🔧 RESPALDO TÉCNICO - Sistema ERP Completo

📋 CONFIGURACIÓN DEL ENTORNO

Variables de Entorno (.env)

```
# Base de datos
DATABASE_URL="postgresql://user:password@localhost:5432/erp_db"

# NextAuth
NEXTAUTH_SECRET="your-secret-here"
NEXTAUTH_URL="http://localhost:3000"

# WhatsApp (Evolution API)
EVOLUTION_API_URL="http://localhost:8080"
EVOLUTION_INSTANCE_NAME="mi-instancia"
EVOLUTION_API_TOKEN="your-token"

# SMS (LabsMobile)
LABSMOBILE_USERNAME="your-username"
LABSMOBILE_TOKEN="your-token"

# n8n Webhook
N8N_WEBHOOK_URL="http://localhost:5678/webhook"
```

🗄️ ESQUEMA COMPLETO DE BASE DE DATOS

Prisma Schema Actual

```
generator client {
    provider = "prisma-client-js"
    binaryTargets = ["native", "linux-musl-arm64-openssl-3.0.x"]
    output = "/home/ubuntu/sistema_erp_completo/app/node_modules/.prisma/client"
}

datasource db {
    provider = "postgresql"
    url      = env("DATABASE_URL")
}

// Modelos implementados:
// - User (✅ Completo)
// - Cliente (✅ Completo)
// - Configuracion (✅ Completo)
// - Account/Session (✅ NextAuth)
// - AuditLog (🟡 Estructura base)
// - Producto/Categoria (🟡 Estructura base)
// - Venta/VentaItem (🟡 Estructura base)
// - Pago (🟡 Estructura base)
```

🔌 APIS IMPLEMENTADAS

Autenticación

- `POST /api/auth/signin` - Login
- `POST /api/auth/signup` - Registro
- `GET /api/auth/session` - Sesión actual

Clientes

- `GET /api/clientes` - Listar (✅)
- `POST /api/clientes` - Crear (✅)
- `PUT /api/clientes/[id]` - Actualizar (✅)
- `DELETE /api/clientes/[id]` - Eliminar (✅)
- `POST /api/clientes/import` - Importar CSV (✅)
- `GET /api/users/by-role` - Usuarios por rol (✅)

Cobranza Móvil

- `GET /api/cobranza-movil/clientes/search` - Búsqueda (✅)
- `POST /api/cobranza-movil/pagos` - Registro pagos (✅)
- `GET /api/cobranza-movil/sync` - Sincronización (✅)

Comunicaciones

- `POST /api/whatsapp/send` - WhatsApp (✅)
- `GET /api/whatsapp/status` - Estado WhatsApp (✅)
- `POST /api/sms/send` - SMS individual (✅)
- `POST /api/sms/bulk` - SMS masivo (✅)

🎨 COMPONENTES UI DISPONIBLES

Componentes Base (Shadcn/UI)

```
// Formularios
- Button, Input, Textarea
- Select, Checkbox, RadioGroup
- Form, Label, ErrorMessage

// Layout
- Card, Dialog, Sheet, Tabs
- Table, Accordion, Collapsible
- Separator, ScrollArea

// Navegación
- Breadcrumb, Pagination
- Command, DropdownMenu

// Feedback
- Toast, Alert, Progress
- Badge, Avatar, Skeleton

// Data Display
- DataTable, Chart, Calendar
```

Componentes Personalizados

```
// Navegación
- Sidebar (/components/navigation/sidebar.tsx)
- Header (/components/navigation/header.tsx)

// Clientes
- ClienteCard (/components/clientes/cliente-card.tsx)
- ClienteForm (/components/clientes/cliente-form.tsx)
- ClienteModal (/components/clientes/cliente-modal.tsx)
- ClienteImport (/components/clientes/cliente-import.tsx)

// Cobranza Móvil
- ClientSearch (/components/cobranza-movil/client-search.tsx)
- PaymentForm (/components/cobranza-movil/payment-form.tsx)
- SyncStatus (/components/cobranza-movil/sync-status.tsx)
- TicketPrint (/components/cobranza-movil/ticket-print.tsx)

// Comunicaciones
- WhatsAppSender (/components/comunicacion/whatsapp-sender.tsx)
- SMSSender (/components/comunicacion/sms-sender.tsx)
```

🔄 FUNCIONALIDADES OFFLINE

IndexedDB Structure

```
// Stores disponibles:
- clients: Información de clientes
- payments: Cola de pagos pendientes
- sync_queue: Operaciones para sincronizar
- app_state: Estado de la aplicación

// Funciones implementadas:
- saveToIndexedDB(store, data)
- getFromIndexedDB(store, key)
- syncWithServer()
- queueOperation(operation)
```

Service Worker

- Cache de recursos estáticos
- Cache de APIs críticas
- Background sync para pagos
- Notificaciones push (preparado)

🖨️ SISTEMA DE IMPRESIÓN

Bluetooth Thermal Printing

```
// Funciones disponibles:
- connectToPrinter()
- printReceipt(paymentData)
- formatESCPOS(content)
- checkPrinterStatus()

// Comandos ESC/POS implementados:
- Corte de papel
- Negrita y subrayado
- Alineación de texto
- Códigos QR básicos
```

📊 SISTEMA DE PERMISOS

Roles y Permisos

```
const RolePermissions = {
  SUPERADMIN: [
    'dashboard', 'clientes', 'productos', 'ventas',
    'cobranza', 'almacen', 'credito', 'configuracion',
    'usuarios', 'reportes'
  ],
  ADMIN: [
    'dashboard', 'clientes', 'productos', 'ventas',
    'cobranza', 'reportes'
  ],
  GESTOR: [
    'dashboard', 'clientes', 'cobranza'
  ],
  VENTAS: [
    'dashboard', 'clientes', 'productos', 'ventas'
  ],
  ANALISTA: [
    'dashboard', 'reportes', 'clientes'
  ],
  CLIENTE: [
    'dashboard'
  ]
};
```

🔐 SEGURIDAD IMPLEMENTADA

Autenticación

- Hash de passwords con bcrypt (salt rounds: 12)
- JWT tokens seguros
- Sesiones con expiración
- Protección CSRF

Validación

- Zod schemas para validación

- Sanitización de inputs
- Rate limiting básico
- SQL injection protection (Prisma)

Permisos

- Middleware de autenticación
- Validación de roles por ruta
- Protección de APIs sensibles

📱 PWA CONFIGURATION

Manifest.json

```
{
  "name": "Sistema ERP Completo",
  "short_name": "ERP Sistema",
  "description": "Sistema integral de gestión empresarial",
  "start_url": "/dashboard",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#3B82F6",
  "orientation": "portrait-primary",
  "categories": ["business", "finance", "productivity"]
}
```

Service Worker Features

- Caché de recursos críticos
- Offline fallback pages
- Background synchronization
- Push notifications (preparado)

🌐 INTEGRATIONS PREPARADAS

n8n Webhooks

```
// Endpoints preparados para n8n:
POST /api/webhooks/n8n/payment
POST /api/webhooks/n8n/client-update
POST /api/webhooks/n8n/communication

// Eventos que se envían:
- Nuevo pago registrado
- Cliente actualizado
- Mensaje enviado
```

Evolution API (WhatsApp)

```
// Configuración:
const evolutionConfig = {
  baseURL: process.env.EVOLUTION_API_URL,
  instanceName: process.env.EVOLUTION_INSTANCE_NAME,
  token: process.env.EVOLUTION_API_TOKEN
};

// Funciones implementadas:
- sendTextMessage(number, message)
- sendFileMessage(number, fileUrl, caption)
- getInstanceStatus()
- getContacts()
```

LabsMobile SMS

```
// Configuración:
const labsMobileConfig = {
  username: process.env.LABSMOBILE_USERNAME,
  token: process.env.LABSMOBILE_TOKEN,
  apiUrl: 'https://api.labsmobile.com/rest/sms'
};

// Funciones implementadas:
- sendSMS(number, message)
- sendBulkSMS(numbers, message)
- checkBalance()
- getDeliveryStatus(messageId)
```

🧪 TESTING Y DEBUGGING

Comandos de Testing

```
# Verificar compilación TypeScript
npx tsc --noEmit

# Ejecutar linting
yarn lint

# Verificar build de producción
yarn build

# Probar APIs con curl
curl -X GET http://localhost:3000/api/clientes
```

Debugging Tools

- Console.log habilitado en desarrollo
- Error boundaries implementados
- Toast notifications para feedback
- Network tab monitoring

📈 MÉTRICAS Y MONITOREO

Preparado para Analytics

```
// Eventos listos para tracking:
- User login/logout
- Client creation/update
- Payment processed
- Message sent
- Module accessed
```

Performance Monitoring

- Next.js built-in analytics preparado
- Core Web Vitals tracking
- API response time logging

🔄 BACKUP Y RECOVERY

Database Backup

```
# Crear respaldo manual
pg_dump $DATABASE_URL > backup_$(date +%Y%m%d_%H%M%S).sql

# Restaurar desde respaldo
psql $DATABASE_URL < backup_file.sql
```

Code Backup

- Git repository configurado
- .gitignore optimizado
- Branches para features recomendadas

🚀 DEPLOYMENT READY

Production Checklist

- [x] Environment variables configuradas
- [x] Database schema actualizado
- [x] Build process funcionando
- [x] Static assets optimizados
- [x] PWA manifest válido
- [x] Service worker registrado
- [x] Security headers básicos
- [x] Error handling robusto

Deployment Commands

```
# Build para producción
yarn build

# Start production server
yarn start

# Database setup
npx prisma migrate deploy
npx prisma generate
```

🎯 Sistema técnicamente sólido y listo para escalamiento