciones educativas.
- **Aplicaciones Personalizadas:** Sistemas adaptados a necesidades específicas de empresas o industrias.

Finalmente, se utilizará OpenXava en este proyecto precisamente por su enfoque en la creación de sistemas fáciles orientados a la gestión de datos, ya que, al estar basado en java, un lenguaje conocido y de estructura muy simple de entender y aplicar, hace que OpenXava sea una gran herramienta para poder implementar este sistema.

**d) ¿Cómo Funciona?**

OpenXava se basa en Java y utiliza anotaciones JPA (Java Persistence API) para definir el modelo de datos, lo que permite generar automáticamente la lógica de negocio y la interfaz de usuario. Al describir los datos mediante anotaciones en las clases del modelo, OpenXava genera formularios, listas y gráficos sin necesidad de escribir código adicional.

El framework también permite la integración con otros sistemas y servicios, como bases de datos relacionales (MySQL, PostgreSQL, etc.), API REST y sistemas de autenticación como OAuth.

**e) Datos Curiosos OpenXava**

- **Alta Productividad:** Se dice que el desarrollo con OpenXava es hasta cuatro veces más rápido que con frameworks tradicionales como Spring o Hibernate, gracias a su capacidad de generar gran parte del código repetitivo de forma automática.
- **Internacionalización:** Soporta múltiples idiomas de manera nativa, lo que facilita su uso en proyectos globales.
- **Extensibilidad:** Aunque su configuración inicial es sencilla, ofrece un alto grado de personalización y extensibilidad para cubrir necesidades específicas.

- **Comunidad Activa:** Existen foros, tutoriales y documentación extensa que respaldan su uso.
- **Casos de Éxito:** Ha sido adoptado por universidades, pymes y grandes corporaciones alrededor del mundo.

**f) Ventajas de OpenXava**

- **Curva de Aprendizaje Corta:** Ideal para desarrolladores con experiencia básica en Java.
- **Reducción de Código:** Se concentra en el modelo de negocio, dejando la generación de la interfaz al framework.
- **Compatibilidad:** Funciona con cualquier servidor de aplicaciones Java, como Tomcat o JBoss.

**g) Desafíos de OpenXava**

- **Dependencia del Framework:** Aunque es muy útil, puede resultar limitante para proyectos con necesidades muy específicas fuera del alcance del marco.
- **Documentación en Constante Evolución:** La rapidez con que se actualiza el framework puede dejar obsoletos algunos recursos.

## **2.4. Descripciones Técnicas de Prueba**

Se han optado por realizar 4 distintos tipos de prueba para la validación y verificación del sistema *ChauchiSoft*. Estas técnicas son, pruebas de caja blanca, clases de equivalencia (pruebas de caja negra), valores limite (pruebas de caja negra) y casos de uso (pruebas de caja negra), pero ¿Qué son estas pruebas?

En primer lugar, debemos saber **¿Qué son las técnicas dinámicas?**

Las técnicas dinámicas de pruebas son métodos utilizados para validar el correcto funcionamiento de un sistema al evaluar su comportamiento durante la ejecución. Estas incluyen tanto técnicas de caja blanca, centradas en la estructura interna del software, como técnicas de caja negra, que se enfocan en la funcionalidad observable.

**a) Pruebas de Caja Blanca**

La técnica de pruebas de caja blanca se basa en analizar el código fuente para diseñar casos de prueba que aseguren una cobertura estructural completa. Esta técnica se enfoca en validar los elementos internos del sistema, como



sentencias, decisiones, condiciones y caminos, buscando inconsistencias o código muerto. Entre los tipos de cobertura utilizados en esta técnica se encuentran:

- **Cobertura de Sentencias:** Asegura que cada sentencia ejecutable en el programa sea ejecutada al menos una vez. Por ejemplo, si un programa contiene 100 sentencias y 80 de ellas son ejecutadas durante la prueba, la cobertura será del 80%.
- **Cobertura de Decisiones:** Evalúa todas las decisiones (como estructuras if-else) para garantizar que cada posible resultado (verdadero o falso) se haya ejecutado al menos una vez. Este tipo de cobertura incluye la de sentencias, pero también valida el flujo de control.
- **Cobertura de Caminos:** Comprueba todos los caminos posibles en un diagrama de flujo del programa, desde el inicio hasta el fin. Aunque proporciona una cobertura más exhaustiva, puede ser difícil de alcanzar en sistemas complejos debido al número de combinaciones posibles.

Adicionalmente, la complejidad ciclomática es un concepto importante en las pruebas de caja blanca. Es una métrica utilizada para determinar la cantidad de caminos linealmente independientes en un programa. Esta complejidad se calcula mediante diferentes métodos, entre los que destacan:

- **Fórmula de McCabe:** Complejidad ciclomática ( $V$ ) =  $E - N + 2$ , donde  $E$  representa el número de aristas en el grafo de control de flujo y  $N$  el número de nodos.
- **Formula Nodos Predicados:** Complejidad ciclomática ( $V$ ) =  $NP + 1$ , donde  $NP$  son Nodos Predicados es decir nodos donde se realiza una decisión, visualmente estos nodos tienen 2 o más líneas que salen de él.
- **Método de Regiones:** Contando las regiones cerradas en el grafo de control del programa.

Un alto valor de complejidad ciclomática indica que el código puede ser difícil de entender, mantener o probar. Diseñar pruebas que cubran caminos independientes es crucial para garantizar la calidad del software.

La implementación de estas coberturas requiere identificar los elementos del código a probar, diseñar casos de prueba específicos y también utilizar herramientas externas como SonarQube para medir el nivel de cobertura alcanzado. Aunque es una técnica muy útil, su principal desventaja es que

requiere conocimientos del código y la creación de grafos que representen los algoritmos de este para poder realizar las evaluaciones del código

#### **b) Pruebas de Caja Negra (Clases de Equivalencia)**

La técnica de clases de equivalencia consiste en agrupar los valores de entrada en categorías o clases que comparten comportamientos esperados similares. Esta agrupación permite reducir significativamente la cantidad de casos de prueba necesarios, mientras se asegura una buena cobertura. Para implementarla, se identifican los rangos de valores válidos e inválidos y se selecciona un representante de cada clase como base para un caso de prueba. Por ejemplo, si un sistema acepta números entre 1 y 10, los valores podrían agruparse en clases válidas ( $1 \leq x \leq 10$ ) e inválidas ( $x < 1$ ,  $x > 10$ ), y se probaría un representante de cada clase. Una vez que se tengan los representantes de cada clase, se podrán ejecutar las pruebas para validar el funcionamiento correcto o incorrecto del sistema

#### **c) Pruebas Caja Negra (Valores Límite)**

El análisis de valores límite complementa a las clases de equivalencia, concentrándose en los límites de los rangos donde los errores son más probables. Esta técnica prueba los valores justo en el límite, dentro del límite y fuera del límite. Por ejemplo, para un rango de  $0 \leq x \leq 100$ , los casos de prueba incluirían valores como  $x = 0$ ,  $x = 1$ ,  $x = 99$ ,  $x = 100$ ,  $x = -1$  y  $x = 101$ . Este enfoque ayuda a identificar defectos relacionados con la implementación incorrecta de restricciones. La prueba con valores límites es de suma importancia ya que nos permite buscar comportamientos no esperados del sistema tomando en cuenta las restricciones que se han colocado en el mismo, y estas al estar al límite pueden generar algún tipo de falla o error.

#### **d) Pruebas Caja Negra (Casos de Uso)**

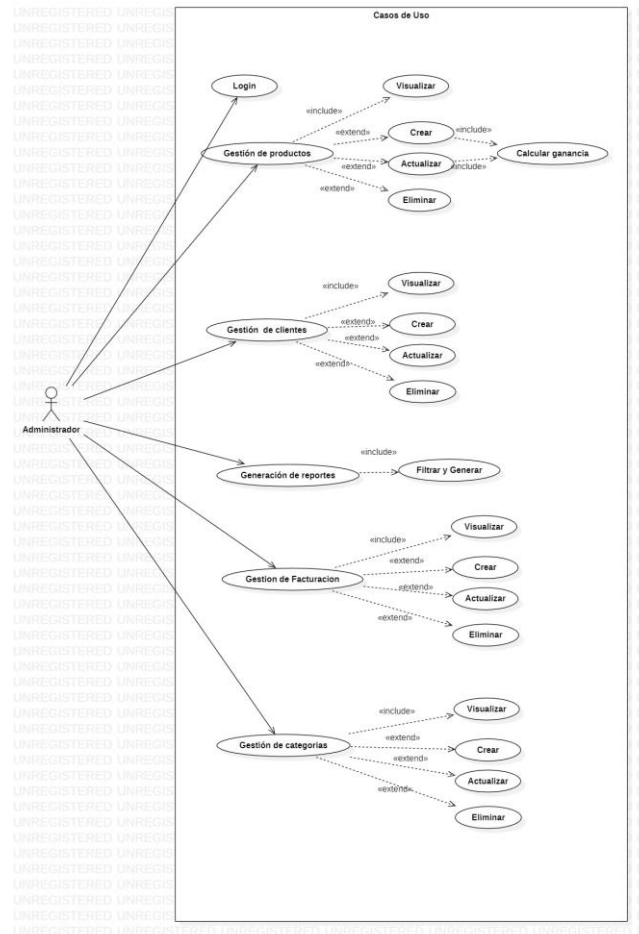
Las pruebas basadas en casos de uso se derivan directamente de los escenarios descritos en los casos de uso del sistema. Cada caso de uso detalla una interacción entre el usuario y el sistema, incluyendo precondiciones y resultados esperados. Para implementar esta técnica, se extraen casos de uso del diseño del sistema y se crean casos de prueba que cubran tanto los flujos principales como los alternativos. Por ejemplo, en un sistema de biblioteca, un caso de uso podría ser "Tomar prestado un libro", con una precondición de que

el usuario esté registrado y el libro disponible. Los pasos incluirían seleccionar el libro y solicitar el préstamo, mientras que el resultado esperado sería una confirmación del préstamo y la actualización del inventario. Estas pruebas son útiles para validar la funcionalidad desde la perspectiva del usuario

### 3. Propuesta Técnica:

#### 3.1. Casos de Uso

El sistema *ChauchiSoft* presenta varios distintos casos de uso, que nos permitirán conocer como el usuario interactúa con el sistema. Para eso se ha realizado el siguiente diagrama de casos de uso con todas las acciones que puede realizar el usuario.



Cada uno de estos casos de uso serán explicados de mejor manera en los siguientes cuadros de casos de uso.

Identificador	Login		
Descripción	El administrador puede ingresar al sistema con sus credenciales de administrador para poder acceder a todas las demás funcionalidades.		
Actores	Usuario (Administrador)		
Pre condiciones	El administrador debe conocer sus credenciales para iniciar sesión		
Post condiciones	El sistema ingresa al usuario correctamente al sistema		
Secuencia Normal	#	Acción (actor)	Reacción (sistema)
	1	El administrador ingresa su usuario correcto	El sistema valida el usuario
	2	El administrador ingresa su contraseña correcta	El sistema deberá ingresar al dashboard inicial
Excepciones	#	Acción (actor)	Reacción (sistema)
	p.1	Si el administrador ingresa credenciales incorrectas. (usuario o contraseña)	El sistema muestra un mensaje de error y no permite el ingreso
Rendimiento	El sistema deberá realizar las acciones descritas en un máximo de <b>2 segundos</b> .		
Frecuencia	Este caso de uso se espera que se ejecute un promedio de <b>5 veces por día</b> .		

<b>Importancia</b>	<b>Vital.</b> Este caso de uso es crucial para el sistema ya que su ingreso seguro garantiza la seguridad de la data y sobre todo sin el login no se puede acceder al sistema completamente
<b>Urgencia</b>	<b>Inmediatamente.</b> El inicio de sesión es crucial para la aplicación.
<b>Comentarios</b>	En este caso al tener únicamente un usuario administrador él tiene sus credenciales únicas para ingresar al sistema.

Identificador	Gestión de Clientes		
<b>Descripción</b>	El administrador puede gestionar la creación, edición, visualización y eliminación de clientes. Este caso de uso permite registrar a los clientes que desean adquirir un producto o servicio		
<b>Actores</b>	Usuario (Administrador), Cliente		
<b>Pre condiciones</b>	El administrador debe haber iniciado sesión correctamente en el sistema El administrador debe tener la información de un cliente que desea registrar		
<b>Post condiciones</b>	El sistema guarda correctamente los datos del usuario creado, editado o eliminado.		
<b>Secuencia Normal</b>	<b>#</b>	<b>Acción (actor)</b>	<b>Reacción (sistema)</b>
	1	El administrador inicia sesión en el sistema	El sistema verifica las credenciales y permite el acceso
			Si el usuario ingresa correctamente sus credenciales el sistema deberá ingresar al

			dashboard del sistema.
	2	El administrador selecciona la pestaña de clientes	El sistema muestra la lista de clientes actuales y opción para crear uno nuevo
		<ul style="list-style-type: none"> <li>2.1 Si el administrador selecciona "Editar"</li> <li>2.2 Si el administrador selecciona "Eliminar"</li> </ul>	<ul style="list-style-type: none"> <li>2.1 El sistema despliega un formulario con los datos actuales del usuario para poder editarlos</li> <li>2.2 El sistema pide confirmación para eliminar y elimina al usuario seleccionado</li> </ul>
	3	El administrador elige crear un nuevo vendedor o comprador	El sistema despliega un formulario para introducir datos del cliente
	4	El administrador completa los datos requeridos y los guarda	El sistema valida la información y guarda los datos del cliente
	5	El sistema confirma la creación del cliente	El administrador puede visualizar al cliente usuario en la lista con sus demás clientes
<b>Excepciones</b>	<b>#</b>	<b>Acción (actor)</b>	<b>Reacción (sistema)</b>
	p.1	Si el administrador ingresa credenciales incorrectas.	El sistema muestra un mensaje de error.
	p.4	Si faltan datos obligatorios al crear un usuario.	El sistema notifica los campos faltantes y no guarda los cambios.
	p. 2.1	Si al editar se ingresan datos erróneos o incompletos	El sistema notifica los campos faltantes y no guarda los cambios.
<b>Rendimiento</b>	El sistema deberá realizar las acciones descritas en los pasos 1 al 5 en un máximo de <b>3 segundos</b> .		

<b>Frecuencia</b>	Este caso de uso se espera que se ejecute un promedio de <b>10 veces por semana</b> .
<b>Importancia</b>	<b>Vital.</b> Este caso de uso es crucial para la funcionalidad del sistema ya que se basa en los clientes que son un eje fundamental de la aplicación.
<b>Urgencia</b>	<b>Inmediatamente.</b> La gestión de clientes debe ejecutarse sin interrupciones.
<b>Comentarios</b>	Este caso de uso facilita la administración de roles y accesos, garantizando el funcionamiento adecuado del sistema con diferentes tipos de usuarios.

Identificador	Gestión de Productos		
<b>Descripción</b>	Este caso de uso permite al <b>Administrador</b> realizar acciones de gestión de productos, como agregar, editar, eliminar y visualizar productos en el sistema.		
<b>Actores</b>	<p>👤 Administrador: Usuario con permisos para gestionar productos.</p> <p>👤 Sistema: Aplicación donde se almacena y gestiona la información de productos.</p>		
<b>Pre condiciones</b>	El administrador debe haber iniciado sesión correctamente en el sistema		
<b>Post condiciones</b>	El sistema guarda correctamente los datos del producto creado, editado o eliminado.		
<b>Secuencia Normal</b>	<b>#</b>	<b>Acción (actor)</b>	<b>Reacción (sistema)</b>
	1	El administrador inicia sesión en el sistema	El sistema verifica las credenciales y permite el acceso

			<ul style="list-style-type: none"> <li>1.1 Si el usuario ingresa mal sus credenciales el sistema deberá notificarle del ingreso incorrecto de credenciales.</li> <li>1.2 Si el usuario ingresa correctamente sus credenciales el sistema deberá ingresar al dashboard del sistema.</li> </ul>
	2	El administrador selecciona la pestaña de productos	El sistema muestra la lista de productos y opciones para agregar, editar o eliminar.
	3	El administrador selecciona "Agregar Producto"	El sistema despliega un formulario para introducir los datos del producto.
	4	El administrador completa los datos requeridos y guarda	El sistema valida la información y guarda el producto en la base de datos.
	5	Si el usuario presiona en editar	El sistema mostrará los campos del producto que desea editar llenos para modificar lo que se deba modificar
	6	El administrador selecciona un producto y elige "Eliminar Producto"	El sistema pide confirmación antes de eliminar el producto.
	7	El administrador confirma la eliminación	El sistema elimina el producto seleccionado de la base de datos.



Excepciones	#	Acción (actor)	Reacción (sistema)
	p.1	Si el administrador ingresa credenciales incorrectas.	El sistema muestra un mensaje de error.
	p.4	Si faltan datos obligatorios al agregar o editar un producto.	El sistema notifica los campos faltantes y no guarda los cambios.
	p.6	Si el administrador intenta eliminar un producto que no existe.	El sistema muestra un mensaje de error notificando que el producto no se encuentra.
<b>Rendimiento</b>	El sistema deberá realizar las acciones descritas en los pasos 1 al 7 en un máximo de 5 segundos.		
<b>Frecuencia</b>	Este caso de uso se espera que se ejecute un promedio de 20 veces por semana.		
<b>Importancia</b>	Crucial. Este caso de uso es esencial para la administración de inventario ya que los productos forman parte de él y esto es esencial para el giro de negocio		
<b>Urgencia</b>	Inmediata. La gestión de productos debe ser fluida y accesible para el buen funcionamiento del sistema.		
<b>Comentarios</b>	Este caso de uso facilita la administración de los productos dentro del sistema, garantizando que el administrador pueda agregar, editar, eliminar y gestionar adecuadamente los productos sin dificultades.		

Identificador	Gestión de Reporte de Venta
<b>Descripción</b>	Este caso de uso permite al Administrador generar un reporte detallado de las ventas realizadas, incluyendo información sobre los productos vendidos, cantidades, precios, cliente y el total de la venta. El reporte también puede incluir filtros como fecha,

	cliente y productos.		
<b>Actores</b>	<p>⌚ Administrador: Usuario con permisos para generar reportes de ventas.</p> <p>⌚ Sistema: Aplicación que procesa las ventas y genera reportes.</p>		
<b>Pre condiciones</b>	El administrador debe haber iniciado sesión correctamente en el sistema y debe haber realizado al menos una venta.		
<b>Post condiciones</b>	El sistema genera correctamente un reporte de ventas, que puede ser exportado o visualizado en el formato solicitado (página para ver reportes).		
<b>Secuencia Normal</b>	<b>#</b>	<b>Acción (actor)</b>	<b>Reacción (sistema)</b>
	1	El administrador inicia sesión en el sistema	El sistema verifica las credenciales y permite el acceso
			<ul style="list-style-type: none"> <li>1.1 Si el usuario ingresa mal sus credenciales el sistema deberá notificarle del ingreso incorrecto de credenciales.</li> <li>1.2 Si el usuario ingresa correctamente sus credenciales el sistema deberá ingresar al dashboard del sistema.</li> </ul>
	2	El administrador selecciona la opción de "Generar reporte de ventas"	El sistema despliega opciones para seleccionar el rango de fechas, productos, clientes, etc.

	3	El administrador selecciona los filtros necesarios (por fecha, producto, etc.)	El sistema muestra una vista preliminar del reporte con los datos filtrados.
	5	El administrador elige la opción de exportar el reporte (página de reporte)	El sistema genera el reporte con la información seleccionada.
	6	El sistema confirma la generación del reporte	El administrador puede visualizar el reporte generado.
<b>Excepciones</b>	<b>#</b>	<b>Acción (actor)</b>	<b>Reacción (sistema)</b>
	p.1	Si no hay ventas registradas en el sistema.	El sistema muestra un mensaje notificando que no se encontraron ventas.
	p.4	Si el rango de fechas seleccionado no es válido.	El sistema muestra un mensaje de error indicando que el rango de fechas es inválido.
	p.6	Si el sistema encuentra un error al generar el reporte.	El sistema muestra un mensaje de error notificando la falla en la generación del reporte.
<b>Rendimiento</b>	El sistema deberá generar el reporte en un máximo de 10 segundos, dependiendo de la cantidad de datos a procesar.		
<b>Frecuencia</b>	Este caso de uso se espera que se ejecute un promedio de 5 veces por semana.		
<b>Importancia</b>	Alta. Este caso de uso es fundamental para el análisis de ventas y toma de decisiones dentro de la empresa, mas no crítico para las actividades cotidianas.		

<b>Urgencia</b>	Moderada. La generación de reportes debe realizarse rápidamente, pero no es una acción urgente que interrumpa otras funcionalidades críticas del sistema.
<b>Comentarios</b>	Este caso de uso facilita la toma de decisiones dentro de la empresa al proporcionar reportes detallados y bien estructurados, lo que permite al administrador analizar el desempeño de las ventas.

Identificador	Gestión de Facturación		
<b>Descripción</b>	El administrador puede generar facturas de ventas. El sistema permite almacenar la factura en el software.		
<b>Actores</b>	<ul style="list-style-type: none"> <li>Administrador: Usuario con permisos para generar facturas.</li> <li>Sistema: Aplicación encargada de gestionar la facturación.</li> </ul>		
<b>Pre condiciones</b>	El administrador debe haber iniciado sesión correctamente en el sistema y tener por lo menos un cliente y un producto registrados		
<b>Post condiciones</b>	El sistema almacena correctamente la factura generada.		
<b>Secuencia Normal</b>	<b>#</b>	<b>Acción (actor)</b>	<b>Reacción (sistema)</b>
	1	El administrador inicia sesión en el sistema	El sistema verifica las credenciales y permite el acceso.
			<ul style="list-style-type: none"> <li>1.1 Si el usuario ingresa mal sus credenciales el sistema deberá notificarle del ingreso incorrecto de credenciales.</li> <li>1.2 Si el usuario ingresa</li> </ul>

			correctamente sus credenciales el sistema deberá ingresar al dashboard del sistema.
	2	El administrador selecciona la opción "Generar Factura".	El sistema muestra la lista de productos disponibles.
	3	El administrador selecciona los productos y luego selecciona el cliente de la factura	El sistema muestra el cálculo del total de la venta y lo muestra
	4	El administrador confirma los datos de la facturación.	El sistema genera la factura y la almacena en la base de datos.
	5	El administrador elige la opción de exportarla (página de reporte)	El sistema genera el reporte con la información seleccionada.
<b>Excepciones</b>	#	<b>Acción (actor)</b>	<b>Reacción (sistema)</b>
	p.3	Si no hay productos seleccionados.	El sistema muestra un mensaje de error.
	p.5	Si ocurre un error en el cálculo	El sistema notifica la falla y solicita reintentar.
<b>Rendimiento</b>	El proceso de facturación debe completarse en un máximo de 5 segundos.		
<b>Frecuencia</b>	Este caso de uso se espera que se ejecute un promedio de 15 veces por semana.		
<b>Importancia</b>	Vital. Es esencial para la gestión de ventas de todo el aplicativo. Esta parte del sistema es crucial para poder formalizar el servicio de ventas y tener un registro de lo que se ha vendido además dando trazabilidad a las ventas y al dinero.		

<b>Urgencia</b>	Inmediata. Debe estar siempre disponible.		
<b>Comentarios</b>	Este caso de uso garantiza una gestión de ventas eficiente, permitiendo generar facturas detalladas y bien estructuradas con opciones de exportación y almacenamiento.		
<b>Identificador</b>	Gestión de reporte de Inventario		
<b>Descripción</b>	El administrador puede gestionar el inventario, actualizando la cantidad de productos disponibles y verificando el stock en tiempo real.		
<b>Actores</b>	<ul style="list-style-type: none"> <li>Administrador: Usuario con permisos para gestionar el inventario.</li> <li>Sistema: Aplicación encargada de registrar y monitorear el inventario.</li> </ul>		
<b>Pre condiciones</b>	El administrador debe haber iniciado sesión correctamente en el sistema y debe tener productos registrados en el sistema		
<b>Post condiciones</b>	El sistema actualiza correctamente el inventario y muestra el estado actualizado.		
<b>Secuencia Normal</b>	<b>#</b>	<b>Acción (actor)</b>	<b>Reacción (sistema)</b>
	1	El administrador inicia sesión en el sistema	El sistema verifica las credenciales y permite el acceso.
			<ul style="list-style-type: none"> <li>1.1 Si el usuario ingresa mal sus credenciales el sistema deberá notificarle del ingreso incorrecto de credenciales.</li> <li>1.2 Si el usuario ingresa correctamente sus credenciales el sistema deberá ingresar al</li> </ul>

			dashboard del sistema.
	2	El administrador selecciona la opción "Gestión de Inventario".	El sistema muestra la lista de productos con sus cantidades actuales.
	3	El administrador selecciona un producto para actualizar su cantidad.	El sistema muestra el producto con la cantidad que hay en el inventario
	4	El administrador ingresa la cantidad actualizada y confirma.	El sistema valida los datos e informa del éxito de la operación.
	5	El administrador revisará su inventario	El sistema mostrará la cantidad actualizada
	6	El administrador realiza una venta	El sistema debe validar automáticamente la disponibilidad
	7		Si existe disponibilidad, el sistema resta la cantidad vendida y actualiza el inventario
	8	El administrador revisa su inventario	El sistema muestra el inventario actualizado
<b>Excepciones</b>	<b>#</b>	<b>Acción (actor)</b>	<b>Reacción (sistema)</b>
	p.3	Si la cantidad ingresada es inválida	El sistema muestra un mensaje de error.
	p.5	Si ocurre un error en la base de datos	El sistema muestra un mensaje de falla

	p.7	Si no hay disponibilidad para realizar la venta	El sistema muestra un mensaje indicando el error.
<b>Rendimiento</b>	La actualización debe realizarse en un máximo de 3 segundos.		
<b>Frecuencia</b>	Este caso de uso se espera que se ejecute un promedio de 20 veces por semana.		
<b>Importancia</b>	Vital. Es esencial para la gestión de ventas.		
<b>Urgencia</b>	Inmediata. Debe estar siempre disponible para evitar interrupciones en ventas y operaciones.		
<b>Comentarios</b>	Este caso de uso asegura una gestión efectiva del inventario, permitiendo mantener el stock actualizado y evitar quiebres en el inventario.		

Identificador	Agregar Inventario		
<b>Descripción</b>	El administrador puede agregar nuevas existencias de productos al inventario, manteniendo el stock actualizado en tiempo real.		
<b>Actores</b>	<ul style="list-style-type: none"> <li>Administrador: Usuario con permisos para agregar productos al inventario.</li> <li>Sistema: Aplicación encargada de registrar las nuevas existencias.</li> </ul>		
<b>Pre condiciones</b>	El administrador debe haber iniciado sesión correctamente en el sistema y debe haber productos registrados.		
<b>Post condiciones</b>	El sistema registra correctamente las nuevas existencias y actualiza el stock.		
<b>Secuencia</b>	#	Acción (actor)	Reacción (sistema)



Normal	1	El administrador inicia sesión en el sistema	El sistema verifica las credenciales y permite el acceso.
			<ul style="list-style-type: none"> <li>1.1 Si el usuario ingresa mal sus credenciales el sistema deberá notificarle del ingreso incorrecto de credenciales.</li> <li>1.2 Si el usuario ingresa correctamente sus credenciales el sistema deberá ingresar al dashboard del sistema.</li> </ul>
	2	El administrador selecciona la opción "Agregar Inventario".	El sistema muestra un formulario para seleccionar un producto e ingresar su cantidad
	3	El administrador ingresa la información necesaria y la cantidad de existencias.	El sistema valida los datos ingresados.
	4	El administrador confirma la adición del inventario.	El sistema actualiza el stock y muestra la cantidad actualizada.
Excepciones	#	Acción (actor)	Reacción (sistema)
	p.3	Si la cantidad ingresada es inválida	El sistema muestra un mensaje de error.
	p.5	Si ocurre un error al registrar el inventario	El sistema muestra un mensaje de falla
Rendimiento	La adición debe completarse en un máximo de 3 segundos.		
Frecuencia	Este caso de uso se espera que se ejecute un promedio de 20 veces por semana.		
Importancia	Alta. Es crucial para mantener el inventario actualizado.		
Urgencia	Inmediata. Debe estar siempre disponible para evitar interrupciones en ventas y operaciones.		

<b>Comentarios</b>	Este caso de uso facilita el mantenimiento continuo del inventario, asegurando que las existencias se registren correctamente y en tiempo real.
--------------------	-------------------------------------------------------------------------------------------------------------------------------------------------

Identificador	Gestión de Ganancia		
<b>Descripción</b>	El administrador puede calcular la ganancia de los productos esperada ingresando el precio base del producto y el porcentaje de ganancia deseado. El sistema realiza el cálculo automáticamente.		
<b>Actores</b>	<ul style="list-style-type: none"> <li>Administrador: Usuario con permisos para calcular ganancias.</li> <li>Sistema: Aplicación encargada de realizar cálculos automáticos.</li> </ul>		
<b>Pre condiciones</b>	El administrador debe haber iniciado sesión correctamente en el sistema y debe conocer el precio y el % de ganancia que desea del mismo		
<b>Post condiciones</b>	El sistema muestra el precio final del producto y la ganancia esperada del mismo		
<b>Secuencia Normal</b>	<b>#</b>	<b>Acción (actor)</b>	<b>Reacción (sistema)</b>
	1	El administrador inicia sesión en el sistema	El sistema verifica las credenciales y permite el acceso.
			<ul style="list-style-type: none"> <li>1.1 Si el usuario ingresa mal sus credenciales el sistema deberá notificarle del ingreso incorrecto de credenciales.</li> <li>1.2 Si el usuario ingresa correctamente sus credenciales el sistema deberá ingresar al dashboard del sistema.</li> </ul>

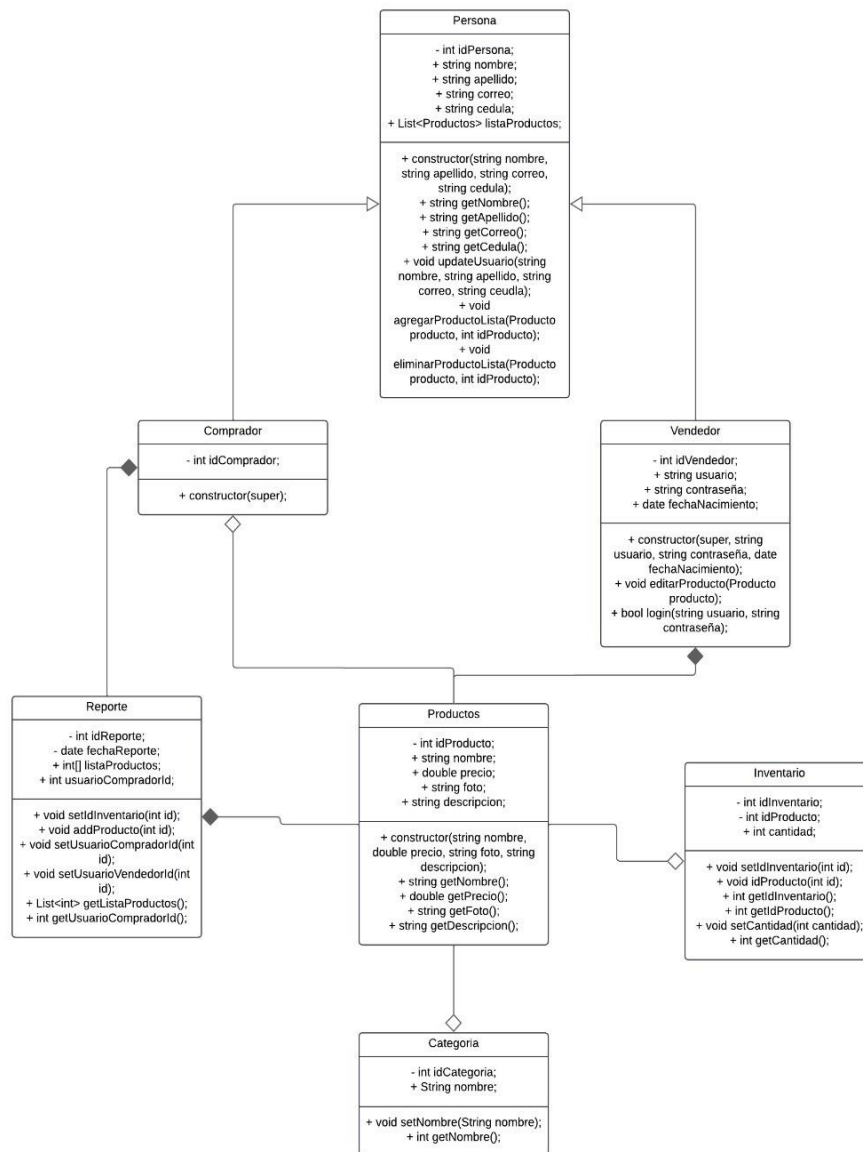
	2	El administrador selecciona la opción "Productos".	El sistema muestra un formulario para del producto donde se puede ver el precio de este y el % de ganancia
	3	El administrador ingresa los datos requeridos.	El sistema realiza el cálculo automáticamente y muestra el precio final y la ganancia esperada
<b>Excepciones</b>	<b>#</b>	<b>Acción (actor)</b>	<b>Reacción (sistema)</b>
	p.3	Si la cantidad ingresada es inválida	El sistema muestra un mensaje de error.
	p.3	Si el % de ganancia es menor a 0% o mayor a 100%	El sistema muestra un mensaje de error
<b>Rendimiento</b>	El cálculo debe completarse en un máximo de 2 segundos.		
<b>Frecuencia</b>	Este caso de uso se espera que se ejecute un promedio de 20 veces por semana.		
<b>Importancia</b>	Media. Es útil para planificar precios y estrategias de venta.		
<b>Urgencia</b>	Moderada. Puede realizarse según sea necesario.		
<b>Comentarios</b>	Este caso de uso facilita una planificación efectiva de precios y estrategias de venta mediante cálculos precisos y automáticos.		

Identificador	Gestión de Categorías		
Descripción	El administrador puede crear categorías a las cuales los productos pertenecerán con el fin de agruparlos en cierto orden tanto para organización interna como para filtrar al momento de intentar realizar una venta		
Actores	<ul style="list-style-type: none"> <li>Administrador: Usuario con permisos para calcular ganancias.</li> <li>Sistema: Aplicación encargada poder realizar un CRUD de categorías</li> </ul>		
Pre condiciones	El administrador debe haber iniciado sesión correctamente en el sistema		
Post condiciones	El sistema muestra las categorías creadas por el usuario.		
Secuencia Normal	#	Acción (actor)	Reacción (sistema)
	1	El administrador inicia sesión en el sistema	El sistema verifica las credenciales y permite el acceso.
			<ul style="list-style-type: none"> <li>1.1 Si el usuario ingresa mal sus credenciales el sistema deberá notificarle del ingreso incorrecto de credenciales.</li> <li>1.2 Si el usuario ingresa correctamente sus credenciales el sistema deberá ingresar al dashboard del sistema.</li> </ul>
	2	El administrador selecciona la opción "Categorías".	El sistema muestra todas las categorías que existen en el sistema

	3	El administrador desea agregar una nueva categoría	El sistema presenta un formulario para llenar la información de las categorías y se guarda en el sistema
	4	El administrador desea editar una categoría	El sistema mostrará al usuario los datos de la categoría seleccionada para editarlos.
	5.	El administrador desea eliminar una categoría	El sistema solicitará confirmación del usuario y procederá con la eliminación
<b>Excepciones</b>	<b>#</b>	<b>Acción (actor)</b>	<b>Reacción (sistema)</b>
	p.3	Si se ingresan datos inválidos en la creación	El sistema muestra un mensaje de error.
	p.5	Si se intenta eliminar una categoría vinculada a productos	El sistema mostrará un mensaje de error por las dependencias que existen.
<b>Rendimiento</b>	Las acciones deben completarse en un máximo de 3 segundos.		
<b>Frecuencia</b>	Este caso de uso se espera que se ejecute un promedio de 5 veces por semana.		
<b>Importancia</b>	Media. Es útil para organizar los productos y poder buscarlos más fácilmente.		
<b>Urgencia</b>	Crítica. Todos los productos deben pertenecer a una categoría por lo que el funcionamiento del sistema depende de esto		
<b>Comentarios</b>	Este caso de uso facilita una categorización y búsqueda de productos ya sea para poder analizar el inventario o realizar una venta.		

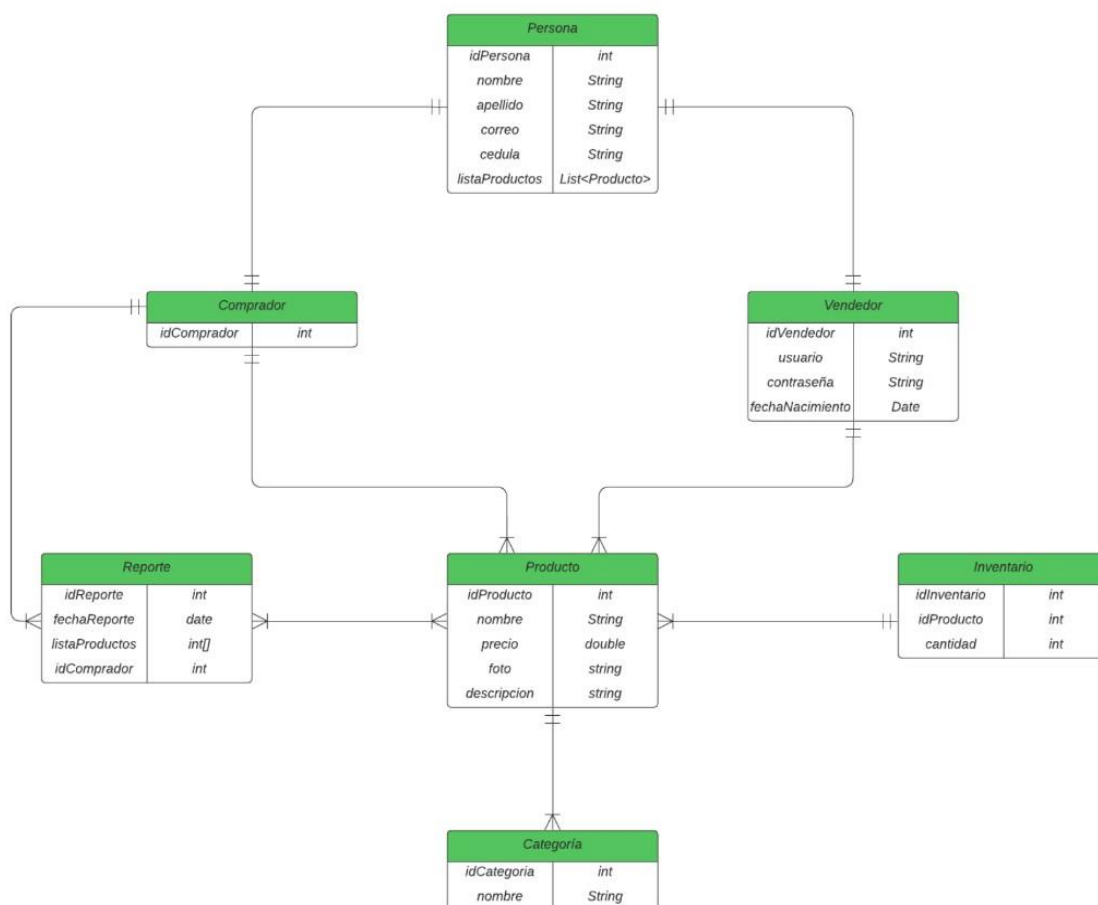
### 3.2. Diseño de Clases del Sistema

Para proceder con la implementación del sistema es de gran importancia realizar un diseño de clases del sistema, los cuales serán la base para su construcción utilizando la herramienta *OpenXava*.



### 3.3. Implementación Sistema Informático

Para implementar el sistema *ChauchiSoft* se han realizado una cantidad de pasos organizados. En primera instancia se ha creado un diagrama de datos, o también conocido como un diagrama de Entidad Relación, el cual describirá como se relacionan las distintas entidades del sistema.



Tras realizar este gráfico, se tiene una mayor claridad de las entidades y como estas se relacionarán para la implementación del sistema.

Tomando en cuenta esto se ha procedido en primer lugar a implementar a la clase personas, que representa tanto a compradores como a vendedores.

```
1 package com.example.chauchisoft.model;
2
3 import javax.persistence.*;
4
5 import org.openxml.annotations.*;
6
7 import lombok.*;
8
9 ...
10 @MappedSuperclass
11 @Getter @Setter
12 abstract public class Persona {
13     @Column(length=50)
14     @Required
15     String nombre;
16
17     @Column(length=50)
18     @Required
19     String apellido;
20
21     @Column(length=50)
22     @Required
23     String email;
24
25     @Column(length=50)
26     @Required
27     String cedula;
28 }
```

Una vez que ha sido implementada esta clase, se extiende de personas la clase comprador, y por igual la clase vendedor, cada una con ciertas características específicas que lo diferencian uno de otro.

```
1 package com.example.chauchisoft.model;
2
3 import javax.persistence.*;
4
5 import org.openxml.annotations.*;
6
7 import lombok.*;
8
9 ...
10 @Entity
11 @Getter @Setter
12 public class Comprador {
13     @Column(length=50)
14     @Required
15     String nombre;
16
17     @Column(length=50)
18     @Required
19     String apellido;
20
21     @Column(length=50)
22     @Required
23     String email;
24
25     @Id
26     @Column(length=50, unique=true)
27     @Required
28     String cedula;
29
30 }
31
32 public class Vendedor extends Persona {
33     @Id
34     @Column(length=32)
35     @GeneratedValue(generator="system-uuid")
36     @Hidden
37     String oid;
38
39     @Column(length=50)
40     @Required
41     String usuario;
42
43     @Column(length=50)
44     @Required
45     String contrasena;
46
47     @DateTime
48     @Required
49     Date fechaNacimiento;
50
51     @OneToMany(mappedBy="vendedor")
52     private List<Productos> productos;
53
54     public boolean isOlderThan18() {
55         Calendar calendar = Calendar.getInstance();
56         int currentYear = calendar.get(Calendar.YEAR);
57         calendar.setTime(fechaNacimiento);
58         int birthYear = calendar.get(Calendar.YEAR);
59         return (currentYear - birthYear) > 18;
60     }
61
62     @PrePersist
63     @PreUpdate
64     private void validateAge() {
65         if (!isOlderThan18()) {
66             throw new IllegalArgumentException("El vendedor debe ser mayor de 18 años.");
67         }
68     }
69 }
```



Tras la implementación de estas clases iniciales se procedieron a implementar distintas clases como productos y las categorías en las cuales los productos estarán agrupados. Todas estas clases se vinculan luego para implementar la clase inventario.

```

12 public class Productos {
13     @Id
14     @Column(length = 32)
15     @GeneratedValue(generator = "system-uuid")
16     @GenericGenerator(name = "system-uuid", strategy = "uuid")
17     @Hidden
18     String oid;
19
20     @Column(length=50)
21     @Required
22     String nombre;
23
24     @Column(length=50)
25     @Required
26     double precio;
27
28     @Files
29     @Column(length = 32)
30     String fotos;
31
32     @Column(length=50)
33     @Required
34     String descripcion;
35
36     @DescriptionsList
37     @ManyToOne
38     private Categoria categoria;
39
40     @ManyToOne
41     @Required
42     private Vendedor vendedor;
43 }

```

```

1 package com.example.chauchisoft.model;
2
3 import javax.persistence.*;
4 import org.openxava.annotations.*;
5 import lombok.*;
6
7 ...
8 @Entity @Setter
9 public class Categoria {
10     @Id
11     @Column(length = 32)
12     @GeneratedValue(strategy = GenerationType.AUTO)
13     @Hidden
14     Long oid;
15
16     @Column(length=50)
17     @Required
18     String nombre;
19 }

```

```

1 package com.example.chauchisoft.model;
2
3 import javax.persistence.*;
4
5 import org.hibernate.annotations.GenericGenerator;
6 import org.openxava.annotations.*;
7
8 import lombok.*;
9
10 ...
11 @Entity
12 @Getter @Setter
13 public class Inventario {
14     @Id
15     @GeneratedValue(generator = "system-uuid")
16     @GenericGenerator(name = "system-uuid", strategy = "uuid")
17     @Column(length = 32)
18     @Hidden
19     String oid;
20
21     @ManyToOne(fetch=FetchType.LAZY)
22     @Required
23     Productos producto;
24
25     @Column
26     @Required
27     int cantidad;
28 }

```

Finalmente, tras implementar todas estas clases podremos implementar una clase reporte y reporte body, los cuales como su nombre lo dice servirán para poder generar reportes del sistema.

```

15 public class Reporte {
16     @Id
17     @GeneratedValue(strategy = GenerationType.IDENTITY)
18     Long id;
19
20     @Required
21     @DefaultValueCalculator(CurrentLocalDateCalculator.class)
22     LocalDate fecha;
23
24     @ManyToOne(optional = false)
25     Comprador comprador;
26
27     @OneToMany(mappedBy = "reporte")
28     Collection<ReporteBody> reporteBody;
29
30     public Reporte() {
31         this.fecha = LocalDate.now();
32     }
33 }

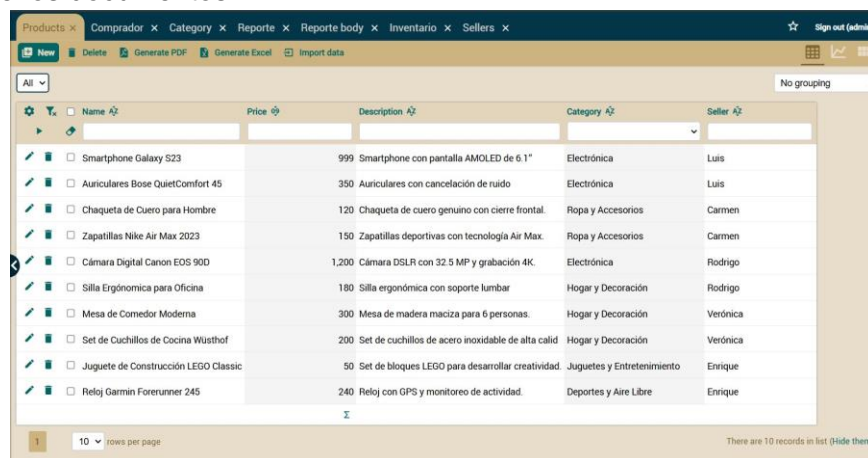
```

```

1 package com.example.chauchisoft.model;
2
3 import javax.persistence.*;
4
5 import org.openxava.annotations.*;
6
7 import lombok.*;
8
9 ...
10 @Entity
11 @Getter @Setter
12 public class ReporteBody {
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     Long id;
16
17     @ManyToOne(optional = false)
18     Reporte reporte;
19
20     @ManyToOne(optional = false)
21     Productos producto;
22
23     @Column
24     @Required
25     int cantidad;

```

Toda esta implementación de clases, vinculada correctamente, es decir siguiendo lo expuesto en los diagramas de clase y entidad relación, gracias al uso de *OpenXava* permiten ya tener un sistema listo con *CRUDs* de las distintas clases para su gestión, además dando un front end por default el cual ya posee todas las clases, entidades, relaciones y funcionalidades que han sido descritas previamente en los diagramas y en el resto de los documentos.



The screenshot shows the OpenXava web application interface. At the top, there is a navigation bar with tabs for 'Products', 'Comprador', 'Category', 'Reporte', 'Reporte body', 'Inventario', and 'Sellers'. Below the navigation bar, there is a toolbar with buttons for 'New', 'Delete', 'Generate PDF', 'Generate Excel', and 'Import data'. The main content area displays a table of products with columns for 'Name', 'Price', 'Description', 'Category', and 'Seller'. The table contains 10 rows of product data. At the bottom of the table, there is a pagination bar showing '10 rows per page' and a message 'There are 10 records in list (Hide them)'.

Name	Price	Description	Category	Seller
<input type="checkbox"/> Smartphone Galaxy S23	999	Smartphone con pantalla AMOLED de 6.1"	Electrónica	Luis
<input type="checkbox"/> Auriculares Bose QuietComfort 45	350	Auriculares con cancelación de ruido	Electrónica	Luis
<input type="checkbox"/> Chaqueta de Cuero para Hombre	120	Chaqueta de cuero genuino con cierre frontal	Ropa y Accesorios	Carmen
<input type="checkbox"/> Zapatillas Nike Air Max 2023	150	Zapatillas deportivas con tecnología Air Max	Ropa y Accesorios	Carmen
<input type="checkbox"/> Cámara Digital Canon EOS 90D	1,200	Cámara DSLR con 32.5 MP y grabación 4K	Electrónica	Rodrigo
<input type="checkbox"/> Silla Ergonómica para Oficina	180	Silla ergonómica con soporte lumbar	Hogar y Decoración	Rodrigo
<input type="checkbox"/> Mesa de Comedor Moderna	300	Mesa de madera maciza para 6 personas	Hogar y Decoración	Verónica
<input type="checkbox"/> Set de Cuchillos de Cocina Wüsthof	200	Set de cuchillos de acero inoxidable de alta calidad	Hogar y Decoración	Verónica
<input type="checkbox"/> Juguete de Construcción LEGO Classic	50	Set de bloques LEGO para desarrollar creatividad	Juguetes y Entretenimiento	Enrique
<input type="checkbox"/> Reloj Garmin Forerunner 245	240	Reloj con GPS y monitoreo de actividad	Deportes y Aire Libre	Enrique

### 3.4. Casos de Prueba Técnica 1 – Prueba de Caja Blanca (Cobertura de Caminos)

A continuación, se analizará la cobertura de caminos de cuatro algoritmos específicos, implementados como parte del sistema informático, para garantizar la calidad y la correcta ejecución del código. La técnica de cobertura de caminos permite identificar y validar todos los flujos posibles en los diagramas de control de cada algoritmo, asegurando que cada camino independiente sea ejecutado al menos una vez durante las pruebas. Los algoritmos seleccionados incluyen:

- Validación del precio antes de persistir o actualizar un objeto en la base de datos.
- Determinación de si una persona es mayor de 18 años.
- Cálculo de la ganancia de un producto en función de su precio y porcentaje de ganancia.
- Verificación de la validez de un número de cédula.

Se presentan los grafos de control de flujo, los casos de prueba diseñados, y los resultados obtenidos para cada algoritmo, resaltando la importancia de cubrir todos los caminos posibles para garantizar un funcionamiento confiable y robusto del sistema.

- **Validación del Precio**

El método *validatePrice* asegura que el precio de un objeto sea válido antes de que se guarde o actualice en la base de datos. Utiliza las anotaciones *@PrePersist* y *@PreUpdate* de JPA para ejecutar una validación previa. Si el precio no cumple con la condición de ser mayor a 0, se lanza una excepción *IllegalArgumentException*.

#### Código

```
@PrePersist
@PreUpdate
private void validatePrice() {
    if (!isPriceValid(precio)) {
        throw new IllegalArgumentException(s:"El precio debe ser mayor a 0");
    }
}
```

### Grafo de Control de Flujo



El grafo correspondiente al método se compone de los siguientes nodos y aristas:

- **Nodo I:** Inicio del método.
- **Nodo 1:** Evaluación de la condición *!isPriceValid(precio)*.
- **Nodo 2:** Lanzamiento de la excepción *IllegalArgumentException*.
- **Nodo F:** Finalización del método.

### Aristas del Grafo

- **De I a 1:** Ingreso al método.
- **De 1 a 2:** La condición *!isPriceValid(precio)* es verdadera.
- **De 1 a F:** La condición *!isPriceValid(precio)* es falsa y finalización del método.
- **De 2 a F:** Finalización del método

**Complejidad Ciclomática**

**McCabe:**  $V = 4 - 4 + 2 = 2$

**Nodos Predicados:**  $1 + 1 = 2$

**Regiones = 2**

**Casos de Prueba Diseñados**

	Entrada	Ejecución Esperada	Camino Cubierto
<b>Caso 1</b>	precio = 100	Se evalúa la condición <i>!isPriceValid(precio)</i> , resulta falsa, y el método finaliza sin lanzar una excepción.	I - 1 - F
<b>Caso 2</b>	precio = -10	Se evalúa la condición <i>!isPriceValid(precio)</i> , resulta verdadera, y se lanza una excepción <i>IllegalArgumentException</i> .	I - 1 - 2 - F

**Resultados de las Pruebas**

Ambos casos de prueba garantizan una cobertura completa de caminos para este algoritmo, validando tanto la ejecución de la excepción como el flujo de terminación exitoso del método. El análisis de cobertura de caminos para el método *validatePrice* confirma que todos los flujos posibles han sido evaluados mediante pruebas específicas, asegurando su correcto comportamiento en condiciones válidas e inválidas.

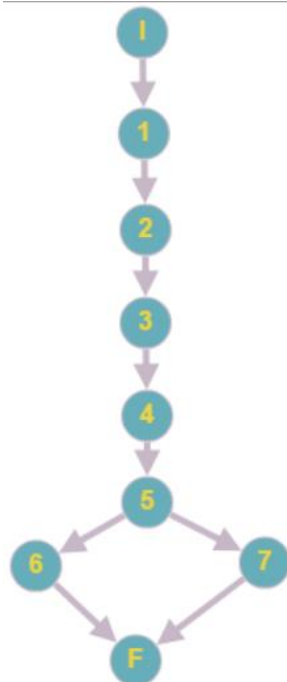
- **Determinación de Mayoría de Edad**

El método *isOlderThan18* determina si una persona es mayor de 18 años. Calcula la diferencia entre el año actual y el año de nacimiento de la persona utilizando la clase *Calendar*. Si la diferencia es mayor a 18, retorna true; en caso contrario, retorna false.

**Código**

```
public boolean isOlderThan18() {  
    Calendar calendar = Calendar.getInstance();  
    int currentYear = calendar.get(Calendar.YEAR);  
    calendar.setTime(fechaNacimiento);  
    int birthYear = calendar.get(Calendar.YEAR);  
    return (currentYear - birthYear) > 18;  
}
```

**Grafo de Control de Flujo**



El grafo correspondiente al método tiene los siguientes nodos y aristas:

- **Nodo I:** Inicio del método.
- **Nodo 1:** Instanciar *Calendar*.
- **Nodo 2:** Obtención del año actual (*currentYear*).
- **Nodo 3:** *Calendar Set Time*
- **Nodo 4:** Obtención del año de nacimiento (*birthYear*).
- **Nodo 5:** Evaluación de la condición (*currentYear* - *birthYear*) > 18.
- **Nodo 6:** Retorno del valor *true*.
- **Nodo 7:** Retorno del valor *false*.
- **Nodo F:** Finalización del método

#### Aristas del Grafo

- **De I a 1:** Ingreso al método.
- **De 1 a 2:** Instanciar *Calendar*.
- **De 2 a 3:** Obtención del año actual.
- **De 3 a 4:** *Calendar Set Time*.
- **De 4 a 5:** Obtención del año de nacimiento.
- **De 5 a 6:** Evaluación de la condición como *True*.
- **De 5 a 7:** Evaluación de la condición como *False*
- **De 6 a F:** Finalización del método.
- **De 7 a F:** Finalización del método.

#### Complejidad Ciclomática

**McCabe:**  $V = 9 - 9 + 2 = 2$

**Nodos Predicados:**  $1 + 1 = 2$

**Regiones = 2**

#### Casos de Prueba Diseñados

	Entrada	Ejecución Esperada	Camino Cubierto
<b>Caso 1</b>	<i>fechaNacimiento</i> = "2000-01-01"	La condición ( <i>currentYear</i> - <i>birthYear</i> ) > 18 es verdadera, y el método retorna <i>true</i> .	I - 1 - 2 - 3 - 4 - 5 - 6 - F
<b>Caso 2</b>	<i>fechaNacimiento</i> = "2010-01-01"	La condición ( <i>currentYear</i> - <i>birthYear</i> ) > 18 es falsa, y el método retorna <i>false</i> .	I - 1 - 2 - 3 - 4 - 5 - 7 - F

### Resultados de las Pruebas

Los casos de prueba abarcan todas las posibles rutas del grafo, validando los escenarios para personas mayores, y menores de 18 años. El análisis de cobertura de caminos asegura que el método *isOlderThan18* funciona correctamente para todos los casos posibles, garantizando la precisión en la determinación de la mayoría de edad.

- **Cálculo de la Ganancia de un Producto**

El método *calculate* calcula la ganancia de un producto multiplicando su precio (precio) por el porcentaje de ganancia (percentageGanancia) dividido entre 100. Luego, imprime el resultado en consola y lo retorna. Este método no contiene decisiones ni estructuras de control condicional.

### Código

```
@Override
public Object calculate() throws Exception {
    double gananciaFinal = precio * (percentageGanancia / 100);
    System.out.println("La ganancia es: " + gananciaFinal);
    return gananciaFinal;
}
```

### Grafo de Control de Flujo





El grafo correspondiente al método es lineal, con los siguientes nodos y aristas:

- **Nodo I:** Inicio del método.
- **Nodo 1:** Cálculo de *gananciaFinal*.
- **Nodo 2:** Impresión de la ganancia en consola.
- **Nodo 3:** Retorno de *gananciaFinal*.
- **Nodo F:** Finalización del método.

#### Aristas del Grafo

- **De I a 1:** Inicio del método.
- **De 1 a 2:** Cálculo de *gananciaFinal*.
- **De 2 a 3:** Impresión de la ganancia en consola.
- **De 3 a F:** Finalización del método

#### Complejidad Ciclomática

**McCabe:**  $V = 4 - 5 + 2 = 1$

**Nodos Predicados:**  $0 + 1 = 1$

**Regiones = 1**

#### Casos de Prueba Diseñados

	Entrada	Ejecución Esperada	Camino Cubierto
<b>Caso 1</b>	precio = 100 percentageGanancia = 20	Retorno: 20.	I - 1 - 2 - 3 - F
<b>Caso 2</b>	precio = 0 percentageGanancia = 20	Retorno: 0.	I - 1 - 2 - 3 - F

#### Resultados de las Pruebas

Todos los casos de prueba siguen el único camino posible en el grafo, cubriendo las variaciones de los valores de entrada (precio y percentageGanancia). El análisis de cobertura de caminos confirma que el método calculate realiza correctamente los cálculos de ganancia para todos los escenarios probados, asegurando su funcionalidad.

- **Verificación de la Validez de un Número de Cédula**

El método *validarCedula* verifica la validez de un número de cédula basado en su longitud y cálculos matemáticos en sus dígitos.

**Código**

```
private boolean validarCedula(String x) {
    int suma = 0;
    if (x.length() == 9) {
        return false;
    } else {
        int a[] = new int[x.length() / 2];
        int b[] = new int[(x.length() / 2)];
        int c = 0;
        int d = 1;
        for (int i = 0; i < x.length() / 2; i++) {
            a[i] = Integer.parseInt(String.valueOf(x.charAt(c)));
            c = c + 2;
            if (i < (x.length() / 2) - 1) {
                b[i] = Integer.parseInt(String.valueOf(x.charAt(d)));
                d = d + 2;
            }
        }

        for (int i = 0; i < a.length; i++) {
            a[i] = a[i] * 2;
            if (a[i] > 9) {
                a[i] = a[i] - 9;
            }
            suma = suma + a[i] + b[i];
        }
        int aux = suma / 10;
        int dec = (aux + 1) * 10;
        if ((dec - suma) == Integer.parseInt(String.valueOf(x.charAt(x.length() - 1))))
            return true;
        else if (suma % 10 == 0 && x.charAt(x.length() - 1) == '0') {
            return true;
        } else {
            return false;
        }
    }
}
```

### Grafo de Control de Flujo



El grafo correspondiente al método tiene los siguientes nodos y aristas:

- **Nodo I:** Inicio del método
- **Nodo 1:** Inicialización variable suma
- **Nodo 2:** Evaluación de la condición  $(x.length() == 9)$
- **Nodo 3:** Inicialización variable a, b, c, d.
- **Nodo 4:** Evaluación de la condición  $(i < x.length() / 2)$  en un for loop.
- **Nodo 5:** Cálculo variable  $a[i]$  y c.
- **Nodo 6:** Evaluación de la condición  $(i < (x.length() / 2) - 1)$
- **Nodo 7:** Cálculo variable  $b[i]$  y d
- **Nodo 8:** Evaluación de la condición  $(i < a.length)$  en un for loop.
- **Nodo 9:** Cálculo variable  $a[i]$
- **Nodo 10:** Evaluación de la condición  $(a[i] > 9)$ .
- **Nodo 11:** Cálculo variable  $a[i]$
- **Nodo 12:** Cálculo variable suma
- **Nodo 13:** Cálculo variable aux y dec

- **Nodo 14:** Evaluación de la condición `(dec - suma) == Integer.parseInt(String.valueOf(x.charAt(x.length() - 1)))`
- **Nodo 15:** Evaluación de la condición `(suma % 10 == 0 && x.charAt(x.length() - 1) == '0')`
- **Nodo 16:** Evaluación en else.
- **Nodo F:** Finalización del método.

#### Aristas del Grafo

- **De I a 1:** Inicio del método
- **De 1 a 2:** Inicialización variable suma
- **De 2 a F:** Evaluación de la condición `(x.length() == 9)` como false.
- **De 2 a 3:** Evaluación de la condición `(x.length() == 9)` como true
- **De 3 a 4:** Inicialización variable a, b, c, d.
- **De 4 a 5:** Evaluación de la condición `(i < x.length() / 2)` en un for loop como true.
- **De 5 a 6:** Cálculo variable `a[i]` y c.
- **De 6 a 4:** Evaluación de la condición `(i < (x.length() / 2) - 1)` como false.
- **De 6 a 7:** Evaluación de la condición `(i < (x.length() / 2) - 1)` como false.
- **De 7 a 4:** Cálculo variable `b[i]` y d.
- **De 4 a 8:** Evaluación de la condición `(i < x.length() / 2)` en un for loop como false.
- **De 8 a 9:** Evaluación de la condición `(i < a.length)` en un for loop como true.
- **De 9 a 10:** Cálculo variable `a[i]`
- **De 10 a 11:** Evaluación de la condición `(a[i] > 9)` como true
- **De 10 a 12:** Evaluación de la condición `(a[i] > 9)` como false
- **De 11 a 12:** Cálculo variable `a[i]`
- **De 12 a 8:** Cálculo variable suma
- **De 8 a 13:** Evaluación de la condición `(i < a.length)` en un for loop como false.
- **De 13 a 14:** Cálculo variable aux y dec
- **De 14 a F:** Evaluación de la condición `(dec - suma) == Integer.parseInt(String.valueOf(x.charAt(x.length() - 1)))` como true.
- **De 14 a 15:** Evaluación de la condición `(dec - suma) == Integer.parseInt(String.valueOf(x.charAt(x.length() - 1)))` como false.

- **De 15 a F:** Evaluación de la condición (suma % 10 == 0 && x.charAt(x.length() - 1) == '0') como true.
- **De 15 a 16:** Evaluación de la condición (suma % 10 == 0 && x.charAt(x.length() - 1) == '0') como false.
- **De 16 a F:** Retornar false

### Complejidad Ciclomática

**McCabe:**  $V = 24 - 18 + 2 = 8$

**Nodos Predicados:**  $7 + 1 = 8$

**Regiones = 8**

### Casos de Prueba Diseñados

	Entrada	Descripción	Ejecución Esperada	Camino Cubierto
<b>Caso 1</b>	123456789	Este caso debe activar la condición que valida la longitud del número de cédula y devolver false inmediatamente, saltándose el resto del código.	false	I-1-2-F
<b>Caso 2</b>	1234567890	La longitud de la cédula es mayor a 9, por lo que el flujo continuará procesando las condiciones y cálculos.	true	I-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-F
<b>Caso 3</b>	12345678901	En este caso, el número de cédula es válido en cuanto a su longitud y la estructura interna, pero el último dígito no	false	I-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-F

		concuera con la suma.		
<b>Caso 4</b>	12345678123	Este caso de prueba valida un número de cédula correcto.	true	I-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-F
<b>Caso 5</b>	12345678000	Este caso de prueba debería fallar porque el número de cédula no es válido, ya que la suma no concuerda con el último dígito.	false	I-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-F

### Resultados de las Pruebas

Los casos cubren todos los caminos posibles, incluyendo las condiciones principales y los flujos alternativos. El análisis de cobertura de caminos garantiza que todas las posibles ejecuciones del método validarCedula están correctamente probadas y verificadas, asegurando su funcionalidad y fiabilidad.

### 3.5. Casos de Prueba Técnica 2 - Caja Negra Cobertura en Equivalencia

Aquí se está realizando un diseño de pruebas basado en clases de equivalencia, utilizando el enfoque de pruebas de caja negra para validar una restricción específica en el programa: asegurar que el vendedor sea mayor de 18 años al momento de ser creado. A continuación, te proporciono una descripción detallada que puedes usar como introducción a estas pruebas:

Caso de prueba	Clase de equivalencia	Representante	Resultado esperado
1	EC1: $x > 18$	1/1/2005	True
2	EC2: $x < 18$	1/1/2010	False
3	EC3: $x = 18$	1/1/2006	True
4	EC4: $x = \text{null}$	Null	False
5	EC5: $x < 0$	31/12/2030	False
6	EC4: $x = \text{texto}$	"Hola"	False

Luego de tener los casos de prueba, se analiza con el caso de equivalencia para poder observar qué condiciones se necesitan cumplir para que el caso de prueba se apruebe dónde:

Las clases de equivalencia se agrupan en dos categorías principales:

**Valores Numéricos:** Representan edades válidas e inválidas.

- **EC1:** Edades mayores a 18 (válido).
- **EC2:** Edades menores a 18 (no válido).
- **EC3:** Edad exactamente igual a 18 (válido).

**Valores No Reales:** Representan datos que no corresponden a una edad numérica válida.

- **EC4:** Edad nula (no válido).
- **EC5:** Edad negativa (no válido).
- **EC6:** Entrada no numérica, como texto (no válido).

Variable	CE	Estado	Representante	TC 1	TC 2	TC 3	TC 4	TC 5
<b>VALOR NUMÉRICO</b>	EC1: $x > 18$	Válido	1/1/2005	*	*	*		
	EC2: $x < 18$	No válido	1/1/2010				*	
	EC3: $x = 18$	Válido	1/1/2006					*
<b>VALORES NO REALES</b>	EC4: $x = \text{Null}$	No válido	Null	*			*	*
	EC5: $x < 0$	No válido	31/12/2030		*			
	EC6: $x = \text{texto}$	No válido	"Hola"			*		

Aquí se está realizando un diseño de pruebas basado en clases de equivalencia para validar una restricción específica: asegurar que solo se acepten cédulas ecuatorianas válidas. A continuación, te proporciono una descripción que puedes usar como introducción a estas pruebas de caja negra:

Caso de prueba	Clase de equivalencia	Representante	Resultado esperado
1	EC1: $x > 9$	123456789	False
2	EC2: $x = 9$	123456789	False
3	EC3: $x < 9$	12345678	False
4	EC4: $x \text{ SUM}(x > 9)$	1750611541	True
5	EC5: $x \text{ !SUM}(x > 9)$	1750611540	False
6	EC4: $x = \text{texto}$	abctdasd	False



Después se puede ver cómo es que se complementan para realizar los casos de prueba, esta vez, primero debe analizar el valor inicial que son 9 dígitos y luego llega a verificar su suma, entonces por eso abajo empieza a verse su equivalencia, pero arriba se ve al revés

Donde las clases de equivalencia para las cédulas ecuatorianas se dividen en dos grandes categorías:

**Validación Numérica Inicial:** Representa el formato de la cédula.

- **EC1:** Números con más de 10 dígitos (no válido).
- **EC2:** Números con exactamente 9 dígitos (no válido).
- **EC3:** Números con menos de 9 dígitos (no válido).

**Validación de la Suma de Dígitos:** Determina si la cédula cumple con la lógica específica de validación.

- **EC4:** Cédula con suma válida y 10 dígitos (válido).
- **EC5:** Cédula con suma inválida, aunque tenga 10 dígitos (no válido).
- **EC6:** Cédula con caracteres no numéricos, como texto (no válido).

Variable	CE	Estado	Representante	TC 1	TC 2	TC 3	TC 4	TC 5
VALOR NUMÉRICO	EC4: $x \text{ SUM}(x > 9)$	Valido	175061 1541	*			*	*
	EC5: $x \text{ !SUM}(x > 9)$	No valido	175061 1540		*			
	EC6: $x = \text{texto}$	No valido	abctdas d			*		
VERIFICACIÓN INICIAL	EC1: $x > 9$	No valido	123456 789	*	*	*		
	EC2: $x = 9$	No valido	123456 789				*	

	EC3: $x < 9$		No valido		123456 78						*
--	--------------	--	-----------	--	--------------	--	--	--	--	--	---

A continuación, presento una descripción para las pruebas diseñadas bajo clases de equivalencia para el cálculo de ganancias basado en el precio y el porcentaje de ganancia:

Caso de prueba	Clase de equivalencia	Representante	Resultado esperado
1	EC1: $x > 0$	10000	True
2	EC2: $x \leq 0$	-10	False
3	EC3: $x = \text{texto}$	"Test"	False
4	EC4: $0 \leq x \leq 100$	15	True
5	EC5: $x < 0$	-20	False
6	EC6: $x > 100$	150	False
7	EC7: $x$ no numerico	"hola"	False

Donde las clases de equivalencia se dividen en dos categorías principales:

#### Validación Numérica Inicial (Precio y Porcentaje):

- **EC1:** Precio mayor a 0 (valido).
- **EC2:** Precio menor o igual a 0 (no válido).
- **EC3:** Precio no numérico (texto o caracteres) (no válido).

#### Validación de Porcentajes:

- **EC4:** Porcentaje entre 0 y 100 (inclusive) (válido).
- **EC5:** Porcentaje menor que 0 (no válido).
- **EC6:** Porcentaje mayor que 100 (no válido).
- **EC7:** Porcentaje no numérico (texto o caracteres) (no válido).

Variable	CE	Estado	Representante	TC 1	TC 2	TC 3	TC 4	TC 5	TC 6
<b>VALOR NUMÉRICO</b>	EC1: $x > 0$	Valido	10000	*	*	*	*		
	EC2: $x \leq 0$	No valido	-10					*	
	EC3: $x = \text{texto}$	No valido	"Test"						*
<b>VALORES NO REALES</b>	EC4: $0 \leq x < 100$	Valido	15	*				*	*
	EC5: $x < 0$	No valido	-20		*				
	EC6: $x > 100$	No valido	150			*			
	EC7: $x$ no numerico	No valido	"hola"				*		

A continuación, se presenta un análisis detallado de los casos de prueba para un programa que verifica que los precios no sean negativos antes de crear o actualizar un producto:

Caso de prueba	Clase de equivalencia	Representante	Resultado esperado
1	EC1: $x > 0$	10000	True
2	EC2: $x \leq 0$	-10	False
3	EC3: $x = \text{texto}$	"Test"	False
4	EC5: $x < 0$	15	True
5	EC6: $x > 0$	-20	False

6		EC7: x no numerico		sadsad		False	
---	--	--------------------	--	--------	--	-------	--

Donde las clases de equivalencia se dividen en dos categorías:

**Validación de Precios Numéricos:**

- **EC1:** Precio mayor a 0 (válido).
- **EC2:** Precio menor o igual a 0 (no válido).
- **EC3:** Precio no numérico (texto o caracteres) (no válido).

**Validación de Valores No Reales:**

- **EC5:** Precio menor a 0 (no válido).
- **EC6:** Precio mayor que 0 (válido).
- **EC7:** Precio no numérico (texto o caracteres) (no válido).

Variable	CE	Estado	Representante	TC 1	TC 2	TC 3	TC 4	TC 5
<b>VALOR NUMÉRICO</b>	EC1: $x > 0$	Válido	10000	*	*	*		
	EC2: $x \leq 0$	No válido	-10				*	
	EC3: $x = \text{texto}$	No válido	"Test"					*
<b>VALORES NO REALES</b>	EC5: $x < 0$	Válido	15	*			*	*
	EC6: $x > 0$	No válido	-20		*			
	EC7: x no numerico	No válido	asdsa			*		

### 3.6. Casos de Prueba Técnica 3 - Caja Negra Cobertura de Valores Límites

El enfoque de la cobertura de valores límites es realizar pruebas en los puntos justo por debajo y por encima de los valores límite esperados. Aquí, el límite es 18 años. En el caso de prueba, las clases de equivalencia y los valores a probar serían los siguientes:

Caso de prueba		Clase de equivalencia		Representante		Resultado esperado		Observaciones
TC	1	$x \geq$	18	18		True		Es mayor o igual a 18
TC	2	$x <$	18	17		False		Es menor a 18
TC	3	$x <$	0	-1		False		Es un numero negativo
TC	4	x	es texto	asd		False		Son letras antes que números

La cobertura de valores límites también puede aplicarse cuando se manejan entradas específicas, como en el caso de la validación de números de cédulas ecuatorianas. Este tipo de validación requiere que la cédula sea un número mayor a 9 y que cumpla con una regla adicional relacionada con la suma de sus dígitos.

Caso de prueba		Clase de equivalencia		Representante		Resultado esperado		Observaciones
TC	1	$x >$	9	123456789		False		Es mayor a 9 pero no es suma
TC	2	$x <$	9	12345678		False		Es menor a 9
TC	3	x	es letras	abcdefgh		False		Son letras antes que números
TC	4	x suma	es verdadera	1750611541		True		Es suma y es mayor a 9
TC	5	x suma	es falsa	1750611540		False		Es mayor a 9 pero no es suma

TC	6	x	es alfanumérica	175061a541		False		Hay letras en su interior
----	---	---	-----------------	------------	--	-------	--	---------------------------

También se aplica cuando se manejan condiciones matemáticas o de cálculo, como en este caso, donde el sistema calcula la ganancia a partir de un precio y un porcentaje de ganancia. La cobertura de valores límites asegura que el sistema maneje correctamente los valores de entrada tanto dentro como fuera de los rangos esperados, identificando posibles errores o resultados incorrectos en los bordes de esos rangos.

Caso de prueba		Precio		Porcentaje de ganancia		Resultado esperado		Observación
TC	1	x =	0.01	x =	0	Calculo exitoso		Ambos limites son validos
TC	2	x =	0	x =	50	Error	precio invalido	Precio igual a 0
TC	3	x =	-0.01	x =	50	Error	precio invalido	Precio negativo
TC	4	x =	10	x =	100	Calculo exitoso		Ambos limites son validos
TC	5	x =	1,000	x =	100.01	Error	ganancia > 100	Porcentaje mayor a 100
TC	6	x =	1,000	x =	-0.01	Error	Ganancia invalida	Porcentaje negativo

Es útil cuando se manejan condiciones de entrada como valores numéricos que deben cumplir ciertas restricciones, como es el caso de un programa que calcula la ganancia en función de un precio. Esta técnica se enfoca en asegurar que el programa gestione correctamente los valores de entrada en los límites de los parámetros definidos, así como aquellas entradas que no deberían ser válidas, como las que contienen caracteres no numéricos.

Caso de prueba		Precio		Resultado esperado		Observación
TC	1	x =	0.01	Agregar exitoso		Precio es correcto
TC	2	x =	0	Error	precio invalido	Precio igual a 0
TC	3	x =	-0.01	Error	precio invalido	Precio es negativo
TC	4	x =	abc	Error	precio invalido	Precio son letras

### 3.7. Casos de Prueba Técnica 4 – Casos de Prueba

En esta sección se presentarán los casos de prueba técnica que se derivan directamente de los casos de uso del sistema, lo que nos permitirá validar la funcionalidad desde la perspectiva del usuario. Los casos de prueba se han elaborado a partir de los escenarios descritos en los casos de uso, teniendo en cuenta las precondiciones necesarias para que la interacción sea válida, los pasos a seguir en el proceso y los resultados esperados en cada escenario. A través de estas pruebas, se evaluará si el sistema se comporta de acuerdo con las expectativas y si todas las interacciones posibles, tanto las principales como las alternativas, están cubiertas correctamente.

#### Descripción de cada Campo

- **ID:** Un identificador único para cada caso de prueba.
- **Nombre:** El título del caso de prueba.
- **Descripción:** Breve descripción del objetivo del caso de prueba.
- **Precondiciones:** Los requisitos previos necesarios para ejecutar la prueba.
- **Pasos:** La secuencia de acciones que el tester debe seguir para realizar la prueba.
- **Resultado Esperado:** El comportamiento esperado del sistema en base a los pasos realizados.
- **Resultado Actual:** El comportamiento real del sistema observado durante la prueba.
- **Estado:** El estado actual de la prueba (aprobada, fallida, en progreso, etc.).
- **Referencias:** Referencias adicionales que explican o enlazan al caso de prueba con otras pruebas o documentación relevante.

A continuación, se detallan los casos de prueba asociados con los casos de uso definidos para el sistema:

## INGENIERÍA DE SOFTWARE VALIDACIÓN DE SOFTWARE

ID	Nombre	Descripción	Precondiciones	Pasos	Resultado esperado	Resultado Actual	Estado	Referencias
CP_Ven1	Agregar Vendedor dejando los campos vacíos	Se agrega un nuevo vendedor con todos los campos vacíos	Iniciar Sesión	1. Presionar el botón de guardar	Debe salir varias notificaciones indicando que se llenen los campos requeridos	Salen varias notificaciones indicando los campos requeridos para guardar	Aprobado	Documento de Casos de Uso
CP_Ven2	Agregar Vendedor con una cédula inválida	Se agrega un nuevo vendedor con una cédula inválida	Iniciar Sesión	1. Llenar los campos de nombre, apellido, correo, usuario, contraseña y fecha de nacimiento. 2. Llenar el campo de cédula con un número inválido 3. Presionar el botón de guardar	Debe salir una notificación indicando que la cédula ingresada no es válida	Sale una notificación indicando que la cédula no es válida	Aprobado	Documento de Casos de Uso
CP_Ven3	Agregar Vendedor con una fecha de nacimiento menor a 18 años	Se agrega un nuevo vendedor con una fecha de nacimiento menor a 18 años	Iniciar Sesión	1. Llenar los campos de nombre, apellido, correo, cédula, usuario y contraseña. 2. Llenar el campo de fecha de nacimiento para un usuario menor a 18 años 3. Presionar el botón de guardar	Debe salir una notificación indicando que el vendedor debe ser mayor a 18 años	Sale una notificación indicando que el vendedor debe ser mayor a 18 años	Aprobado	Documento de Casos de Uso
CP_Ven4	Agregar Vendedor con todos los campos correctos	Se agrega un nuevo vendedor con todos los campos llenados de manera correcta	Iniciar Sesión	1. Llenar los campos de nombre, apellido, correo, cédula, usuario, contraseña y fecha de nacimiento 2. Presionar el botón de guardar	Debe salir una notificación indicando que el vendedor fue creado exitosamente	Sale una notificación indicando que el vendedor fue creado exitosamente	Aprobado	Documento de Casos de Uso
CP_Ven5	Actualizar Vendedor con todos los campos correctos	Se actualiza un vendedor con todos los campos llenados de manera correcta	Tener un vendedor creado	1. Ir a la lista de vendedores 2. Seleccionar un vendedor 3. Actualizar los campos requeridos 4. Presionar el botón de guardar	Debe salir una notificación indicando que el vendedor fue actualizado exitosamente	Sale una notificación indicando que el vendedor fue actualizado exitosamente	Aprobado	Documento de Casos de Uso
CP_Ven5	Actualizar Vendedor con campos vacíos	Se actualiza un vendedor con campos vacíos	Tener un vendedor creado	1. Ir a la lista de vendedores 2. Seleccionar un vendedor 3. Actualizar los campos con datos vacíos 4. Presionar el botón de guardar	Debe salir una notificación indicando que falta algún campo requerido	Sale una notificación indicando que falta algún campo requerido	Aprobado	Documento de Casos de Uso
CP_Ven5	Actualizar Vendedor con una fecha de nacimiento menor a 18 años	Se actualiza un vendedor con una fecha de nacimiento menor a 18 años	Tener un vendedor creado	1. Ir a la lista de vendedores 2. Seleccionar un vendedor 3. Actualizar el campo de fecha de nacimiento, con una fecha menor a 18 años 4. Presionar el botón de guardar	Debe salir una notificación que el vendedor debe ser mayor a 18 años	Sale una notificación indicando que el vendedor debe ser mayor a 18 años	Aprobado	Documento de Casos de Uso
CP_Ven6	Actualizar Vendedor con una cédula inválida	Se actualiza el vendedor con una cédula inválida	Tener un vendedor creado	1. Ir a la lista de vendedores 2. Seleccionar un vendedor 3. Actualizar la cédula con un dato inválido 4. Presionar el botón de guardar	Debe salir una notificación indicando que la cédula ingresada no es válida	Sale una notificación indicando que la cédula no es válida	Aprobado	Documento de Casos de Uso
CP_Ven7	Eliminar un vendedor	Se elimina un vendedor	Tener un vendedor creado	1. Ir a la lista de vendedores. 2. Presionar el botón de eliminar (basurero) 3. Aceptar la confirmación de eliminación	Debe eliminarse el vendedor	Se elimina el vendedor	Aprobado	Documento de Casos de Uso
CP_Ven8	Eliminar vendedor con productos vinculados	Se elimina un vendedor con productos vinculados	Tener un vendedor creado	1. Ir a la lista de vendedores 2. Presionar el botón de eliminar (basurero) 3. Aceptar la confirmación de eliminación	No se debe eliminar el vendedor	No se elimina el vendedor	Aprobado	Documento de Casos de Uso
CP_Com1	Agregar Comprador dejando los campos vacíos	Se agrega un nuevo comprador con todos los campos vacíos	Iniciar Sesión	1. Presionar el botón de guardar	Debe salir varias notificaciones indicando que se llenen los campos requeridos	Salen varias notificaciones indicando los campos requeridos para guardar	Aprobado	Documento de Casos de Uso
CP_Com2	Agregar Comprador con una cédula inválida	Se agrega un nuevo comprador con una cédula inválida	Iniciar Sesión	1. Llenar los campos de nombre, apellido y correo. 2. Llenar el campo de cédula con un número inválido. 3. Presionar el botón de guardar	Debe salir una notificación indicando que la cédula ingresada no es válida	Sale una notificación indicando que el comprador fue creado exitosamente	Fallido	Documento de Casos de Uso
CP_Com3	Agregar Comprador con todos los campos correctos	Se agrega un nuevo comprador con todos los campos llenados de manera correcta	Iniciar Sesión	1. Llenar los campos de nombre, apellido, correo y cédula. 2. Presionar el botón de guardar	Debe salir una notificación indicando que el comprador fue creado exitosamente	Sale una notificación indicando que el comprador fue creado exitosamente	Aprobado	Documento de Casos de Uso
CP_Com4	Actualizar comprador con todos los campos correctos	Se actualiza un comprador con todos los campos correctos	Tener un comprador creado	1. Ir a la lista de compradores 2. Seleccionar un comprador 3. Actualizar los campos requeridos 4. Presionar el botón de guardar	Debe salir una notificación indicando que el comprador fue actualizado exitosamente	Sale una notificación indicando que el comprador fue actualizado exitosamente	Aprobado	Documento de Casos de Uso
CP_Com5	Actualizar comprador con campos vacíos	Se actualiza un comprador con campos vacíos	Tener un comprador creado	1. Ir a la lista de compradores 2. Seleccionar un comprador 3. Actualizar con campos vacíos 4. Presionar el botón de guardar	Debe salir una notificación indicando que falta algún campo requerido	Sale una notificación indicando que falta algún campo requerido	Aprobado	Documento de Casos de Uso
CP_Com6	Actualizar comprador con cédula inválida	Se actualiza el comprador con una cédula inválida	Tener un comprador creado	1. Ir a la lista de compradores 2. Seleccionar un comprador 3. Actualizar con una cédula inválida 4. Presionar el botón de guardar	Debe salir una notificación indicando que se debe ingresar una cédula válida	El sistema no permite actualizar la cédula	Fallido	Documento de Casos de Uso
CP_Com7	Eliminar comprador	Se elimina un comprador	Tener un comprador creado	1. Ir a la lista de compradores 2. Presionar el botón de eliminar (basurero) 3. Aceptar la confirmación de eliminación	Debe eliminarse el comprador	Se elimina el comprador	Aprobado	Documento de Casos de Uso
CP_Cat1	Agregar Categoría dejando los campos vacíos	Se agrega una nueva categoría con todos los campos vacíos	Iniciar Sesión	1. Presionar el botón de guardar	Debe salir una notificación indicando que se llenen los campos requeridos	Sale una notificación indicando que se llenen los campos requeridos	Aprobado	Documento de Casos de Uso



CP_Cat2	Agregar Categoría con todos los campos correctos	Se agrega una nueva categoría con todos los campos llenados de manera correcta	Iniciar Sesión	1. Llenar el campo de nombre. 2. Presionar el botón de guardar.	Debe salir una notificación indicando que la categoría fue creada exitosamente	Sale una notificación indicando que la categoría fue creada exitosamente	Aprobado	Documento de Casos de Uso
CP_Cat3	Actualizar Categoría con todos los campos correctos	Se actualiza la categoría con todos los campos correctos	Tener una categoría creada	1. Ir a la lista de categorías 2. Seleccionar una categoría 3. Actualizar los campos requeridos 4. Presionar el botón de guardar	Debe salir una notificación indicando que la categoría fue actualizada exitosamente	Sale una notificación indicando que la categoría fue actualizada exitosamente	Aprobado	Documento de Casos de Uso
CP_Cat4	Actualizar Categoría con campos vacíos	Se actualiza la categoría con campos vacíos	Tener una categoría creada	1. Ir a la lista de categorías 2. Seleccionar una categoría 3. Actualizar con campos vacíos 4. Presionar el botón de guardar	Debe salir una notificación indicando que falta algún campo requerido	Sale una notificación indicando que falta algún campo requerido	Aprobado	Documento de Casos de Uso
CP_Cat5	Eliminar categoría	Se elimina la categoría	Tener una categoría creada	1. Ir a la lista de categorías 2. Presionar el botón de eliminar (basurero) 3. Aceptar la confirmación de eliminación	Debe eliminarse la categoría	Se elimina la categoría	Aprobado	Documento de Casos de Uso
CP_Cat6	Eliminar Categoría que tenga productos vinculados	Se elimina una categoría con productos vinculados	Tener una categoría creada	1. Ir a la lista de categorías 2. Presionar el botón de eliminar (basurero) 3. Aceptar la confirmación de eliminación	No se debe eliminar la categoría	No se elimina la categoría	Aprobado	Documento de Casos de Uso
CP_Prod1	Agregar Producto dejando los campos vacíos	Se agrega un nuevo producto con todos los campos vacíos	Tener previamente creado al menos un vendedor y una categoría	1. Presionar el botón de guardar	Debe salir una notificación indicando que se llenen los campos requeridos	Sale una notificación indicando que se llenen los campos requeridos	Aprobado	Documento de Casos de Uso
CP_Prod2	Agregar Producto sin Vendedor	Se agrega un nuevo producto sin un vendedor	Tener previamente creado al menos un vendedor y una categoría	1. Llenar el campo de nombre, precio, porcentaje de ganancia, ganancia calculada, descripción y categoría. 2. Presionar el botón de guardar.	Debe salir una notificación indicando que falta ingresar un vendedor	Sale una notificación indicando que falta ingresar un vendedor	Aprobado	Documento de Casos de Uso
CP_Prod3	Agregar Producto con precio negativo	Se agrega un nuevo producto con un precio menor a cero	Tener previamente creado al menos un vendedor y una categoría	1. Llenar el campo de nombre, porcentaje de ganancia, ganancia calculada, descripción, categoría y vendedor. 2. Ingresar un precio negativo 3. Presionar el botón de guardar.	Debe salir una notificación indicando que no se pueden ingresar precios negativos	Sale una notificación indicando que el precio debe ser mayor a 0	Aprobado	Documento de Casos de Uso
CP_Prod4	Agregar Producto con porcentaje de ganancia negativo	Se agrega un nuevo producto con un porcentaje de ganancia menor a cero	Tener previamente creado al menos un vendedor y una categoría	1. Llenar el campo de nombre, precio, ganancia calculada, descripción, categoría y vendedor. 2. Ingresar un porcentaje de ganancia negativo 3. Presionar el botón de guardar.	Debe salir una notificación indicando que no se pueden ingresar porcentajes de ganancia negativos	Sale una notificación indicando que el producto fue creado exitosamente	Fallido	Documento de Casos de Uso
CP_Prod5	Agregar Producto con todos los campos correctos	Se agrega un nuevo producto con todos los campos llenados de manera correcta	Tener previamente creado al menos un vendedor y una categoría	1. Llenar el campo de nombre, precio, porcentaje de ganancia, ganancia calculada, descripción, categoría y vendedor.	Debe salir una notificación indicando que el producto fue creado exitosamente	Sale una notificación indicando que el producto fue creado exitosamente	Aprobado	Documento de Casos de Uso
CP_Prod6	Actualizar Producto con los campos correctos	Se actualiza el producto con los campos correctos	Tener creado un producto	1. Ir a la lista de productos 2. Seleccionar un producto 3. Actualizar los campos requeridos 4. Presionar el botón de guardar	Debe salir una notificación indicando que el producto fue actualizado exitosamente	Sale una notificación indicando que el producto fue actualizado exitosamente	Aprobado	Documento de Casos de Uso
CP_Prod7	Actualizar Producto con campos vacíos	Se actualiza el producto con campos vacíos	Tener creado un producto	1. Ir a la lista de productos 2. Seleccionar un producto 3. Actualizar con campos vacíos 4. Presionar el botón de guardar	Debe salir una notificación indicando que hay campos vacíos	Sale una notificación indicando que hay campos vacíos	Aprobado	Documento de Casos de Uso
CP_Prod8	Actualizar Producto con un nuevo vendedor	Se actualiza el producto con un nuevo vendedor	Tener creado un producto	1. Ir a la lista de productos 2. Seleccionar un producto 3. Seleccionar un nuevo vendedor para el producto 4. Presionar el botón de guardar	Debe salir una notificación indicando que el producto fue actualizado exitosamente	Sale una notificación indicando que el producto fue actualizado exitosamente	Aprobado	Documento de Casos de Uso
CP_Prod9	Eliminar un producto	Se elimina el producto	Tener creado un producto	1. Ir a la lista de productos 2. Presionar el botón de eliminar (basurero) 3. Aceptar la confirmación de eliminación	Debe eliminarse el producto	Se elimina el producto	Aprobado	Documento de Casos de Uso
CP_Prod10	Eliminar un producto vinculado a un reporte de venta	Se elimina el producto asociado a un reporte de venta	Tener creado un producto	1. Ir a la lista de productos 2. Presionar el botón de eliminar (basurero) 3. Aceptar la confirmación de eliminación	No debe eliminarse el producto	No se elimina el producto	Aprobado	Documento de Casos de Uso
CP_Invt1	Agregar inventario dejando los campos vacíos	Se agrega un nuevo inventario con todos los campos vacíos	Tener previamente creado al menos un vendedor y un producto	1. Presionar el botón de guardar	Debe salir una notificación indicando que se llenen los campos requeridos	Sale una notificación indicando que se llenen los campos requeridos	Aprobado	Documento de Casos de Uso
CP_Invt2	Agregar inventario dejando producto y vendedor vacío	Se agrega un nuevo inventario sin producto ni vendedor	Tener previamente creado al menos un vendedor y un producto	1. Llenar el campo de cantidad. 2. Presionar el botón de guardar.	Debe salir una notificación indicando que falta ingresar un producto y un vendedor	Sale una notificación indicando que falta ingresar un producto y un vendedor	Aprobado	Documento de Casos de Uso
CP_Invt3	Agregar inventario con todos los campos correctos	Se agrega un nuevo inventario con todos los campos llenados de manera correcta	Tener previamente creado al menos un vendedor y un producto	1. Llenar el campo de producto, vendedor y cantidad. 2. Presionar el botón de guardar.	Debe salir una notificación indicando que el inventario fue creado exitosamente	Sale una notificación indicando que el inventario fue creado exitosamente	Aprobado	Documento de Casos de Uso
CP_Invt4	Agregar inventario con cantidad negativa	Se agrega un nuevo inventario con cantidad negativa	Tener previamente creado al menos un vendedor y un producto	1. Llenar el campo de producto, vendedor y cantidad con un valor negativo. 2. Presionar el botón de guardar.	Debe salir una notificación indicando que se debe ingresar una cantidad positiva	Sale una notificación indicando que el inventario fue creado exitosamente	Fallido	Documento de Casos de Uso
CP_Invt5	Actualizar inventario dejando los campos vacíos	Se actualiza el inventario dejando campos vacíos	Tener previamente creado el inventario	1. Ir a la lista de inventario 2. Seleccionar un inventario 3. Actualizar con campos vacíos 4. Presionar el botón de guardar	Debe salir una notificación indicando que se llenen los campos requeridos	Sale una notificación indicando que se llenen los campos requeridos	Aprobado	Documento de Casos de Uso
CP_Invt6	Actualizar inventario dejando los campos correctos	Se actualiza el inventario con los campos correctos	Tener previamente creado el inventario	1. Ir a la lista de inventario 2. Seleccionar un inventario 3. Actualizar con campos correctos 4. Presionar el botón de guardar	Debe salir una notificación indicando que el inventario fue actualizado exitosamente	Sale una notificación indicando que el inventario fue actualizado exitosamente	Aprobado	Documento de Casos de Uso

CP_Inv7	Eliminar inventario	Se elimina un inventario	Tener previamente creado el inventario	1. Ir a la lista de inventario 2. Presionar el botón de eliminar (basurero) 3. Aceptar la confirmación de eliminación	Debe eliminarse el inventario	Se elimina el inventario	Aprobado	Documento de Casos de Uso
CP_Rep1	Agregar Reporte dejando los campos vacíos	Se agrega un nuevo reporte con todos los campos vacíos	Tener previamente creado un comprador	1. Presionar el botón de guardar	Debe salir una notificación indicando que se llenen los campos requeridos	Salen una notificación indicando que se llenen los campos requeridos	Aprobado	Documento de Casos de Uso
CP_Rep2	Agregar Reporte con ID repetido	Se agrega un nuevo reporte con un ID repetido	Tener previamente creado un comprador	1. Llenar el campo de ID y comprador. 2. Presionar el botón de guardar.	Debe salir una notificación indicando que ya existe un reporte con ese ID	Salen una notificación indicando que ya existe un reporte con ese ID	Aprobado	Documento de Casos de Uso
CP_Rep3	Agregar Reporte con todos los campos llenados de manera correcta	Se agrega un nuevo reporte con todos los campos llenados de manera correcta	Tener previamente creado un comprador	1. Llenar el campo de ID y comprador. 2. Presionar el botón de guardar.	Debe salir una notificación indicando que el reporte fue creado exitosamente	Salen una notificación indicando que el reporte fue creado exitosamente	Aprobado	Documento de Casos de Uso
CP_Rep4	Actualizar Reporte con todos los campos correctos	Se actualiza el reporte con todos los campos correctos	Tener previamente creado un reporte	1. Ir a la lista de reporte 2. Seleccionar un reporte 3. Actualizar con campos vacíos 4. Presionar el botón de guardar	Debe salir una notificación indicando que se llenen los campos requeridos	Salen una notificación indicando que se llenen los campos requeridos	Aprobado	Documento de Casos de Uso
CP_Rep5	Actualizar Reporte con campos vacíos	Se actualiza el reporte con campos vacíos	Tener previamente creado un reporte	1. Ir a la lista de reporte 2. Seleccionar un reporte 3. Actualizar con campos correctos 4. Presionar el botón de guardar	Debe salir una notificación indicando que el reporte fue actualizado exitosamente	Salen una notificación indicando que el reporte fue actualizado exitosamente	Aprobado	Documento de Casos de Uso
CP_Rep6	Eliminar Reporte	Se elimina el reporte	Tener previamente creado un reporte	1. Ir a la lista de reporte 2. Presionar el botón de eliminar (basurero) 3. Aceptar la confirmación de eliminación	Debe eliminarse el reporte	Debe eliminarse el reporte	Aprobado	Documento de Casos de Uso
CP_Rep7	Eliminar Reporte con Cuerpo del Reporte Vinculado	Se elimina el reporte con cuerpo del reporte vinculado	Tener previamente creado un reporte	1. Ir a la lista de reporte 2. Presionar el botón de eliminar (basurero) 3. Aceptar la confirmación de eliminación	No debe eliminarse el reporte	No se elimina el reporte	Aprobado	Documento de Casos de Uso
CP_RepBod1	Agregar Cuerpo del Reporte dejando los campos vacíos	Se agrega un nuevo cuerpo del reporte con todos los campos vacíos	Tener previamente creado un reporte y un producto	1. Presionar el botón de guardar	Debe salir una notificación indicando que se llenen los campos requeridos	Salen una notificación indicando que se llenen los campos requeridos	Aprobado	Documento de Casos de Uso
CP_RepBod2	Agregar Cuerpo del Reporte con ID repetido	Se agrega un nuevo cuerpo del reporte con un ID repetido	Tener previamente creado un reporte y un producto	1. Llenar el campo de ID, reporte, producto y cantidad. 2. Presionar el botón de guardar.	Debe salir una notificación indicando que ya existe un cuerpo del reporte con ese ID	Salen una notificación indicando que ya existe un cuerpo del reporte con ese ID	Aprobado	Documento de Casos de Uso
CP_RepBod3	Agregar Cuerpo del Reporte con todos los campos correctos	Se agrega un nuevo cuerpo del reporte con todos los campos llenados de manera correcta	Tener previamente creado un reporte y un producto	1. Llenar el campo de ID, reporte, producto y cantidad. 2. Presionar el botón de guardar.	Debe salir una notificación indicando que el cuerpo del reporte fue creado exitosamente	Salen una notificación indicando que el cuerpo del reporte fue creado exitosamente	Aprobado	Documento de Casos de Uso
CP_RepBod3	Actualizar Cuerpo del Reporte con todos los campos correctos	Se actualiza el cuerpo del reporte con todos los campos correctos	Tener previamente creado el cuerpo del reporte	1. Ir a la lista de reporte 2. Seleccionar un reporte 3. Actualizar con campos correctos 4. Presionar el botón de guardar	Debe salir una notificación indicando que el cuerpo del reporte fue actualizado exitosamente	Salen una notificación indicando que el cuerpo del reporte fue actualizado exitosamente	Aprobado	Documento de Casos de Uso
CP_RepBod4	Actualizar Cuerpo del Reporte con todos los campos correctos	Se actualiza el cuerpo del reporte con todos los campos correctos	Tener previamente creado el cuerpo del reporte	1. Ir a la lista de reporte 2. Seleccionar un reporte 3. Actualizar con campos correctos 4. Presionar el botón de guardar	Debe salir una notificación indicando que el cuerpo del reporte fue actualizado exitosamente	Salen una notificación indicando que el cuerpo del reporte fue actualizado exitosamente	Aprobado	Documento de Casos de Uso
CP_RepBod5	Actualizar Cuerpo del Reporte con campos vacíos	Se actualiza el reporte con campos vacíos	Tener previamente creado el cuerpo del reporte	1. Ir a la lista de reporte 2. Seleccionar un reporte 3. Actualizar con campos vacíos 4. Presionar el botón de guardar	Debe salir una notificación indicando que se llenen los campos requeridos	Salen una notificación indicando que se llenen los campos requeridos	Aprobado	Documento de Casos de Uso
CP_RepBod6	Eliminar Cuerpo del Reporte	Se elimina el cuerpo del reporte	Tener previamente creado el cuerpo del reporte	1. Ir a la lista de cuerpo de reporte 2. Presionar el botón de eliminar (basurero) 3. Aceptar la confirmación de eliminación	Debe eliminarse el cuerpo del reporte	Debe eliminarse el cuerpo del reporte	Aprobado	Documento de Casos de Uso

De un total de 53 casos de prueba ejecutados, 49 casos fueron exitosos, mientras que 4 casos resultaron fallidos. Esto representa un porcentaje de éxito del 92.45%. Los casos fallidos fueron analizados en detalle para identificar las causas subyacentes y proponer las acciones correctivas necesarias. Este resultado refleja una alta eficacia en la implementación y funcionalidad del sistema, aunque se identificaron áreas específicas que requieren atención para garantizar una cobertura completa y un rendimiento óptimo.

### 3.8. Resumen de Pruebas Ejecutadas y Recomendaciones.

#### Caja Blanca – Cobertura de Caminos

En esta sección, se ejecutaron una serie de pruebas basadas en la técnica de caja blanca, enfocándose en la cobertura de los caminos de los algoritmos y las validaciones internas dentro del código. Estas pruebas se diseñaron para asegurar que todas las condiciones, bucles y excepciones posibles en el código fueran correctamente evaluadas. A continuación, se presenta el resumen de las pruebas ejecutadas y sus resultados:

- **Validación de Precio:** Se verificó que el sistema valida correctamente el precio antes de persistir o actualizar un objeto en la base de datos. Se probó con precios válidos y precios inválidos (igual a 0 o negativos), asegurándose de que la excepción *IllegalArgumentException* se lanzará cuando el precio no fuera válido. La cobertura del flujo fue completa, cubriendo la validación antes de la persistencia y actualización.
- **Determinación de Edad:** Se probó el método que determina si una persona es mayor de 18 años. Se verificaron diferentes fechas de nacimiento para garantizar que el cálculo de la edad y la comparación con el valor 18 se ejecutaran correctamente. El resultado fue el esperado, validando correctamente si la persona es mayor de 18 años según la fecha de nacimiento proporcionada.
- **Cálculo de Ganancia:** Se validó el cálculo de la ganancia de un producto en función de su precio y porcentaje de ganancia. Las pruebas se realizaron con diferentes combinaciones de precio y porcentaje de ganancia, y se verificó que el sistema calculó correctamente la ganancia final. El sistema mostró los resultados correctos de la ganancia, considerando valores de entrada positivos.

- **Validación de Cédula:** Se ejecutaron múltiples pruebas sobre el algoritmo de validación de cédulas, evaluando tanto números válidos como inválidos. El sistema verificó correctamente el formato y las reglas para determinar si una cédula era válida o no, y se lanzó el resultado esperado dependiendo de la validación del número de cédula.

#### **Equivalencia:**

En esta serie de pruebas de caja negra, hemos validado el funcionamiento de un programa que maneja precios de productos, validaciones de cédula, validaciones de edad y cálculo de ganancias según un porcentaje, asegurándose de que los valores sean adecuados antes de permitir su creación o actualización. Para ello, hemos utilizado el concepto de Clases de Equivalencia (CE), que se emplean para agrupar las entradas en clases que deben ser tratadas de la misma manera, facilitando la creación de casos de prueba y asegurando que el programa maneje correctamente diferentes tipos de entradas.

#### **Valores Límites:**

Las pruebas que hemos analizado se centran en la validación de entradas de un programa que calcula la ganancia a partir de un precio y un porcentaje de ganancia. Estas pruebas están basadas en la cobertura de valores límites, que se enfoca en verificar que el sistema maneje correctamente los valores en los bordes de los rangos definidos y las entradas no válidas.

- En resumen, estas pruebas incluyen:
- 1. **Validación del Precio:** Se asegura que el precio sea un número positivo, mayor que cero, y que esté en un formato numérico adecuado.
- 2. **Validación del Porcentaje de Ganancia:** En algunos casos, también se verifican las restricciones sobre el porcentaje de ganancia (debe ser positivo y no superior a 100).
- 3. **Validación de Entradas no Numéricas:** Se prueba con valores que no son numéricos (como texto o caracteres alfanuméricos), asegurando que el sistema los rechace correctamente.
- 4. **Validaciones de Verificaciones de Cédula Ecuatoriana:** Son valores que se deben cumplir con dos condiciones a la vez para poder realizar la validación de esta

### Casos de Prueba

Durante la ejecución de los casos de prueba, se diseñaron y ejecutaron un total de 53 escenarios, enfocados en validar las funcionalidades principales del sistema, incluyendo validaciones de precio, cálculo de ganancias, determinación de edad, y verificación de cédulas. Estas pruebas se dividieron en casos que cubrieron diferentes clases de equivalencia, valores límites, y flujos alternativos en los algoritmos implementados. De los 53 casos, 49 se completaron exitosamente, confirmando el correcto funcionamiento de la mayoría de las funcionalidades evaluadas. Los 4 casos restantes identificaron áreas de mejora en las validaciones del sistema, proporcionando una base para ajustes y optimizaciones futuras.

Considerando las pruebas realizadas, se puede evidenciar que, en las pruebas de caja blanca, clases de equivalencia y valores límite se alcanzó un porcentaje de éxito del 100%, lo que valida el correcto funcionamiento del sistema. Por otro lado, los casos de prueba presentaron una tasa de éxito del 92.45%, un resultado muy positivo. En general, el sistema muestra una tasa de éxito total del 96.19%. Aunque se recomienda validar el sistema en las áreas donde se han presentado fallos para corregirlos, el sistema está listo para ser utilizado por los clientes, ya que dichas fallas no afectan su uso por parte de los usuarios.

Tipo de Prueba	Pruebas1	Exito	Error	% Exito
Caja Blanca	11	11	0	100.00
Casos de Prueba	53	49	4	92.45
Clases de Equivalencia	21	21	0	100.00
Valores Limite	20	20	0	100.00
Totalizado de pruebas	105	101	4	96.19

#### 3.9. Link Video

<https://youtu.be/cZne77TohHw>

#### 3.10. Link GitHub

<https://github.com/JossueJativa/chauchisoft>

#### 4. Referencias:

- Perfil del Proyecto: Documento de perfil del proyecto validado por Juan Aristizábal, Enrique Merizalde y Jossue Játiva.
- El Comercio (22/05/2024): Análisis del mercado de vehículos usados: Astudillo, G. (2024, May 22). ¿Cómo está la demanda de vehículos usados en Ecuador?. El Comercio. <https://www.elcomercio.com/actualidad/negocios/como-demanda-vehiculos-usados-ecuador.html>.
- El Comercio (27/12/2022): Opciones para generación de ingresos online: Tapia, X. (2022, December 27). ¿Cómo sacar un dinero extra Gracias a internet?. El Comercio. <https://www.elcomercio.com/tecnologia/dinero-extra-trabajos-internet-opciones.html>
- El Comercio (04/03/2023): Transformación digital en Quito: Medina, A. (2023, March 4). Los Negocios SE adaptan a lo virtual y presencial en Quito. El Comercio. <https://www.elcomercio.com/actualidad/los-negocios-se-adaptan-a-lo-virtual-y-presencial-en-quito.html>
- Paniza, J. (2005). OpenXava: Framework para aplicaciones empresariales. Recuperado de <https://openxava.org>
- Oracle. (2023). Java Persistence API (JPA). Recuperado de <https://oracle.com/java>
- OpenXava Team. (2023). Documentación oficial de OpenXava. Recuperado de <https://openxava.org/documentation>
- Stack Overflow. (2023). OpenXava discussions and solutions. Recuperado de <https://stackoverflow.com>
- InfoQ. (2022). Productividad en frameworks Java. Recuperado de <https://infoq.com>
- ISTQB. (2023). Foundations of Software Testing. International Software Testing Qualifications Board.
- Pressman, R. S. (2020). Software Engineering: A Practitioner's Approach. McGraw-Hill Education.
- Kaner, C., Bach, J., & Pettichord, B. (2002). Lessons Learned in Software Testing: A Context-Driven Approach. Wiley.
- Sommerville, I. (2016). Software Engineering. Pearson.
- Luna, N. (2024). Introducción al Testing de Software. Recuperado de <https://www.certificaristqb.com>.