

**Icono

Descripción generada automáticamente**Icono

Descripción generada automáticamente

**ESCUELA SUPERIOR POLITÉCNICA**

**DE CHIMBORAZO**

**PAO 8**

**18/07/2025**

**Nombre:** Angélica Narváez (6779)

## Tarea: Análisis estático y dinámico(Ingeniería Inversa)

**Ingeniería en software**

**APLICACIONES INFORMATICAS II**

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.ESPECÍFÍCOS**

-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.

1. **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.

1. **EQUIPOS Y MATERIALES:**

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

1. **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 aplicar 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.

1. **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).

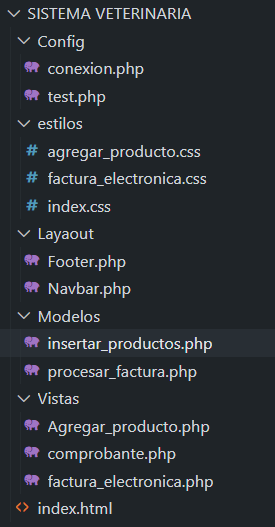
**Estilos:** Carpeta donde se encuentran los archivos CSS para los diferentes módulos o páginas del sistema, como agregar productos, facturación electrónica e index.

**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.



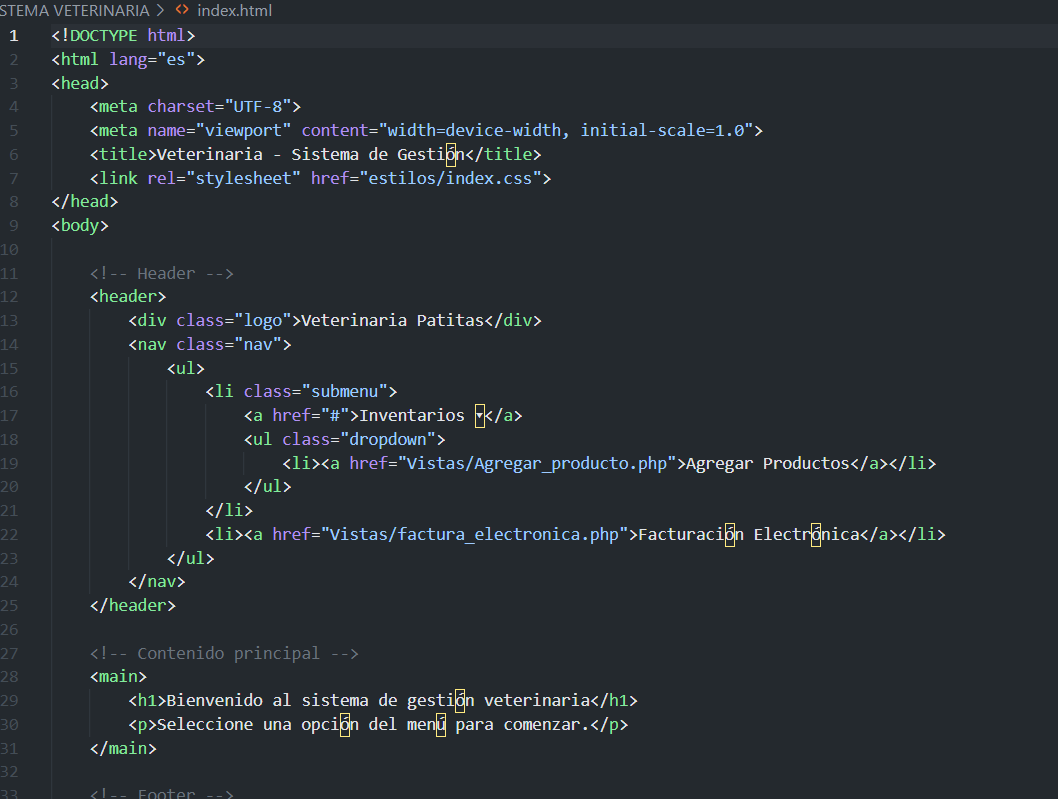
***Fig 1. Estructura del código***

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

**Funcionalidades principales**

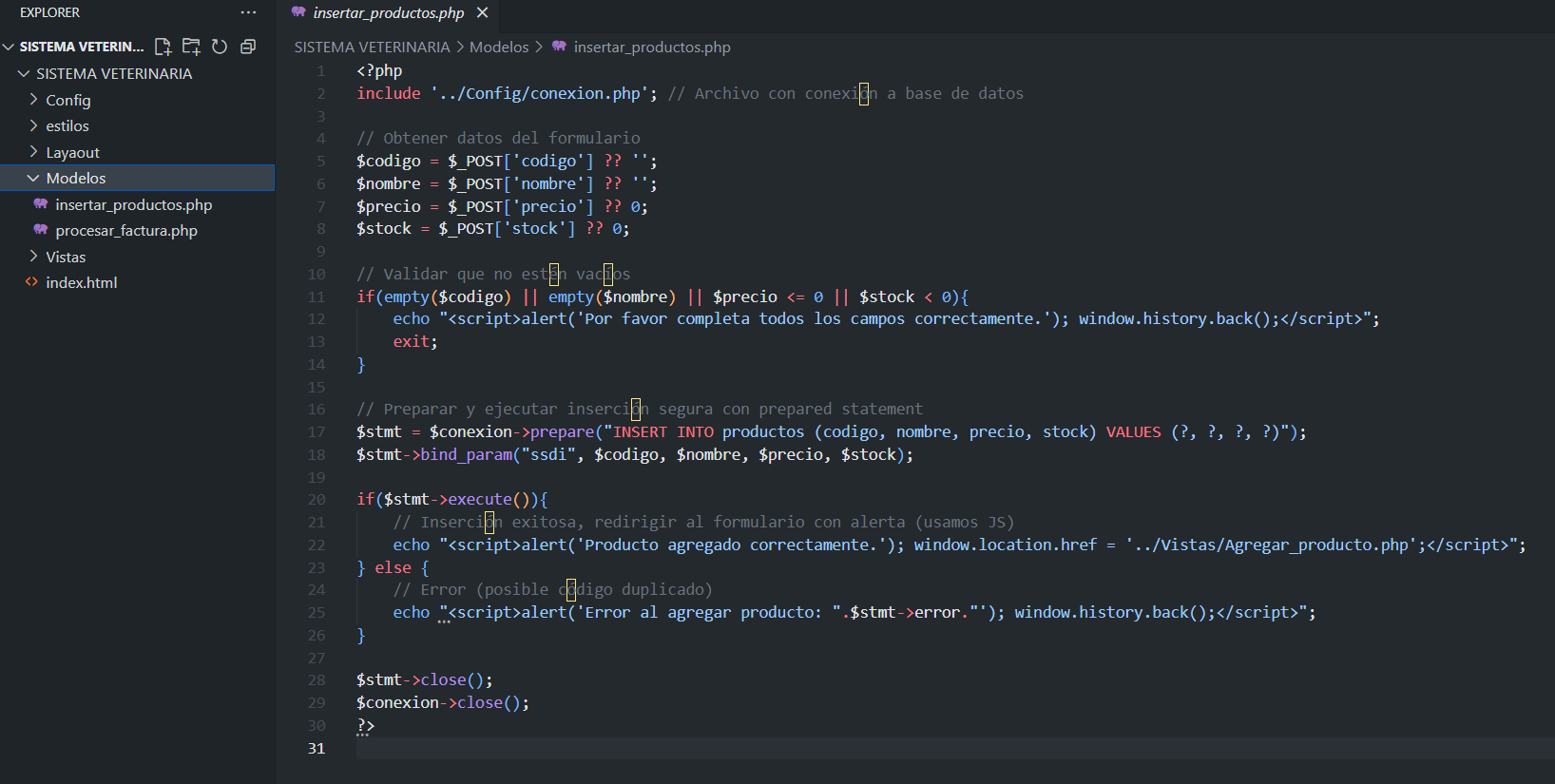
* **Ruta principal:**

Muestra la página de inicio (index.html).



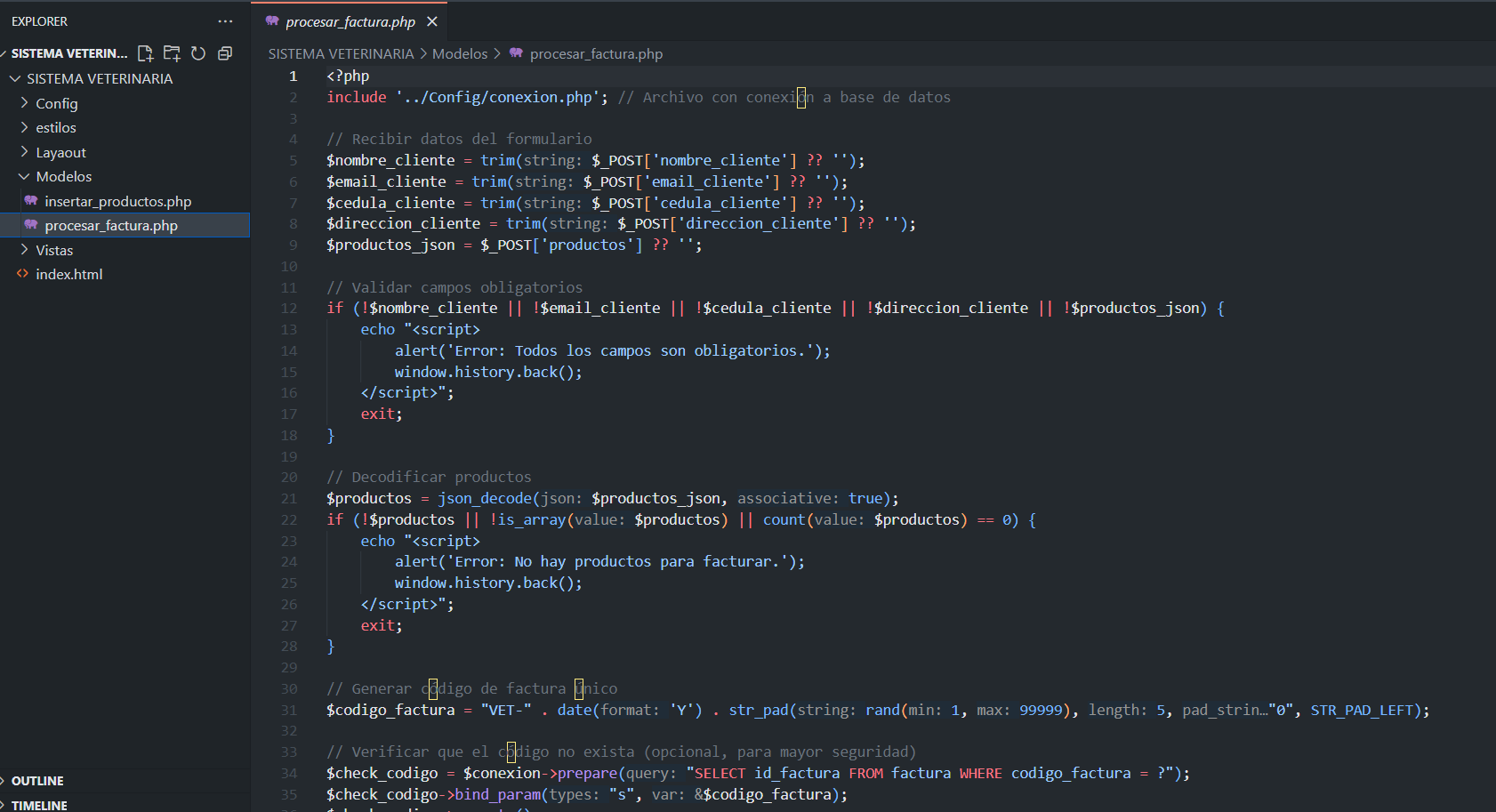
* **Agregar producto:**

Formulario para agregar productos, que envía datos a *Modelos/insertar\_productos.php* (POST), donde se procesan y almacenan en la base de datos.



* **Facturación electrónica:**

Formulario para generar facturas, que envía datos *a Modelos/procesar\_factura.php* (POST).



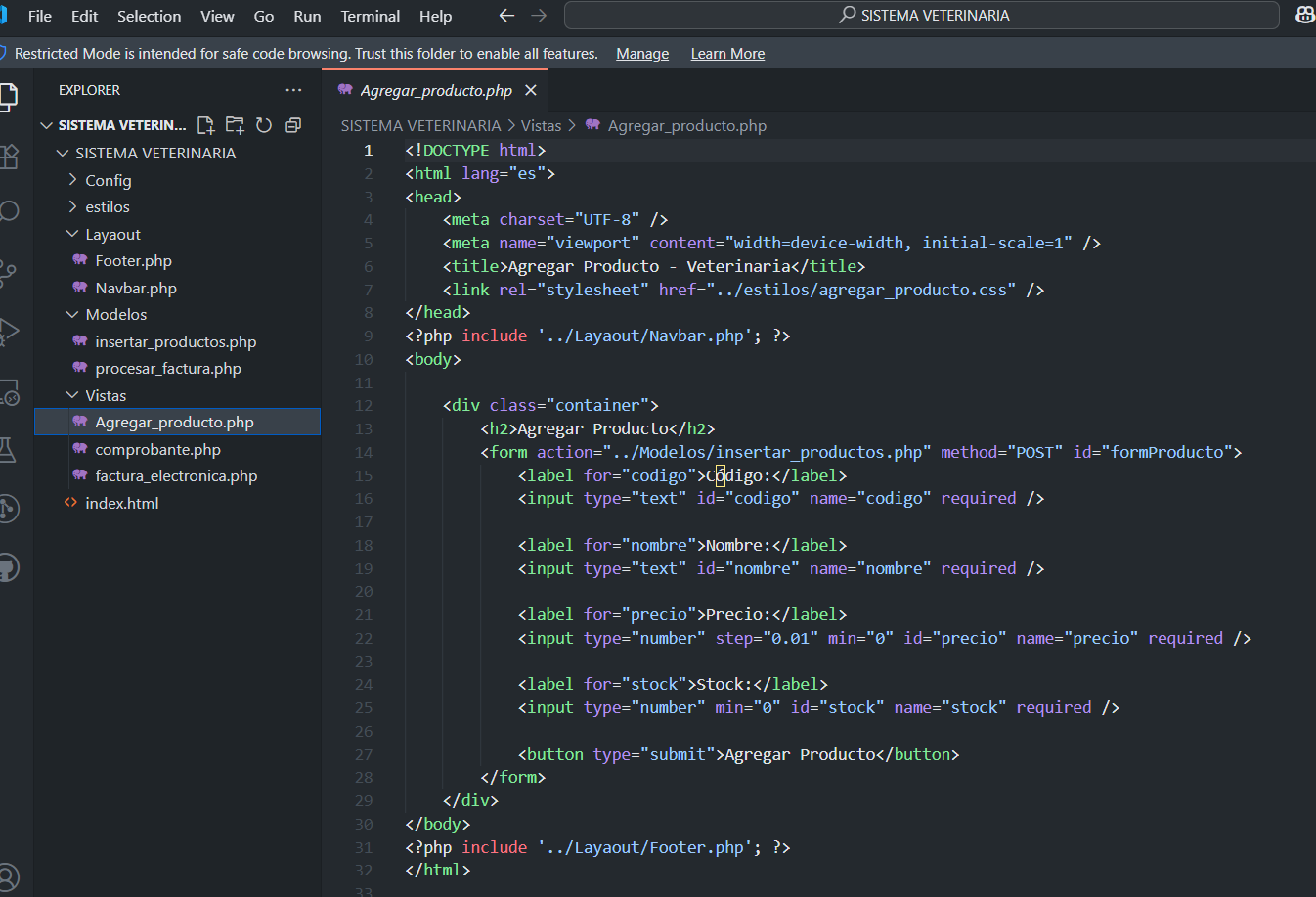
Permite generar una factura electrónica, ingresando los datos del cliente y los productos a facturar.

**Operaciones que maneja el Backend**

El backend maneja las siguientes operaciones principales:

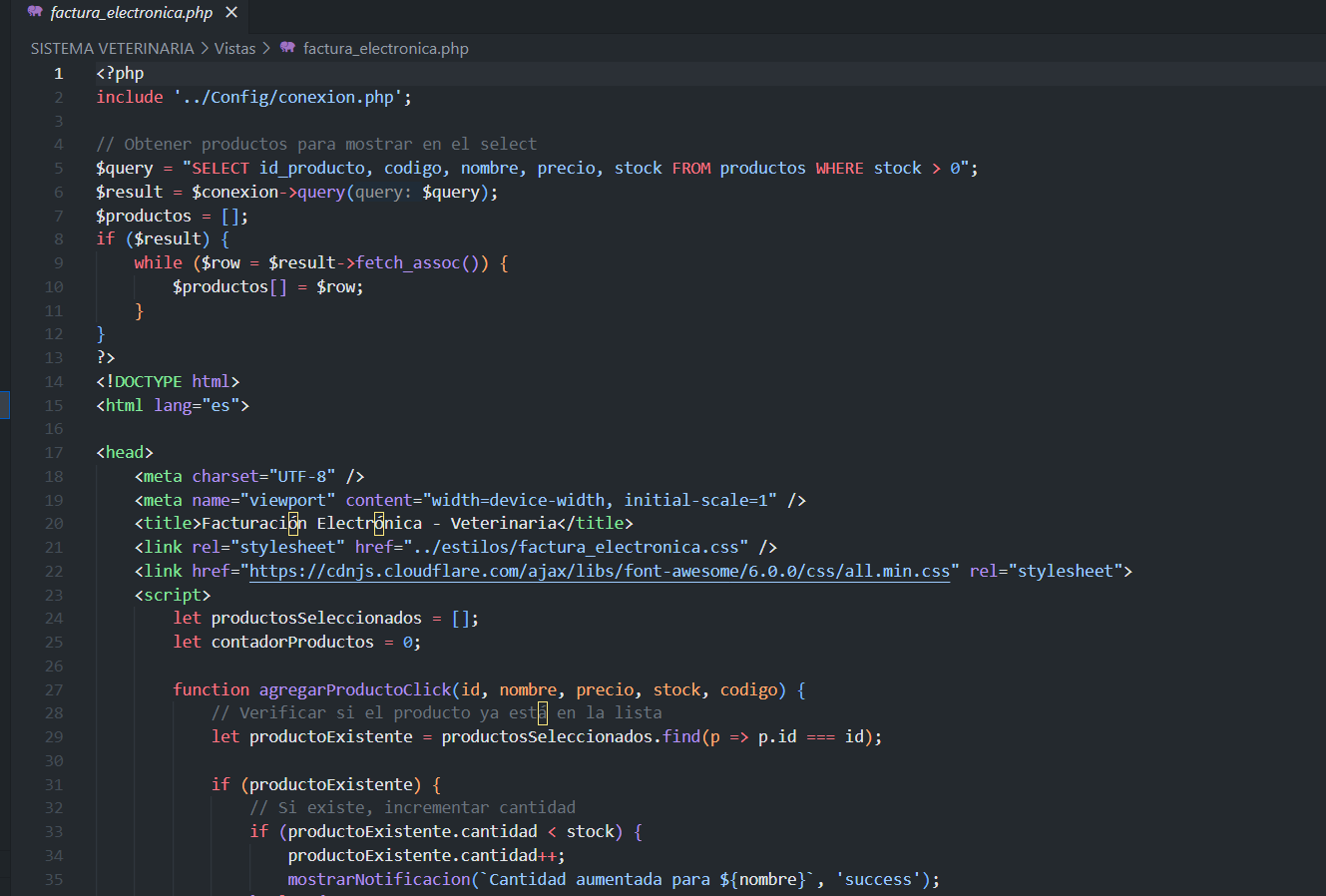
**Agregar productos:**

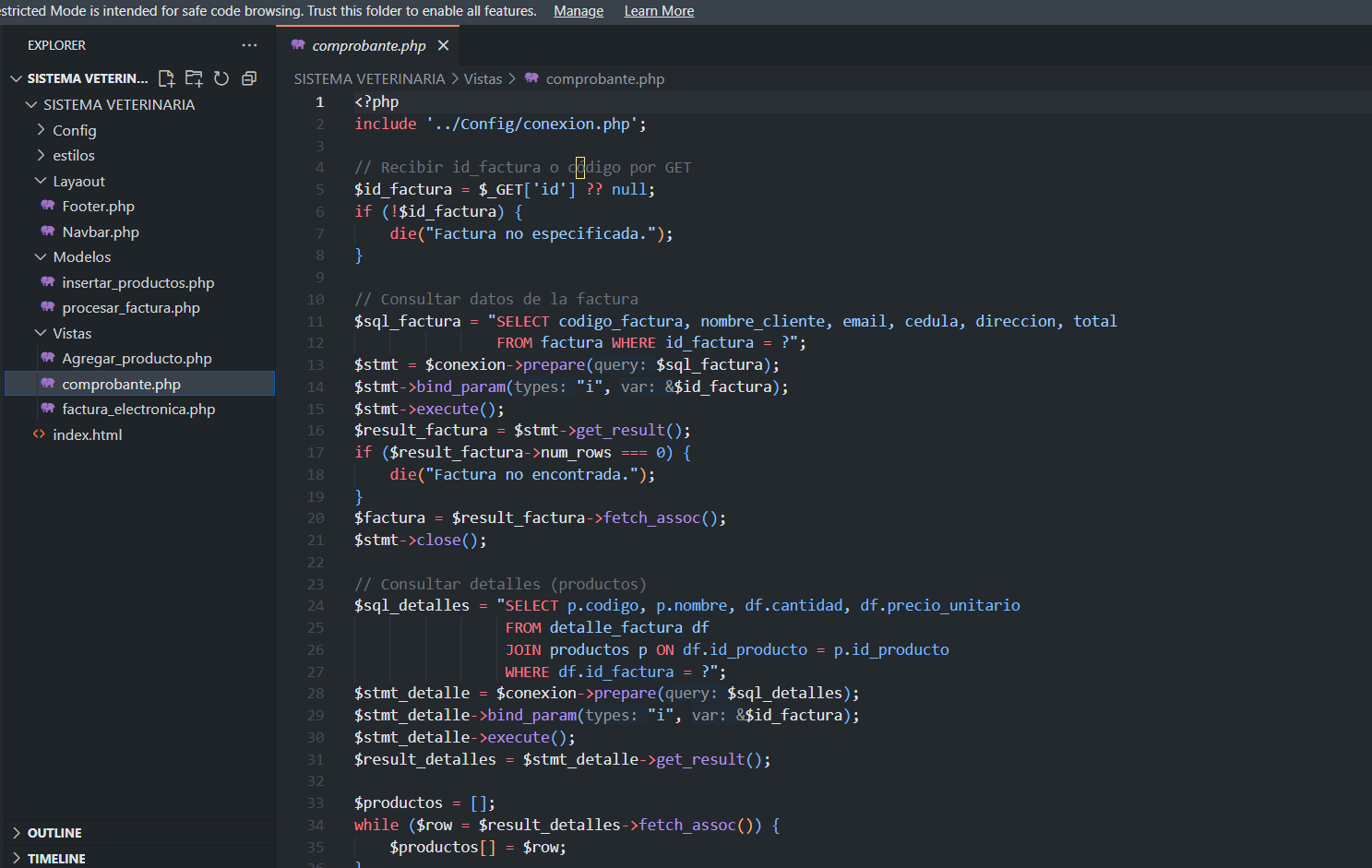
* Valida que los campos no estén vacíos.
* Inserta el producto en la base de datos usando consultas preparadas.
* Muestra mensajes de éxito o error.



**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.





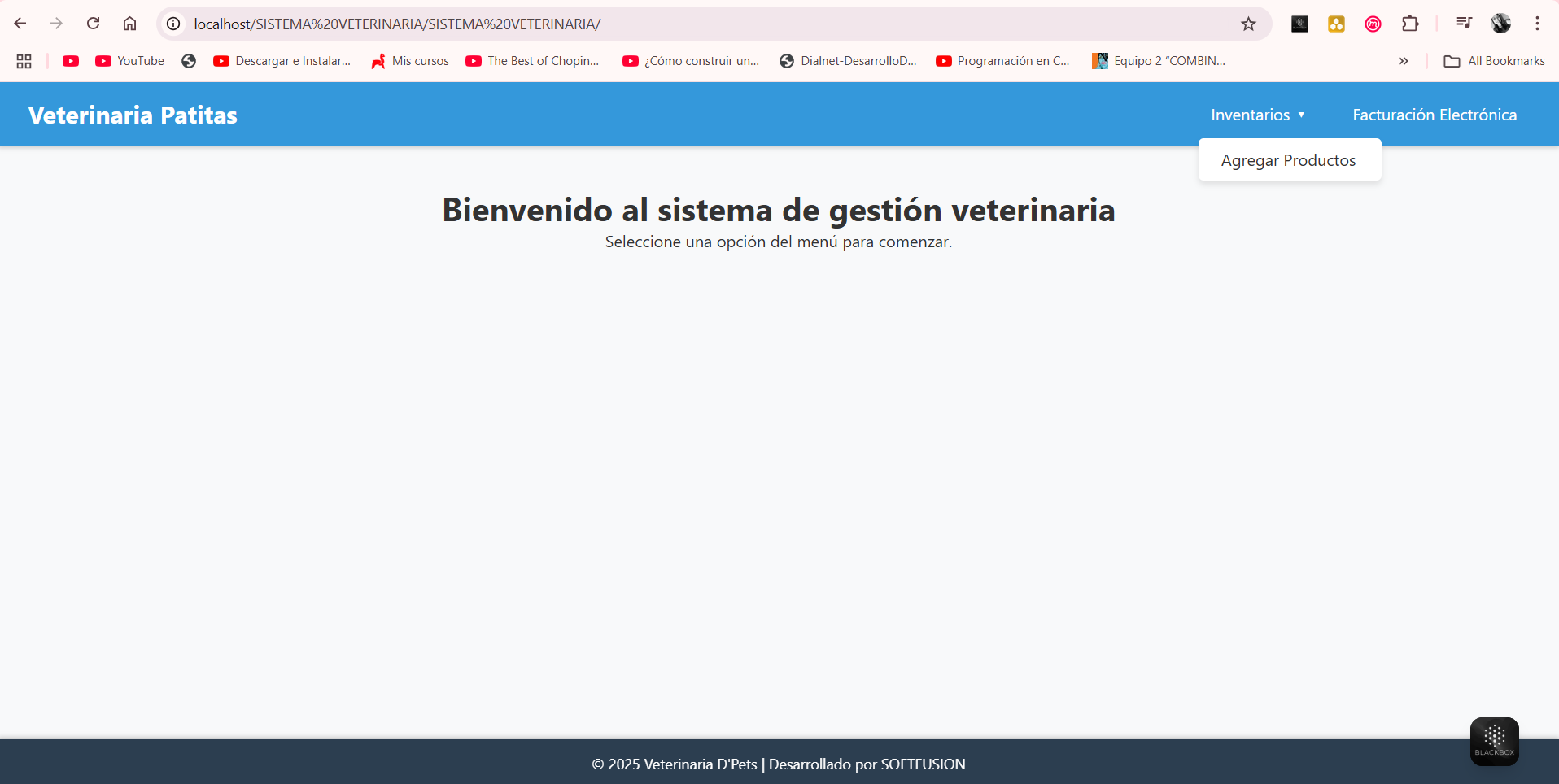
**ANÁLISIS DINÁMICO**

**Este tipo de análisis se centra en observar el comportamiento del sistema en tiempo real.**

**Ejecución de la aplicación**

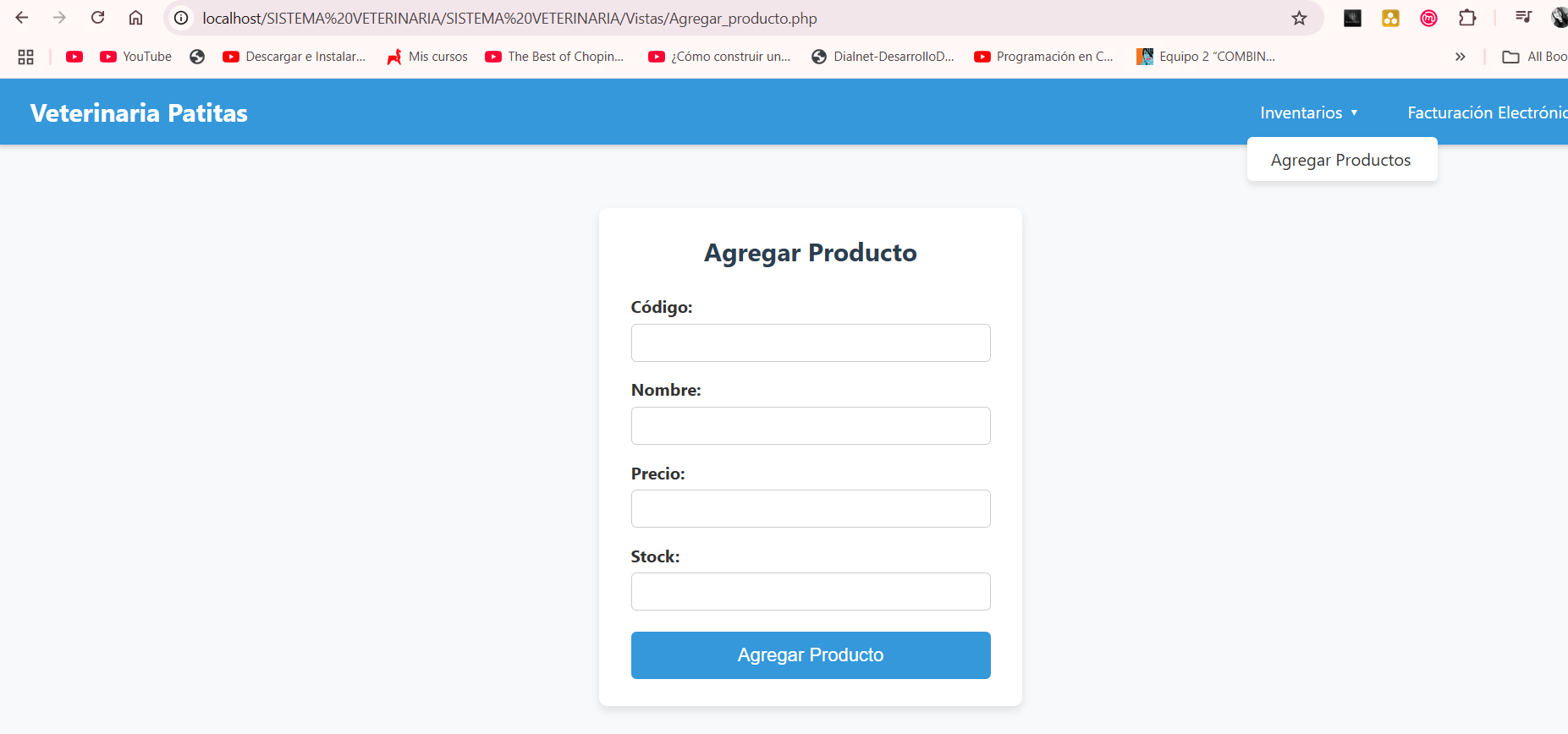
**Pruebas funcionales**

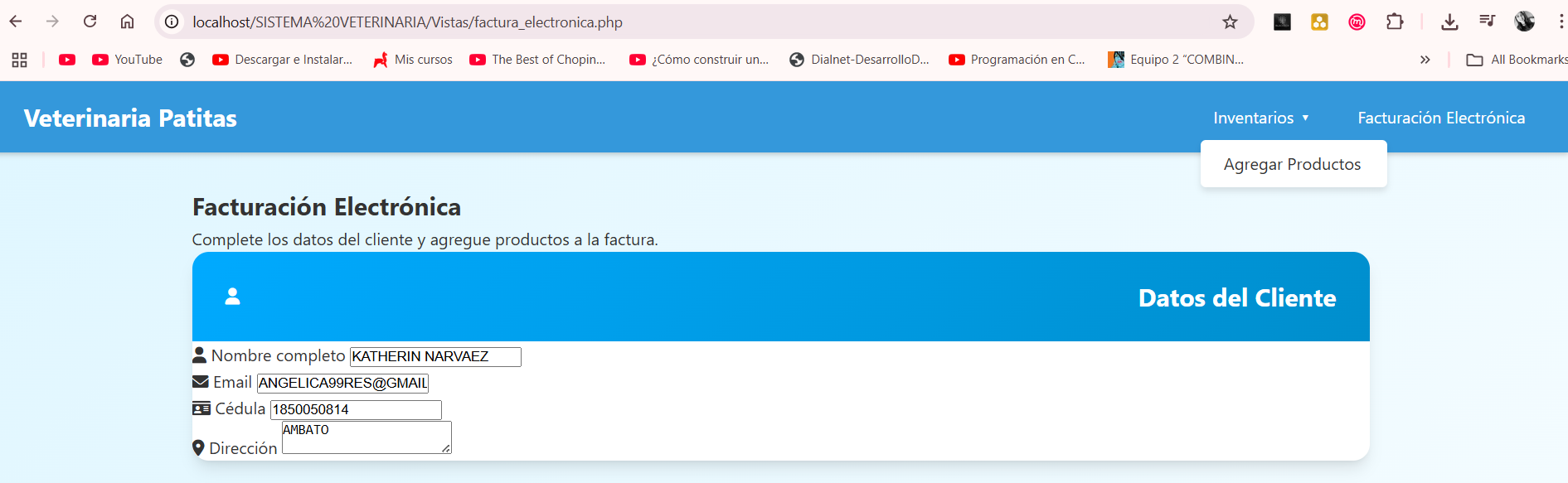
**Inicio:** Se abre index.html y se verifica que carga correctamente.

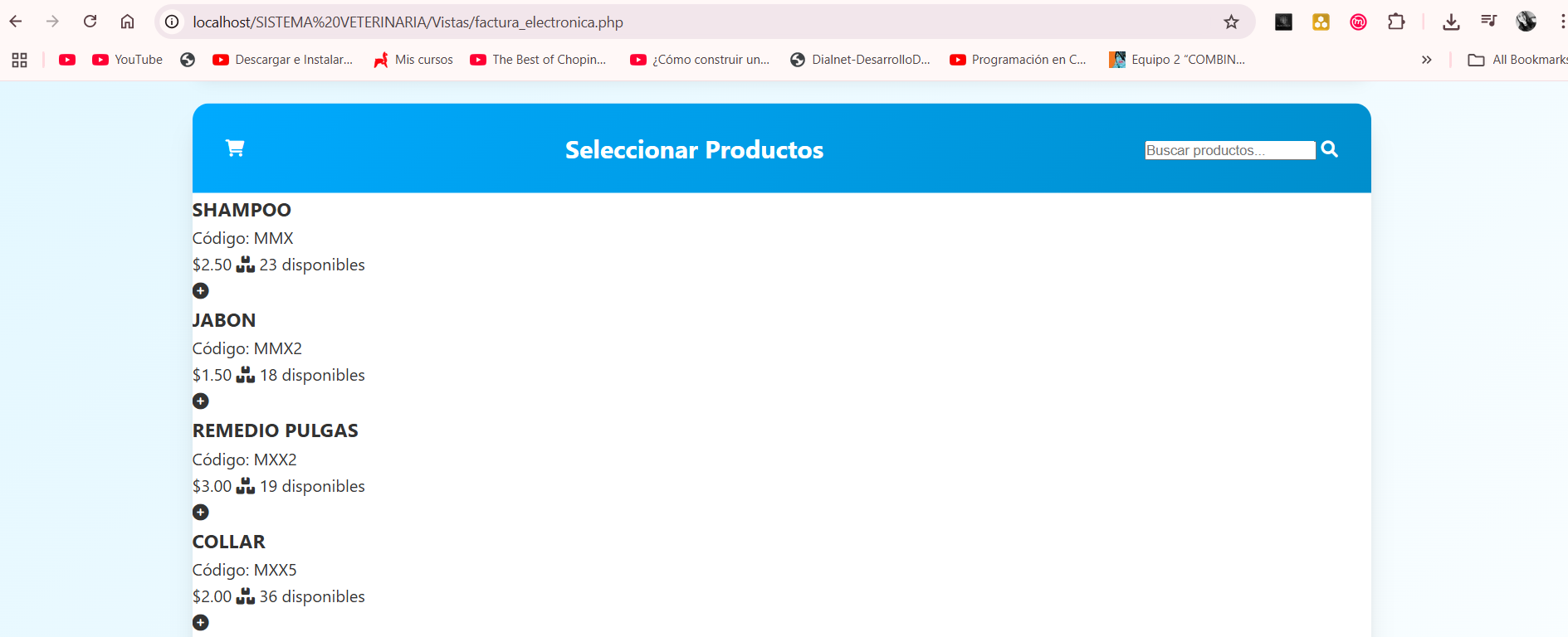


**Agregar producto:**

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







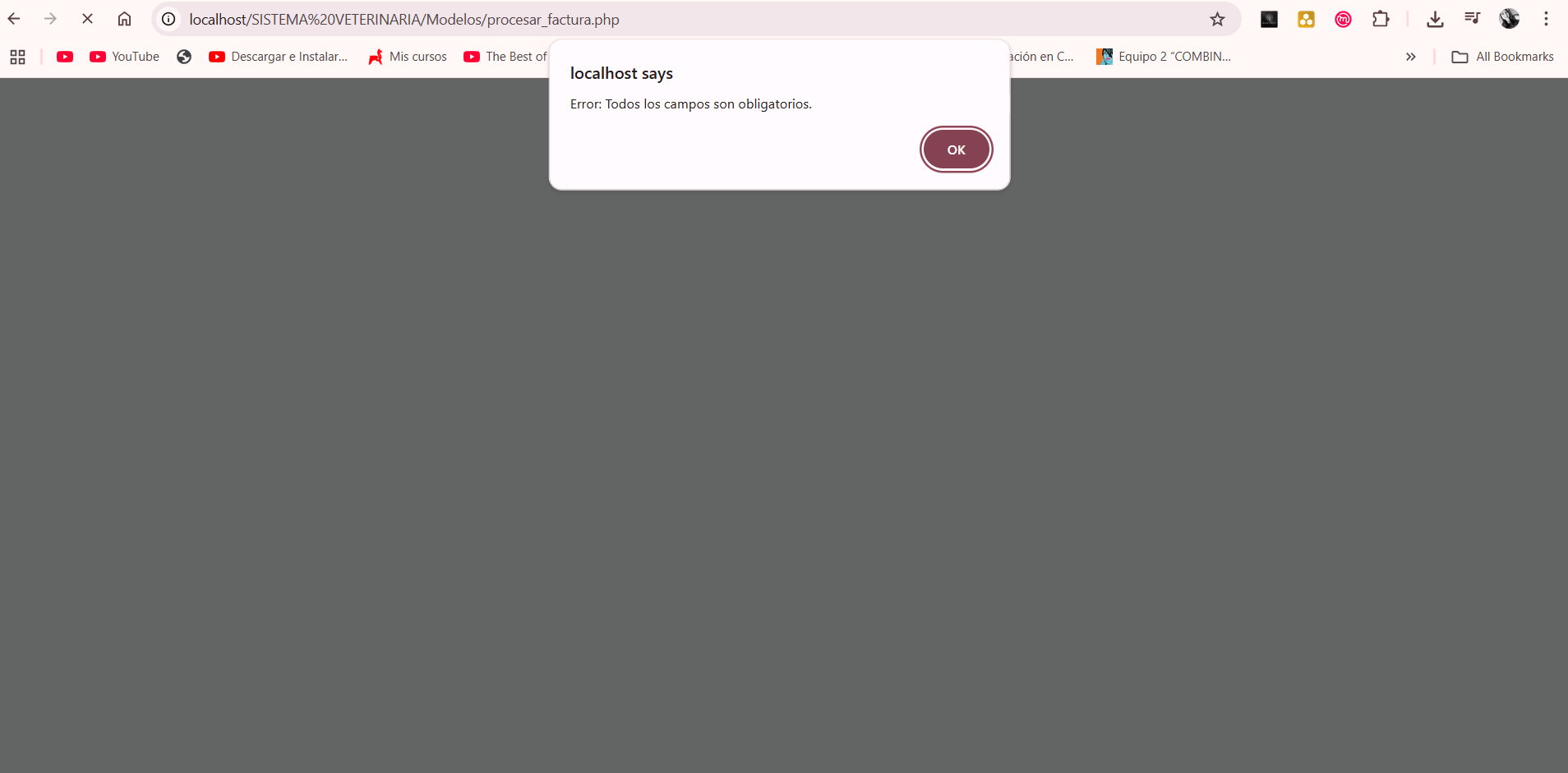
**Facturación electrónica:**

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



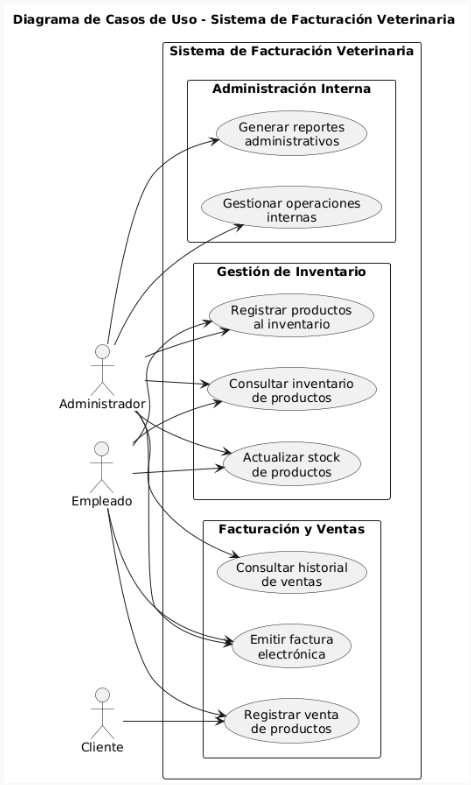
**Comprobantes:**

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



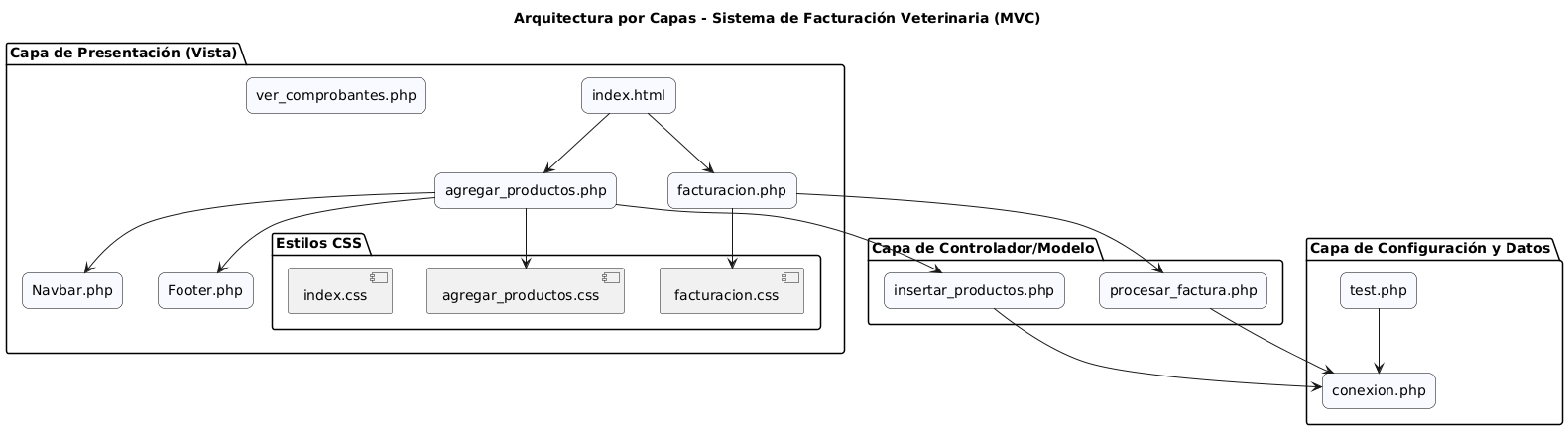
**Diagramas :**

* Diagrama de Casos de Uso :

****

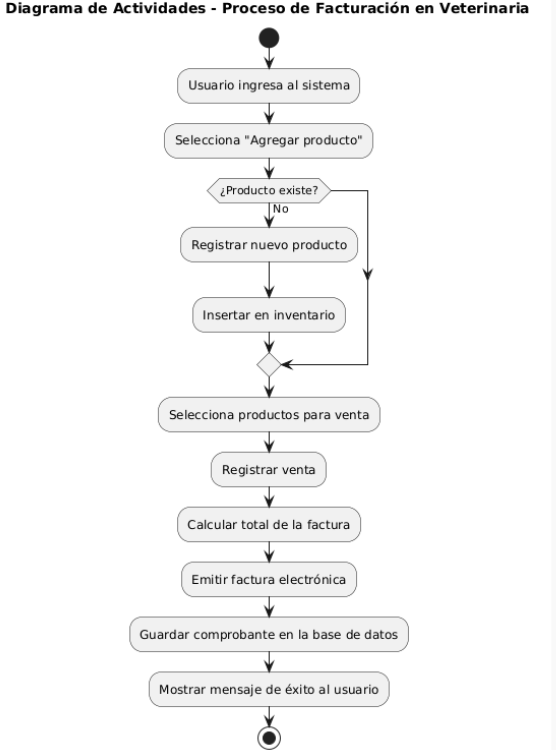
Este diagrama muestra las principales funcionalidades que el usuario puede realizar al interactuar con la aplicación. En este caso, representa cómo el usuario. Este diagrama es útil para entender las necesidades del usuario y los servicios que el sistema debe ofrecer desde su perspectiva.

* **Diagrama de la Arquitectura**

****

* **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.

****

1. **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.

1. **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")