

Entrega 3: UI + Integración con Backend

Fecha: 06/11/2025

Entregables

1. **Maquetado** de todas las pantallas que el equipo considere necesarias para dar cumplimiento a todos los requerimientos funcionales de la iteración 2, con la correspondiente navegabilidad entre ellas y ajuste al cumplimiento de los requerimientos no funcionales listados en esta iteración.
2. **Implementación** de funcionalidad completa de “*Búsqueda y Visualización de Productos*”, incluyendo su integración con el backend.
3. **Implementación** del carrito de compras del lado cliente.
4. **Justificación** acerca del cumplimiento de los requerimientos no funcionales.

Tecnologías a utilizar

- Todas las mencionadas en las anteriores entregas.
- [Next.js](#) como Framework para el desarrollo de nuestro Frontend.
- [React](#) como biblioteca de generación de componentes para nuestro Frontend.
- [Axios](#) como Cliente HTTP para integrar nuestro Frontend con nuestro Backend.

Justificaciones iniciales

1. Introducción

Esta documentación presenta la justificación de diseño de la interfaz de usuario de la plataforma "Tienda Sol", desarrollada en la tercera iteración del proyecto. La interfaz permite al usuario interactuar con todas las funcionalidades que la tienda ofrece, tales como visualizar productos, filtrar búsquedas, armado de carrito y gestión de pedidos. Todas las pantallas buscan ofrecer una experiencia y usabilidad adecuada, con botones visibles, acciones claras, paleta de colores coherente y tipografía correcta. Se incorpora paginación para consultas eficientes, filtros y ordenamientos para mejorar la usabilidad, y validaciones en los esquemas para garantizar la integridad de los datos. Además, se prevé la implementación de tests e2e para validar la lógica de negocio.

2. Maquetado de interfaces

2.1 Home

La pantalla principal de Tienda Sol tiene como finalidad ofrecer al usuario una primera experiencia atractiva, clara y funcional al ingresar a la plataforma. Desde ella se promueve la exploración de productos, se destaca la navegación hacia el catálogo completo y se refuerza la identidad de marca con un diseño visual coherente.

Diseño de Interfaz

La pantalla se organiza mediante una estructura de tipo layout con componentes reutilizables (Navbar, Footer y ProductCard), asegurando consistencia visual y modularidad. Se emplea Container y Box de Material UI para mantener alineación, márgenes y espaciado uniforme.

- Elementos visuales:
 - Se utiliza una paleta de colores neutra y coherente (grises y azul Oxford) que transmite profesionalismo y legibilidad.
 - La tipografía mantiene jerarquías visuales claras mediante Typography con tamaños adaptativos (xs, sm, md) que garantizan la correcta visualización en distintos dispositivos.
 - Los botones cuentan con labels descriptivos y atributos de accesibilidad (aria-label) para favorecer la inclusión y la comprensión de las acciones.
- Usabilidad y accesibilidad:
 - Se aplican roles y etiquetas ARIA (role="region", aria-label) para permitir la navegación asistida por lectores de pantalla.
 - El diseño es responsive, adaptando grillas y carruseles según el ancho del dispositivo gracias a las propiedades responsive en el componente Slider.
 - Se minimiza la sobrecarga visual mediante una disposición limpia y uso de espacio en blanco.

Integración con el Backend

- Se utiliza Axios como cliente HTTP para integrar el frontend con el backend de productos. El hook *useEffect* ejecuta una llamada GET a /productos, lo que permite visualizar dinámicamente los productos más vendidos en la Home.
- El estado local (*useState*) almacena los datos recibidos y controla los mensajes de carga (Cargando productos...) y error (No se pudieron cargar los productos.), asegurando feedback inmediato al usuario.
- Cada producto se renderiza mediante el componente ProductCard, garantizando una visualización homogénea y una arquitectura de componentes desacoplada.



2.2 Products (comprador)

La pantalla Productos permite al usuario explorar el catálogo completo de Tienda Sol, con herramientas de búsqueda, filtrado y ordenamiento que optimizan la experiencia de compra. Su diseño busca equilibrar claridad visual, eficiencia en la carga y fácil interacción con los productos.

Diseño de Interfaz

La vista combina un panel lateral de filtros y un grid central de productos, priorizando la organización y legibilidad. Se emplean componentes de Material UI (Container, Box, Select, Typography) junto con clases utilitarias de Tailwind para garantizar un diseño adaptable y limpio. La interfaz es responsive, ocultando el panel lateral en pantallas pequeñas y reorganizando los productos mediante un grid flexible.

Filtros y ordenamientos: Incluye controles para filtrar por precio, categoría y orden de resultados, además de un buscador integrado en el Navbar. Los filtros están acompañados de etiquetas y placeholders claros, manteniendo coherencia visual con el resto del sistema.

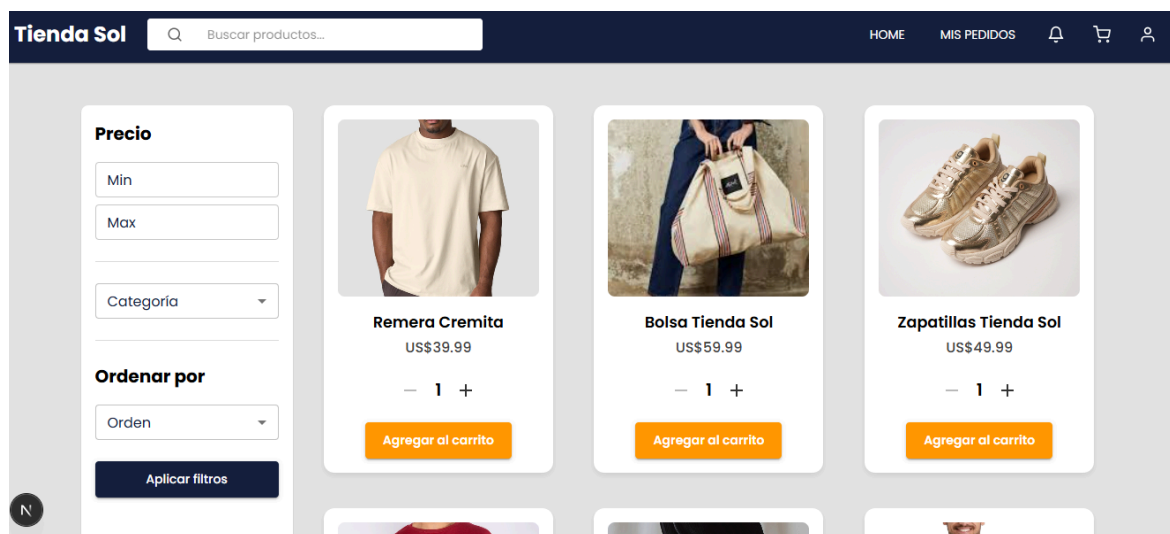
Integración con el Backend

Se implementa la comunicación con el servidor mediante Axios, consumiendo los endpoints:

- GET /productos para obtener productos con soporte de paginación, búsqueda y filtros dinámicos.
- GET /productos/categorías para cargar categorías disponibles desde el backend.

Los parámetros enviados (paginación, categoría, precio, orden) se construyen dinámicamente en la consulta, garantizando una búsqueda precisa y optimizada.

Los datos recibidos se almacenan en estado local y actualizan el DOM de forma reactiva, sin recargar la página.



2.2 Products (vendedor)

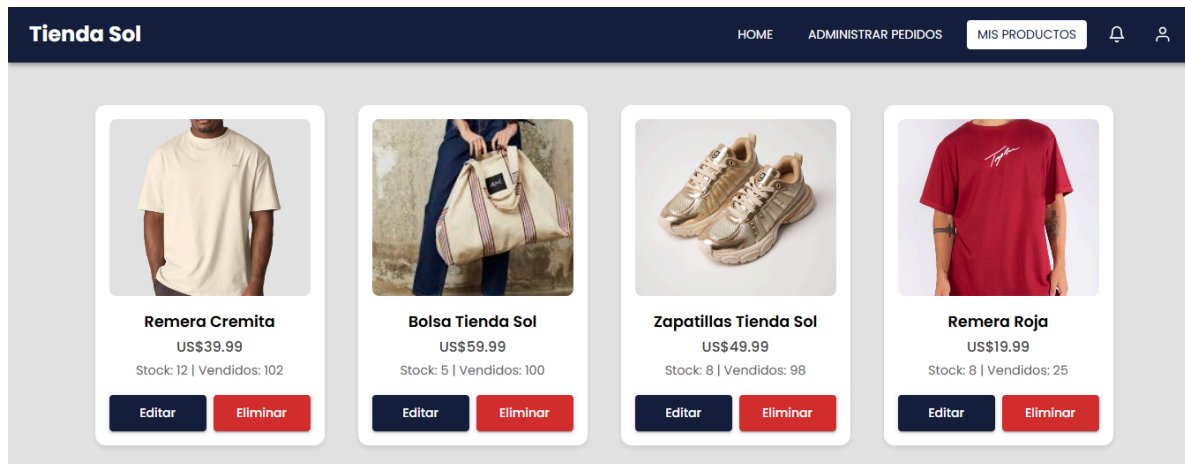
La pantalla Productos del Vendedor permite al usuario con rol de vendedor visualizar, editar y eliminar los productos que tiene publicados en Tienda Sol. Su propósito es ofrecer una interfaz sencilla para la gestión del catálogo propio, facilitando la administración del stock, precios y descripciones.

Diseño de Interfaz

- Los botones de acción (editar y eliminar) están claramente diferenciados, reforzando la usabilidad y evitando errores.
- Diálogo de edición: El formulario de modificación se implementa con un Dialog de Material UI, que permite editar en contexto sin abandonar la pantalla. Incluye campos validados de texto, número y descripción, con un diseño consistente y etiquetas descriptivas.

Interacción y Navegación

- El vendedor puede editar un producto haciendo clic en el botón correspondiente, lo que abre un modal donde se modifican los campos de manera controlada.
- La opción eliminar y editar producto simulan la acción con confirmación en consola, pensadas para integrarse posteriormente con la API real.
- Se incluye paginación para mejorar la legibilidad cuando la lista de productos es extensa, evitando sobrecarga visual.



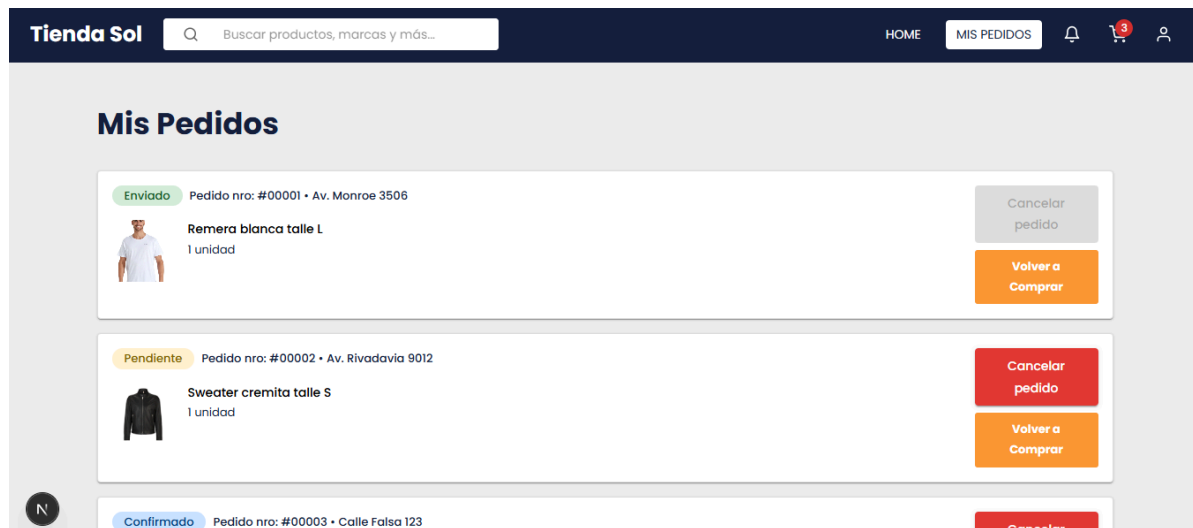
2.3 Pedidos (comprador)

La pantalla Pedidos del Comprador permite al usuario visualizar el historial y estado de sus compras dentro de Tienda Sol. Desde esta sección, puede consultar los detalles de cada pedido, su dirección de entrega y los productos adquiridos.

El objetivo principal es ofrecer **transparencia y trazabilidad** del proceso de compra, mostrando de forma clara el estado de cada pedido (pendiente, confirmado, enviado, cancelado, etc.).

Visualización de pedidos: Cada pedido se muestra como un bloque con número de orden, dirección de entrega, estado actual y los productos asociados. El diseño mantiene una estética minimalista y limpia, priorizando la legibilidad de la información y la coherencia visual.

Adaptabilidad: El diseño es totalmente responsive, adaptándose a distintos tamaños de pantalla sin perder legibilidad ni funcionalidad.

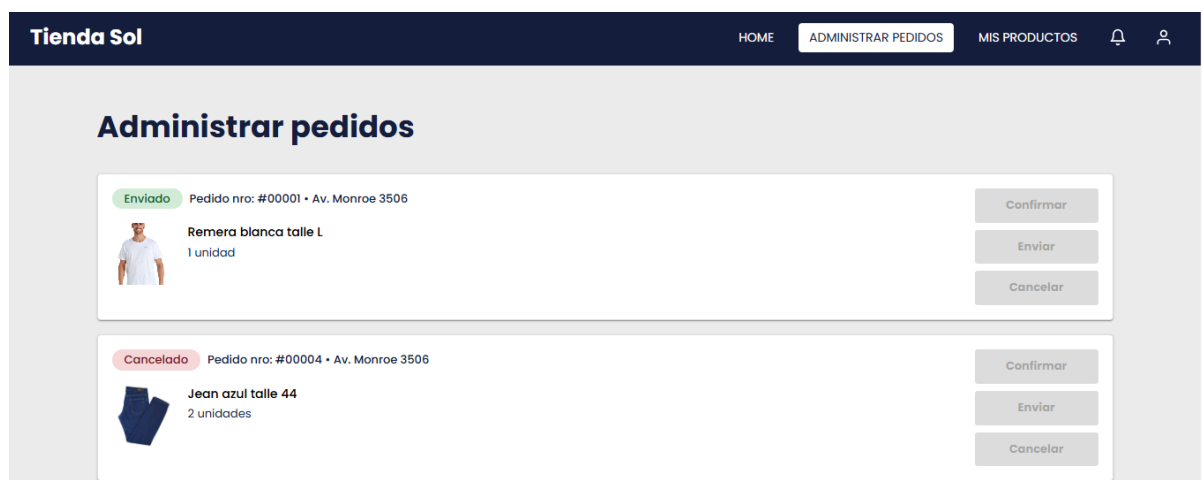


Pantalla del Vendedor

La pantalla del Vendedor comparte la misma base estructural y lógica de presentación (OrderRow y Pagination), pero se diferencia de las acciones disponibles:

- El vendedor gestiona los pedidos (confirmar, enviar, cancelar).
- El comprador consulta o cancela sus propios pedidos.

De este modo, ambas pantallas mantienen consistencia visual y funcional, adaptando las acciones según el rol autenticado en la sesión.



2.4 Notificaciones

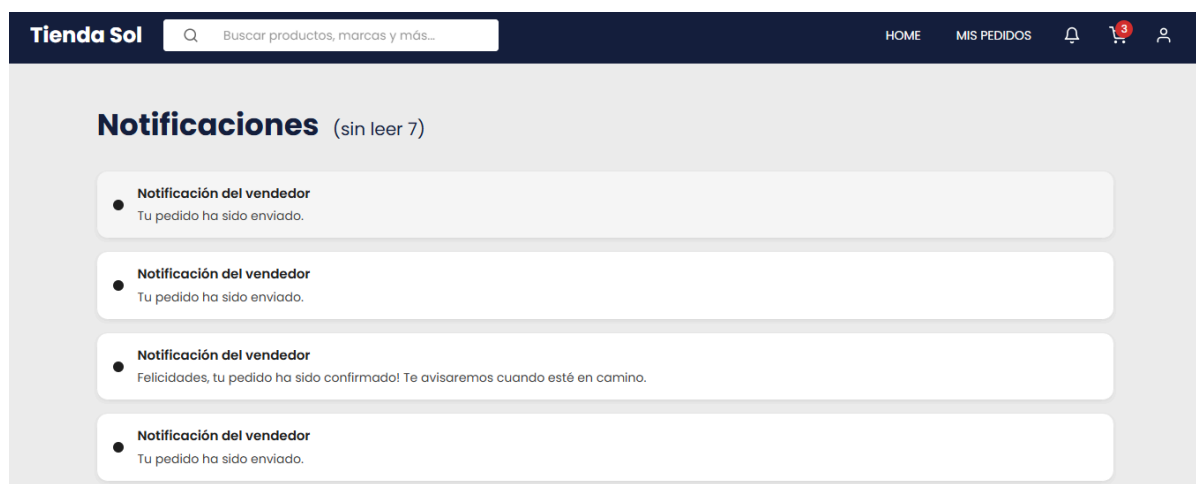
La pantalla Notificaciones permite al usuario visualizar los avisos relacionados con su actividad en la plataforma, como actualizaciones de pedidos, confirmaciones, envíos o cancelaciones. Su propósito es mantener una comunicación clara y centralizada entre el sistema y el usuario, evitando que deba revisar manualmente cada pedido o acción.

Diseño de Interfaz

La vista mantiene la estructura general del sitio con Navbar, Footer y contenido principal centrado (Container). Cada notificación se muestra mediante el componente NotificationCard, con un diseño limpio, espaciado y jerarquía visual clara. El contador dinámico de “sin leer” resalta la información más relevante, favoreciendo la atención del usuario.

Interacción y Navegación

- El usuario puede marcar notificaciones como leídas o no leídas mediante interacción directa.
- Se utiliza paginación para navegar cómodamente entre múltiples avisos.
- La información se actualiza en tiempo real en la interfaz, sin recargar la página.
- Las notificaciones se obtendrán según la sesión, mostrando solo los avisos correspondientes a cada usuario.



2.5 Carrito

La pantalla de Carrito permite al usuario administrar los productos seleccionados antes de avanzar al checkout. Su implementación se basa en un estado global mediante **Context API (useContext)**, lo que evita el paso excesivo de props y permite acceder al carrito desde cualquier parte de la aplicación (Navbar, ProductCard, Checkout, etc.).

Implementación del Estado (CartContext)

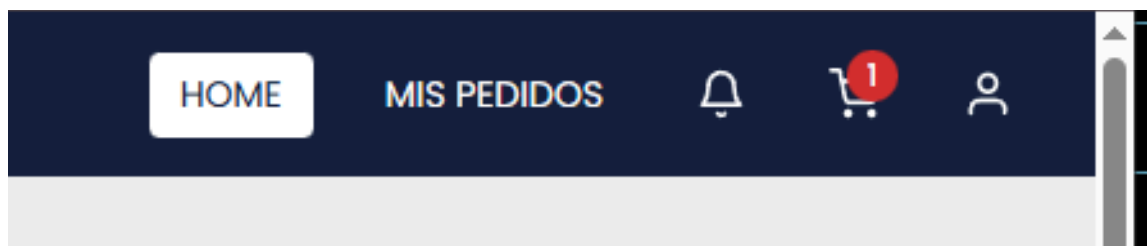
El carrito se gestiona con un **CartContext**, que centraliza todas las operaciones:

- addToCart(product, quantity)
- updateQuantity(productId, qty)
- removeFromCart(productId)
- clearCart()

El contexto almacena los ítems en un arreglo de objetos que extienden Product con el campo cantidad.

Para persistencia local se utiliza **localStorage**, cargando el carrito al iniciar la app y guardando los cambios en cada actualización del state.

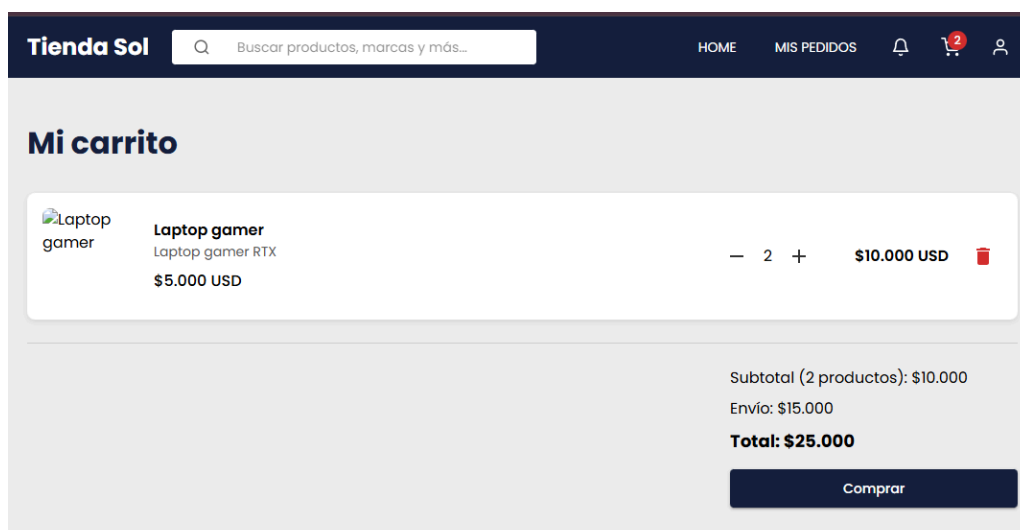
Esto permite mantener el carrito aunque el usuario recargue la página o cambie de vista.



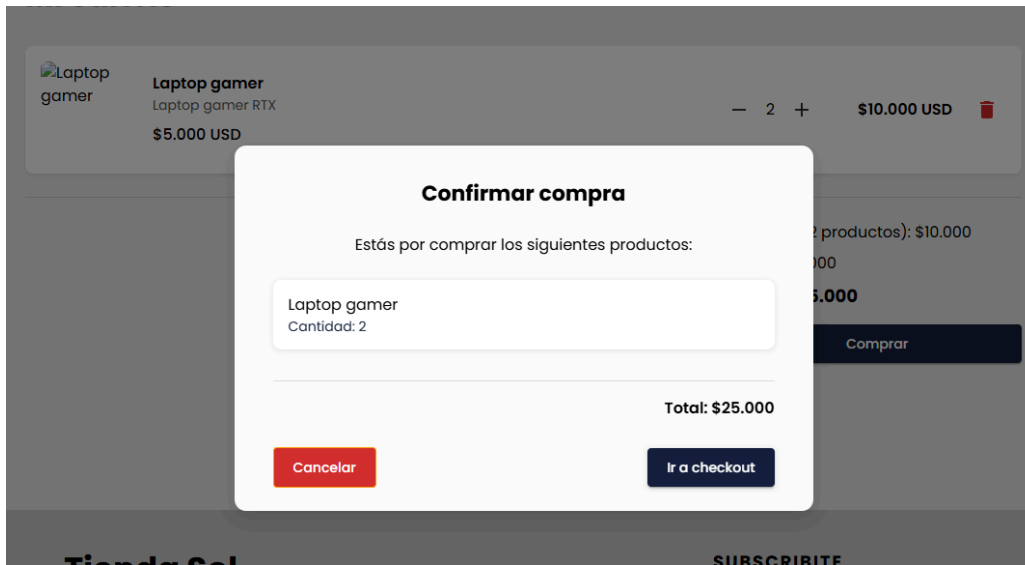
Pantalla del Carrito (CarritoPage)

La vista consume el contexto mediante useCart() y ofrece las funcionalidades principales:

- **Incrementar y decrementar cantidades**, con validaciones según stock disponible.
- **Eliminar productos** con un botón accesible.
- **Cálculo automático de subtotal, envío y total** directamente desde el estado del contexto.
- **Modal de confirmación**, para revisar los productos antes de pasar al checkout.
- **Redirección a /checkout** mediante router.



Luego de tocar comprar, sale un modal para confirmar la misma que permite asegurar que el usuario realmente quiere realizar esa acción



Al confirmar la compra, te envía a la pantalla de checkout, que muestra un resumen del pedido y al enviar los datos personales realiza un POST a pedidos

Resumen del pedido	Datos del comprador
Laptop gamer × 2 \$10.000	Nombre
Total: \$25.000	Apellido
	Email
	Repetir Email
	Finalizar compra

2.6 Navbar

El sistema utiliza un componente de navegación adaptable según el tipo de usuario (**buyer** o **seller**) y según el dispositivo (**mobile** o **desktop**). Está construido con **Material UI**, **Next.js** y componentes reutilizables.

BuyerNavbar

Es la barra de navegación para compradores. Sus funciones principales:

- **Búsqueda integrada**, con formulario que redirige a `/products?search=...`

- **Badge del carrito**, conectado al **CartContext**, mostrando en tiempo real el total de productos.
- **Menú de usuario** (UsuarioMenu), accesible desde cualquier dispositivo.
- **Notificaciones** con indicador visual si estás en la ruta correspondiente.
- **Responsividad completa**:
 - En mobile se usa un **Drawer lateral** para los enlaces.
 - En desktop se muestran los enlaces directamente en la barra.
- **Enlaces dinámicos** y estilos que resaltan el link activo usando `usePathname()`.



SellerNavbar

Versión simplificada enfocada en el vendedor:

- No incluye buscador ni carrito.
- Contiene los enlaces del vendedor (HOME, ADMINISTRAR PEDIDOS, MIS PRODUCTOS).
- También alterna entre vista compacta en mobile con Drawer y vista completa en desktop.
- Incluye notificaciones y menú de usuario.



Navbar (wrapper)

Es un componente unificador que decide **qué Navbar renderizar** según el tipo de usuario:

- Si `userType === 'buyer'` → muestra BuyerNavbar.
- Si `userType === 'seller'` → muestra SellerNavbar.
- Soporta un modo minimal usado en pantallas especiales (login, registro, etc.), donde solo se muestra el logo.

3. Manejo de sesiones (TODO próxima entrega)

El backend utiliza **Clerk** para gestionar autenticación y control de acceso. El middleware global (`clerkMiddleware`) se configura en el servidor y expone la información de sesión en cada request.

Para rutas que requieren login se usa `requireAuth()`, que valida automáticamente la sesión y devuelve 401 si no existe.

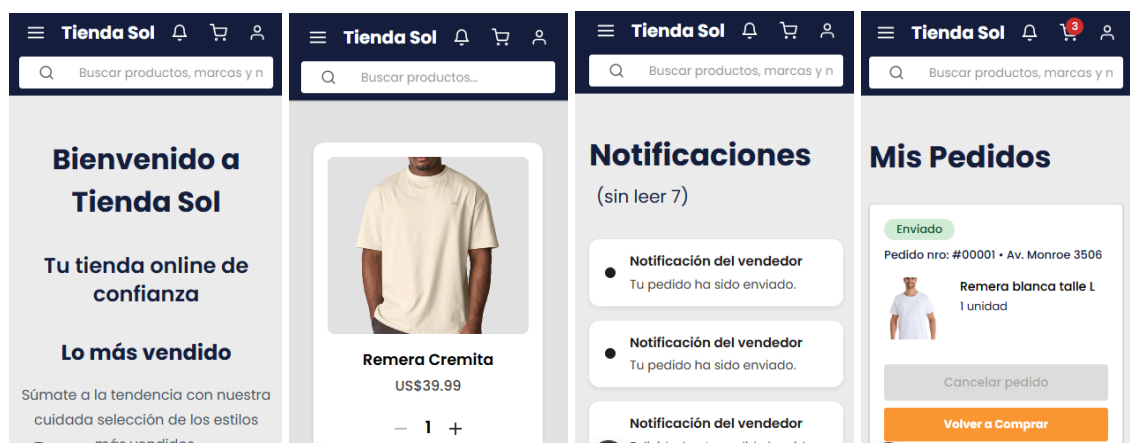
Algunas rutas permiten autenticación opcional, leyendo `req.auth?.userId` para adaptar la respuesta según el usuario.

Además, se implementó un middleware propio `requireRole`, que verifica los roles asignados al usuario (buyer, seller, etc.) y restringe el acceso según corresponda. De esta forma se pueden crear endpoints exclusivos para vendedores, compradores o ambos.

Esta estructura asegura rutas públicas, rutas protegidas por sesión y rutas protegidas por rol, manteniendo una arquitectura clara y escalable.

4. Responsive

Vistas de la interfaz desde un dispositivo móvil



5. Conclusión

Requerimientos no funcionales cubiertos en la entrega:

- Rendimiento: Se limita la carga inicial de productos (`limit=3`) y se usa paginación para evitar sobrecargar el renderizado.
- Usabilidad: Interfaz intuitiva, elementos reconocibles y botones visibles.
- Accesibilidad: Uso de etiquetas ARIA y roles semánticos en los elementos principales.
- Reutilización de componentes: Navbar, Footer y ProductCard se emplean en múltiples pantallas, promoviendo la modularidad.
- Escalabilidad: La estructura basada en componentes y hooks facilita la ampliación del proyecto.