💼 Mejoras de Negocio y UX - Mueblería La Económica

Fecha: 11 de Octubre, 2025
Enfoque: Funcionalidades de negocio y experiencia de usuario

🎯 Mejoras de Funcionalidad de Negocio

1. Dashboard con KPIs Mejorados

Métricas Actuales (Supuestas)

- Total clientes
- Total cobrado hoy
- Clientes visitados

Métricas Adicionales Recomendadas

```typescript
interface DashboardKPIs {
  // Financieras
  cobranzaDelDia: number;
  cobranzaSemanal: number;
  cobranzaMensual: number;
  metaDiaria: number;
  porcentajeMeta: number;

  // Operacionales
  clientesPendientes: number;
  clientesVisitados: number;
  clientesMorosos: number;
  rutasCompletadas: number;

  // Comparativas
  cobranzaVsDiaAnterior: number;
  cobranzaVsSemanaAnterior: number;
  tendenciaCobranza: 'up' | 'down' | 'stable';

  // Alertas
  clientesConMora3Dias: number;
  clientesConMora7Dias: number;
  clientesSinContacto15Dias: number;
}
```

Visualización Sugerida

- **Gráfica de tendencias** (últimos 30 días)
- **Comparativa por cobrador** (top performers)
- **Mapa de calor** (días de la semana con más cobranza)
- **Embudo de conversión** (visitados → pagaron)

---

2. Sistema de Notificaciones Inteligentes

Notificaciones para Cobradores

```
interface Notificacion {
  tipo: 'cliente_moroso' | 'ruta_pendiente' | 'meta_alcanzada';
  prioridad: 'alta' | 'media' | 'baja';
  mensaje: string;
  accion?: {
    texto: string;
    url: string;
  };
}

// Ejemplos
const notificaciones = [
  {
    tipo: 'cliente_moroso',
    prioridad: 'alta',
    mensaje: 'Juan Pérez tiene 3 días de mora',
    accion: {
      texto: 'Ver cliente',
      url: '/clientes/123'
    }
  },
  {
    tipo: 'ruta_pendiente',
    prioridad: 'media',
    mensaje: 'Tienes 12 clientes pendientes hoy',
    accion: {
      texto: 'Ver ruta',
      url: '/rutas/hoy'
    }
  }
];
```

Notificaciones Push (PWA)

```
// lib/notifications.ts
export async function requestNotificationPermission() {
  if ('Notification' in window) {
    const permission = await Notification.requestPermission();
    return permission === 'granted';
  }
  return false;
}

export async function sendPushNotification(
  title: string,
  body: string,
  url: string
) {
  if ('serviceWorker' in navigator) {
    const registration = await navigator.serviceWorker.ready;
    registration.showNotification(title, {
      body,
      icon: '/icon-192.png',
      badge: '/badge-72.png',
      data: { url },
      actions: [
        { action: 'open', title: 'Ver' },
        { action: 'close', title: 'Cerrar' }
      ]
    });
  }
}
```

Notificaciones por WhatsApp (Integración)

```
// Para enviar recordatorios de pago a clientes
interface WhatsAppMessage {
  to: string; // Número del cliente
  message: string;
  template?: 'recordatorio_pago' | 'confirmacion_pago';
}

async function enviarRecordatorioPago(clienteId: string) {
  const cliente = await prisma.cliente.findUnique({
    where: { id: clienteId }
  });

  await sendWhatsApp({
    to: cliente.telefono,
    template: 'recordatorio_pago',
    variables: {
      nombre: cliente.nombreCompleto,
      monto: cliente.montoPago,
      diaPago: getDiaNombre(cliente.diaPago)
    }
  });
}
```

---

3. Reportes Avanzados

Reporte de Cartera Vencida

```
interface ReporteCarteraVencida {
  totalClientes: number;
  montoTotal: Decimal;

  // Segmentado por antigüedad
  mora1a7dias: { clientes: number; monto: Decimal };
  mora8a15dias: { clientes: number; monto: Decimal };
  mora16a30dias: { clientes: number; monto: Decimal };
  moraMas30dias: { clientes: number; monto: Decimal };

  // Por cobrador
  porCobrador: Array<{
    cobrador: string;
    clientesMorosos: number;
    montoVencido: Decimal;
  }>;

  // Clientes críticos
  clientesCriticos: Array<{
    id: string;
    nombre: string;
    diasMora: number;
    montoVencido: Decimal;
    ultimoContacto: Date;
  }>;
}
```

Reporte de Efectividad de Cobranza

```
interface ReporteEfectividad {
  periodo: { desde: Date; hasta: Date };

  // Por cobrador
  porCobrador: Array<{
    cobradorId: string;
    nombre: string;

    // Métricas
    clientesAsignados: number;
    clientesVisitados: number;
    clientesPagaron: number;

    // Ratios
    tasaVisita: number; // visitados/asignados
    tasaConversion: number; // pagaron/visitados
    tasaRecuperacion: number; // pagaron/asignados

    // Montos
    metaCobranza: Decimal;
    cobranzaReal: Decimal;
    cumplimientoMeta: number; // %

    // Promedio
    ticketPromedio: Decimal;
    tiempoPromedioVisita: number; // minutos
  }>;

  // Comparativa vs mes anterior
  comparativa: {
    mejorCobrador: string;
    mayorMejora: string;
    alertas: string[];
  };
}
```

Exportación de Reportes

```
// app/api/reportes/export/route.ts
export async function GET(request: NextRequest) {
  const { searchParams } = new URL(request.url);
  const format = searchParams.get('format'); // 'excel' | 'pdf' | 'csv'
  const tipo = searchParams.get('tipo'); // 'cartera' | 'efectividad' | 'cobranza'

  const data = await generateReport(tipo);

  if (format === 'excel') {
    const buffer = await generateExcel(data);
    return new Response(buffer, {
      headers: {
        'Content-Type': 'application/vnd.openxmlformats-officedocu-
ment.spreadsheetml.sheet',
        'Content-Disposition': `attachment; filename="reporte-${tipo}-${Date.now()}.xl
sx"`
      }
    });
  }

  // Similar para PDF y CSV
}
```

---

4. Gestión de Rutas Optimizada

Algoritmo de Optimización de Rutas

```typescript
interface PuntoRuta {
  clienteId: string;
  nombre: string;
  direccion: string;
  coordenadas: { lat: number; lng: number };
  prioridad: number; // 1-5
  tiempoEstimado: number; // minutos
}

async function optimizarRuta(
  cobradorId: string,
  fecha: Date
): Promise<PuntoRuta[]> {
  // 1. Obtener clientes del cobrador para ese día
  const clientes = await prisma.cliente.findMany({
    where: {
      cobradorAsignadoId: cobradorId,
      diaPago: getDiaPago(fecha),
      statusCuenta: 'activo'
    }
  });

  // 2. Geocodificar direcciones (si no están geocodificadas)
  const puntos = await geocodificarClientes(clientes);

  // 3. Aplicar algoritmo TSP (Traveling Salesman Problem)
  // Usando aproximación con nearest neighbor + 2-opt
  const rutaOptimizada = algoritmoTSP(puntos, {
    puntoInicio: { lat: 0, lng: 0 }, // Ubicación cobrador
    considerarPrioridad: true,
    considerarHorarios: true,
  });

  return rutaOptimizada;
}
```

Visualización en Mapa

```tsx
// components/mapa-ruta.tsx
import 'mapbox-gl/dist/mapbox-gl.css';
import mapboxgl from 'mapbox-gl';

export function MapaRuta({ puntos }: { puntos: PuntoRuta[] }) {
  useEffect(() => {
    const map = new mapboxgl.Map({
      container: 'map',
      style: 'mapbox://styles/mapbox/streets-v11',
      center: [puntos[0].coordenadas.lng, puntos[0].coordenadas.lat],
      zoom: 12
    });

    // Agregar marcadores
    puntos.forEach((punto, index) => {
      new mapboxgl.Marker({
        color: punto.prioridad > 3 ? '#ef4444' : '#3b82f6'
      })
        .setLngLat([punto.coordenadas.lng, punto.coordenadas.lat])
        .setPopup(
          new mapboxgl.Popup().setHTML(
            `<h3>${index + 1}. ${punto.nombre}</h3>
             <p>${punto.direccion}</p>
             <p>Tiempo est: ${punto.tiempoEstimado}min</p>`
          )
        )
        .addTo(map);
    });

    // Dibujar ruta
    map.addLayer({
      id: 'route',
      type: 'line',
      source: {
        type: 'geojson',
        data: {
          type: 'Feature',
          geometry: {
            type: 'LineString',
            coordinates: puntos.map(p => [p.coordenadas.lng, p.coordenadas.lat])
          }
        }
      },
      paint: {
        'line-color': '#3b82f6',
        'line-width': 4
      }
    });

  }, [puntos]);

  return <div id="map" className="w-full h-[600px]" />;
}
```

---

5. Sistema de Metas y Gamificación

Definición de Metas

```typescript
interface MetaCobrador {
  cobradorId: string;
  periodo: 'diaria' | 'semanal' | 'mensual';
  tipo: 'monto' | 'clientes' | 'tasa_recuperacion';
  objetivo: number;
  actual: number;
  progreso: number; // %
  recompensa?: {
    tipo: 'bonus' | 'badge' | 'puntos';
    valor: number;
  };
}

interface Logro {
  id: string;
  nombre: string;
  descripcion: string;
  icono: string;
  requisito: {
    metrica: string;
    valor: number;
  };
  obtenido: boolean;
  fechaObtencion?: Date;
}

// Ejemplos de logros
const logros: Logro[] = [
  {
    id: 'cobrador_100',
    nombre: 'Cobrador Maestro',
    descripcion: 'Alcanza 100% de tu meta mensual',
    icono: '🏆',
    requisito: { metrica: 'cumplimiento_meta', valor: 100 },
    obtenido: false
  },
  {
    id: 'racha_7',
    nombre: 'Racha de Fuego',
    descripcion: '7 días consecutivos cumpliendo meta',
    icono: '🔥',
    requisito: { metrica: 'dias_consecutivos', valor: 7 },
    obtenido: false
  },
  {
    id: 'visitante_pro',
    nombre: 'Visitante Profesional',
    descripcion: 'Visita 50 clientes en un día',
    icono: '🚀',
    requisito: { metrica: 'clientes_visitados_dia', valor: 50 },
    obtenido: false
  }
];
```

Tablero de Líderes

```tsx
interface LeaderboardEntry {
  posicion: number;
  cobradorId: string;
  nombre: string;
  avatar?: string;

  // Métricas
  metrica: number; // monto cobrado, clientes visitados, etc.
  progreso: number;

  // Badges
  logrosObtenidos: number;
  rachaActual: number;

  // Comparativa
  diferenciaPrimero: number;
  cambioSemanaAnterior: number; // +2, -1, etc.
}

// Component
export function Leaderboard({ periodo }: { periodo: 'dia' | 'semana' | 'mes' }) {
  const { data } = useSWR<LeaderboardEntry[]>(
    `/api/leaderboard?periodo=${periodo}`,
    fetcher
  );

  return (
    <div className="space-y-2">
      {data?.map((entry) => (
        <div key={entry.cobradorId} className="flex items-center gap-4 p-4 bg-white
rounded-lg">
          {/* Medalla para top 3 */}
          {entry.posicion <= 3 && (
            <span className="text-3xl">
              {entry.posicion === 1 ? '🥇' : entry.posicion === 2 ? '🥈' : '🥉'}
            </span>
          )}

          <div className="flex-1">
            <div className="flex items-center gap-2">
              <span className="font-bold">#{entry.posicion}</span>
              <span>{entry.nombre}</span>
              {entry.rachaActual > 3 && (
                <span className="text-orange-500">🔥 {entry.rachaActual}</span>
              )}
            </div>
            <div className="text-sm text-gray-500">
              ${entry.metrica.toLocaleString()} cobrado
            </div>
          </div>

          {entry.cambioSemanaAnterior !== 0 && (
            <div className={entry.cambioSemanaAnterior > 0 ? 'text-green-500' : 'text-
red-500'}>
              {entry.cambioSemanaAnterior > 0 ? '↑' : '↓'} {Math.abs(entry.cambioSeman
aAnterior)}
            </div>
          )}
        </div>
      ))}
    </div>
```

```
  );
}
```

---

6. Historial y Auditoría Completa

Log de Cambios

```
model AuditLog {
  id          String    @id @default(cuid())
  userId      String
  accion      String    // 'crear', 'actualizar', 'eliminar'
  entidad     String    // 'Cliente', 'Pago', 'User'
  entidadId   String
  cambios     Json      // { campo: { anterior: X, nuevo: Y } }
  ipAddress   String?
  userAgent   String?
  timestamp   DateTime  @default(now())

  user        User      @relation(fields: [userId], references: [id])

  @@index([entidad, entidadId])
  @@index([userId])
  @@index([timestamp])
}

// Middleware para capturar cambios
export async function logChange(
  userId: string,
  accion: string,
  entidad: string,
  entidadId: string,
  cambios: any,
  request: NextRequest
) {
  await prisma.auditLog.create({
    data: {
      userId,
      accion,
      entidad,
      entidadId,
      cambios,
      ipAddress: request.ip || 'unknown',
      userAgent: request.headers.get('user-agent') || 'unknown',
    }
  });
}
```

Vista de Historial

```
// app/dashboard/historial/page.tsx
export default function HistorialPage() {
  const { data: logs } = useSWR('/api/audit-logs');

  return (
    <div>
      <h1>Historial de Cambios</h1>
      <table>
        <thead>
          <tr>
            <th>Fecha</th>
            <th>Usuario</th>
            <th>Acción</th>
            <th>Entidad</th>
            <th>Cambios</th>
          </tr>
        </thead>
        <tbody>
          {logs?.map((log) => (
            <tr key={log.id}>
              <td>{formatDate(log.timestamp)}</td>
              <td>{log.user.name}</td>
              <td>{log.accion}</td>
              <td>{log.entidad} #{log.entidadId}</td>
              <td>
                <button onClick={() => showChanges(log.cambios)}>
                  Ver detalles
                </button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
}
```

🎨 Mejoras de UX/UI

1. Modo Offline Mejorado

Service Worker con Sincronización

```
// public/sw.js
const CACHE_NAME = 'muebleria-v1';
const urlsToCache = [
  '/',
  '/dashboard',
  '/clientes',
  '/pagos',
  '/offline.html'
];

// Estrategia: Network First, fallback to Cache
self.addEventListener('fetch', (event) => {
  event.respondWith(
    fetch(event.request)
      .then(response => {
        // Guardar en cache si es exitoso
        if (response.status === 200) {
          const responseClone = response.clone();
          caches.open(CACHE_NAME).then(cache => {
            cache.put(event.request, responseClone);
          });
        }
        return response;
      })
      .catch(() => {
        // Si falla, usar cache
        return caches.match(event.request)
          .then(response => response || caches.match('/offline.html'));
      })
  );
});

// Background Sync para pagos
self.addEventListener('sync', (event) => {
  if (event.tag === 'sync-pagos') {
    event.waitUntil(syncPagos());
  }
});

async function syncPagos() {
  const db = await openDB();
  const pagos = await db.getAll('pagos-pendientes');

  for (const pago of pagos) {
    try {
      await fetch('/api/pagos', {
        method: 'POST',
        body: JSON.stringify(pago)
      });
      await db.delete('pagos-pendientes', pago.localId);
    } catch (error) {
      console.error('Error syncing pago:', error);
    }
  }
}
```

2. Búsqueda Avanzada con Filtros

Componente de Búsqueda

```tsx
interface FiltrosCliente {
  search?: string;
  cobrador?: string;
  diaPago?: string;
  statusCuenta?: 'activo' | 'inactivo';
  periodicidad?: 'semanal' | 'quincenal' | 'mensual';
  rangoPago?: { min: number; max: number };
  rangoSaldo?: { min: number; max: number };
  conMora?: boolean;
}

export function BusquedaAvanzada() {
  const [filtros, setFiltros] = useState<FiltrosCliente>({});
  const [mostrarFiltros, setMostrarFiltros] = useState(false);

  return (
    <div className="space-y-4">
      {/* Búsqueda rápida */}
      <div className="relative">
        <Search className="absolute left-3 top-3 h-5 w-5 text-gray-400" />
        <input
          type="text"
          placeholder="Buscar por nombre, código o teléfono..."
          className="w-full pl-10 pr-4 py-3 border rounded-lg"
          onChange={(e) => setFiltros({ ...filtros, search: e.target.value })}
        />
      </div>

      {/* Botón filtros avanzados */}
      <button
        onClick={() => setMostrarFiltros(!mostrarFiltros)}
        className="flex items-center gap-2"
      >
        <Filter className="h-4 w-4" />
        Filtros avanzados
        {Object.keys(filtros).length > 1 && (
          <Badge>{Object.keys(filtros).length - 1}</Badge>
        )}
      </button>

      {/* Panel de filtros */}
      {mostrarFiltros && (
        <div className="grid grid-cols-2 gap-4 p-4 bg-gray-50 rounded-lg">
          <Select
            label="Cobrador"
            value={filtros.cobrador}
            onChange={(value) => setFiltros({ ...filtros, cobrador: value })}
          >
            {/* Opciones */}
          </Select>

          <Select
            label="Día de pago"
            value={filtros.diaPago}
            onChange={(value) => setFiltros({ ...filtros, diaPago: value })}
          >
            {/* Opciones */}
          </Select>

          <div className="col-span-2">
            <label>Rango de pago</label>
            <div className="flex gap-2">
```

```
              <input
                type="number"
                placeholder="Mínimo"
                onChange={(e) => setFiltros({
                  ...filtros,
                  rangoPago: { ...filtros.rangoPago, min: +e.target.value }
                })}
              />
              <input
                type="number"
                placeholder="Máximo"
                onChange={(e) => setFiltros({
                  ...filtros,
                  rangoPago: { ...filtros.rangoPago, max: +e.target.value }
                })}
              />
            </div>
          </div>

          <label className="flex items-center gap-2">
            <input
              type="checkbox"
              checked={filtros.conMora}
              onChange={(e) => setFiltros({ ...filtros, conMora: e.target.checked })}
            />
            Solo clientes con mora
          </label>
        </div>
      )}

      {/* Aplicar filtros */}
      <button
        onClick={() => aplicarFiltros(filtros)}
        className="w-full bg-blue-500 text-white py-2 rounded-lg"
      >
        Aplicar filtros
      </button>
    </div>
  );
}
```

3. Acciones Rápidas y Shortcuts

Comandos con Teclado

```
// hooks/use-shortcuts.ts
export function useKeyboardShortcuts() {
  useEffect(() => {
    const handleKeyPress = (e: KeyboardEvent) => {
      // Ctrl/Cmd + K = Búsqueda rápida
      if ((e.ctrlKey || e.metaKey) && e.key === 'k') {
        e.preventDefault();
        openSearchModal();
      }

      // Ctrl/Cmd + N = Nuevo cliente
      if ((e.ctrlKey || e.metaKey) && e.key === 'n') {
        e.preventDefault();
        router.push('/clientes/nuevo');
      }

      // Ctrl/Cmd + P = Registrar pago rápido
      if ((e.ctrlKey || e.metaKey) && e.key === 'p') {
        e.preventDefault();
        openPagoRapidoModal();
      }

      // / = Focus en búsqueda
      if (e.key === '/' && !isInputFocused()) {
        e.preventDefault();
        document.querySelector('input[type="search"]')?.focus();
      }
    };

    window.addEventListener('keydown', handleKeyPress);
    return () => window.removeEventListener('keydown', handleKeyPress);
  }, []);
}
```

Command Palette (Como VS Code)

```
// components/command-palette.tsx
export function CommandPalette() {
  const [open, setOpen] = useState(false);
  const [search, setSearch] = useState('');

  const comandos = [
    { id: 'nuevo-cliente', label: 'Nuevo cliente', icon: UserPlus, action: () => route
r.push('/clientes/nuevo') },
    { id: 'nuevo-pago', label: 'Registrar pago', icon: DollarSign, action: () => open-
PagoModal() },
    { id: 'ver-rutas', label: 'Ver rutas del día', icon: Map, action: () => router.pus
h('/rutas/hoy') },
    { id: 'reportes', label: 'Generar reporte', icon: FileText, action: () => router.p
ush('/reportes') },
    { id: 'perfil', label: 'Mi perfil', icon: User, action: () => router.push('/per-
fil') },
    { id: 'logout', label: 'Cerrar sesión', icon: LogOut, action: () => signOut() },
  ];

  const filteredComandos = comandos.filter(cmd =>
    cmd.label.toLowerCase().includes(search.toLowerCase())
  );

  return (
    <Dialog open={open} onOpenChange={setOpen}>
      <DialogContent className="p-0">
        <div className="border-b">
          <input
            type="text"
            placeholder="Escribe un comando..."
            className="w-full px-4 py-3 outline-none"
            value={search}
            onChange={(e) => setSearch(e.target.value)}
            autoFocus
          />
        </div>

        <div className="max-h-[400px] overflow-auto">
          {filteredComandos.map((comando) => (
            <button
              key={comando.id}
              onClick={() => {
                comando.action();
                setOpen(false);
              }}
              className="w-full flex items-center gap-3 px-4 py-3 hover:bg-gray-100"
            >
              <comando.icon className="h-5 w-5" />
              <span>{comando.label}</span>
            </button>
          ))}
        </div>

        <div className="border-t px-4 py-2 text-xs text-gray-500">
          Presiona <kbd>Esc</kbd> para cerrar
        </div>
      </DialogContent>
    </Dialog>
  );
}
```

4. Vista de Cliente Mejorada

Perfil de Cliente Completo

```tsx
// app/clientes/[id]/page.tsx
export default function ClientePage({ params }: { params: { id: string } }) {
  const { data: cliente } = useSWR(`/api/clientes/${params.id}`);
  const { data: pagos } = useSWR(`/api/pagos?clienteId=${params.id}`);
  const { data: motorarios } = useSWR(`/api/motorarios?clienteId=${params.id}`);

  return (
    <div className="grid grid-cols-3 gap-6">
      {/* Columna 1: Info básica */}
      <div className="col-span-2 space-y-6">
        {/* Header con foto y datos principales */}
        <Card>
          <div className="flex items-start gap-6">
            <Avatar size="xl">
              {cliente?.nombreCompleto?.charAt(0)}
            </Avatar>

            <div className="flex-1">
              <h1 className="text-2xl font-bold">{cliente?.nombreCompleto}</h1>
              <p className="text-gray-500">Código: {cliente?.codigoCliente}</p>

              <div className="flex gap-4 mt-4">
                <Badge status={cliente?.statusCuenta}>
                  {cliente?.statusCuenta}
                </Badge>
                {cliente?.diasMora > 0 && (
                  <Badge variant="destructive">
                    {cliente?.diasMora} días de mora
                  </Badge>
                )}
              </div>
            </div>

            <div className="text-right">
              <div className="text-3xl font-bold">
                ${cliente?.saldoActual}
              </div>
              <div className="text-sm text-gray-500">Saldo actual</div>
            </div>
          </div>
        </Card>

        {/* Tabs con historial */}
        <Tabs defaultValue="pagos">
          <TabsList>
            <TabsTrigger value="pagos">
              Historial de pagos ({pagos?.length})
            </TabsTrigger>
            <TabsTrigger value="motorarios">
              Motorarios ({motorarios?.length})
            </TabsTrigger>
            <TabsTrigger value="notas">
              Notas
            </TabsTrigger>
          </TabsList>

          <TabsContent value="pagos">
            <HistorialPagos pagos={pagos} />
          </TabsContent>

          <TabsContent value="motorarios">
            <HistorialMotorarios motorarios={motorarios} />
```

```
          </TabsContent>
        </Tabs>
      </div>

      {/* Columna 2: Sidebar con acciones */}
      <div className="space-y-6">
        {/* Acciones rápidas */}
        <Card>
          <h3 className="font-semibold mb-4">Acciones rápidas</h3>
          <div className="space-y-2">
            <Button fullWidth onClick={() => registrarPago(cliente)}>
              <DollarSign className="h-4 w-4 mr-2" />
              Registrar pago
            </Button>
            <Button fullWidth variant="outline" onClick={() => registrar-
Motarario(cliente)}>
              <AlertCircle className="h-4 w-4 mr-2" />
              Registrar motarario
            </Button>
            <Button fullWidth variant="outline" onClick={() => llamarCliente(cliente)}
>
              <Phone className="h-4 w-4 mr-2" />
              Llamar
            </Button>
            <Button fullWidth variant="outline" onClick={() => enviarWhats-
App(cliente)}>
              <MessageSquare className="h-4 w-4 mr-2" />
              WhatsApp
            </Button>
          </div>
        </Card>

        {/* Info de contacto */}
        <Card>
          <h3 className="font-semibold mb-4">Información</h3>
          <dl className="space-y-3">
            <div>
              <dt className="text-sm text-gray-500">Teléfono</dt>
              <dd className="font-medium">{cliente?.telefono}</dd>
            </div>
            <div>
              <dt className="text-sm text-gray-500">Dirección</dt>
              <dd className="font-medium">{cliente?.direccionCompleta}</dd>
              <button className="text-blue-500 text-sm mt-1">
                Ver en mapa →
              </button>
            </div>
            <div>
              <dt className="text-sm text-gray-500">Día de pago</dt>
              <dd className="font-medium">{getDiaNombre(cliente?.diaPago)}</dd>
            </div>
            <div>
              <dt className="text-sm text-gray-500">Cobrador asignado</dt>
              <dd className="font-medium">{cliente?.cobradorAsignado?.name}</dd>
            </div>
          </dl>
        </Card>

        {/* Timeline de actividad reciente */}
        <Card>
          <h3 className="font-semibold mb-4">Actividad reciente</h3>
          <Timeline items={cliente?.actividadReciente} />
        </Card>
```

```
      </div>
    </div>
  );
}
```

---

5. Tema Oscuro (Dark Mode)

```tsx
// components/theme-toggle.tsx
import { useTheme } from 'next-themes';

export function ThemeToggle() {
  const { theme, setTheme } = useTheme();

  return (
    <button
      onClick={() => setTheme(theme === 'dark' ? 'light' : 'dark')}
      className="p-2 rounded-lg hover:bg-gray-100 dark:hover:bg-gray-800"
    >
      {theme === 'dark' ? <Sun /> : <Moon />}
    </button>
  );
}

// tailwind.config.ts
module.exports = {
  darkMode: 'class',
  theme: {
    extend: {
      colors: {
        // Custom colors for dark mode
        dark: {
          bg: '#0f172a',
          card: '#1e293b',
          border: '#334155',
        }
      }
    }
  }
}
```

📊 Resumen de Prioridades

Impacto Alto + Esfuerzo Bajo (Quick Wins)

1. ✅ Dashboard con KPIs mejorados (2 días)
2. ✅ Notificaciones push (1 día)
3. ✅ Búsqueda avanzada (1 día)
4. ✅ Dark mode (4 horas)

Impacto Alto + Esfuerzo Medio

1. ✅ Sistema de reportes (1 semana)
2. ✅ Gestión de rutas optimizada (1 semana)

3. ✅ Vista de cliente mejorada (3 días)
4. ✅ Command palette (2 días)

Impacto Alto + Esfuerzo Alto

1. ✅ Sistema de metas y gamificación (2 semanas)
2. ✅ Modo offline mejorado (2 semanas)
3. ✅ Integración WhatsApp (1 semana)
4. ✅ Historial y auditoría (1 semana)

🎯 ROI Estimado

Mejoras de Negocio

- **KPIs mejorados:** +30% visibilidad de métricas clave
- **Rutas optimizadas:** -20% tiempo de traslado
- **Notificaciones:** +25% tasa de cobro
- **Gamificación:** +15% motivación del equipo

Mejoras de UX

- **Búsqueda avanzada:** -50% tiempo encontrar cliente
- **Command palette:** -40% clicks para acciones comunes
- **Vista de cliente:** +60% información útil visible
- **Modo offline:** 100% disponibilidad en campo

Timestamp: 20251011_092000_BUSINESS_UX_IMPROVEMENTS
Estado: ✅ Análisis completo - Listo para priorización