Actividad |1| Instalación XCode/Programa 1.

Desarrollo de Aplicaciones Móviles IV.

Ingeniería en Desarrollo de Software.

TUTOR: Marco Alonso Rodriguez Tapia.

ALUMNO: Ramón Ernesto Valdez Felix.

FECHA: 15/11/2025.

Introducción.

En esta primera actividad de la materia Desarrollo de Aplicaciones Móviles IV, nos enfocamos en la creación de una aplicación para la boutique Norma que busca modernizar su negocio y expandir su alcance a través de una tienda de ropa en línea. Este proyecto implica el desarrollo de una aplicación de e-commerce robusta y amigable, utilizando el lenguaje de programación Swift, ideal para la creación de aplicaciones nativas en el ecosistema de Apple. El objetivo principal es ofrecer una experiencia de compra fluida. La aplicación deberá mostrar un catálogo inicial con al menos cuatro artículos, incluyendo su nombre, precio, y el stock disponible. Además, se implementará un menú interactivo que permita al cliente navegar, seleccionar un producto específico y proceder a la compra, o bien, salir de la aplicación.

La fase inicial se centrará en la configuración del entorno de desarrollo (como la instalación de Xcode), la definición de la estructura de datos para los productos, y la lógica esencial para el manejo del inventario y la transacción de compra. Este programa sentará las bases digitales para el futuro crecimiento de la boutique Norma.

Descripción.

En esta primera actividad de la materia Desarrollo de Aplicaciones Móviles IV, la boutique Norma requiere el desarrollo de una aplicación funcional de tienda de ropa en línea, marcando su incursión en el comercio electrónico. Este proyecto exige la utilización del lenguaje de programación Swift, reconocido por su seguridad y rendimiento en el ecosistema de Apple, asegurando una plataforma robusta y escalable. La Actividad principal consiste en construir el core del programa de ventas. Esto comienza con la configuración del entorno, ya sea mediante la instalación de Xcode o el uso de un compilador online. Posteriormente, el programa en Swift debe ser capaz de mostrar un catálogo inicial presentando 4 artículos con sus atributos clave: nombre, precio y la cantidad en stock.

El componente esencial de interacción es un menú de selección que guiará al cliente. Este menú debe

permitir al usuario elegir si desea realizar una compra seleccionando un artículo del catálogo, o si prefiere salir del programa. La lógica de compra es crítica: al realizar la compra, el programa deberá descontar la unidad del stock y confirmar la transacción, completando así el ciclo de venta.

Justificación.

La digitalización es crucial para la supervivencia y el crecimiento de la boutique Norma. La justificación de este proyecto radica en la necesidad de expandir el mercado más allá de la ubicación física, ofreciendo a los clientes la comodidad de una tienda en línea. Esto no solo aumenta las ventas potenciales, sino que también moderniza la imagen de la marca. La elección de Swift como lenguaje de programación está justificada por su eficiencia y su fuerte enfoque en la creación de aplicaciones de alta calidad, garantizando una futura migración a plataformas móviles (iOS) más sencillas y un rendimiento óptimo.

La implementación inicial mostrando 4 artículos con precio y stock, y un menú interactivo de compra valida la viabilidad del modelo de e-commerce. Esta actividad permite establecer las bases de la lógica de inventario y transacción. Al crear un ciclo de compra funcional que descuente el stock, se prueba la capacidad del sistema para manejar transacciones reales, preparando a Norma para una completa transformación digital y una mejor gestión de su inventario.

Desarrollo.

En esta primera parte de la materia Desarrollo de Aplicaciones Móviles IV, realizaremos el programa solicitado. Documentaremos el desarrollo de la aplicación, describiendo cada uno de sus requisitos. La aplicación debe incluir cuatro artículos con su precio y stock, así como un menú interactivo de compra que valide la viabilidad del modelo de e-commerce. Cada una de las funciones de la aplicación será la evidencia que se adjuntará a la documentación, e incluirá una breve explicación de su propósito.

Link: GitHub.

Codificación.

En el código Swift se implementa una simulación simple de una tienda en línea de boutique basada en la consola. Su función principal es gestionar un catálogo de productos, permitir al usuario añadir artículos a un carrito de compras y calcular el total final al pagar.

```swift
import Foundation

// Definición de la estructura para un Artículo
struct Articulo {
    let id: Int
    let nombre: String
    let precio: Double
    var stock: Int
}

// 1. Inicializar el catálogo de 4 artículos
var catalogo: [Articulo] = [
    Articulo(id: 1, nombre: "Vestido de Noche", precio: 850.00, stock: 5),
    Articulo(id: 2, nombre: "Blusa de Seda", precio: 320.50, stock: 12),
    Articulo(id: 3, nombre: "Pantalón de Lino", precio: 599.90, stock: 8),
    Articulo(id: 4, nombre: "Falda Plisada", precio: 450.00, stock: 3)
]

// Estructura para almacenar los ítems en el carrito
struct CarritoItem {
    let articulo: Articulo
    var cantidad: Int
}

// Función para obtener una entrada de número entero segura
func obtenerNumeroValido(mensaje: String) -> Int? {
    print(mensaje, terminator: " ")
    guard let input = readLine(), let numero = Int(input) else {
        print("❌ Entrada inválida. Por favor, introduce un número entero.")
        return nil
    }
    return numero
}

// Función para mostrar el catálogo
func mostrarCatalogo() {
    print("\n--- 👗 Catálogo de Boutique Norma 👚 ---")
    print("ID\t| Nombre\t\t| Precio\t| Stock")
    print("---\t| ----------------------| ----------\t|------")
    for articulo in catalogo {

        let precioFormateado = String(format: "%.2f", articulo.precio)

        let tabParaNombre = (articulo.nombre.count < 16) ? "\t" : ""

        print("\(articulo.id)\t| \(articulo.nombre)\(tabParaNombre)\t| $\(precioFormateado)\t| \(articulo.stock)")
    }
    print("-------------------------------------------------")
}
```

En esta parte del código configura las estructuras de datos para los productos y los ítems del carrito,

nicializa el inventario, y proporciona funciones básicas para validar la entrada de datos y presentar el catálogo de productos para el usuario.

```swift
// Lógica principal de la tienda
func iniciarTienda() {
    var seguirComprando = true
    var carrito: [CarritoItem] = []

    print("👋 ¡Bienvenido a la Tienda en Línea de Norma! 👋")

    while seguirComprando {
        mostrarCatalogo()

        print("\nMenú de Opciones:")
        print("1. Comprar un artículo")
        print("2. Finalizar compra y pagar")
        print("3. Salir (sin finalizar compra)")

        guard let opcion = obtenerNumeroValido(mensaje: "Elige una opción (1, 2 o 3):") else {
            continue
        }

        switch opcion {
        case 1:
            guard let idSeleccionado = obtenerNumeroValido(mensaje: "Introduce el ID del artículo que deseas comprar:") else {
                continue
            }

            guard let indiceCatalogo = catalogo.firstIndex(where: { $0.id == idSeleccionado }) else {
                print("❌ El ID del artículo no existe en el catálogo.")
                continue
            }

            let articuloSeleccionado = catalogo[indiceCatalogo]

            if articuloSeleccionado.stock <= 0 {
                print("🚫 El artículo '\(articuloSeleccionado.nombre)' está agotado.")
                continue
            }

            print("¿Cuántas unidades de '\(articuloSeleccionado.nombre)' deseas comprar? (Stock actual: \(articuloSeleccionado.stock))")

            guard let cantidad = obtenerNumeroValido(mensaje: "Cantidad:") else {
                continue
            }

            if cantidad <= 0 {
                print("⚠️ La cantidad debe ser mayor a cero.")
                continue
            }

            if cantidad <= articuloSeleccionado.stock {
```

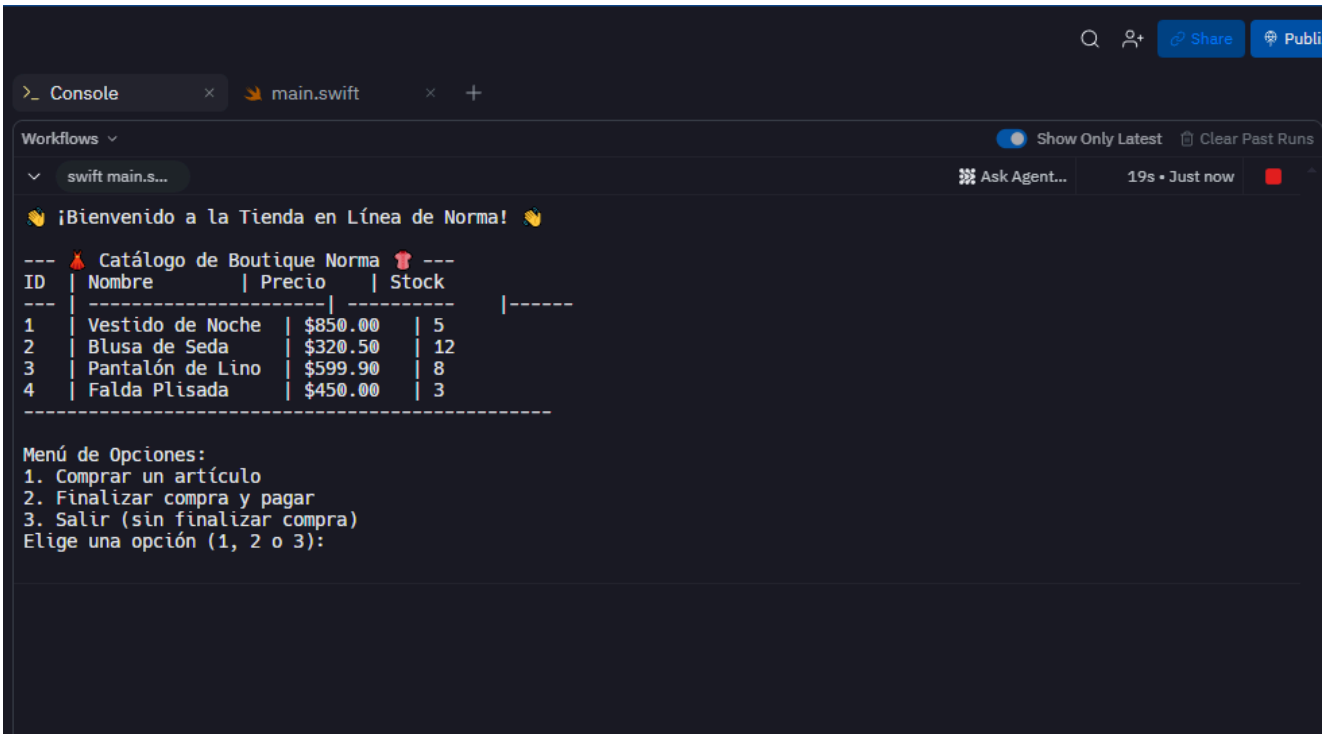
En la función utilizada de iniciarTienda() es el flujo de control que dirige la experiencia del usuario, gestiona la interacción por consola, valida las entradas, y busca y verifica la disponibilidad de los artículos antes de permitir una compra.

```swift
                // Realizar la compra
                if let indiceCarrito = carrito.firstIndex(where: { $0.articulo.id == idSeleccionado }) {
                    carrito[indiceCarrito].cantidad += cantidad
                } else {
                    carrito.append(CarritoItem(articulo: articuloSeleccionado, cantidad: cantidad))
                }

                catalogo[indiceCatalogo].stock -= cantidad

                let costoCompra = articuloSeleccionado.precio * Double(cantidad)

                print("\n✅ ¡Se agrego a tu carrito!")
                print("Artículo(s) añadido(s): \(cantidad)x \(articuloSeleccionado.nombre)")
                print("El costo de esta adición fue: $\(String(format: "%.2f", costoCompra))")

            } else {
                print("⚠️ No hay suficiente stock. Solo quedan \(articuloSeleccionado.stock) unidades.")
            }

        case 2:
            // Finalizar compra y pagar
            if carrito.isEmpty {
                print("🛒 Tu carrito está vacío. ¡Te invitamos a comprar algo!")
                break
            }

            var totalAPagar: Double = 0.0

            print("\n💳 --- Resumen de Compra --- 💳")
            for item in carrito {
                let costoTotalItem = item.articulo.precio * Double(item.cantidad)
                totalAPagar += costoTotalItem
                print("- Compraste \(item.cantidad)x \(item.articulo.nombre). Total por este artículo: $\(String(format: "%.2f", costoTotalItem))")
            }
            print("-----------------------------")
            print("Total Final a Pagar: $\(String(format: "%.2f", totalAPagar))")
            print("¡Gracias por tu compra! Vuelve pronto.")
            seguirComprando = false

        case 3:
            // Salir sin finalizar compra
            print("\n🚪 Has decidido salir. Tu carrito no se guardará. ¡Hasta luego!")
            seguirComprando = false

        default:
            print("❌ Opción no reconocida. Por favor, elige 1, 2 o 3.")
        }
    }
}

// Iniciar la aplicación
iniciarTienda()
```

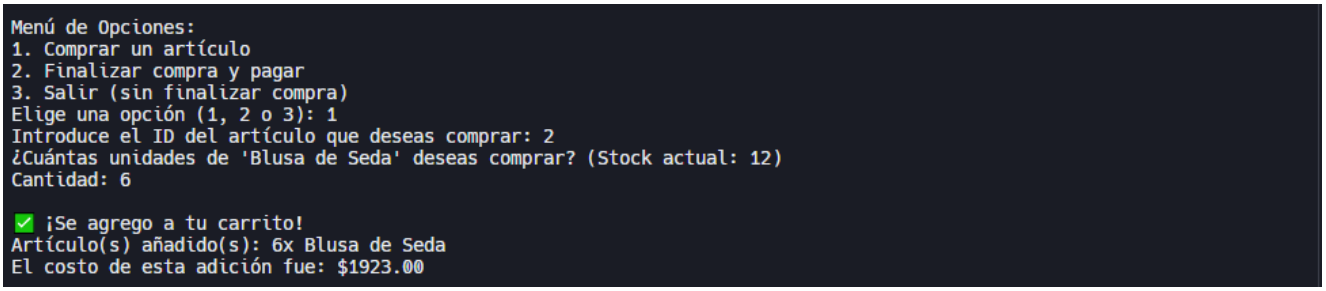
En la función iniciarTienda(), se agregan los artículos al carrito, la finalización de la compra y la opción de salida.

Prueba del Programa.

En esta etapa realizaremos la prueba del funcionamiento de la ejecución del código Swift anterior donde anexaremos los screenshot de evidencia del funcionamiento correcto de aplicación de la tienda en línea.



En esta parte de la ejecución nos muestra la estructuras de datos de los productos con el ID, Nombre del producto, costo y cantidad de productos. Además nos muestra el menú de 3 opciones (Comprar artículo, finalizar compra y salir) este es el menú básico para validar la entrada de datos que se presenta al usuario.

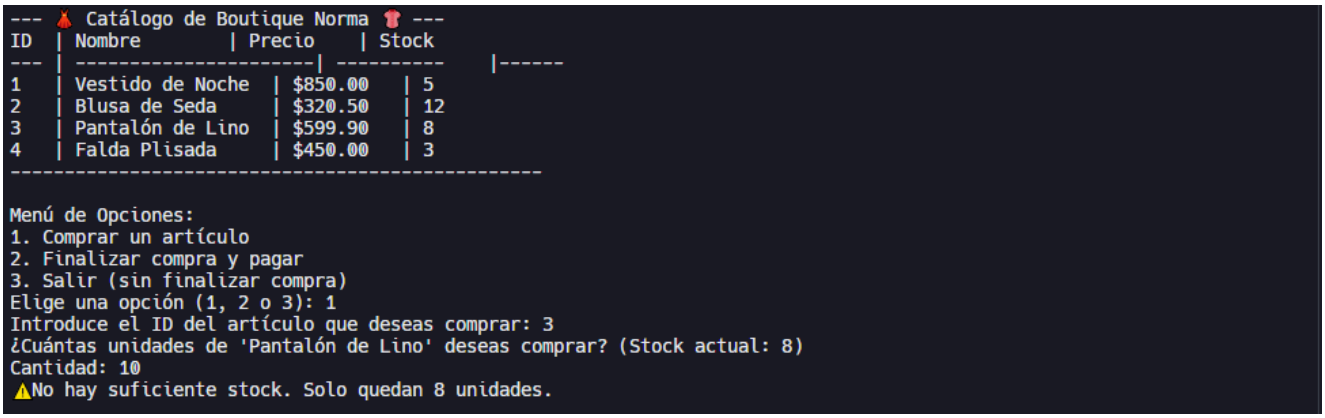


En esta parte de la ejecución realizamos la compra de un artículo, en donde nos pide ingresar el id del artículo la cantidad de artículos a comprar y nos muestra un mensaje de que fue agregado al carrito con la suma de los artículos a comprar.

```
--- 👗 Catálogo de Boutique Norma 👚 ---
ID  | Nombre          | Precio    | Stock
--- | ----------------| ----------    |------
1   | Vestido de Noche  | $850.00   | 5
2   | Blusa de Seda     | $320.50   | 6
3   | Pantalón de Lino  | $599.90   | 8
4   | Falda Plisada     | $450.00   | 3
-------------------------------------------------

Menú de Opciones:
1. Comprar un artículo
2. Finalizar compra y pagar
3. Salir (sin finalizar compra)
Elige una opción (1, 2 o 3): 2

💳 --- Resumen de Compra --- 💳
- Compraste 6x Blusa de Seda. Total por este artículo: $1923.00
------------------------------
Total Final a Pagar: $1923.00
¡Gracias por tu compra! Vuelve pronto.
```

En esta parte final de la ejecución presionamos la opción 2 que es la finalización de la compra y nos manda un mensaje del resumen de la compra y costo mostrando en el inventario una reducción del artículo que se compró, se da agradecimiento por la compra finalizando la ejecución de la aplicación.

A continuación en una segunda ejecución de la aplicación ingresamos desde el inicio una opción no existente en la aplicación y nos muestra un mensaje de error que ese ID del catálogo no existe y nos regresa al menú procinal para seguir utilizando la aplicación.



En esta parte de la evidencia colocamos más artículos de los que existen en stock y nos manda un mensaje que no son suficientes los artículos de stock y que solo se encuentra la cantidad indicada regresar al menú principal para seguir trabajando con la aplicación.

Conclusion.

En conclusión: esta actividad, aparentemente simple, es una piedra angular en la formación de un desarrollador Swift. Al manejar la entrada de usuario, la conversión de datos y la lógica condicional, se establecen las bases para cualquier aplicación interactiva. En el campo laboral, estas habilidades son cruciales para construir interfaces de usuario responsivas, validar datos en formularios o implementar flujos de negocio complejos en apps de banca, comercio o salud. En la vida cotidiana del desarrollador, este ejercicio refuerza el pensamiento lógico y la resolución de problemas, habilidades transferibles a cualquier desafío. Aprender a manejar errores y a estructurar el código de manera limpia son prácticas esenciales que garantizan la creación de software robusto y eficiente. En definitiva, esta primera inmersión en Swift no solo enseña a programar, sino a pensar como un desarrollador. Como dato personal y adicional, se me hizo muy fácil el poder realizar esta actividad debido a que la consola de programación es muy intuitiva y aporta mucho apoyo al desarrollador con el autoayuda o autocompletar al estar desarrollando alguna aplicación.

Referencias.

Google. (n.d.). Gemini. Retrieved November 26, 2025, from https://gemini.google.com/