



INGENIERÍA EN SISTEMAS DE INFORMACIÓN

MODALIDAD EN LÍNEA

Estudiante: Cristian Rojas

Trabajo Autónomo N 3

Fecha: 29 de JUNIO del 2025

TECNOLOGIA DE COMUNICACION DE INFORMACION EN RED 1-ECC-3B



Sistema de Gestión E-commerce

Cronograma de trabajo por semanas

| Semana | Tema trabajado | Actividad realizada |
|--------|--------------------------------------|--|
| 1 | Planeación y estructura inicial | Selección del sistema E-commerce. Creación del repositorio y definición de carpetas. |
| 2 | Clases y estructuras | Implementación de estructuras: Usuario, Producto, Pedido. |
| 3 | Interfaces y métodos | Creación de interfaz Pago y métodos concretos. |
| 4 | Encapsulamiento y control de acceso | Aplicación de atributos privados y getters/setters. |
| 5 | Manejo de errores | Validación de datos en servicios web. |
| 6 | Servicios web REST | Desarrollo de 8 endpoints funcionales con JSON. |
| 7 | Visualización del Futuro y README.md | Redacción del futuro del proyecto y documentación. |
| 8 | Capturas, video, presentación final | Capturas, grabación del video y presentación en Canva. |

◊ Introducción

Este proyecto consiste en el desarrollo de un sistema de gestión E-commerce como parte del trabajo integrador de la asignatura. Abarca desde la planificación, estructura de clases, manejo de errores, encapsulamiento, uso de interfaces y finalmente, la implementación de servicios web funcionales usando el lenguaje de programación Go.

◊ Objetivo del Proyecto

Desarrollar un sistema de comercio electrónico que permita gestionar usuarios, productos, pedidos y pagos, siguiendo principios de la programación orientada a objetos y utilizando servicios web en Go.

◊ Estructura del Proyecto

El proyecto está dividido en varios paquetes:

- **models**: contiene las estructuras de datos como Usuario, Producto, Pedido.
- **controllers**: donde se implementan las funcionalidades que responden a las peticiones.
- **services**: lógica de negocio e interfaces como Pago.
- **utils**: funciones de utilidad.
- **main.go**: punto de entrada donde se ejecuta el servidor y los servicios web.

◊ Clases e Interfaces

Se definieron las siguientes estructuras: Producto, Usuario y Pedido, todas encapsuladas con atributos privados y métodos públicos.

Se implementó la interfaz Pago con dos métodos concretos: PagoTarjeta y PagoEfectivo.

◊ Manejo de Errores

El sistema incluye validaciones básicas y retorno de mensajes controlados en los servicios web. Además, se utilizaron mapas JSON para manejar estados y respuestas claras al cliente.

◊ Servicios Web REST

Se desarrollaron **8 servicios web** funcionales en Go que responden en formato JSON:

1. **/crear-usuario**: Crea un usuario de prueba
2. **/listar-productos**: Lista productos disponibles
3. **/buscar-producto**: Muestra un producto específico
4. **/agregar-carrito**: Simula agregar un producto
5. **/realizar-pedido**: Simula un pedido con productos
6. **/ver-historial**: Muestra pedidos realizados
7. **/realizar-pago**: Simula el pago de un pedido
8. **/cancelar-pedido**: Cancela un pedido

◊ Visualización del Futuro del Proyecto

En el futuro, este sistema E-commerce se puede mejorar mediante la incorporación de tecnologías como **inteligencia artificial** para recomendaciones inteligentes, **blockchain** para pagos seguros y trazabilidad, **IoT** para gestión de stock automático y **diseño accesible** para todo tipo de usuario. Además, se puede escalar a una **API distribuida en microservicios** y ofrecer análisis predictivos sobre comportamiento de compra.

◊ Capturas de Pantalla de los Servicios Web

Servicio 1 – /crear-usuario:



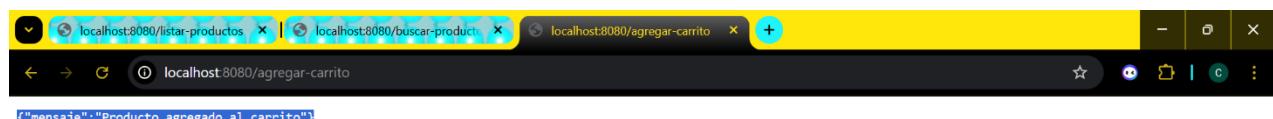
Servicio 2 – /listar-productos:



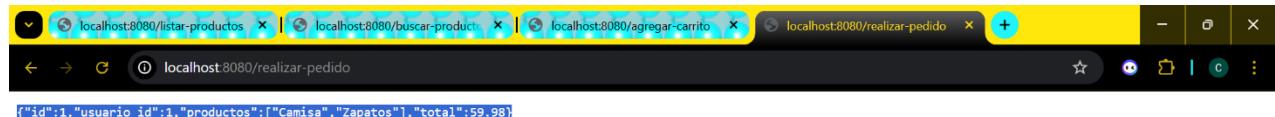
Servicio 3 – /buscar Producto:



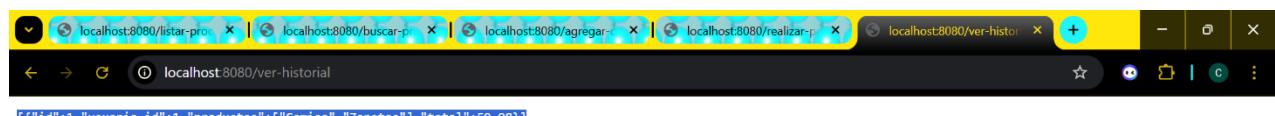
Servicio 4 – /agregar-carrito:



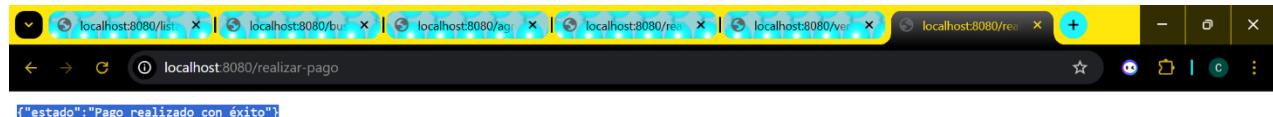
Servicio 5 – /realizar-pedido:



Servicio 6 – /ver-historial:



Servicio 7 – /realizar-pago:



Servicio 8 – /cancelar-pedido:



◊ Conclusión

El desarrollo del sistema E-commerce permitió aplicar los conocimientos adquiridos en programación orientada a objetos, estructuras de datos, interfaces y manejo de errores. A través de la implementación en Go, se logró construir una base sólida para un sistema funcional y escalable. Este proyecto fortaleció el entendimiento de servicios web y la capacidad de integrar múltiples componentes de software de forma estructurada y clara.

Referencias

- Google. (n.d.). *Golang Documentation*. <https://golang.org/doc/>
- Mozilla Developer Network. (n.d.). *JSON – JavaScript Object Notation*.
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON
- REST API Tutorial. (n.d.). *What is REST?*. <https://restfulapi.net>
- Microsoft. (n.d.). *Visual Studio Code*. <https://code.visualstudio.com>
- GitHub Docs. (n.d.). *Working with repositories*.
<https://docs.github.com/en/repositories>