

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO



Nombre: Angélica Narváez (6779)

Tarea: Análisis estático y dinámico(Ingeniería
Inversa)

Ingeniería en software
APLICACIONES INFORMATICAS II



PAO 8
18/07/2025



1. OBJETIVO(S):

2.1.GENERAL

Aplicar la técnica de ingeniería inversa sobre una aplicación informática existente para comprender su estructura, funcionamiento interno y componentes principales, con el fin de generar representaciones útiles para el mantenimiento de software.

2.2.ESPECÍFICOS

- Seleccionar una aplicación para análisis de mantenimiento de software
- Identificar su estructura, como funciona, como esta desarrollada y el flujo lógico en el que se basa la aplicación .
- Interpretar y detectar posibilidades de mejora en la aplicación que ayuden a la mantenibilidad de software.

2. METODOLOGÍA

Se desea implementar la técnica de ingeniería inversa, mediante el análisis de una aplicación que será valorada y analizada, recopilando información de su estructura, como funciona y como esta desarrollada, con la aplicación de un análisis estático donde se realiza una revisión de la codificación y un análisis dinámico para validación de ejecución.

3. EQUIPOS Y MATERIALES:

- Computador
- Entorno integrado de desarrollo (IDE)
- Aula virtual
- Acceso a internet
- Bibliografía

4. MARCO TEORICO:

Partimos de comprender o recordar en que consiste la ingeniería inversa , La ingeniería inversa es un proceso mediante el cual se observa cómo está construido y cómo funciona un objeto, proceso, programa o sistema con la intención de mejorarlo o duplicarlo. La observación se puede basar en aspectos muy diversos, como averiguar cuáles son sus componentes, cómo estos interactúan entre sí o cómo se fabricó el producto. En otras palabras, la ingeniería inversa se basa en algo tan sencillo que parece un juego de niños: desmontar algo y volverlo a montar para ver cómo funciona.

Se destaca por recuperar el diseño, revisar y detectar código innecesario.

1. Mejora del diseño de producto: La ingeniería inversa permite a los fabricantes desmontar y estudiar productos existentes para comprender cómo están diseñados y cómo funcionan. Esto brinda la oportunidad de identificar posibles mejoras en el diseño, aumentando la eficiencia, la funcionalidad y la calidad del producto. Al comprender los aspectos clave del diseño de un producto exitoso, los fabricantes pueden crear versiones mejoradas o incluso desarrollar productos completamente nuevos basados en esa información.

2. Reparación y mantenimiento: En la industria manufacturera, a menudo se enfrentan a la tarea de reparar o mantener productos que ya no están en producción o que carecen de documentación técnica. La ingeniería inversa permite descomponer y comprender el producto para identificar los componentes y su funcionamiento. Esto facilita la reparación y el mantenimiento de productos obsoletos o sin soporte técnico, al permitir que los ingenieros comprendan cómo funciona y qué piezas son necesarias para su reparación.

3. Innovación y desarrollo de productos: La ingeniería inversa puede ser una fuente de inspiración para la innovación y el desarrollo de nuevos productos. Al analizar y comprender los productos existentes, los fabricantes pueden identificar nuevas ideas, características o enfoques que pueden ser aplicados a sus propios productos. Esto puede conducir a la creación de productos innovadores y mejorados que satisfagan las necesidades cambiantes de los consumidores y el mercado.

Técnicas más comunes aplicadas en la Ingeniería Inversa:

- Análisis estático es la inspección del código fuente sin ejecutarlo. Permite analizar su estructura, lógica, organización en módulos, funciones, etc.
- Análisis dinámico es la observación del comportamiento del sistema mientras se ejecuta. Ayuda a entender el flujo de datos y la interacción entre componentes.

A continuación, validaremos como se aplica la ingeniería inversa en una aplicación de software existente y como llevar a cabo el proceso de ingeniería inversa de la misma.

Ingeniería Inversa:

La ingeniería inversa de software se creó con el propósito de ayudar a comprender el funcionamiento interno de algún proceso bajo un entorno de ejecución específico. Si bien, se dificulta el aprendizaje de su aplicación, debido a las pocas fuentes existentes de información, se planea, por ende, realizar una guía con la cual el usuario final podrá asimilar diferentes conceptos sobre el reversing (ingeniería inversa), que le permitirán aplicarlo a pequeños programas para entender su funcionamiento interno con base en su ejecución.

La ingeniería inversa en programación implica:

- **Desmontar el programa:**

Analizar el código máquina o ensamblador de un programa compilado para entender su funcionamiento.

- **Reconstruir el código fuente:**

Intentar generar una representación del código fuente original a partir del código máquina, aunque esto puede ser complejo y no siempre preciso.

- **Comprender la lógica:**

Identificar los algoritmos, estructuras de datos y patrones de diseño utilizados en el programa.

- **Analizar vulnerabilidades:**

Detectar posibles errores, fallos o puntos débiles en el código que podrían ser explotados.

- **Mejorar o adaptar el software:**

Utilizar la información obtenida para corregir errores, agregar nuevas funcionalidades o adaptar el programa a nuevas plataformas.

5. PROCEDIMIENTO:

Descripción de sistema analizado:

Nombre de la Aplicación: **Sistema de Facturación(Veterinaria)**

Dominio: El software que se describe para ser analizado es un sistema de Facturación de una Veterinaria, donde el dominio de este sistema es la administración y operación básica de una veterinaria, con el enfoque en inventarios y facturación, principalmente determinándolo con la lógica de resolver actividades de un software de gestión para negocios veterinarios o comercios de productos para mascotas.

Para realizar la ingeniería inversa se optó en realizar el análisis estático y dinámico:

ANÁLISIS ESTÁTICO

Este tipo de análisis revisa la estructura, lógica y componentes del código fuente:

Arquitectura del Sistema:

El sistema está diseñado y desarrollado con la Arquitectura Modelo, Vista, Controlador (Vistas, Modelos, Config) el cual es un aplicativo web desarrollada en PHP.

Estructura del código:

El análisis estático se centró en examinar el código fuente, la estructura de archivos y la base de datos sin ejecutar la aplicación.

Config: Contiene archivos de configuración, como la conexión a la base de datos (conexion.php) y pruebas de conexión (test.php).

Estilos: Carpeta donde se encuentran los archivos CSS para los diferentes módulos o páginas del sistema, como agregar productos, facturación electrónica e index.

Layaout: Incluye componentes reutilizables de la interfaz, como el Footer.php y el Navbar.php, que se usan en varias páginas para mantener una estructura consistente.

Modelos: Aquí están los scripts PHP encargados de la lógica de negocio, como insertar productos (insertar_productos.php) y procesar facturas (procesar_factura.php). Se encargan de interactuar con la base de datos y manejar los datos principales.

Vistas: Contiene las páginas principales que ve el usuario, como agregar productos, ver comprobantes y la facturación electrónica. Son archivos PHP que muestran la interfaz e incluyen los componentes del layout.

index.html: Archivo principal del sistema.

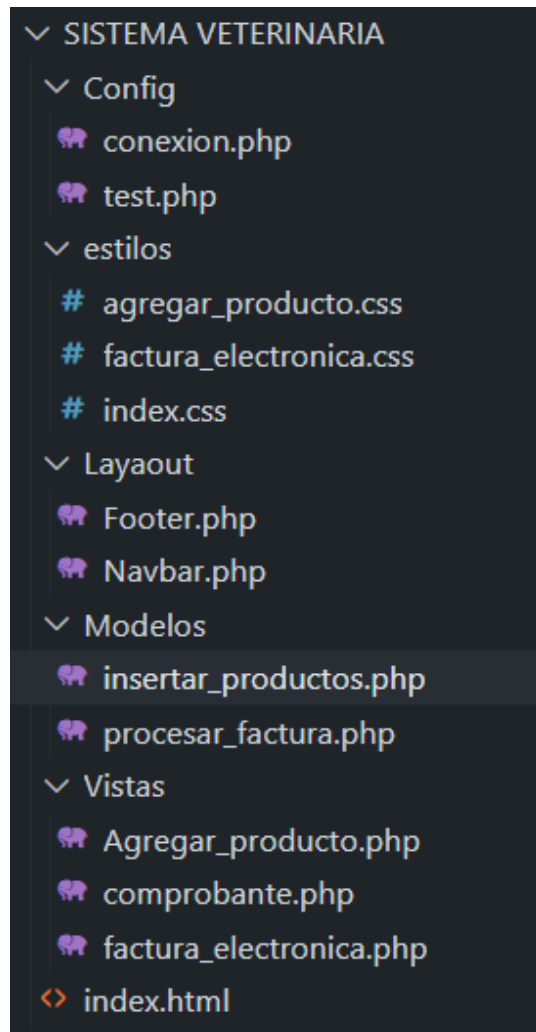


Fig 1. Estructura del código

Se analizaron los siguientes aspectos mediante observación del comportamiento de la app, estructura de archivos, vistas HTML y su conexión.

Funcionalidades principales

✓ Ruta principal:

Muestra la página de inicio (index.html).

```

STEMA VETERINARIA > index.html
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Veterinaria - Sistema de Gestión</title>
7      <link rel="stylesheet" href="estilos/index.css">
8  </head>
9  <body>
10
11      <!-- Header -->
12      <header>
13          <div class="logo">Veterinaria Patitas</div>
14          <nav class="nav">
15              <ul>
16                  <li class="submenu">
17                      <a href="#">Inventarios </a>
18                      <ul class="dropdown">
19                          <li><a href="Vistas/Agregar_producto.php">Agregar Productos</a></li>
20                      </ul>
21                  </li>
22                  <li><a href="Vistas/factura_electronica.php">Facturación Electrónica</a></li>
23              </ul>
24          </nav>
25      </header>
26
27      <!-- Contenido principal -->
28      <main>
29          <h1>Bienvenido al sistema de gestión veterinaria</h1>
30          <p>Seleccione una opción del menú para comenzar.</p>
31      </main>
32
33      <!-- Footer -->

```

✓ Agregar producto:

Formulario para agregar productos, que envía datos a [Modelos/insertar_productos.php](#) (POST), donde se procesan y almacenan en la base de datos.

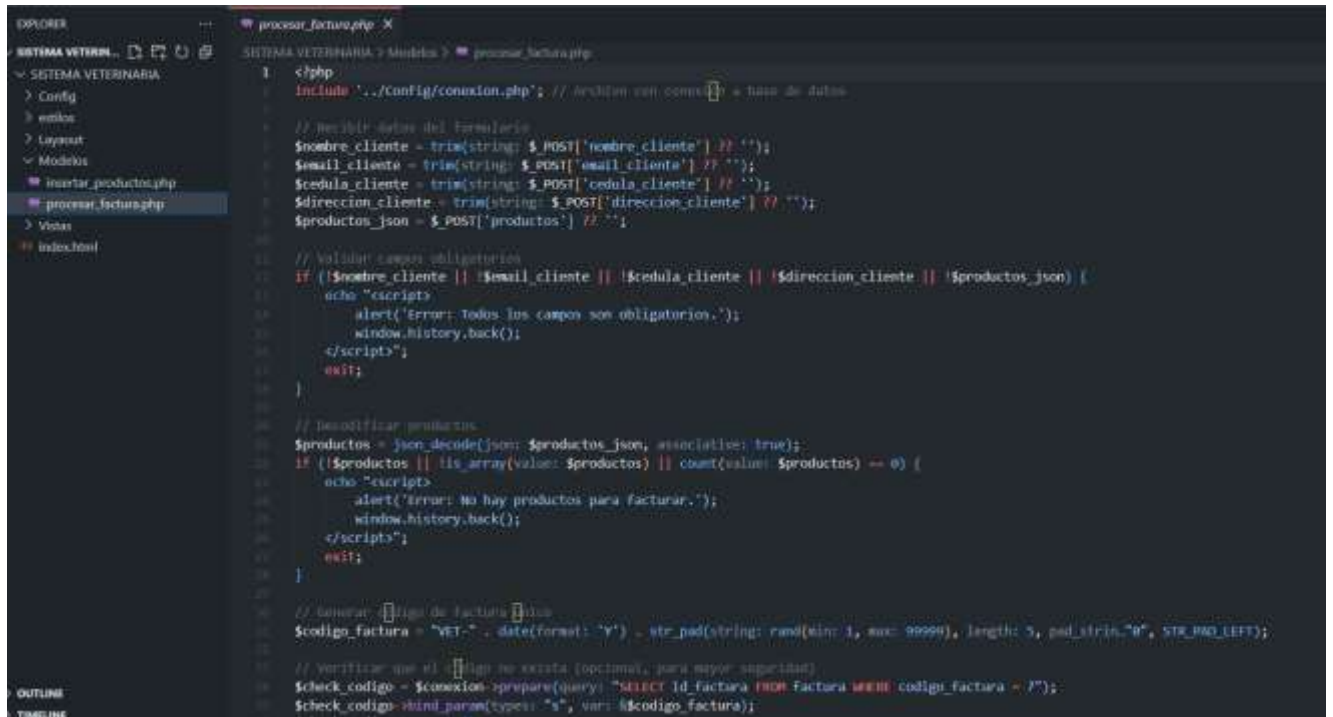
```

EXPLORES
SYSTEMA VETERINARIA > Modelos > insertar_productos.php
SYSTEMA VETERINARIA > Modelos > insertar_productos.php
1  <?php
2  include '../Config/conexion.php'; // Archivo con conexión a base de datos
3
4  // Obtener datos del formulario
5  $codigo = $_POST['codigo'] ?? '';
6  $nombre = $_POST['nombre'] ?? '';
7  $precio = $_POST['precio'] ?? 0;
8  $stock = $_POST['stock'] ?? 0;
9
10 // Validar que no estén vacíos
11 if(empty($codigo) || empty($nombre) || $precio <= 0 || $stock < 0){
12     echo "<script>alert('Por favor completa todos los campos correctamente.');

```

✓ Facturación electrónica:

Formulario para generar facturas,
que envía datos a Modelos/procesar_factura.php (POST).



```
1 <?php
2 include '../Config/conexion.php'; // Archivo con conexión a base de datos
3
4 // Recibir datos del formulario
5 $nombre_cliente = trim(string: $_POST['nombre_cliente'] ?? '');
6 $email_cliente = trim(string: $_POST['email_cliente'] ?? '');
7 $cedula_cliente = trim(string: $_POST['cedula_cliente'] ?? '');
8 $direccion_cliente = trim(string: $_POST['direccion_cliente'] ?? '');
9 $productos_json = $_POST['productos'] ?? '';
10
11 // Validar campos obligatorios
12 if (!$nombre_cliente || !$email_cliente || !$cedula_cliente || !$direccion_cliente || !$productos_json) {
13     echo "<script>
14         alert('Error: Todos los campos son obligatorios.');
15         window.history.back();
16     </script>";
17     exit;
18 }
19
20 // Decodificar productos
21 $productos = json_decode(json: $productos_json, associative: true);
22 if (!$productos || !is_array(value: $productos) || count(value: $productos) == 0) {
23     echo "<script>
24         alert('Error: No hay productos para facturar.');
25         window.history.back();
26     </script>";
27     exit;
28 }
29
30 // Generar código de factura único
31 $codigo_factura = "VET-" . date(format: 'Y') . str_pad(string: rand(min: 1, max: 99999), length: 5, pad: str: "0", STR_PAD_LEFT);
32
33 // Verificar que el código no exista (opcional, para mayor seguridad)
34 $check_codigo = $conexion->prepare(query: "SELECT Id_factura FROM factura WHERE codigo_factura = ?");
35 $check_codigo->bind_param(type: "s", var: $codigo_factura);
```

Permite generar una factura electrónica, ingresando los datos del cliente y los productos a facturar.

Operaciones que maneja el Backend

El backend maneja las siguientes operaciones principales:

Agregar productos:

- Valida que los campos no estén vacíos.
- Inserta el producto en la base de datos usando consultas preparadas.
- Muestra mensajes de éxito o error.


```

File Edit Selection View Go Run Terminal Help
SISTEMA VETERINARIA

Protected Mode is intended for safe code browsing. Trust this folder to enable all features. Manage Learn More

EXPLORER
SISTEMA VETERINARIA
  Config
  estilos
  Layout
  footer.php
  navbar.php
  Modelos
  insertar_productos.php
  procesar_factura.php
  Vistas
    Agregar_producto.php
    comprobante.php
    factura_electronica.php
  index.html

Agregar_producto.php
SISTEMA VETERINARIA > Vistas > Agregar_producto.php

1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4 <meta charset="UTF-8" />
5 <meta name="viewport" content="width=device-width, initial-scale=1" />
6 <title>Agregar Producto - Veterinaria</title>
7 <link rel="stylesheet" href="../../estilos/agregar_producto.css" />
8 </head>
9 <?php include '../../Layout/navbar.php'; ?>
10 <body>
11
12 <div class="container">
13 <h2>Agregar Producto</h2>
14 <form action="../../Modelos/insertar_productos.php" method="POST" id="formProducto">
15 <label for="codigo">Codigo:</label>
16 <input type="text" id="codigo" name="codigo" required />
17
18 <label for="nombre">Nombre:</label>
19 <input type="text" id="nombre" name="nombre" required />
20
21 <label for="precio">Precio:</label>
22 <input type="number" step="0.01" min="0" id="precio" name="precio" required />
23
24 <label for="stock">Stock:</label>
25 <input type="number" min="0" id="stock" name="stock" required />
26
27 <button type="submit">Agregar Producto</button>
28 </form>
29 </div>
30 </body>
31 <?php include '../../Layout/footer.php'; ?>
32 </html>

```

Procesar facturas:

- Valida los datos del cliente y los productos.
- Verifica que los productos existan y haya suficiente stock.
- Inserta la factura y los detalles en la base de datos.
- Actualiza el stock de los productos.
- Muestra mensajes de éxito o error y redirige al comprobante.

```

factura_electronica.php
SISTEMA VETERINARIA > Vistas > factura_electronica.php

1 <?php
2 include '../../Config/conexion.php';
3
4 // Obtener productos para mostrar en el select
5 $query = "SELECT id_producto, codigo, nombre, precio, stock FROM productos WHERE stock > 0";
6 $result = $conexion->query($query);
7 $productos = [];
8 if ($result) {
9     while ($row = $result->fetch_assoc()) {
10         $productos[] = $row;
11     }
12 }
13 ?>
14 <!DOCTYPE html>
15 <html lang="es">
16
17 <head>
18 <meta charset="UTF-8" />
19 <meta name="viewport" content="width=device-width, initial-scale=1" />
20 <title>Facturación Electrónica - Veterinaria</title>
21 <link rel="stylesheet" href="../../estilos/factura_electronica.css" />
22 <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css" rel="stylesheet">
23 <script>
24     let productosSeleccionados = [];
25     let contadorProductos = 0;
26
27     function agregarProductoClick(id, nombre, precio, stock, codigo) {
28         // Verificar si el producto ya está en la lista
29         let productoExistente = productosSeleccionados.find(p => p.id === id);
30
31         if (productoExistente) {
32             // Si existe, incrementar cantidad
33             if (productoExistente.cantidad < stock) {
34                 productoExistente.cantidad++;
35                 mostrarNotificacion('Cantidad aumentada para ${nombre}', 'success');

```



```
stricted Mode is intended for safe code browsing. Trust this folder to enable all features. Manage Learn More

EXPLORER
SISTEMA VETERINARIA
  Config
  estilos
  Layout
  Footer.php
  Navbar.php
  Modelos
  insertar_producto.php
  procesar_factura.php
  Vistas
  Agregar_producto.php
  comprobante.php
  factura_electronica.php
  index.html

comprobante.php
1 <?php
2 include '../Config/conexion.php';
3
4 // Recibir id factura o id tipo por GET
5 $id_factura = $_GET['id'] ?? null;
6 if (!$id_factura) {
7     die("Factura no especificada.");
8 }
9
10 // Consultar datos de la factura
11 $sql_factura = "SELECT codigo_factura, nombre_cliente, email, cedula, direccion, total
12 FROM factura WHERE id_factura = ?";
13 $stmt = $conexion->prepare(query: $sql_factura);
14 $stmt->bind_param(types: "i", var: &$id_factura);
15 $stmt->execute();
16 $result_factura = $stmt->get_result();
17 if ($result_factura->num_rows == 0) {
18     die("Factura no encontrada.");
19 }
20 $factura = $result_factura->fetch_assoc();
21 $stmt->close();
22
23 // Consultar detalles (productos)
24 $sql_detalle = "SELECT p.codigo, p.nombre, df.cantidad, df.precio_unitario
25 FROM detalle_factura df
26 JOIN productos p ON df.id_producto = p.id_producto
27 WHERE df.id_factura = ?";
28 $stmt_detalle = $conexion->prepare(query: $sql_detalle);
29 $stmt_detalle->bind_param(types: "i", var: &$id_factura);
30 $stmt_detalle->execute();
31 $result_detalle = $stmt_detalle->get_result();
32
33 $productos = [];
34 while ($row = $result_detalle->fetch_assoc()) {
35     $productos[] = $row;
36 }
```

ANÁLISIS DINÁMICO

Este tipo de análisis se centra en observar el comportamiento del sistema en tiempo real. Ejecución de la aplicación

Pruebas funcionales

Inicio: Se abre index.html y se verifica que carga correctamente.



Agregar producto:

- Se ingresa datos válidos y verifica que el producto se agrega y aparece en la base de datos.

Veterinaria Patitas

Inventarios + Facturación Electrónica

Agregar Productos

Agregar Producto

Código:

Nombre:

Precio:

Stock:

Agregar Producto

Veterinaria Patitas

Inventarios + Facturación Electrónica

Agregar Productos

Facturación Electrónica

Complete los datos del cliente y agregue productos a la factura.

Datos del Cliente

Nombre completo: KATHERIN NARVAEZ

Email: ANGELICAG99@SIGMAWI

Cédula: 1150000814

Dirección: ZARATO

Veterinaria Patitas

Inventarios + Facturación Electrónica

Agregar Productos

Seleccionar Productos

SHAMPOO
Código: MM8
\$2.50 23 disponibles

JABON
Código: MM62
\$1.50 18 disponibles

REMEDIO PULGAS
Código: M002
\$3.00 19 disponibles

COLLAR
Código: M005
\$2.00 16 disponibles

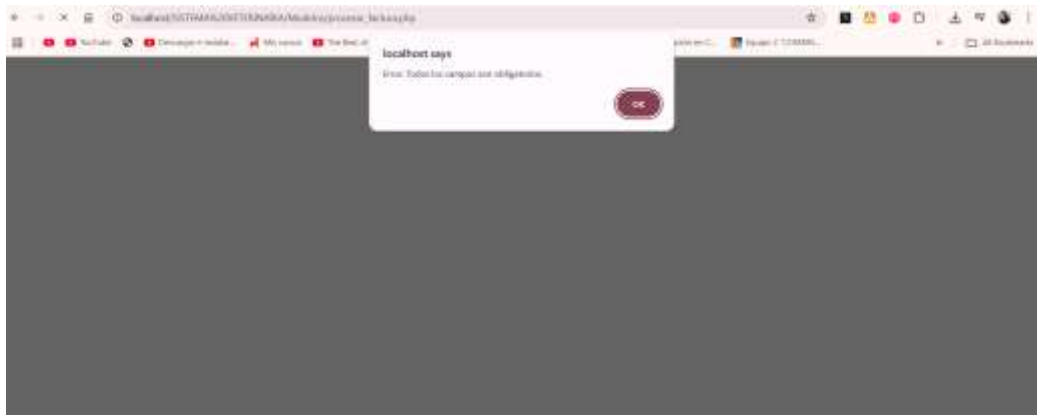
Facturación electrónica:

-Realiza una factura con productos existentes y verifica que se descuenta el stock y se genere el comprobante.

Productos Seleccionados					0 productos
Producto	Cantidad	Precio Unit.	Stock	Total	Acción
PASTILLA BICHOS Código: M005	2	\$0.50	11 disponibles	\$1.00	
PRUEBA Código: M0012	2	\$1.00	12 disponibles	\$2.00	
JABON Código: MM62	1	\$1.50	18 disponibles	\$1.50	
REMEDIO PULGAS Código: M002	3	\$3.00	19 disponibles	\$9.00	
Total: \$13.50					

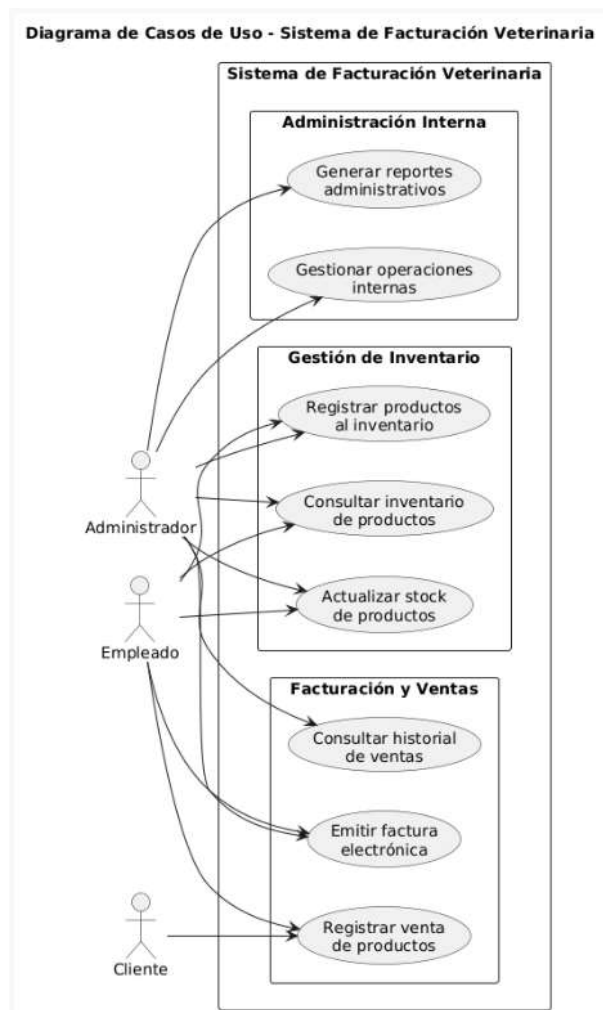
Comprobantes:

Se accede a la vista de comprobante y se verifica que se produce un error no genera el comprobante.



Diagramas :

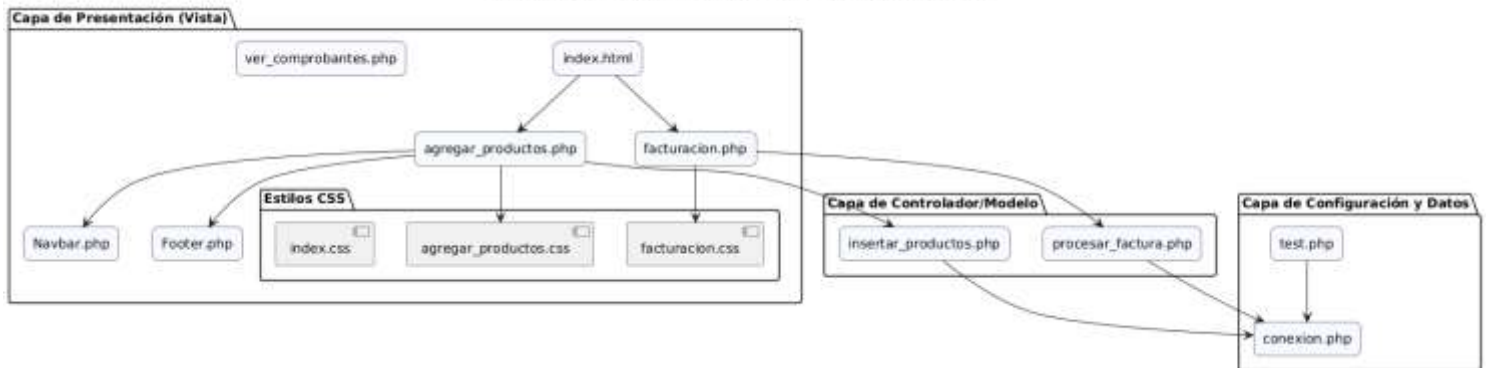
- Diagrama de Casos de Uso :



Este diagrama muestra las principales funcionalidades que el usuario puede realizar al interactuar con la aplicación. En este caso, representa cómo el usuario. Este diagrama es útil para entender las necesidades del usuario y los servicios que el sistema debe ofrecer desde su perspectiva.

- Diagrama de la Arquitectura

Arquitectura por Capas - Sistema de Facturación Veterinaria (MVC)



- Diagrama de Procesos del Sistema

Estos diagramas describen el flujo funcional del sistema, mostrando paso a paso las acciones que se ejecutan desde que el usuario inicia una operación hasta que se presenta el resultado. Incluye la entrada de datos, selección de la operación, validaciones y la visualización del resultado.

Diagrama de Actividades - Proceso de Facturación en Veterinaria



6. CONCLUSIONES Y RECOMENDACIONES:

El análisis mediante ingeniería inversa permitió identificar con claridad los componentes clave del sistema, su arquitectura en capas (presentación, lógica y datos) y las dependencias entre ellos. Sin embargo, el proceso reveló limitaciones en la documentación interna, lo que dificultó la comprensión de ciertos módulos complejos, como el manejo de transacciones en la pasarela de pagos. Además, la falta de comentarios descriptivos en el código y la escasa trazabilidad entre los requisitos y su implementación complicaron el mapeo completo de funcionalidades.

Recomendaciones:

Para optimizar futuros ejercicios de ingeniería inversa, se recomienda implementar una documentación técnica detallada que incluya diagramas de secuencia y flujo de datos críticos. Asimismo, sería valioso incorporar herramientas de análisis estático automatizado (como SonarQube) para identificar rápidamente patrones problemáticos, como código no seguro o redundante. Estas mejoras no solo agilizarían el proceso de ingeniería inversa, sino que también facilitarían el mantenimiento y la evolución del sistema, asegurando una base más sólida para posteriores actualizaciones.

7. BIBLIOGRAFÍA:

- Chikofsky, E. J., & Cross, J. H. (1990). *Reverse Engineering and Design Recovery: A Taxonomy*. IEEE Software, 7(1), 13-17.
- Pressman, R. S. (2010). *Ingeniería del Software: Un Enfoque Práctico* (7ª ed.). McGraw-Hill. (Capítulo 29: "Reengineering and Reverse Engineering")
- Sommerville, I. (2011). *Software Engineering* (9ª ed.). Pearson. (Sección 27.4: "Reverse Engineering")