Documento de Diseño y Funcionalidad: Panel de Gestión para Kiosco y Librería

Este documento detalla la interfaz de usuario (UI), la experiencia de usuario (UX) y la lógica de funcionamiento interno (Data y Flujos) del MVP diseñado para ser rápido, intuitivo y sin fricciones.

1. Disposición General de la Pantalla (Layout)

La aplicación está diseñada para usarse en una computadora de escritorio o notebook en el mostrador. Se divide en dos grandes áreas:

- **Barra Lateral (Menú de Navegación - Izquierda):**
  - **Ancho:** Ocupa aproximadamente un 20% de la pantalla.
  - **Estilo:** Fondo oscuro (azul marino o gris oscuro) con texto blanco para no cansar la vista.
  - **Contenido:** * Título grande arriba: "Mi Librería" junto a un ícono de un libro.
    - 4 Botones grandes de navegación: "Precios", "Libreta (Fiados)", "Fotocopias" y "Encargos Libros".
    - El botón activo se ilumina en un color azul brillante para indicar en qué sección se encuentra el usuario.
- **Área de Trabajo Principal (Centro/Derecha):**
  - **Ancho:** Ocupa el 80% restante de la pantalla.
  - **Estilo:** Fondo gris muy clarito con los elementos de contenido (tablas, tarjetas) en cajas de color blanco puro con sombras suaves.

2. Detalle de Vistas y Flujos Funcionales (El "Cómo Funciona")

Módulo A: Buscador de Precios (La pantalla principal)

Visual y UX:

- **Barra de Búsqueda:** Grande, central, con autofocus. Diseñada para usarse con pistola lectora de códigos de barras o teclado.
- **Tabla de Resultados:** Muestra ISBN, Descripción, Categoría y Precio Público (destacado en verde).
- **Gestor de Categorías:** Un panel oculto que se abre con un botón para editar los porcentajes de ganancia.

Lógica de Funcionamiento (Backend/Datos):

1. **El Motor de Búsqueda:** * La búsqueda debe ser "en tiempo real" (a medida que escribe) y *case-insensitive* (no distingue mayúsculas de minúsculas).
    - Debe buscar coincidencias tanto en el campo `ISBN` como en el campo `Descripción`.
2. **Cálculo Automático de Precio Público:**
    - El sistema **nunca** guarda el precio final en la base de datos. Solo guarda el `Costo Base` y el `ID de la Categoría`.
    - **La Fórmula:** `Precio Final = Costo Base + (Costo Base * (Porcentaje de la Categoría / 100))`
    - *Ejemplo:* Si el cuaderno cuesta $1000 (Costo Base) y la categoría "Librería" tiene 45%, el sistema calcula en el momento: `1000 + (1000 * 0.45) = $1450`.
    - **Ventaja Estructural:** Si la dueña cambia el porcentaje de "Librería" a 50%, todos los productos de esa categoría actualizarán su precio de venta automáticamente en el buscador sin tocar los productos uno por uno.
3. **Actualización de Costos:**
    - Al hacer clic en "Cambiar Costo" en un producto, se lanza un prompt. Al ingresar el nuevo costo, se sobrescribe el `Costo Base` en la base de datos y la vista se refresca calculando el nuevo precio final al instante.

Módulo B: Libreta (Fiados)

Visual y UX:

- **Pantalla 1 (Directorio):** Cuadrícula de tarjetas con nombres y deudas totales (Rojo = debe, Verde = saldo a favor/cero).
- **Pantalla 2 (Perfil):** Botones gigantes rojo (+) y verde (-) para anotar cargos o pagos, y un historial detallado debajo.

Lógica de Funcionamiento (Backend/Datos):

1. **La Estructura de la Deuda (Ledger):**
    - El "Saldo Total" de un cliente no es un número estático que se edita a mano. Es el resultado matemático de todas sus transacciones.
    - La base de datos guarda un historial (Array/Tabla de transacciones) para cada cliente. Cada transacción tiene: `Fecha`, `Descripción`, `Monto` y `Tipo` (Cargo o Pago).
2. **Flujo de "Anotar Fiado" (Cargo):**
    - El sistema pide "Descripción" y "Monto".

- Registra una nueva transacción de tipo "Cargo" y SUMA ese monto al "Saldo Total" del cliente.

3. **Flujo de "Registrar Pago" (Abono):**
   - El sistema pide el "Monto".
   - Registra una transacción de tipo "Pago" y RESTA ese monto al "Saldo Total".
4. **Cálculos seguros:** Si un cliente debe $1000 y paga $1500, el Saldo Total pasará a ser -$500 (Saldo a favor), mostrado en color verde.

Módulo C: Tablero de Fotocopias (Kanban)

Visual y UX:

- 3 Columnas: Pendiente de Hacer (Naranja), Listo/Esperando Retiro (Azul), Entregados (Verde).
- Tarjetas con detalles del trabajo y botones para mover de estado.

Lógica de Funcionamiento (Backend/Datos):

1. **Máquina de Estados (State Machine):**
   - Cada pedido de fotocopia nace con el estado predeterminado: `estado = "pendiente"` .
   - La interfaz grafica filtra los pedidos según su estado para ubicarlos en la columna correcta.
2. **Transiciones:**
   - Al hacer clic en "¡Listo para retirar!", el sistema busca el ID de ese pedido y actualiza en la base de datos: `estado = "listo"` . La interfaz reacciona moviendo la tarjeta a la columna del medio.
   - Al hacer clic en "Entregado", se actualiza a `estado = "entregado"` .
3. **Limpieza Automática (Opcional pero recomendada):**
   - Para no saturar el sistema, los pedidos con estado "entregado" podrían ocultarse automáticamente después de 48 horas o pasar a una pestaña de "Historial de Archivo".

Módulo D: Encargos de Libros

Visual y UX:

- Tabla con Libro, Cliente, Seña, Selector de Estado, y un Botón Mágico de WhatsApp.

Lógica de Funcionamiento (Backend/Datos):

1. **Manejo de Señas:**

- Es un campo numérico simple. Sirve únicamente como referencia visual para saber que el cliente ya adelantó dinero y no cobrarle el total al momento de la entrega.

2. **Selector de Estado:**
   - Al cambiar el valor en el menú desplegable (Ej. de "En espera" a "En el local"), la base de datos se actualiza instantáneamente (`onChange` event).
3. **El Botón Mágico de WhatsApp (Deep Linking):**
   - Este botón tiene una lógica condicional: Solo aparece (`renderiza`) si el estado del pedido es exactamente `"en_local"` .
   - **El motor del enlace:** Al hacer clic, el sistema toma el número de teléfono del cliente (ej: `1123456789`) y el nombre del libro.
   - Limpia el número quitando espacios o guiones.
   - Genera una URL dinámica usando la API pública de WhatsApp: `https://wa.me/549[NUMERO_LIMPIO]?text=[MENSAJE_CODIFICADO]`
   - *Ejemplo de mensaje codificado:* "¡Hola! Te aviso de la librería que ya llegó el libro que encargaste: [TITULO_DEL_LIBRO]."
   - Abre esa URL en una pestaña nueva, disparando WhatsApp Web o la app de escritorio automáticamente.

3. Arquitectura de Datos Sugerida (Para el Programador)

Si esto se construye con una base de datos (como Firebase, Supabase o un JSON local), la estructura debería ser relacional o no-SQL estructurada de la siguiente forma:

- **Tabla** `Categorias` **:** `{ id, nombre, margen_porcentaje }`
- **Tabla** `Productos` **:** `{ id, isbn, descripcion, costo_base, categoria_id }`
- **Tabla** `Clientes` **:** `{ id, nombre, telefono, saldo_total }`
- **Tabla** `Transacciones_Fiados` **:** `{ id, cliente_id, fecha, detalle, tipo_operacion, monto }`
- **Tabla** `Trabajos_Fotocopiadora` **:** `{ id, solicitante, descripcion_material, telefono, estado_actual }`
- **Tabla** `Pedidos_Libros` **:** `{ id, titulo_isbn, nombre_cliente, telefono, monto_senia, estado_pedido }`

4. Consideraciones Técnicas Cruciales

1. **Cero Pantallas de Carga:** El frontend debe mantener el estado en memoria (React State, por ejemplo) para que al navegar por el menú lateral no haya recargas de página.

2. **Redondeo de Precios:** En la fórmula del "Buscador de Precios", el resultado matemático puede dar decimales (ej. $1450.75). Se debe implementar una función matemática que redondee siempre al número entero superior más cercano (o usar la regla de redondeo comercial estándar) para facilitar el cambio físico en la caja registradora.
3. **Prevención de Errores Humanos:** En todos los `prompts` o campos donde se ingresa dinero (Cambiar Costo, Anotar Fiado), el sistema debe validar obligatoriamente que la entrada sea un **número válido** antes de guardar, ignorando letras o símbolos accidentales.