ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD: INFORMÁTICA Y ELECTRÓNICA
ESCUELA DE INGENIERÍA EN SISTEMAS
CARRERA: SOFTWARE

GUÍA DE LABORATORIO DE APLICACIONES INFORMÁTICAS II

1. DATOS GENERALES:

NOMBRE DEL ESTUDIANTE	CODIGO DEL ESTUDIANTE
Katherin Narváez	6779
FECHA DE REALIZACIÓN:	**FECHA DE ENTREGA:**
2025/07/18	**2025/07/18**

2. OBJETIVO(S):

2.1.GENERAL

Desarrollar la fase de documentación adjunto al proceso de ingeniería inversa sore una aplicación de software ya existente la cual fue analizada y valorada, con el fin de comprender sus componentes, arquitectura, su funcionamiento y el análisis de consideraciones para que faciliten su mantenimiento.

2.2.ESPECÍFÍCOS

-Recopilar información del análisis anteriormente realizado y la aplicación de ingeniería inversa.

- Documentar la aplicación de ingeniería inversa sobre el aplicativo informático elegido en base a lo visto en clases.

3. METODOLOGÍA

En el presente documento se realizara la documentación de la aplicación de la técnica de ingeniería inversa aplicada anteriormente sobre un software existente. Mediante el análisis estático y dinámico.

4. EQUIPOS Y MATERIALES:

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

5. MARCO TEORICO:

Proceso de ingeniería inversa:

La ingeniería inversa puede ser un proceso complejo, pero generalmente sigue algunos pasos clave:

La ingeniería inversa de software es una disciplina que consiste en analizar un sistema informático existente para identificar sus componentes, estructura y comportamiento, sin tener acceso directo a su documentación original o especificaciones.
Este enfoque es útil en contextos donde se requiere mantenimiento, actualización o documentación de sistemas heredados, facilitando la comprensión de cómo funciona una aplicación desde el código hacia su estructura general.
El objetivo principal de la ingeniería inversa no es rediseñar el software, sino entender cómo fue construido, cómo interactúan sus componentes y qué funcionalidades ofrece. Con ello, se puede generar documentación, modelos y representaciones gráficas que orienten el trabajo de nuevos desarrolladores o equipos de mantenimiento.

Técnicas principales:

1. Análisis estático: Consiste en revisar el código fuente sin necesidad de ejecutarlo. Se examinan archivos, estructuras, módulos, funciones, clases, lógica condicional y organización general del proyecto. Esta técnica es muy útil en lenguajes como Python, donde el código suele ser más legible y directo.

2. Análisis dinámico: Se enfoca en observar cómo se comporta el sistema cuando está en ejecución. Permite ver cómo fluye la información, cómo se reciben y procesan los datos, cómo responde el sistema ante errores o entradas inesperadas, y cómo interactúan el frontend y el backend.

Ambos enfoques son complementarios: mientras que el análisis estático ofrece una visión estructural, el dinámico proporciona información sobre el rendimiento y funcionamiento real del sistema.

Diagramas para documentación:

Como parte del proceso de documentación de la ingeniería inversa, se recurre a diagramas UML (Unified Modeling Language), los cuales permiten representar gráficamente distintos aspectos del sistema.
• Casos de uso: representan las interacciones del usuario con el sistema.
• Diagramas de secuencia: muestran el flujo de mensajes entre los distintos componentes durante una operación.
• Diagramas de actividades: reflejan los pasos y decisiones dentro de un proceso funcional
- Diagramas de arquitectura: presentan la estructura general del sistema, sus capas o componentes y cómo se comunican.

Estos modelos facilitan la lectura técnica del sistema, la comprensión del flujo de trabajo y la identificación de oportunidades de mejora.

Importancia de documentar mantenimiento de software:

• Facilita el mantenimiento y evolución del software.
• Reduce el tiempo necesario para comprender sistemas complejos.
• Mejora la calidad del código mediante la detección de redundancias o errores lógicos.
• Permite la capacitación de nuevos integrantes del equipo de desarrollo.

MANTENIMIENTO ADAPTATIVO

Este mantenimiento se encarga de mantener operativo un programa cuando se realizan cambios en el entorno de producción. Es importante mencionar que este tipo de mantenimiento es uno de los más usuales ya que se debe a los cambios que se producen en la tecnología informática, dejando sin operar a los productos software que han sido desarrollados por la competitividad que se da entre empresas, dando como resultado que el software sea más utilizado mientras ocurran más modificaciones.

Objetivo: Modificar un programa en los diferentes cambios en el entorno, es decir, que se realicen cambios en el hardware o software en el que está siendo ejecutado.

6. PROCEDIMIENTO DOCUMENTACION:

Información del Proyecto

Empresa / Organización	ESPOCH
Proyecto	**Sistema de Facturación(Veterinaria)**
Fecha de preparación	07/18/2025

1. HISTORIAL DE CAMBIOS

Fecha	Versión	Organización	Descripción
07/17/2025	1.0	ESPOCH	Análisis del proyecto existente
07/18/2025	1.0	ESPOCH	Aplicación de ingeniería inversa, análisis estático y dinámico.
07/18/2025	1.0	ESPOCH	Elaboración de diagramas
07/18/2025	1.0	ESPOCH	Elaboración del proceso de documentación de la ingeniería inversa.
07/18/2025	1.0	ESPOCH	Retroalimentación de información, análisis y resultados
07/23/2025	1.0	ESPOCH	Ejecución del Mantenimiento Adaptativo
07/23/2025	1.0	ESPOCH	Ejecución del mantenimiento en el código Refactorizacion.
07/23/2025	1.0	ESPOCH	Documentación detallada del mantenimiento adaptativo de las nuevas funcionalidades del sistema.
07/24/2025	1.1	ESPOCH	Planteamiento de los nuevos diagramas.
07/24/2025	1.1	ESPOCH	Planteamiento de la visualización de los cambios realizados en el mantenimiento aplicado.

6.1 Proceso de Análisis

Como anteriormente realizamos el análisis del dominio mediante la aplicación de ingeniería inversas del programa se pudo comprender que el software que se describe 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.

Una vez que se realizo el respectivo proceso de comprensión del programa establecido, se determinó un análisis estático y dinámico, identificamos:

- Que se presentaron varios aspectos que se podrían considerar como deuda técnica, como es el caso de la generación en pdf o impresión de el comprobante.
- Editar y eliminar productos a una venta para ser facturada.
- Generar comprobante.
- Adaptar la facturación a requisitos establecidos por las autoridades tributarias de la facturación nacional ecuatoriana.

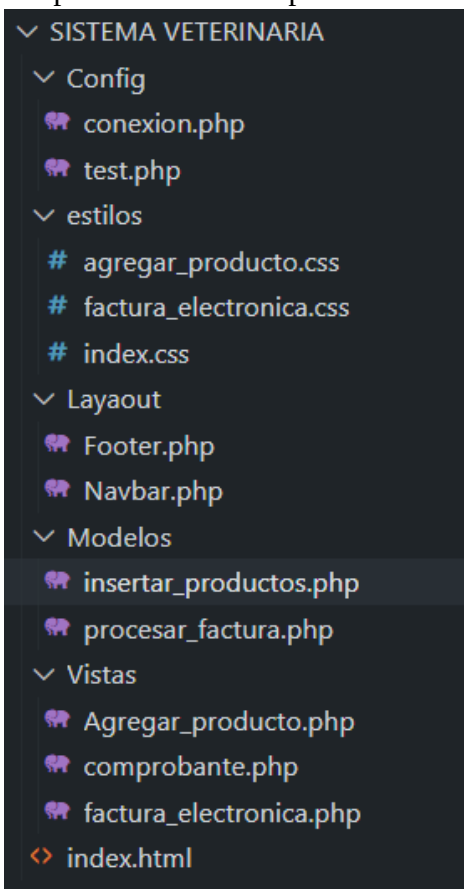
Ejecución del Mantenimiento Adaptativo:

Codificación/Implementación:

→ **Desarrollo de cambios.**

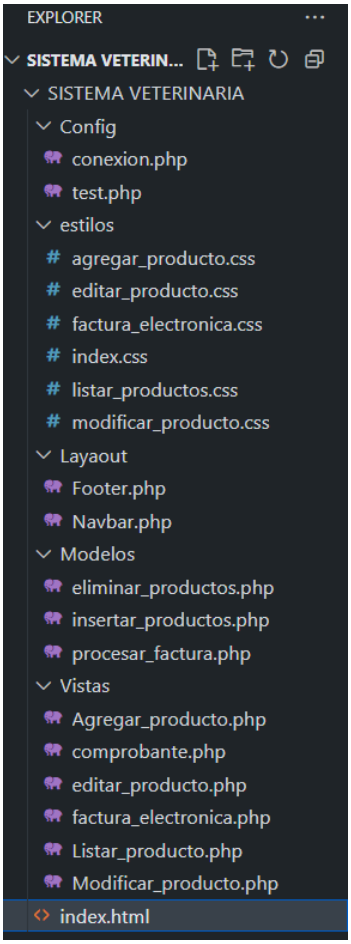
- Que se presentaron varios aspectos que se podrían considerar como deuda técnica, como es el caso de la generación en pdf o impresión de el comprobante.
- Editar y eliminar productos a una venta para ser facturada.
- Generar comprobante.
- Adaptar la facturación a requisitos establecidos por las autoridades tributarias de la facturación nacional ecuatoriana.

En la Primera versión SITEMA VETERINARIA podemos visualizar la estructura de su Arquitectura MVC y sus componentes como se puede ver a continuación.



→ **Implementación de nuevas funcionalidades.**

En la Segunda versión SITEMA VETERINARIA V2 podemos visualizar la estructura de su Arquitectura MVC y sus nuevas funcionalidades y el desarrollo de sus cambios como se puede ver a continuación



DETALLE DE LOS CAMBIOS REALIZADOS:

Refactorización:

→ **Mejora interna sin alterar comportamiento.**

Archivos nuevos en la carpeta **estilos/**

- ✓ **editar_producto.css**
- ✓ **listar_productos.css**
- ✓ **modificar_producto.css**

En estos archivos de estilos se añadido funcionalidades nuevas para editar, listar y modificar productos, y que ahora se pueda priorizar una mejor estructura y organización para que cada vista correspondiente tenga su propia hoja de estilos CSS específica. Esto es una buena práctica para mantener una mejor organización y modularidad del diseño.

→ **Reducción de deuda técnica acumulada.**

Archivos nuevos en la carpeta **Modelos/**

- ✓ **eliminar_productos.php**

En este archivo se ha incorporado una funcionalidad nueva para **eliminar productos** desde el backend o base de datos.

Archivos nuevos en la carpeta **Vistas/**

- ✓ **editar_producto.php**
- ✓ **Listar_producto.php**
- ✓ **Modificar_producto.php**

Posteriormente se añadieron nuevas **vistas** (interfaces de usuario) para editar, listar y modificar productos, don de estas vistas se apoyan en los nuevos archivos CSS.

→ **Optimización del código para eficiencia.**

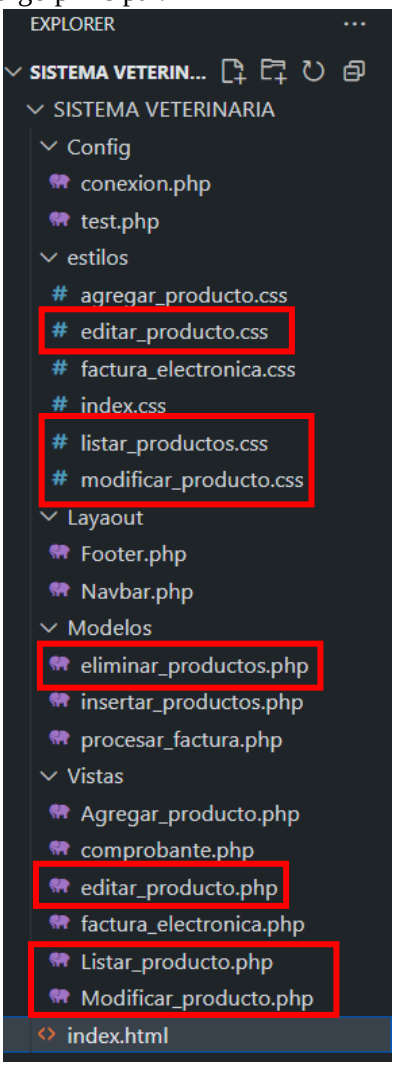
Con la implementación del mantenimiento adaptativo se logro obtener una mejor organización y nomenclatura del código.

En la segunda versión se implementó cambios leves como agregar nuevas funcionalidades en el mismo código en donde se respeta mejor una nomenclatura consistente, por ejemplo: Listar_producto.php (mayúscula inicial), aunque no es completamente uniforme (mezcla mayúsculas y minúsculas como editar_producto.php vs Listar_producto.php).

Se mantiene la misma estructura de carpetas (Config, estilos, Layout, Modelos, Vistas), lo que indica una mejora incremental sin romper la arquitectura original.

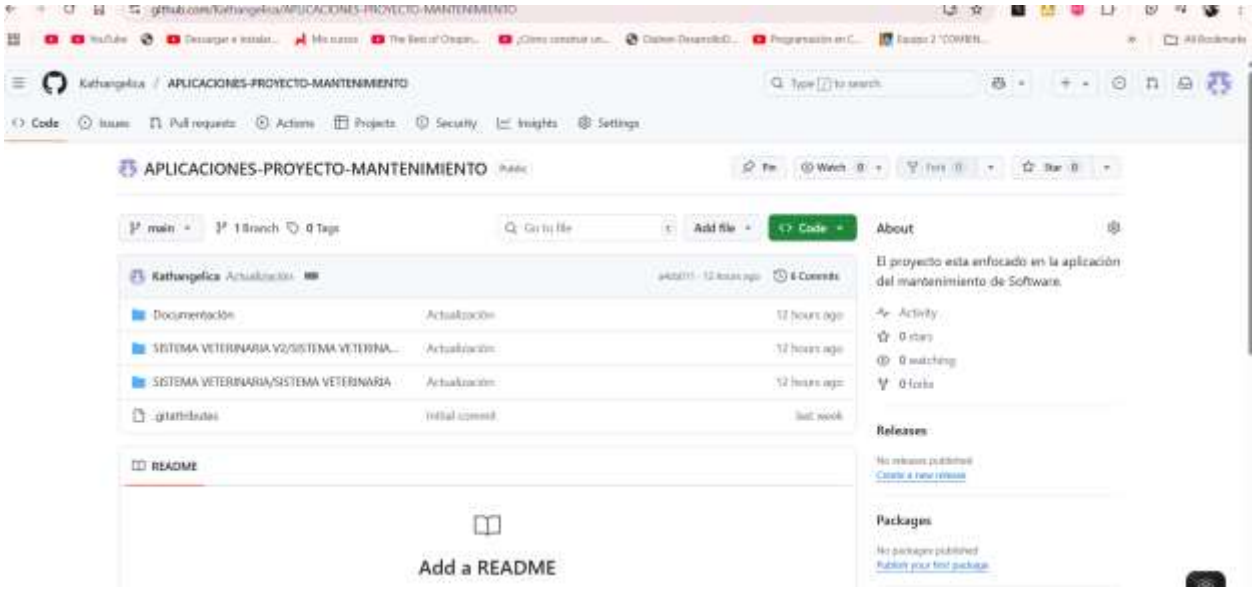
Integración:

→ Fusión de cambios con el código principal.



Sin eliminar archivos anteriores
Ningún archivo de la primera versión ha sido eliminado en la segunda versión. Todos los archivos existentes siguen presentes en fusión de cambios con el código principal , lo cual muestra que los cambios han sido **adaptados**.

→ Resolución de conflictos de versión.



2. CONFIGURACIONES CRITICAS.

Documentacion:

→ Actualización de especificaciones técnicas.

Entorno de Desarrollo:

Elemento	Descripción
Servidor	Localhost ejecutado con servidor local de PHP (XAMPP).
IDE	Visual Studio Code (VS Code) con extensiones para PHP, HTML, y CSS
Lenguaje/Framework	PHP 7.x/8.x con arquitectura modular (Vistas, Modelos, Config, Layout)
Base de Datos	MySQL gestionada con phpMyAdmin (base de datos: sistema_veterinaria)
Servicios	Navegador web (Chrome, Firefox), phpMyAdmin para gestión de datos
Observaciones	Entorno completo para desarrollo local, permite pruebas y ajustes rápidos

Entorno de pruebas:

Elemento	Descripción
Servidor	Servidor local con Apache
Base de Datos	MySQL (sistema_veterinaria_test)
Servicios	Pruebas de inserción, edición, eliminación y facturación
Seguridad	Validaciones básicas en formularios

Observaciones	Permite verificar funcionalidades como gestión de productos y facturas sin afectar datos reales

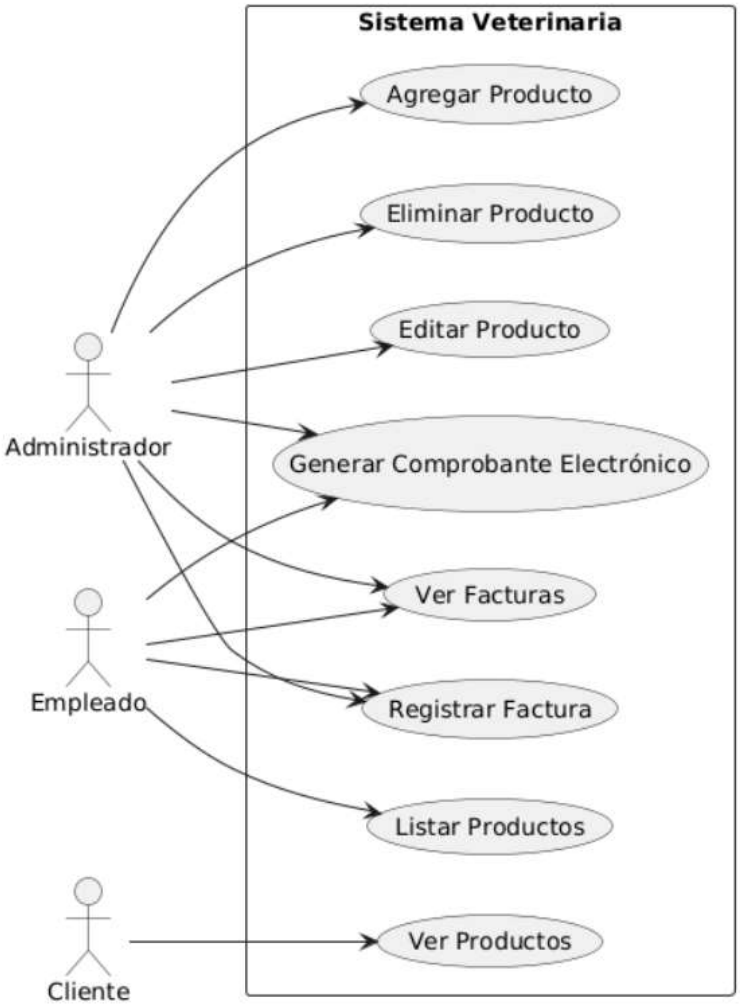
Entorno de producción:

Elemento	Descripción
Servidor	Hosting compartido o VPS con Apache
Proxy Reverso	Apache
Base de Datos	MySQL PhPyAdmin gestionada remotamente
Seguridad	HTTPS con SSL, variables sensibles en archivo .env o fuera del root público
Servicios	Monitoreo básico, escalabilidad,control de errores en producción
Observaciones	Preparado para clientes o usuarios reales, con control de acceso y mantenimiento regular.

3. DIAGRAMAS DE FLUJO

6.1.1 Reconstrucción de la Arquitectura

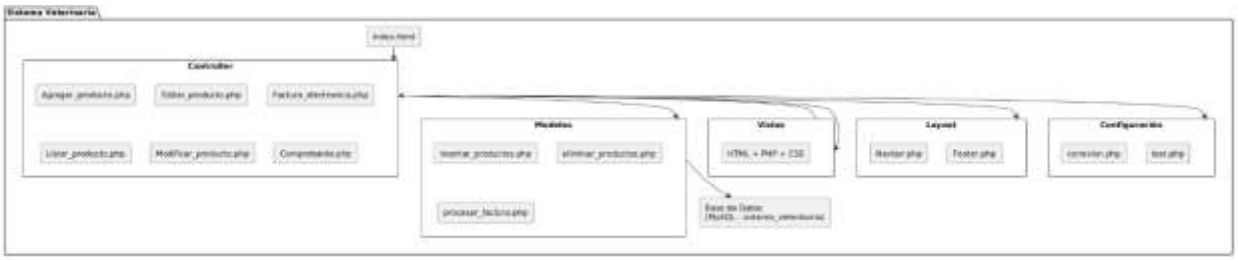
- **Diagrama de Casos de Uso :**



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

Su arquitectura MVC, con separación de vistas, modelos y componentes reutilizables, facilita el mantenimiento y la escalabilidad inicial. El uso de PHP y MySQL lo hace accesible y fácil de desplegar en entornos comunes.



- **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 general del sistema veterinario, el cual cubre los principales módulos que se han implementado:

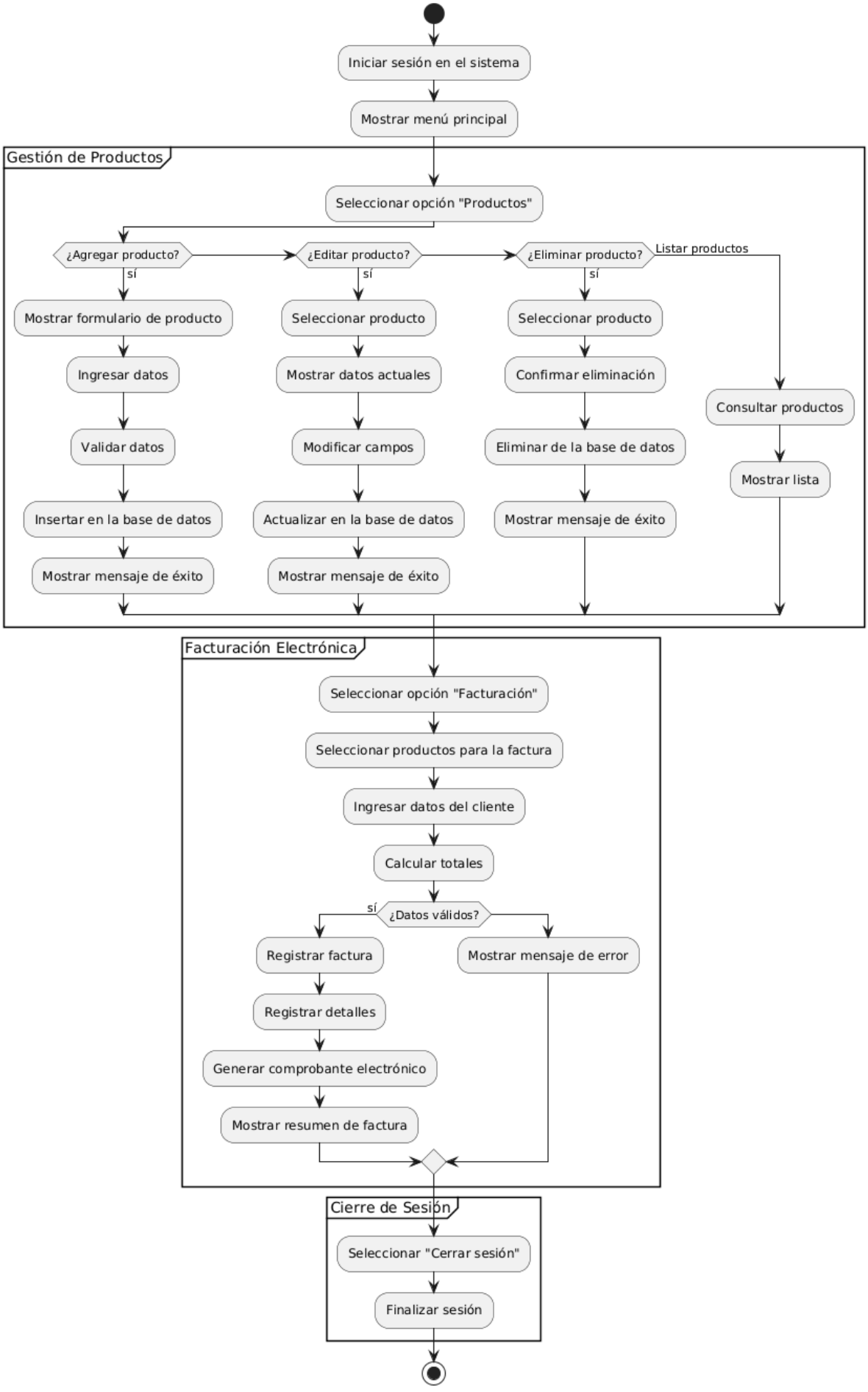
- Gestión de productos (agregar, editar, eliminar, listar)
- Facturación electrónica
- Visualización general del sistema

→ Flujo desde el inicio de sesión hasta el cierre de sesión.

Subprocesos para:

- Agregar, editar, eliminar y listar productos.
- Crear una factura con validación.
- Generar el comprobante electrónico.

Flujo condicional según la acción que el usuario realiza.

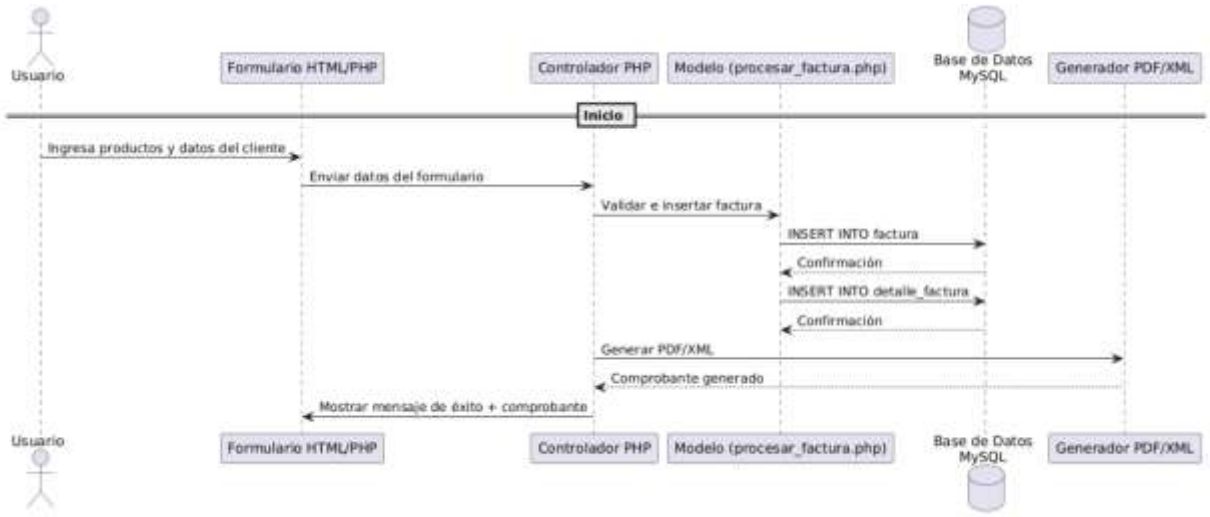

Iniciar sesión en el sistema
Mostrar menú principal
Gestión de Productos
Seleccionar opción "Productos"
¿Agregar producto?
sí
Mostrar formulario de producto
Ingresar datos
Validar datos
Insertar en la base de datos
Mostrar mensaje de éxito
¿Editar producto?
sí
Seleccionar producto
Mostrar datos actuales
Modificar campos
Actualizar en la base de datos
Mostrar mensaje de éxito
¿Eliminar producto?
sí
Seleccionar producto
Confirmar eliminación
Eliminar de la base de datos
Mostrar mensaje de éxito
Listar productos
Consultar productos
Mostrar lista
Facturación Electrónica
Seleccionar opción "Facturación"
Seleccionar productos para la factura
Ingresar datos del cliente
Calcular totales
¿Datos válidos?
sí
Registrar factura
Registrar detalles
Generar comprobante electrónico
Mostrar resumen de factura
Mostrar mensaje de error
Cierre de Sesión
Seleccionar "Cerrar sesión"
Finalizar sesión

- **Diagrama de Secuencias**

El diagrama de secuencia ilustra la interacción temporal entre el usuario, la interfaz de el sistema de ventas, facturación y la base de datos con la ejecución de una operación. Describe cómo el usuario introduce los datos, cómo se envía la solicitud , cómo se procesa la operación y finalmente cómo se devuelve el resultado para ser mostrado en la pantalla. Este diagrama ayuda a visualizar el flujo de mensajes y la comunicación entre componentes en tiempo real. Modela la interacción entre objetos o componentes del sistema en el tiempo.

Escenario: Generar Factura Electrónica

- El usuario (Empleado o Administrador)
- La interfaz (formulario)
- El controlador PHP
- El modelo (operaciones con la base de datos)
- La base de datos MySQL
- La generación del comprobante



Este flujo refleja cómo el sistema responde cuando el usuario genera una factura:

1. El usuario completa el formulario.
2. Se envían los datos al **Controlador PHP**.
3. El controlador llama al **modelo** que inserta los datos en la base de datos.
4. Se generan los **detalles de la factura**.
5. Se invoca un módulo para **crear un PDF** del comprobante.
6. El resultado se muestra al usuario.

- **Diagrama de la BASE DE DATOS**

Tabla: factura

Esta tabla almacena la información general de cada factura emitida. Incluye los datos del cliente y totales monetarios.

- **Campos principales:**
  - id_factura: Identificador único de la factura (clave primaria).
  - codigo_factura: Código único de la factura.
  - nombre_cliente, cedula, email, direccion: Datos personales del cliente.
  - fecha: Fecha y hora de emisión.

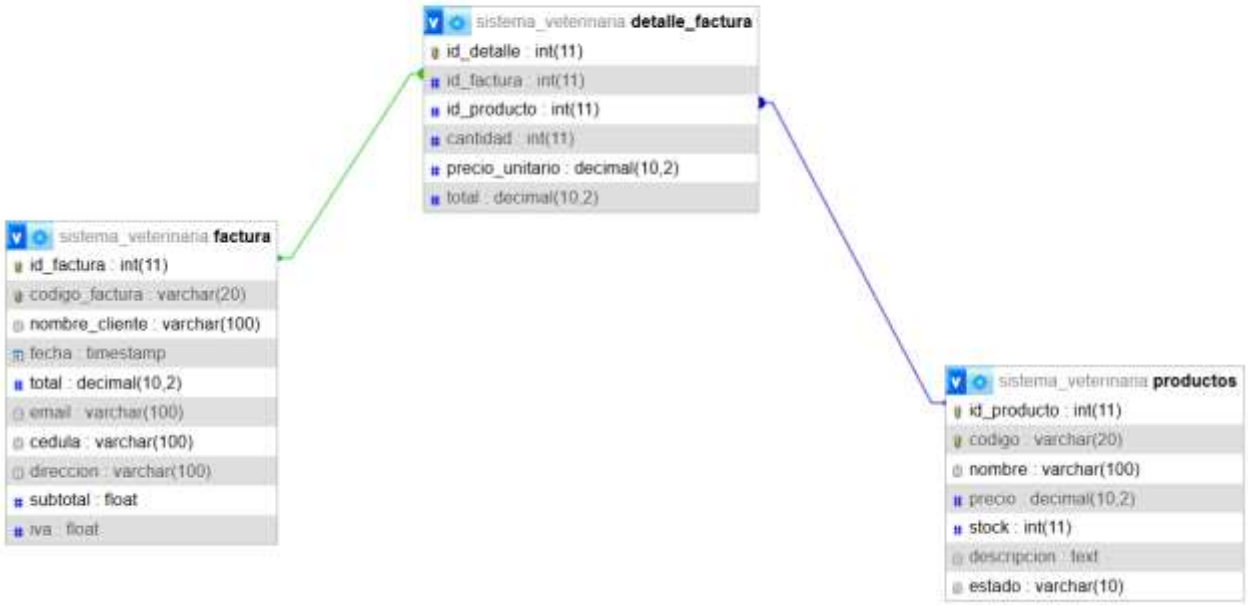
- subtotal, iva, total: Cálculos financieros de la factura según requisitos tributarios del ecuador.

Tabla: detalle_factura
Representa los ítems o productos individuales dentro de una factura específica.

- **Campos principales:**
  id_detalle: Clave primaria del detalle.
  id_factura: Clave foránea que relaciona con la tabla factura.
  id_producto: Clave foránea que referencia a productos.
  cantidad, precio_unitario, total: Información específica del producto facturado.
- **Relaciones:**
  - Muchas filas de detalle_factura pueden estar asociadas a una sola factura (relación 1 a muchos).
  - Cada ítem de detalle está asociado a un producto registrado en productos.

Tabla: productos
Almacena el catálogo de productos disponibles en la veterinaria.

- **Campos principales:**
  - id_producto: Identificador único del producto (clave primaria).
  - codigo, nombre: Identificadores del producto.
  - precio, stock: Información financiera y de inventario.
  - descripcion, estado: Información adicional sobre el producto.



4. GLOSARIO DE TÉRMINOS

Término	Definición
MVC (Modelo-Vista-Controlador)	Patrón de arquitectura de software que separa la lógica del negocio (Modelo), la interfaz de usuario (Vista) y el flujo de control (Controlador).

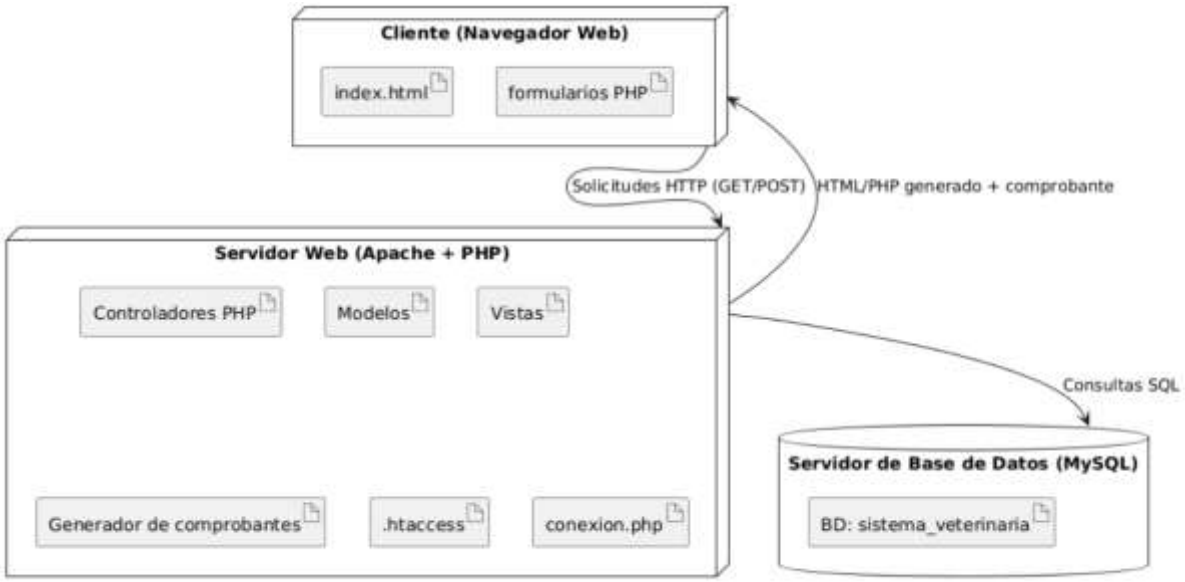
Controlador	Componente encargado de recibir solicitudes del usuario, procesarlas y devolver una respuesta adecuada, conectando la vista con el modelo.
Modelo	Componente encargado de gestionar la lógica de negocio y el acceso a la base de datos.
Vista	Parte del sistema que presenta los datos al usuario, generalmente usando HTML, PHP y CSS.
Formulario	Elemento de la interfaz de usuario que permite capturar datos para el sistema, como productos o datos del cliente.
Factura Electrónica	Documento digital generado por el sistema que representa una transacción de venta de productos o servicios.
Detalle de Factura	Información específica de cada producto incluido en una factura, como cantidad, precio unitario y subtotal.
PHP	Lenguaje de programación del lado del servidor utilizado en el desarrollo del sistema.
MySQL	Sistema de gestión de bases de datos relacional usado para almacenar la información del sistema.
phpMyAdmin	Herramienta web para administrar bases de datos MySQL de forma visual.
Servidor Web (Apache)	Software que entrega páginas web al navegador del usuario, ejecutando archivos PHP en el servidor.
Base de Datos	Conjunto estructurado de datos almacenados electrónicamente, como productos, facturas y usuarios.
Comprobante Electrónico	Archivo PDF o XML generado automáticamente como evidencia de la venta realizada.
Validación	Proceso para comprobar que los datos ingresados por el usuario son correctos antes de ser almacenados.
CRUD	Acrónimo de Crear, Leer, Actualizar y Eliminar: operaciones básicas sobre los datos.
Layout	Plantilla base que incluye elementos comunes de todas las páginas, como el encabezado, menú o pie de página.
conexion.php	Archivo PHP que contiene las credenciales y configuración para conectarse a la base de datos.
PlantUML	Herramienta que permite generar diagramas de forma sencilla utilizando código de texto.
Diagrama de Casos de Uso	Representación gráfica de las funcionalidades del sistema desde la perspectiva del usuario.
Diagrama de Secuencia	Muestra la interacción temporal entre distintos componentes del sistema.
Diagrama de Actividades	Representación de flujo de procesos o tareas en el sistema.
Diagrama de Despliegue	Muestra cómo los componentes del sistema se distribuyen físicamente en los distintos nodos (cliente/servidor).
index.html	Archivo principal o inicial que el navegador carga al entrar al sistema.

5. PROCEDIMIENTO DE DESPLIEGUE Y RECUPERACIÓN

→ **Despliegue**

Este diagrama de despliegue representa cómo está distribuida la aplicación.

- **Nodos físicos/lógicos:**
  - Cliente: usuario accede vía navegador.
  - Servidor: donde están alojados los archivos PHP (controladores, modelos, vistas, configuración).
  - Base de datos: motor MySQL con la base sistema_veterinaria.
- **Artefactos:** son los componentes del software desplegados en cada nodo.
- **Conexiones:** indican la comunicación entre cliente-servidor y servidor-base de datos.



Procedimiento de Recuperación ante Desastre

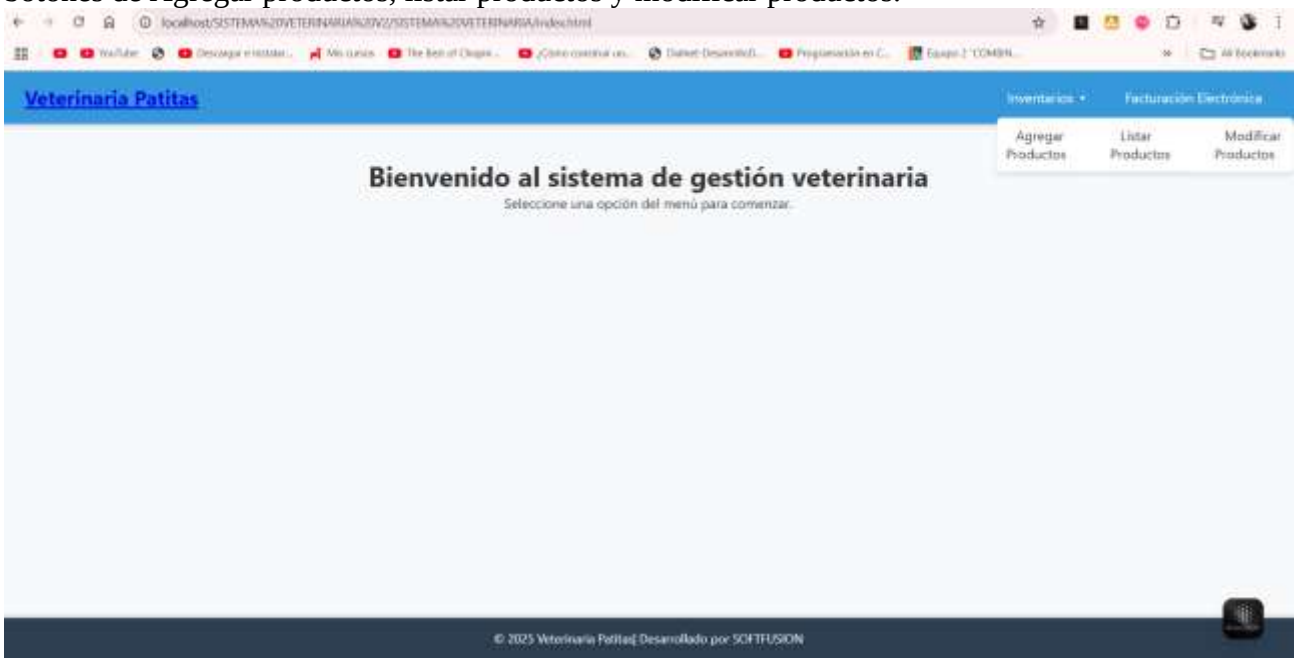
Paso	Descripción
Respaldo del código fuente	Mantener el sistema versionado en un repositorio remoto como **GitHub**. Realizar **copias periódicas del código** fuente (.php, .css, .html) y archivos de configuración.
Respaldo de la base de datos	Programar respaldos automáticos o manuales usando herramientas como mysqldump. Guardar copias del archivo .sql en la nube o servidor externo.
Restauración del entorno	Clonar el repositorio o copiar los archivos a un nuevo equipo o servidor. Configurar el entorno PHP (instalar Apache/Nginx, PHP y MySQL). Importar la base de datos desde el respaldo.
Configuración de entorno	Configurar archivos como conexion.php con las nuevas credenciales. Verificar puertos, rutas, permisos de carpetas y el correcto enlace de estilos CSS.
Verificación funcional	Ejecutar el sistema en el navegador. Probar todas las funcionalidades: agregar, editar, eliminar y listar productos, facturación y cierre de sesión. Validar que la base de datos se conecte y procese bien los datos.
Prevención futura	Implementar **monitoreo básico** (logs de errores y acciones del sistema). Establecer políticas de respaldo automático y mantener la **documentación técnica** del sistema siempre actualizada.

Resumen de cambios:

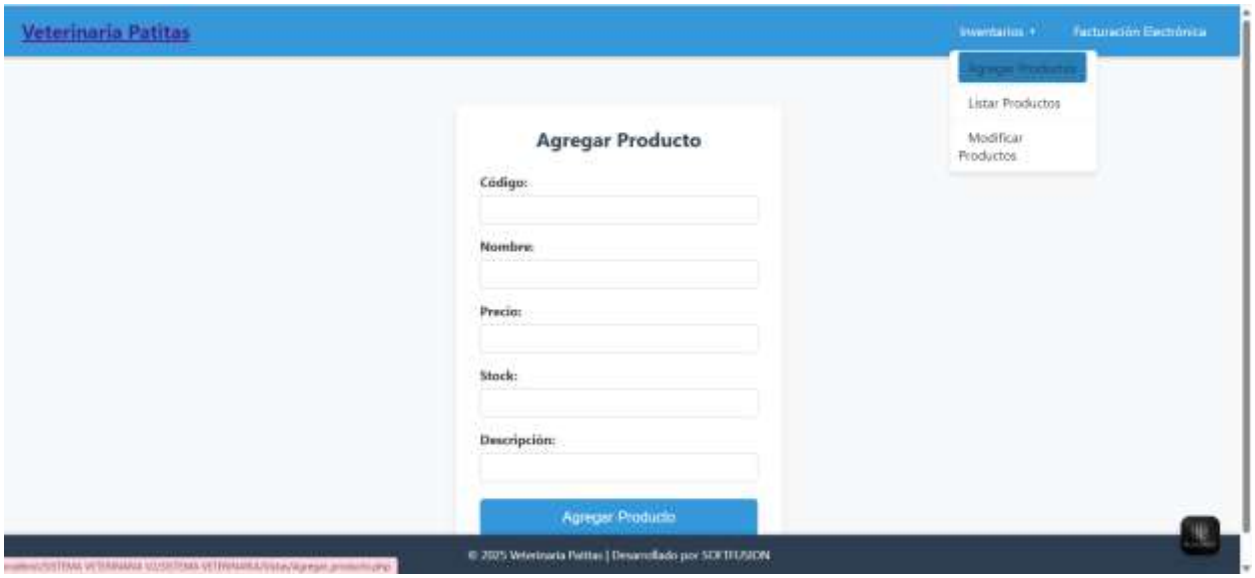
Tipo de Cambio	Descripción
Nuevos estilos CSS	Se añadieron estilos específicos para nuevas funcionalidades: editar, listar, modificar productos
Nuevos modelos	Se añadió eliminar_productos.php para gestión de productos en el backend
Nuevas vistas	Se añadieron vistas nuevas para editar, listar y modificar productos
Mejor organización	Mayor modularidad y separación por responsabilidades en los archivos

Interfaz del Sistema de Facturación aplicado el Mantenimiento Adaptativo.

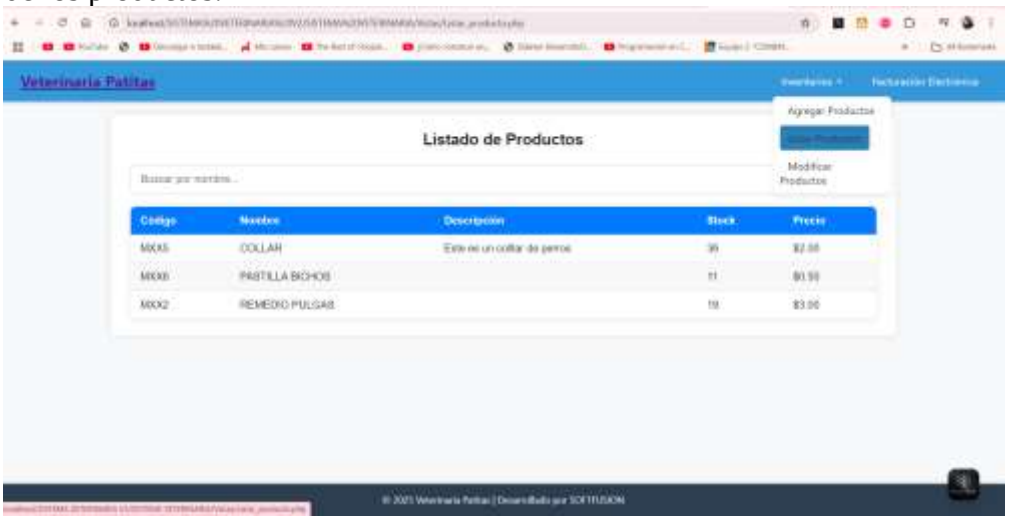
Visualización de la interfaz de la página principal del sistema de facturación con los nuevos botones de Agregar productos, listar productos y modificar productos.



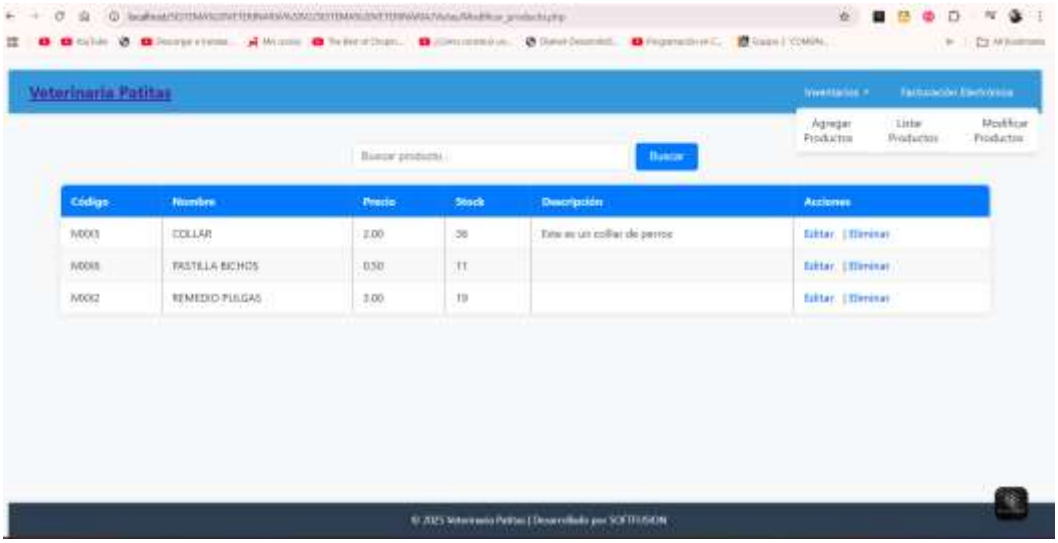
Se puede observar la vista de agregar producto.



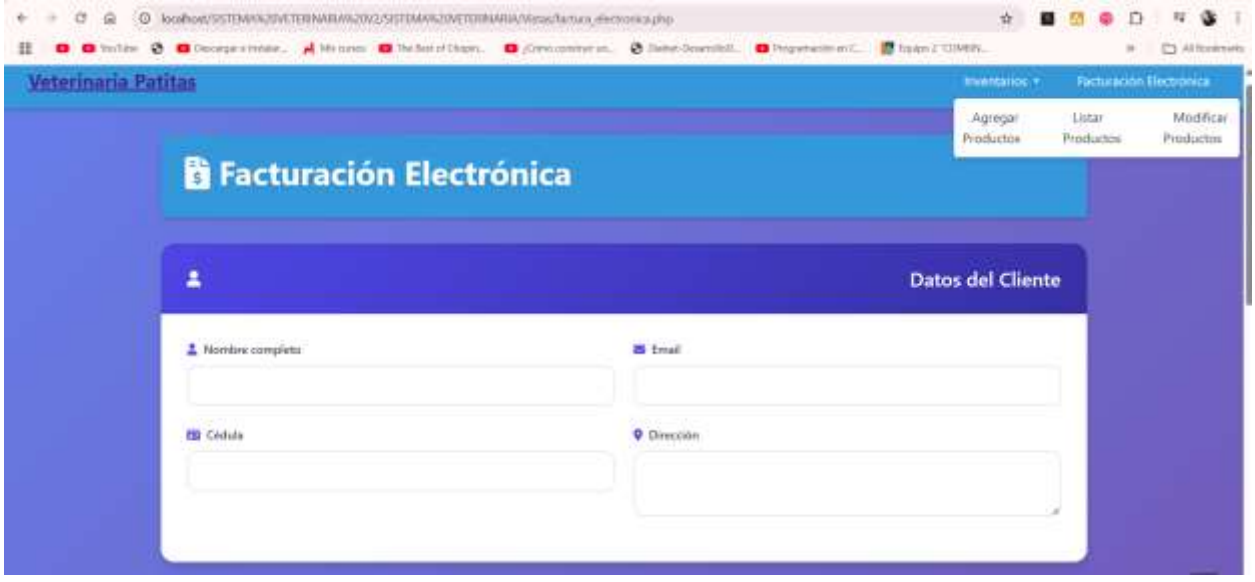
A continuación, tenemos la vista de la pagina de Listar producto donde se observa un detalle de el listado de los productos.



Se obtiene la opción de Modificar productos donde se podrán editar y eliminar los productos anteriormente agregados.

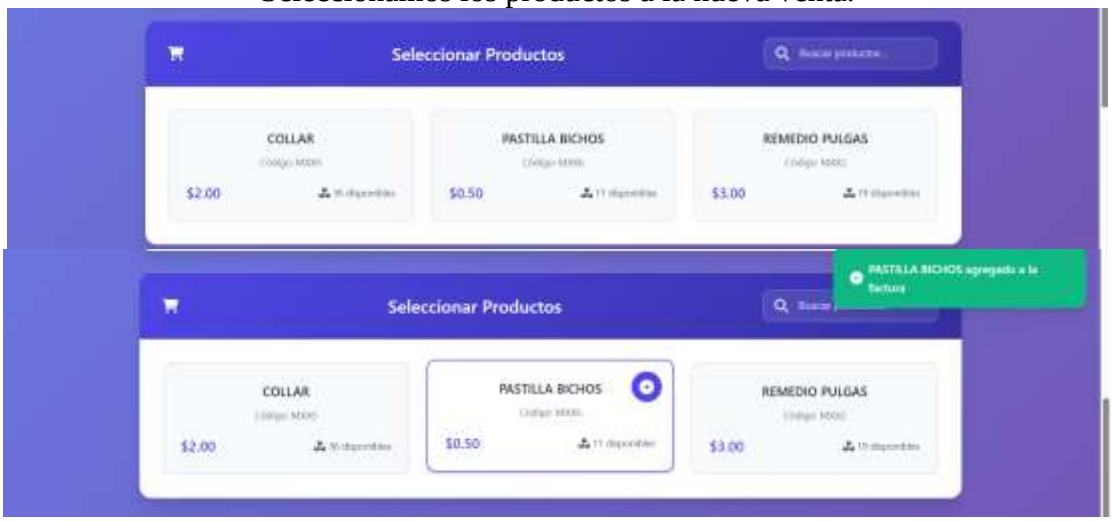


Ingresamos a la vista de Facturación Electrónica iniciando con los datos del cliente.

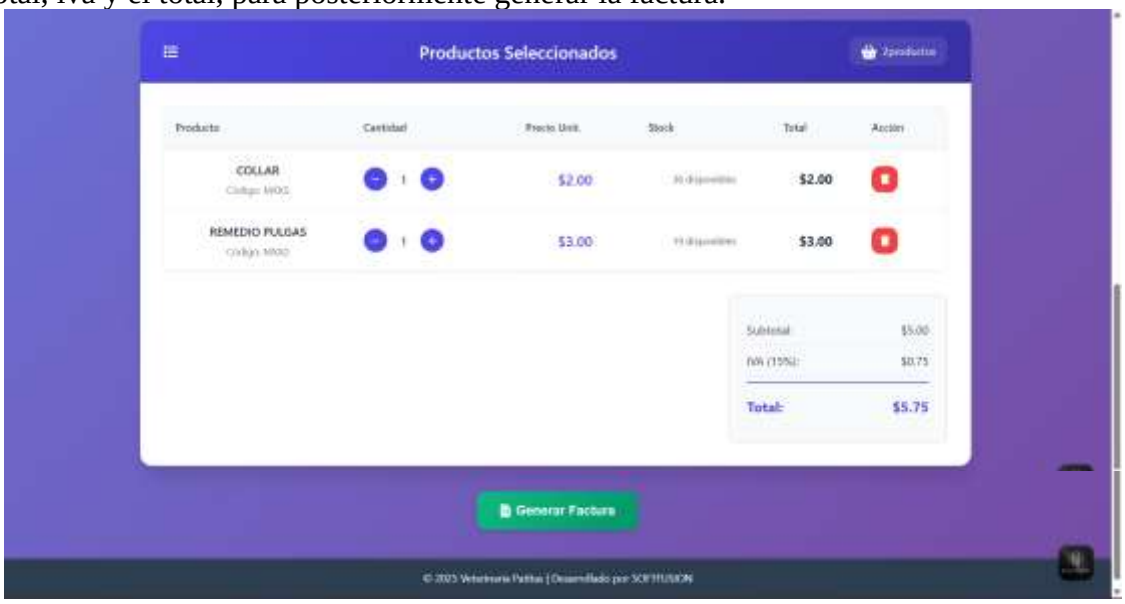


Ingresamos los datos del cliente.

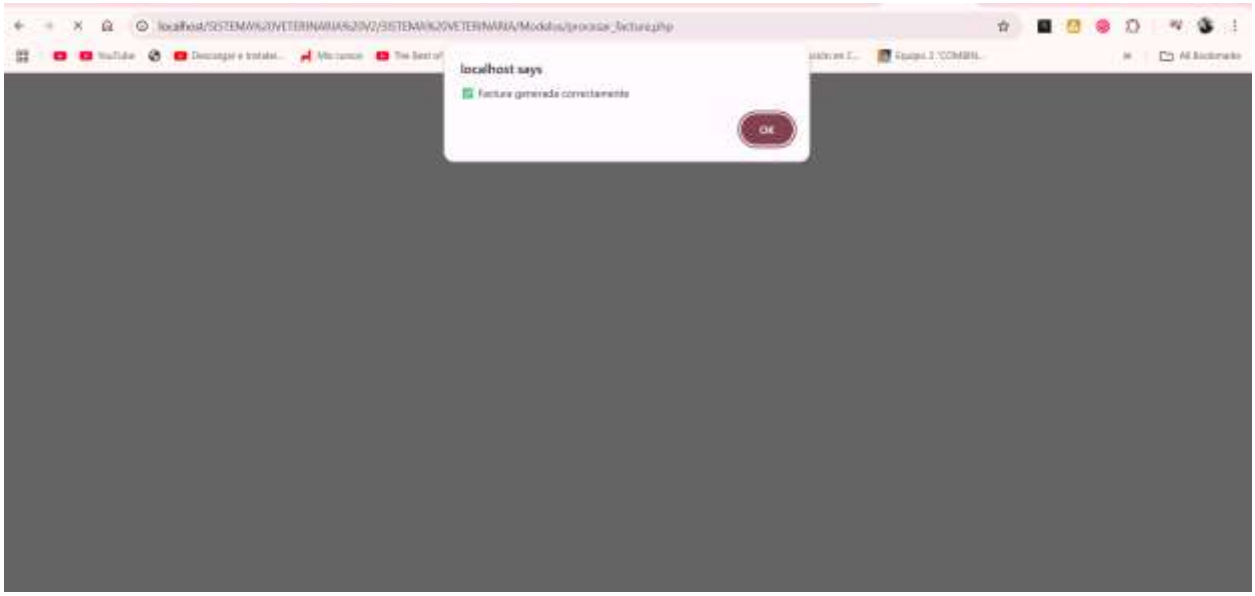
Seleccionamos los productos a la nueva venta.



Visualizamos los productos agregados a la nueva venta y se puede visualizar la cantidad, el subtotal, iva y el total, para posteriormente generar la factura.



Seleccionamos en generar factura y visualizamos que se a generado correctamente.



Finalmente visualizamos y verificamos la factura generada correctamente con todos los datos específicos y principalmente con la adaptación de la facturación con los requisitos establecidos por las autoridades tributarias del Ecuador.

7. CONCLUSIONES Y RECOMENDACIONES:

Conclusiones:

-En conclusión, se fortalecieron habilidades técnicas y de análisis aplicadas en la ejecución del proceso del mantenimiento adaptativo aplicado al sistema de facturación. El mantenimiento adaptativo permitió extender la vida útil del sistema
A lo largo del desarrollo del proyecto, se demostró que aplicar mantenimiento adaptativo es fundamental para garantizar que el sistema pueda seguir funcionando correctamente ante cambios en el entorno tecnológico o normativo, sin necesidad de ser rediseñado completamente.

-Se construyó documentación técnica clara y útil, como en los diagramas generados (casos de uso, actividades, despliegue, etc.) junto con el glosario de términos y estructura de entornos, brindan una visión integral del mantenimiento aplicado y los cambios importantes realizados del sistema y que facilitara su comprensión para otros desarrolladores o usuarios.

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.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.

Mantener actualizada la documentación y planificar revisiones periódicas del código y la arquitectura, garantizando que el sistema evolucione de manera controlada y sostenible.

8. BIBLIOGRAFÍA:

- Eilam, E. (2005). *Reversing: Secrets of reverse engineering*. Wiley. https://doi.org/10.1002/0471756978
- 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")
- Chikofsky, E. J. y Cross, J. H. (1990). Ingeniería inversa y recuperación de diseño: Una taxonomía. *IEEE Software*, *7*(1), 13-17. https://doi.org/10.1109/52.43044
- Flask Documentation. Flask Web Development. https://flask.palletsprojects.com/en/latest/ 5. Fowler, Martin. UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition. Addison-Wesley, 200