Facultad de Ingeniería
Escuela de Ingeniería en Computación

Asignatura: Desarrollo de Aplic. Web con Soft. Interpret. en el Servidor.

Ciclo / Grupo: DSS404 G01T Ciclo III

Investigación aplicada I
DSS

Facilitador: Kevin Jiménez.

Elaborado por:

Integrante	Carnet
Paola Matilde Orellana Castillo	OC250609
Aleshka Priscila Gámez Navidad	GN252409
Gabriela Giselle Vásquez Rodríguez	VR250295
Francisco Miguel Serrano Orellana	SO252952
Mariana Maytee López Gómez	LG252169

Link al repositorio de Github:
https://github.com/itspao529/Sistema-Productos

Fecha de entrega: 23 de febrero 2026

Índice

Definición de Server-Side

La programación del lado del servidor (Server-Side) es un modelo de desarrollo web en el cual el procesamiento principal de la aplicación se ejecuta en el servidor antes de enviar una respuesta al usuario. Esto significa que cuando el cliente realiza una solicitud a través del navegador, el servidor recibe la petición, ejecuta el código correspondiente (por ejemplo, en PHP), procesa la información y genera dinámicamente el contenido que será enviado al navegador.

En el **Sistema de Gestión de Productos desarrollado**, el lenguaje PHP se ejecuta en el servidor para validar los datos ingresados por el usuario, gestionar los productos y actualizar la información almacenada en la variable de sesión ($_SESSION). El usuario no tiene acceso directo al código del servidor, ya que solo recibe el resultado final en forma de página web.

Rol del Servidor

El servidor cumple un papel fundamental dentro de una aplicación web, ya que es el encargado de procesar las solicitudes realizadas por los usuarios y garantizar el correcto funcionamiento del sistema.

En el **Sistema de Gestión de Productos**, el servidor realiza las siguientes funciones:

- Recibir los datos enviados desde los formularios.
- Validar que los campos no estén vacíos y que cumplan las condiciones establecidas..
- Verificar que no existan identificadores (ID) repetidos.
- Validar que los valores numéricos (precio y stock) no sean negativos.
- Guardar la información en la variable de sesión ($_SESSION).
- Permitir la edición y eliminación de productos.
- Registrar ventas de productos existentes.
- Actualizar automáticamente el stock después de cada venta.

El servidor actúa como el núcleo lógico del sistema, asegurando que los datos sean procesados correctamente antes de mostrarse al usuario.

Flujo de Petición Web

El flujo de petición web describe el proceso que ocurre desde que el usuario realiza una acción hasta que recibe una respuesta del sistema.

En el **Sistema de Gestión de Productos**, el proceso funciona de la siguiente manera:

1. El usuario completa un formulario en el navegador (agregar, editar o vender un producto).
2. El navegador envía una solicitud HTTP al servidor.
3. El servidor recibe la petición y ejecuta el archivo PHP correspondiente.
4. Se procesan y validan los datos.
5. Se actualiza la información almacenada en la variable de sesión ($_SESSION).
6. El servidor genera una respuesta en formato HTML.
7. El navegador muestra la página actualizada al usuario.

Este flujo se repite cada vez que se crea, edita, elimina o registra una venta de un producto, garantizando una interacción dinámica entre el cliente y el servidor.

El diagrama muestra el proceso que sigue una solicitud desde que el usuario interactúa con el sistema hasta que el servidor procesa la información y devuelve una respuesta al navegador.

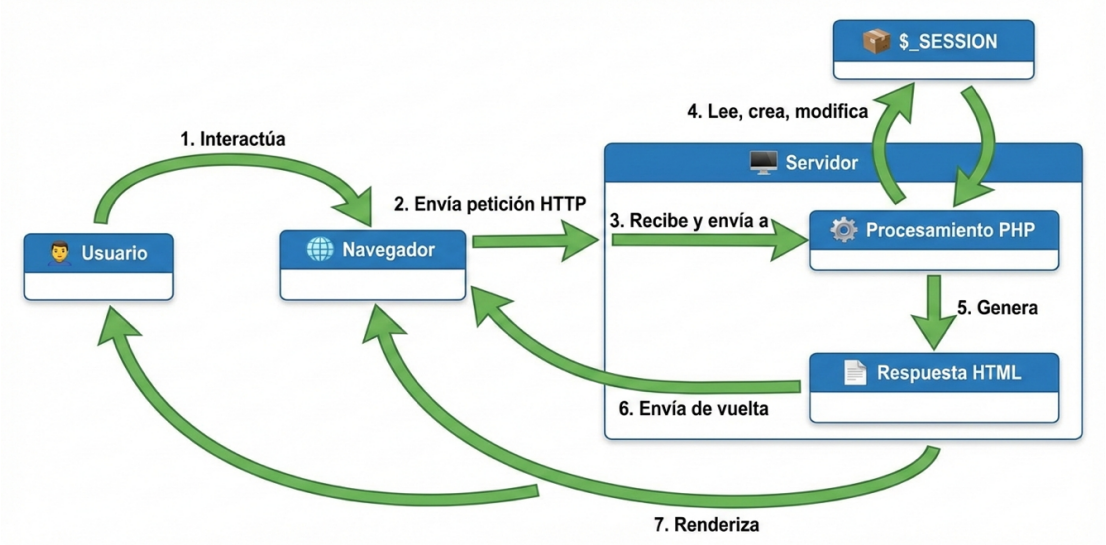
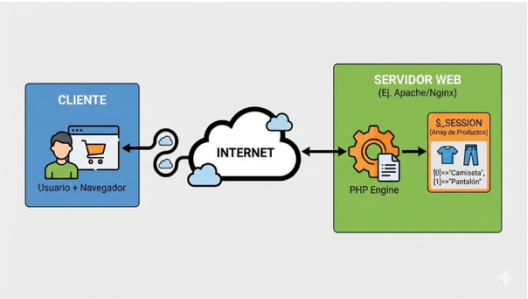


Ilustración 1.Flojo de peticion web del sistema de gestion de productos

Diferencias entre client-side y Server-side

Del lado del cliente se refiere a los procesos que se llevan a cabo en el dispositivo del usuario, generalmente en el navegador web del usuario. Estos procesos se ejecutan después de que el sitio web o la aplicación web se han entregado al dispositivo del usuario, y pueden incluir tareas como renderizar y mostrar una página web, manejar interacciones del usuario o ejecutar código JavaScript.



El lado del servidor se refiere a los procesos que se llevan a cabo en el servidor web, donde se aloja el sitio web o la aplicación web. Estos procesos suelen ser ejecutados por el servidor antes de que el sitio web o la aplicación web se entregue al dispositivo del usuario, y pueden incluir tareas como recuperar y manipular información almacenada en sesión o base de datos, renderizar una página web o manejar la entrada del usuario.

- Client-Side (lado del cliente)

- Se ejecuta en el **navegador del usuario**.
- Depende del dispositivo del usuario.
- Ejemplo de tecnologías:
  - JavaScript
  - HTML
  - CSS

- Server-Side (lado del servidor)

- Se ejecuta en el **servidor web**.
- No depende del dispositivo del usuario.
- Ejemplo de tecnologías:
  - PHP
  - Python
  - Java
  - Node.js

Función principal

Client-Side	Server-Side
Interfaz de usuario	Procesamiento de datos
Validaciones rápidas	Acceso a bases de datos
Animaciones y efectos visuales	Autenticación de usuarios
Experiencia interactiva	Lógica de negocio

Ventajas y desventajas de Client-Side y Server-Side

Tipo	Ventajas	Desventajas
Client-Side	Respuesta rápida en la interfaz Menor carga al servidor Mejor experiencia interactiva Validaciones inmediatas	Menor seguridad (código visible) Depende del navegador del usuario No ideal para datos sensibles Puede afectar rendimiento en equipos lentos
Server-Side	Mayor seguridad Mejor manejo de bases de datos Control total de la lógica del sistema Ideal para datos sensibles	Puede ser más lento Mayor carga en el servidor Requiere conexión constante

Php

PHP es un lenguaje de programación interpretado y orientado principalmente al desarrollo web del lado del servidor. Es uno de lenguajes más utilizados en aplicaciones web dinámicas.

Node.js

Node.js es un entorno de ejecución que permite ejecutar JavaScript en el servidor.

Python

Python es un lenguaje de programación Versátil, conocido por su sintaxis clara y legible. En el desarrollo web se utiliza principalmente mediante frameworks como Django y Flask.

- Django

  Framework de alto nivel.
  Incluye ORM, sistema de autenticación y panel administrativo.
  Sigue el patrón MTV (Modelo-Template-Vista).

- *Flask*

  Framework ligero y minimalista.
  Mayor flexibilidad y control del desarrollador.
  Ideal para APIs y proyectos pequeños.

Java, Ruby, ASP.NET, NET CORE

Java

Java es un lenguaje de programación de propósito general, orientado a objetos y fuertemente tipado. En el desarrollo del lado del servidor, emplea tecnologías como JavaServer Pages (JSP) y Servlets para procesar solicitudes y generar contenido web dinámico. Su ejecución sobre la Máquina Virtual de Java (JVM) le otorga portabilidad multiplataforma. Es reconocido por su robustez, seguridad y excelente rendimiento en arquitecturas empresariales de gran escala, aunque presenta una curva de aprendizaje pronunciada.

Ruby

Ruby es un lenguaje de programación interpretado y dinámico, enfocado en la simplicidad y la productividad del desarrollador. En el entorno de servidores, destaca mediante el framework Ruby on Rails, el cual implementa el patrón de arquitectura Modelo-Vista-Controlador (MVC). Este framework se basa en el principio de "Convención sobre Configuración", optimizando los tiempos de desarrollo, lo que lo hace ideal para la creación rápida de prototipos y aplicaciones en entornos de startups.

ASP.NET

ASP.NET es un framework de desarrollo web consolidado, creado por Microsoft para la construcción de aplicaciones y servicios dinámicos mediante el lenguaje C#. Se caracteriza por su profunda integración con el ecosistema de productos de Microsoft, proporcionando altos niveles de seguridad y soporte corporativo. Tradicionalmente, ha presentado una fuerte dependencia hacia el sistema operativo Windows y el servidor de Internet Information Services (IIS).

.NET Core

.NET Core representa la evolución moderna, multiplataforma y de código abierto del framework ASP.NET tradicional. Ha sido rediseñado para ofrecer un rendimiento superior, una arquitectura modular y compatibilidad nativa con múltiples sistemas operativos. Es una tecnología altamente eficiente, optimizada para el desarrollo de aplicaciones web contemporáneas, despliegues basados en la nube y arquitecturas de microservicios.

Cuadro comparativo de PHP, Node.js, Python, Java, Ruby, ASP.NET y .NET Core

Característica	PHP	Node.js	Python (Django/Flask)	Java (JSP/Servlets)	Ruby (RoR)	ASP.NET (C#)	.NET Core
Tipo	Lenguaje interpretado	Entorno para ejecutar JavaScript	Lenguaje multiparadigma	Lenguaje compilado / Orientado a objetos	Lenguaje interpretado dinámico	Framework web de Microsoft	Framework web multiplataforma
Modelo	Sincrónico	Asincrónico (no bloqueante)	Principalmente sincrónico	Multihilo (Multithreading) robusto	Sincrónico (Convención sobre configuración)	Sincrónico y asincrónico	Asincrónico y modular
Frameworks	Laravel, CodeIgniter	Express.js	Django, Flask	Spring Boot, JSF	Ruby on Rails, Sinatra	ASP.NET MVC	ASP.NET Core
Rendimiento	Bueno para webs tradicionales	Muy alto en tiempo real	Bueno y estable	Muy alto (sistemas a gran escala)	Medio (optimizado para agilidad)	Alto	Excepcionalmente alto
Facilidad	Fácil	Media	Fácil	Difícil	Fácil / Media	Media	Media / Difícil
Uso común	CRUD y páginas dinámicas	APIs y apps en tiempo real	Web compleja e IA	Sistemas bancarios y empresariales	Startups y prototipos rápidos	Intranets y corporativos	Microservicios y nube

Conclusión General

En conclusión, el desarrollo web del lado del servidor constituye un componente esencial en la creación de aplicaciones dinámicas, seguras y funcionales. A través del procesamiento de datos, validación de información y control de la lógica de negocio, el servidor garantiza la correcta interacción entre el usuario y el sistema.

El análisis del Sistema de Gestión de Productos permitió comprender cómo funciona el flujo de petición web y cómo el servidor actúa como eje central en la gestión de información utilizando una matriz almacenada en la variable de sesión ($_SESSION) sin el uso de base de datos.

Finalmente, el estudio de tecnologías como PHP, Node.js y Python demuestra que existen múltiples alternativas para implementar soluciones del lado del servidor, cada una con ventajas específicas según el tipo de proyecto. La correcta elección de la tecnología dependerá de los requerimientos funcionales, el rendimiento esperado y la complejidad del sistema a desarrollar.