COMPONENTES FRONTEND DETALLADOS

Sistema de Inventario - Documentación React

RESUMEN DE COMPONENTES

Tipo	Cantidad	Archivos
Páginas	6	Auth, Dashboard, ForgotPassword, ResetPassword, Login, Signup
Modales	9	Product, Category, Supplier, Sale, Movement, Password, Profile, UserRole, Confirm
Utilidades	5	ProtectedRoute, ToastProvider, ThemeSwitch, LottieIcon, SearchInput
Utils	3	api.js, export.js, company.js

COMPONENTE: ProtectedRoute.jsx

Propósito: Proteger rutas que requieren autenticación

Código Completo:

```
import React from 'react'
import { Navigate } from 'react-router-dom'

export default function ProtectedRoute({ children }) {
  // Buscar token en localStorage o sessionStorage
  const token = localStorage.getItem('token') || sessionStorage.getItem('token')

  if (!token) {
    // Si no hay token, redirigir a login
    return <Navigate to="/" replace />
  }

  try {
    // Decodificar JWT para verificar expiración
    const payload = JSON.parse(atob(token.split('.')[1]))
    const now = Math.floor(Date.now() / 1000)

    if (payload.exp < now) {
      // Token expirado
      localStorage.removeItem('token')
      sessionStorage.removeItem('token')
      return <Navigate to="/" replace />
    }

    // Token válido, renderizar children
    return children
```

```
  } catch (error) {
    // Token corrupto
    localStorage.removeItem('token')
    sessionStorage.removeItem('token')
    return <Navigate to="/" replace />
  }
}
```

Uso en App.jsx:

```
<Route path="/dashboard" element={
  <ProtectedRoute>
    <Dashboard />
  </ProtectedRoute>
} />
```

¿Cómo funciona?

1. Verifica si existe token en storage
2. Decodifica el payload del JWT (parte del medio)
3. Compara fecha de expiración con fecha actual
4. Si es válido → Muestra el componente hijo
5. Si no → Redirige a login

🔔 COMPONENTE: ToastProvider.jsx

Propósito: Sistema de notificaciones estilo "toast" (esquina superior derecha)

Código Completo:

```
import React, { createContext, useCallback, useContext, useMemo, useState } from
'react'
import { AnimatePresence, motion } from 'framer-motion'

const ToastContext = createContext({ notify: () => {} })

export function useToast() {
  return useContext(ToastContext)
}

export default function ToastProvider({ children }) {
  const [toasts, setToasts] = useState([])

  const notify = useCallback((message, opts = {}) => {
    const id = Math.random().toString(36).slice(2)
    const toast = {
      id,
      message,
```

```
      type: opts.type || 'info',  // 'success', 'error', 'warning', 'info'
      duration: opts.duration ?? 3000
    }

    setToasts((t) => [...t, toast])

    if (toast.duration > 0) {
      setTimeout(() => {
        setToasts((t) => t.filter((x) => x.id !== id))
      }, toast.duration)
    }
  }, [])

  const remove = useCallback((id) => {
    setToasts((t) => t.filter((x) => x.id !== id))
  }, [])

  const value = useMemo(() => ({ notify }), [notify])

  return (
    <ToastContext.Provider value={value}>
      {children}
      <div className="toast-container">
        <AnimatePresence>
          {toasts.map((t) => (
            <motion.div
              key={t.id}
              initial={{ opacity: 0, y: 20 }}
              animate={{ opacity: 1, y: 0 }}
              exit={{ opacity: 0, y: 10 }}
              transition={{ duration: 0.2 }}
              className={`toast toast--${t.type}`}
              onClick={() => remove(t.id)}
              role="status"
            >
              {t.message}
            </motion.div>
          ))}
        </AnimatePresence>
      </div>
    </ToastContext.Provider>
  )
}
```

CSS (styles.css):

```
.toast-container {
  position: fixed;
  top: 20px;
  right: 20px;
  z-index: 9999;
  display: flex;
```

```
  flex-direction: column;
  gap: 10px;
}

.toast {
  background: white;
  padding: 16px 24px;
  border-radius: 8px;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15);
  min-width: 300px;
  cursor: pointer;
}

.toast--success {
  border-left: 4px solid #10b981;
}

.toast--error {
  border-left: 4px solid #ef4444;
}

.toast--warning {
  border-left: 4px solid #f59e0b;
}

.toast--info {
  border-left: 4px solid #3b82f6;
}
```

Uso en Dashboard:

```
import { useToast } from '../components/ToastProvider'

export default function Dashboard() {
  const toast = useToast()

  const handleSave = async () => {
    try {
      await api.post('/api/productos', data)
      toast.notify('Producto creado exitosamente', { type: 'success' })
    } catch (error) {
      toast.notify('Error al crear producto', { type: 'error' })
    }
  }
}
```

Tipos de notificaciones:

```
toast.notify('Operación exitosa', { type: 'success' })
toast.notify('Error al guardar', { type: 'error' })
toast.notify('Ten cuidado', { type: 'warning' })
toast.notify('Información importante', { type: 'info' })
```

COMPONENTE: ThemeSwitch.jsx

Propósito: Switch animado para cambiar entre modo oscuro y claro

Código con Styled Components:

```
import React from 'react'
import styled from 'styled-components'

export default function ThemeSwitch({ checked, onChange }) {
  return (
    <StyledWrapper>
      <label className="switch">
        <input
          type="checkbox"
          checked={!!checked}
          onChange={(e) => onChange(e.target.checked)}
        />
        <div className="slider round">
          {/* Sol/Luna con nubes y estrellas */}
          <div className="sun-moon">
            {/* 3 manchas de luna (círculos grises) */}
            <svg id="moon-dot-1" className="moon-dot" viewBox="0 0 100 100">
              <circle cx={50} cy={50} r={50} />
            </svg>
            {/* ... más elementos */}

            {/* Rayos de sol */}
            <svg id="light-ray-1" className="light-ray" viewBox="0 0 100 100">
              <circle cx={50} cy={50} r={50} />
            </svg>

            {/* Nubes flotantes */}
            <svg id="cloud-1" className="cloud-dark" viewBox="0 0 100 100">
              <circle cx={50} cy={50} r={50} />
            </svg>
          </div>

          {/* Estrellas parpadeantes */}
          <div className="stars">
            <svg id="star-1" className="star" viewBox="0 0 20 20">
              <path d="M 0 10 C 10 10,10 10 ,0 10..." />
            </svg>
          </div>
        </div>
```

```
        </label>
      </StyledWrapper>
    )
  }

  const StyledWrapper = styled.div`
    .switch { ... }

    .sun-moon {
      background-color: yellow; /* Sol por defecto */
      transition: 0.4s;
    }

    input:checked + .slider {
      background-color: black; /* Cielo nocturno */
    }

    input:checked + .slider .sun-moon {
      transform: translateX(26px);
      background-color: white; /* Luna */
      animation: rotate-center 0.6s ease-in-out both;
    }

    .moon-dot {
      fill: gray;
      opacity: 0;
      transition: 0.4s;
    }

    input:checked + .slider .moon-dot {
      opacity: 1; /* Mostrar manchas de luna */
    }

    .stars {
      opacity: 0;
      transform: translateY(-32px);
    }

    input:checked + .slider .stars {
      opacity: 1;
      transform: translateY(0);
    }

    @keyframes star-twinkle {
      0%, 100% { transform: scale(1); }
      40% { transform: scale(1.2); }
      80% { transform: scale(0.8); }
    }

    @keyframes cloud-move {
      0%, 100% { transform: translateX(0px); }
      40% { transform: translateX(4px); }
      80% { transform: translateX(-4px); }
```

```
  }
`
```

Uso en Dashboard:

```
<ThemeSwitch
  checked={isDarkMode}
  onChange={() => setIsDarkMode(!isDarkMode)}
/>
```

Animaciones incluidas:

1. **Sol → Luna:** Transición suave con rotación
2. **Nubes:** Flotan de lado a lado
3. **Estrellas:** Aparecen y parpadean
4. **Manchas lunares:** Fade in al activar modo oscuro

🔍 COMPONENTE: SearchInput.jsx

Propósito: Input de búsqueda con animación de enfoque

Código:

```
import React from 'react'
import LottieIcon from './LottieIcon'

export default function SearchInput({ value, onChange, placeholder = 'Buscar...'
}) {
  return (
    <div className="search-input-wrapper">
      <LottieIcon name="search" size={20} color="#9ca3af" />
      <input
        type="text"
        value={value}
        onChange={(e) => onChange(e.target.value)}
        placeholder={placeholder}
        className="search-input"
      />
      {value && (
        <button
          className="search-clear"
          onClick={() => onChange('')}
        >
          ×
        </button>
      )}
    </div>
```

```
  )
}
```

CSS (searchInput.css):

```
.search-input-wrapper {
  position: relative;
  display: flex;
  align-items: center;
  background: var(--color-bg-secondary);
  border-radius: 12px;
  padding: 10px 16px;
  border: 2px solid transparent;
  transition: all 0.3s ease;
}

.search-input-wrapper:focus-within {
  border-color: var(--color-primary);
  box-shadow: 0 0 0 3px rgba(249, 115, 22, 0.1);
}

.search-input {
  flex: 1;
  border: none;
  background: none;
  margin-left: 12px;
  font-size: 14px;
  color: var(--color-text);
}

.search-clear {
  background: none;
  border: none;
  color: #9ca3af;
  cursor: pointer;
  padding: 4px 8px;
  font-size: 18px;
}

.search-clear:hover {
  color: var(--color-primary);
}
```

Uso:

```
const [searchQuery, setSearchQuery] = useState('')

<SearchInput
  value={searchQuery}
```

```
  onChange={setSearchQuery}
  placeholder="Buscar productos..."
/>
```

COMPONENTE: LottieIcon.jsx

Propósito: Iconos SVG con animaciones CSS

Código (parcial - 30+ iconos):

```
export const LottieIcon = ({ name, size = 24, color = '#f97316', style = {} }) =>
{
  return (
    <div
      className="animated-icon-container"
      style={{ width: size, height: size, ...style }}
    >
      <SVGIcon name={name} size={size} color={color} />
    </div>
  )
}

const SVGIcon = ({ name, size, color }) => {
  const icons = {
    dashboard: (
      <svg className="icon-dashboard" width={size} height={size} viewBox="0 0 24
24" fill="none" stroke={color}>
        <rect x="3" y="3" width="7" height="9" rx="1" />
        <rect x="14" y="3" width="7" height="5" rx="1" />
        <rect x="14" y="12" width="7" height="9" rx="1" />
        <rect x="3" y="16" width="7" height="5" rx="1" />
      </svg>
    ),

    products: (
      <svg className="icon-products" width={size} height={size} viewBox="0 0 24
24">
        <path d="M21 16V8a2 2 0 0 0-1-1.73l-7-4a2 2 0 0 0-2 0l-7 4A2 2 0 0 0 3
8v8..." />
        <polyline points="3.27 6.96 12 12.01 20.73 6.96" />
        <line x1="12" y1="22.08" x2="12" y2="12" />
      </svg>
    ),

    // ... 28 iconos más
  }

  return icons[name] || null
}
```

Animaciones CSS (lottieIcons.css):

```
.icon-dashboard {
  animation: dashboard-pulse 3s ease-in-out infinite;
}

@keyframes dashboard-pulse {
  0%, 100% { transform: scale(1); }
  50% { transform: scale(1.05); }
}

.icon-products {
  animation: products-rotate 4s linear infinite;
}

@keyframes products-rotate {
  0% { transform: rotateY(0deg); }
  100% { transform: rotateY(360deg); }
}

.icon-sales {
  animation: sales-shake 2s ease-in-out infinite;
}

@keyframes sales-shake {
  0%, 100% { transform: translateX(0); }
  25% { transform: translateX(-2px); }
  75% { transform: translateX(2px); }
}
```

Iconos disponibles:

- dashboard, products, categories, suppliers, sales
- inventory, reports, settings, logout, menu, close
- search, filter, edit, delete, add
- pdf, excel, phone, email, location
- calendar, briefcase, user, shield, lock, key
- info, checkCircle, crown

📋 MODAL: ProductModal.jsx

Propósito: Crear/Editar productos

Estructura:

```
export default function ProductModal({
  open,
  onClose,
  product = null,  // null = crear, objeto = editar
```

```
  categories,
  suppliers,
  onSave
}) {
  const [formData, setFormData] = useState({
    codigo: '',
    nombre: '',
    descripcion: '',
    precioUnitario: '',
    stock: '',
    stockMinimo: '',
    idCategoria: '',
    idProveedor: ''
  })

  const [errors, setErrors] = useState({})
  const [loading, setLoading] = useState(false)

  useEffect(() => {
    if (product) {
      // Modo edición - prellenar formulario
      setFormData({
        codigo: product.codigo,
        nombre: product.nombre,
        // ... más campos
      })
    } else {
      // Modo creación - limpiar formulario
      setFormData({
        codigo: '',
        nombre: '',
        // ...
      })
    }
  }, [product, open])

  const handleSubmit = async (e) => {
    e.preventDefault()

    // Validar
    const newErrors = {}
    if (!formData.nombre) newErrors.nombre = 'Nombre requerido'
    if (formData.precioUnitario <= 0) newErrors.precioUnitario = 'Precio debe ser
mayor a 0'

    if (Object.keys(newErrors).length > 0) {
      setErrors(newErrors)
      return
    }

    setLoading(true)

    try {
      if (product) {
```

```
        // Editar
        await apiFetch(`/api/productos/${product.idProducto}`, {
          method: 'PUT',
          body: JSON.stringify(formData)
        })
      } else {
        // Crear
        await apiFetch('/api/productos', {
          method: 'POST',
          body: JSON.stringify(formData)
        })
      }

      onSave()  // Callback para recargar lista
      onClose()
    } catch (error) {
      toast.error('Error al guardar producto')
    } finally {
      setLoading(false)
    }
  }

  if (!open) return null

  return (
    <AnimatePresence>
      <motion.div
        className="modal-backdrop"
        initial={{ opacity: 0 }}
        animate={{ opacity: 1 }}
        exit={{ opacity: 0 }}
        onClick={onClose}
      >
        <motion.div
          className="modal-content"
          initial={{ scale: 0.9, opacity: 0 }}
          animate={{ scale: 1, opacity: 1 }}
          exit={{ scale: 0.9, opacity: 0 }}
          onClick={(e) => e.stopPropagation()}
        >
          <div className="modal-header">
            <h2>{product ? 'Editar Producto' : 'Nuevo Producto'}</h2>
            <button className="modal-close" onClick={onClose}>✕</button>
          </div>

          <form onSubmit={handleSubmit} className="modal-form">
            <div className="form-row">
              <div className="form-group">
                <label>Código *</label>
                <input
                  type="text"
                  value={formData.codigo}
                  onChange={(e) => setFormData({...formData, codigo:
e.target.value})}
```

```
                  className={errors.codigo ? 'error' : ''}
                />
                {errors.codigo && <span className="error-message">{errors.codigo}
</span>}
              </div>

              <div className="form-group">
                <label>Nombre *</label>
                <input
                  type="text"
                  value={formData.nombre}
                  onChange={(e) => setFormData({...formData, nombre:
e.target.value})}
                />
              </div>
            </div>

            {/* ... más campos */}

            <div className="modal-footer">
              <button type="button" onClick={onClose} className="btn-secondary">
                Cancelar
              </button>
              <button type="submit" className="btn-primary" disabled={loading}>
                {loading ? 'Guardando...' : 'Guardar'}
              </button>
            </div>
          </form>
        </motion.div>
      </motion.div>
    </AnimatePresence>
  )
}
```

Uso en Dashboard:

```
const [showProductModal, setShowProductModal] = useState(false)
const [selectedProduct, setSelectedProduct] = useState(null)

// Crear
<button onClick={() => {
  setSelectedProduct(null)
  setShowProductModal(true)
}}>
  Nuevo Producto
</button>

// Editar
<button onClick={() => {
  setSelectedProduct(product)
  setShowProductModal(true)
}}>
```

```
  Editar
</button>

// Modal
<ProductModal
  open={showProductModal}
  onClose={() => setShowProductModal(false)}
  product={selectedProduct}
  categories={categories}
  suppliers={suppliers}
  onSave={() => fetchProducts()}
/>
```

🛒 MODAL: SaleModal.jsx (Complejo)

Propósito: Registrar venta con múltiples productos

Características:

- Agregar múltiples productos a un carrito
- Calcular subtotales y total automáticamente
- Validar stock disponible
- Seleccionar método de pago

Código (simplificado):

```
export default function SaleModal({ open, onClose, products, onSave }) {
  const [cart, setCart] = useState([])
  const [tipoPago, setTipoPago] = useState('Efectivo')
  const [observaciones, setObservaciones] = useState('')

  const addToCart = (product) => {
    const existing = cart.find(item => item.idProducto === product.idProducto)

    if (existing) {
      // Incrementar cantidad
      setCart(cart.map(item =>
        item.idProducto === product.idProducto
          ? { ...item, cantidad: item.cantidad + 1 }
          : item
      ))
    } else {
      // Agregar nuevo
      setCart([...cart, {
        idProducto: product.idProducto,
        nombre: product.nombre,
        precioUnitario: product.precioUnitario,
        cantidad: 1,
        stock: product.stock
      }])
```

```
    }
  }

  const removeFromCart = (idProducto) => {
    setCart(cart.filter(item => item.idProducto !== idProducto))
  }

  const updateQuantity = (idProducto, newQuantity) => {
    if (newQuantity <= 0) {
      removeFromCart(idProducto)
      return
    }

    setCart(cart.map(item =>
      item.idProducto === idProducto
        ? { ...item, cantidad: Math.min(newQuantity, item.stock) }
        : item
    ))
  }

  const calculateTotal = () => {
    return cart.reduce((sum, item) => sum + (item.cantidad * item.precioUnitario),
0)
  }

  const handleSubmit = async () => {
    if (cart.length === 0) {
      toast.error('Agrega al menos un producto')
      return
    }

    const saleData = {
      productos: cart.map(item => ({
        idProducto: item.idProducto,
        cantidad: item.cantidad,
        precioUnitario: item.precioUnitario
      })),
      tipoPago,
      observaciones
    }

    try {
      await apiFetch('/api/ventas', {
        method: 'POST',
        body: JSON.stringify(saleData)
      })

      toast.notify('Venta registrada exitosamente', { type: 'success' })
      setCart([])
      onSave()
      onClose()
    } catch (error) {
      toast.error('Error al registrar venta')
    }
```

```
  }

  return (
    <div className="modal">
      <div className="sale-modal-content">
        <h2>Registrar Venta</h2>

        {/* Buscador de productos */}
        <SearchInput
          placeholder="Buscar producto para agregar..."
          onChange={(q) => {/* filtrar products */}}
        />

        {/* Lista de productos filtrados */}
        <div className="products-list">
          {products.map(product => (
            <div key={product.idProducto} className="product-item">
              <span>{product.nombre}</span>
              <span>S/ {product.precioUnitario}</span>
              <span>Stock: {product.stock}</span>
              <button onClick={() => addToCart(product)}>+</button>
            </div>
          ))}
        </div>

        {/* Carrito de compra */}
        <div className="cart">
          <h3>Carrito ({cart.length} items)</h3>
          {cart.map(item => (
            <div key={item.idProducto} className="cart-item">
              <span>{item.nombre}</span>
              <input
                type="number"
                value={item.cantidad}
                onChange={(e) => updateQuantity(item.idProducto,
parseInt(e.target.value))}
                min="1"
                max={item.stock}
              />
              <span>S/ {(item.cantidad * item.precioUnitario).toFixed(2)}</span>
              <button onClick={() => removeFromCart(item.idProducto)}>×</button>
            </div>
          ))}

          <div className="cart-total">
            <strong>TOTAL: S/ {calculateTotal().toFixed(2)}</strong>
          </div>
        </div>

        {/* Método de pago */}
        <select value={tipoPago} onChange={(e) => setTipoPago(e.target.value)}>
          <option>Efectivo</option>
          <option>Tarjeta</option>
          <option>Transferencia</option>
```

```
            <option>Yape/Plin</option>
          </select>

          {/* Observaciones */}
          <textarea
            placeholder="Observaciones (opcional)"
            value={observaciones}
            onChange={(e) => setObservaciones(e.target.value)}
          />

          <button onClick={handleSubmit} className="btn-primary">
            Registrar Venta
          </button>
        </div>
      </div>
    )
  }
```

Siguiente documento: 05-GRAFICOS-REPORTES.md