Documentación de Pruebas de Calidad

Módulo de Productos - Componentes CatalogoUsuario y CartShop

Introducción

Este documento presenta el plan de pruebas de calidad para los componentes relacionados con el manejo de productos y carrito de compras en el sistema, incluyendo el componente CatalogoUsuario (catálogo de productos) y CartShop (carrito de compras). Se detallan los casos de prueba, criterios de aceptación, y escenarios de prueba para asegurar la correcta funcionalidad del módulo.

Componentes Analizados:

- CatalogoUsuario (Frontend)
- CartShop (Frontend)
- APIs relacionadas con inventarios y carrito

Tecnologías utilizadas:

- Frontend: React, Axios
- Backend: PHP, Laravel (APIs)
- Autenticación: Bearer Token
- Base de datos: Relacional

Fecha de evaluación: 4 de junio de 2025

Descripción de los Componentes

Frontend - CatalogoUsuario

El componente CatalogoUsuario es responsable de mostrar el catálogo de productos disponibles y permitir agregarlos al carrito. Sus características incluyen:

- Visualización de productos en tarjetas
- Mostrar información básica (nombre, descripción, precio)
- Agregar productos al carrito
- Manejo de autenticación con tokens

Frontend - CartShop

El componente CartShop gestiona el carrito de compras del usuario. Sus funcionalidades incluyen:

- Visualización de productos en el carrito
- Modificar cantidades de productos
- Eliminar productos del carrito
- Calcular total de la compra
- Finalizar compra
- Manejo de estado vacío

Funcionalidades a Probar

1. Frontend - CatalogoUsuario

- Carga y visualización de productos
- Agregar productos al carrito
- Manejo de autenticación
- Manejo de estados (carga, error)

2. Frontend - CartShop

- Visualización de productos en carrito
- Modificar cantidades
- Eliminar productos
- Cálculo de totales
- Finalizar compra
- Manejo de carrito vacío

Casos de Prueba

Pruebas Funcionales - CatalogoUsuario

ID	Caso de Prueba	Datos de Entrada	Resultado Esperado	Criterio de Aceptación
CAT-01	Carga de productos con usuario autenticado	Usuario con token válido accede al catálogo	Lista de productos se muestra en tarjetas con nombre, descripción y precio	Los productos se visualizan correctamente organizados en grid
CAT-02	Acceso sin token	Usuario sin token de autenticación	Mensaje en consola "No hay token guardado" y no se cargan productos	El sistema maneja correctamente la falta de autenticación
CAT-03	Error en carga de productos	API devuelve error 500	Error se registra en consola, productos no se cargan	El sistema maneja errores de API sin crashear
CAT-04	Catálogo vacío	API devuelve array vacío	Grid vacío sin productos	El componente maneja correctamente la ausencia de productos
CAT-05	Agregar producto al carrito exitosamente	Usuario autenticado hace clic en "Agregar al carrito"	Alert "Producto agregado" aparece	El usuario recibe confirmación de agregado exitoso
CAT-06	Agregar producto sin autenticación	Usuario sin token hace clic en "Agregar al carrito"	Error se registra en consola	El sistema rechaza peticiones no autenticadas
CAT-07	Error al agregar producto	API devuelve error al agregar	Error se registra en consola	El sistema maneja errores de agregado

Pruebas Funcionales - CartShop

ID	Caso de Prueba	Datos de Entrada	Resultado Esperado	Criterio de Aceptación
CART-01	Visualizar carrito con productos	Usuario autenticado con productos en carrito	Lista de productos con nombre, precio, cantidad y total	Los productos se muestran con información completa
CART-02	Carrito vacío	Usuario autenticado sin productos en carrito	Mensaje "Tu carrito está vacío"	Se muestra mensaje informativo cuando no hay productos
CART-03	Aumentar cantidad de producto	Clic en botón "+" de un producto	Cantidad aumenta en 1, total se recalcula	La cantidad y total se actualizan correctamente
CART-04	Disminuir cantidad de producto	Clic en botón "-" de un producto con cantidad > 1	Cantidad disminuye en 1, total se recalcula	La cantidad y total se actualizan correctamente
CART-05	Prevenir cantidad menor a 1	Clic en botón "-" de un producto con cantidad = 1	Cantidad permanece en 1	El sistema previene cantidades inválidas
CART-06	Eliminar producto del carrito	Clic en botón eliminar (🗑)	Producto se elimina del carrito, total se recalcula	El producto se elimina y el total se actualiza
CART-07	Finalizar compra exitosamente	Clic en "Finalizar compra" con productos en carrito	Alert "Compra finalizada", carrito se vacía	La compra se procesa y el carrito se limpia
CART-08	Error al finalizar compra	API devuelve error al finalizar	Error se registra en consola	El sistema maneja errores de compra
CART-09	Cálculo correcto de total	Múltiples productos con diferentes cantidades	El total refleja precio × cantidad de todos los productos	El cálculo matemático es correcto

Pruebas de Interfaz de Usuario

ID	Caso de Prueba	Pasos	Resultado Esperado	Criterio de Aceptación
UI-01	Diseño responsive del catálogo	Acceder desde diferentes tamaños de pantalla	El grid se adapta al tamaño de pantalla	El componente es usable en móviles, tablets y escritorio
UI-02	Diseño responsive del carrito	Acceder desde diferentes tamaños de pantalla	Los elementos se reorganizan apropiadamente	El carrito es funcional en todos los dispositivos
UI-03	Feedback visual al agregar producto	Clic en "Agregar al carrito"	Alert nativo aparece con mensaje de confirmación	El usuario recibe feedback inmediato
UI-04	Controles de cantidad intuitivos	Interactuar con botones +/-	Los botones responden visualmente al hover/click	Los controles son intuitivos y responsivos
UI-05	Visualización del total	Cambiar cantidades en el carrito	El total se actualiza inmediatamente	El cálculo es visible y se actualiza en tiempo real
UI-06	Estado de carrito vacío	Acceder con carrito sin productos	Mensaje claro y diseño limpio	El estado vacío es informativo y no confuso

Pruebas de Seguridad

ID	Caso de Prueba	Descripción	Resultado Esperado	Criterio de Aceptación
SEC-01	Autenticación en catálogo	Acceder sin token válido	No se cargan productos, se registra en consola	El sistema protege el acceso a productos
SEC-02	Autenticación en carrito	Acceder sin token válido	Error al cargar carrito	El carrito requiere autenticación
SEC-03	Validación de token en operaciones	Usar token expirado/inválido	APIs rechazan peticiones	El sistema valida tokens en todas las operaciones
SEC-04	Protección contra manipulación	Intentar modificar datos desde consola	Las operaciones van a través de APIs	No se puede manipular estado sin validación

Pruebas de Integración

ID	Caso de Prueba	Descripción	Resultado Esperado	Criterio de Aceptación
INT-01	Flujo completo de compra	Navegar catálogo → agregar productos → modificar carrito → finalizar	Flujo completo funciona sin errores	La integración entre componentes es fluida
INT-02	Sincronización catálogo-carrito	Agregar producto desde catálogo y verificar en carrito	El producto aparece correctamente en carrito	Los datos se sincronizan entre componentes
INT-03	Persistencia de carrito	Agregar productos, cerrar y reabrir aplicación	Los productos permanecen en carrito	El carrito persiste entre sesiones
INT-04	Actualización de inventario	Finalizar compra y verificar disponibilidad	Los productos se actualizan en backend	La compra afecta el inventario

Pruebas de Rendimiento

ID	Caso de Prueba	Descripción	Resultado Esperado	Criterio de Aceptación
PERF-01	Carga de catálogo con muchos productos	Cargar catálogo con 100+ productos	Tiempo de carga < 3 segundos	El catálogo mantiene buen rendimiento
PERF-02	Operaciones de carrito rápidas	Realizar múltiples cambios de cantidad	Respuesta inmediata < 500ms	Las operaciones son fluidas
PERF-03	Cálculo de totales eficiente	Carrito con 20+ productos diferentes	Cálculo instantáneo	El cálculo no causa lag
PERF-04	Múltiples usuarios simultáneos	Simular múltiples usuarios comprando	Sistema responde sin degradación	El sistema escala apropiadamente

Pruebas de Accesibilidad

ID	Caso de Prueba	Descripción	Resultado Esperado	Criterio de Aceptación
ACC-01	Navegación con teclado	Navegar catálogo y carrito solo con teclado	Todos los elementos son accesibles	Funcionalidad completa sin ratón
ACC-02	Lectores de pantalla	Probar con lector de pantalla	Información se lee correctamente	Accesible para usuarios con discapacidad visual
ACC-03	Contraste de colores	Evaluar legibilidad de texto	Cumple estándares WCAG	Contraste suficiente en todos los elementos
ACC-04	Etiquetas descriptivas	Verificar atributos alt y aria	Todos los elementos tienen etiquetas	Los elementos son descriptivos

Defectos y Mejoras Sugeridas

Defectos Identificados

CatalogoUsuario:

1. **Uso de localStorage para tokens**: No es la práctica más segura
2. **Alert nativo**: Usa alert() nativo en lugar de componente integrado
3. **Falta de feedback de carga**: No hay indicador mientras se cargan productos
4. **Gestión de errores limitada**: Solo se registra en consola
5. **Falta validación de datos**: No valida si los campos del producto son null

CartShop:

1. **Manejo inconsistente de items**: Usa `items ?? []` en algunas partes
2. **Cálculo de total vulnerable**: No valida que price sea número
3. **Alert nativo**: Usa alert() para confirmación de compra
4. **Falta de confirmación**: No pide confirmación antes de eliminar
5. **Token repetido**: Obtiene token en cada función en lugar de usar contexto

Mejoras Sugeridas

Para CatalogoUsuario:

javascript

```javascript
// 1. Implementar estado de carga
const [loading, setLoading] = useState(false);
const [error, setError] = useState(null);

// 2. Mejorar gestión de errores
const cargarProductos = async () => {
  setLoading(true);
  setError(null);
  try {
    const response = await axios.get('/api/inventarios', {
      headers: { Authorization: `Bearer ${token}` }
    });
    setProductos(response.data);
  } catch (err) {
    setError('Error al cargar productos');
    console.error(err);
  } finally {
    setLoading(false);
  }
};

// 3. Componente de alerta personalizado
const [alerta, setAlerta] = useState(null);

// 4. Validación de datos
<h3>{prod.nombre || 'Producto sin nombre'}</h3>
<p className="precio">${prod.precio ? Number(prod.precio).toFixed(2) : '0.00'}</p>
```

Para CartShop:

```javascript
// 1. Usar contexto para token
const { token } = useAuth();

// 2. Confirmación antes de eliminar
const eliminar = (itemId) => {
  if (window.confirm('¿Estás seguro de eliminar este producto?')) {
    // Lógica de eliminación
  }
};

// 3. Validación segura de cálculos
const total = (items || []).reduce((acc, item) => {
  const price = Number(item.price) || 0;
  const quantity = Number(item.quantity) || 0;
  return acc + (price * quantity);
}, 0);

// 4. Componente de notificación
const [notification, setNotification] = useState(null);

// 5. Manejo de errores mejorado
const finalizarCompra = async () => {
  try {
    await axios.post('/api/cartshop/checkout', {}, {
      headers: { Authorization: `Bearer ${token}` }
    });
    setNotification({ type: 'success', message: 'Compra finalizada exitosamente' });
    setItems([]);
  } catch (err) {
    setNotification({
      type: 'error',
      message: err.response?.data?.message || 'Error al finalizar compra'
    });
  }
};
```

Conclusiones

El módulo de productos tiene una funcionalidad básica bien implementada, pero presenta varias áreas de mejora importantes:

Fortalezas:

- Separación clara de responsabilidades entre catálogo y carrito
- Funcionalidad básica completa (mostrar, agregar, modificar, eliminar)
- Manejo básico de autenticación
- Cálculo automático de totales

Áreas de mejora prioritarias:

1. **Gestión de errores**: Implementar manejo robusto de errores con feedback visual
2. **Componentes de UI**: Reemplazar alerts nativos con componentes integrados
3. **Validación de datos**: Agregar validaciones para prevenir errores
4. **Estados de carga**: Implementar indicadores de carga para mejor UX
5. **Confirmaciones**: Agregar confirmaciones para acciones destructivas

Recomendaciones:

- Implementar un sistema de notificaciones centralizado
- Usar contexto de React para manejo de autenticación
- Agregar pruebas unitarias con React Testing Library
- Implementar manejo de estado más robusto (Redux o Zustand)
- Agregar validación de esquemas (Yup o Zod)

La implementación de estas mejoras aumentaría significativamente la calidad y confiabilidad del módulo de productos.

Anexos

Entorno de Pruebas Recomendado

Navegadores a probar:

- Google Chrome (última versión)
- Mozilla Firefox (última versión)
- Microsoft Edge (última versión)
- Safari (última versión)

Dispositivos a probar:

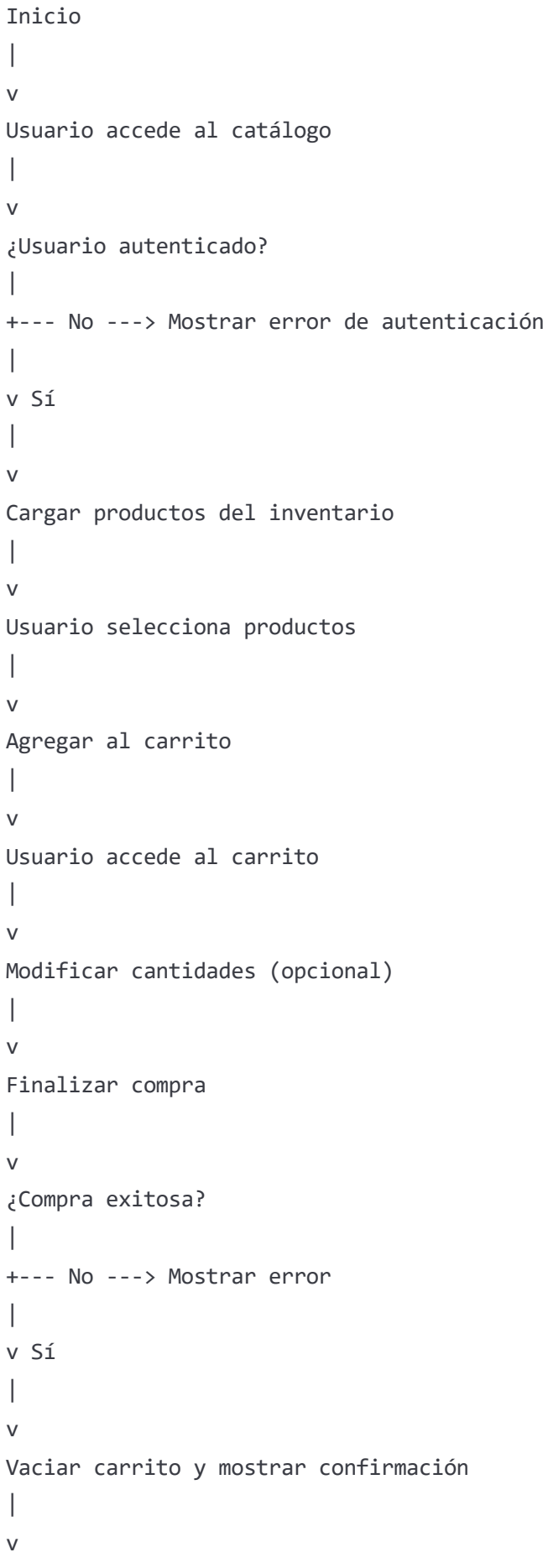
- Desktop (resoluciones 1920x1080, 1366x768)
- Tablet (iPad, resolución 768x1024)

- Móvil (iPhone, Galaxy, resoluciones 375x667, 414x896)

Herramientas recomendadas:

- React Testing Library para pruebas unitarias
- Jest para testing de componentes
- Cypress para pruebas E2E
- Postman para pruebas de API
- Lighthouse para evaluación de rendimiento
- axe para pruebas de accesibilidad

Diagrama de Flujo - Proceso de Compra


Inicio
Usuario accede al catálogo
¿Usuario autenticado?
No ---> Mostrar error de autenticación
Sí
Cargar productos del inventario
Usuario selecciona productos
Agregar al carrito
Usuario accede al carrito
Modificar cantidades (opcional)
Finalizar compra
¿Compra exitosa?
No ---> Mostrar error
Sí
Vaciar carrito y mostrar confirmación
Fin