🛠️ Referencia de Desarrollo - SMS CloudMX

Estado actual del proyecto y guía para futuras mejoras

📊 Estado Actual del Proyecto

✅ Módulos Completamente Implementados

1. Sistema de Autenticación

- 📍 **Ubicación:** `/app/(auth)/`
- 📍 **API:** NextAuth.js configurado en `/api/auth/[...nextauth]/`
- 📍 **Base de datos:** Tablas `User`, `Account`, `Session`
- ✅ Login/Logout funcional
- ✅ Protección de rutas
- ✅ Middleware de autenticación

2. Módulo de Listas de Contactos ⭐

- 📍 **Frontend:** `/app/dashboard/lists/`
- 📍 **API:** `/app/api/dashboard/lists/`
- 📍 **Componentes:** `/app/dashboard/lists/_components/`
- 📍 **Base de datos:** `ContactList`, `Contact`, `ListMembership`

Funcionalidades implementadas:
- ✅ CRUD completo de listas
- ✅ Gestión de contactos por lista
- ✅ Sistema de suscripciones (opt-in/opt-out)
- ✅ Segmentación avanzada con filtros múltiples
- ✅ Estadísticas detalladas por lista
- ✅ Importación masiva de contactos
- ✅ Listas dinámicas vs estáticas
- ✅ UI completa con tablas, formularios y estadísticas

APIs implementadas:

```
GET/POST /api/dashboard/lists - Listar/crear listas
GET/PUT/DELETE /api/dashboard/lists/[id] - CRUD lista específica
POST /api/dashboard/lists/[id]/contacts - Gestionar contactos
POST /api/dashboard/lists/[id]/subscriptions - Gestionar suscripciones
POST /api/dashboard/lists/segment - Aplicar segmentación
GET /api/dashboard/lists/stats - Estadísticas globales
```

3. Módulo de Campañas SMS ⭐

- 📍 **Frontend:** `/app/dashboard/campaigns/`
- 📍 **API:** `/app/api/dashboard/campaigns/`
- 📍 **Componentes:** `/app/dashboard/campaigns/_components/`
- 📍 **Base de datos:** `Campaign`, `Message`

Funcionalidades implementadas:
- ✅ CRUD completo de campañas
- ✅ Creación de campañas con selección de listas
- ✅ Sistema de plantillas de mensajes
- ✅ Programación de envíos
- ✅ Preview de campañas antes del envío
- ✅ Seguimiento en tiempo real
- ✅ Estadísticas de entrega y respuesta
- ✅ Gestión de créditos SMS
- ✅ UI completa con formularios y dashboard

APIs implementadas:

```
GET/POST /api/dashboard/campaigns - Listar/crear campañas
GET/PUT/DELETE /api/dashboard/campaigns/[id] - CRUD campaña específica
POST /api/dashboard/campaigns/[id]/send - Enviar campaña
GET /api/dashboard/campaigns/[id]/stats - Estadísticas de campaña
```

4. Dashboard Principal

- 📍 **Ubicación:** `/app/dashboard/page.tsx`
- ✅ Estadísticas generales
- ✅ Gráficos de actividad reciente
- ✅ Cards con métricas clave
- ✅ Navegación completa

🚧 Módulos Pendientes por Implementar

1. Módulo de Contactos Individual 🔄

- 📍 **Ubicación sugerida:** `/app/dashboard/contacts/`
- 📍 **API sugerida:** `/app/api/dashboard/contacts/`
- 📍 **Estado actual:** Página básica existe (`/dashboard/contacts/page.tsx`) con mensaje “Coming Soon”

Funcionalidades por implementar:
- [ ] Vista de todos los contactos globalmente
- [ ] Perfiles individuales de contactos
- [ ] Historial de interacciones por contacto
- [ ] Importación/exportación de contactos
- [ ] Gestión de campos personalizados
- [ ] Blacklist/whitelist global
- [ ] Deduplicación automática
- [ ] Tags y categorías de contactos

APIs por implementar:

```
GET/POST /api/dashboard/contacts - Listar/crear contactos
GET/PUT/DELETE /api/dashboard/contacts/[id] - CRUD contacto específico
POST /api/dashboard/contacts/import - Importación masiva
GET /api/dashboard/contacts/[id]/history - Historial del contacto
POST /api/dashboard/contacts/deduplicate - Deduplicación
```

Punto de partida:

1. Expandir el modelo `Contact` en `schema.prisma` con campos adicionales
2. Crear componentes base en `/dashboard/contacts/_components/`
3. Implementar `contacts-table.tsx` similar a `lists-table.tsx`
4. Crear formularios para gestión individual

2. Módulo de Plantillas 📝

- 📍 **Ubicación sugerida:** `/app/dashboard/templates/`
- 📍 **API sugerida:** `/app/api/dashboard/templates/`
- 📍 **Estado actual:** Página básica existe con mensaje “Coming Soon”

Funcionalidades por implementar:

- [ ] Creación de plantillas de mensajes
- [ ] Sistema de variables dinámicas ({{nombre}}, {{empresa}}, etc.)
- [ ] Categorización de plantillas
- [ ] Plantillas compartidas vs privadas
- [ ] Preview con datos de ejemplo
- [ ] Biblioteca de plantillas predefinidas
- [ ] Versionado de plantillas

Modelo de base de datos sugerido:

```
model Template {
  id          String   @id @default(cuid())
  name        String
  content     String
  category    String?
  variables   Json?    // Array de variables disponibles
  isPublic    Boolean  @default(false)
  userId      String
  user        User     @relation(fields: [userId], references: [id])
  campaigns   Campaign[]
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt
}
```

3. Módulo de Analytics 📊

- 📍 **Ubicación sugerida:** `/app/dashboard/analytics/`
- 📍 **API sugerida:** `/app/api/dashboard/analytics/`
- 📍 **Estado actual:** Página básica existe con mensaje “Coming Soon”

Funcionalidades por implementar:

- [ ] Dashboard de métricas avanzadas
- [ ] Reportes personalizables
- [ ] Análisis de tendencias temporales
- [ ] Comparativas entre campañas
- [ ] Exportación de reportes (PDF, Excel)

- [ ] Métricas de ROI
- [ ] Análisis de audiencia
- [ ] Heatmaps de actividad

4. Módulo de Configuración ⚙️

- 📍 **Ubicación sugerida:** `/app/dashboard/settings/`
- 📍 **API sugerida:** `/app/api/dashboard/settings/`
- 📍 **Estado actual:** Página básica existe con mensaje “Coming Soon”

Funcionalidades por implementar:
- [ ] Configuración de cuenta de usuario
- [ ] Gestión de proveedores SMS
- [ ] Configuración de webhooks
- [ ] Personalización de marca
- [ ] Configuración de notificaciones
- [ ] Gestión de API keys
- [ ] Límites y cuotas
- [ ] Configuración de facturación

5. Sistema de Notificaciones 🔔

- 📍 **Estado actual:** No implementado

Funcionalidades por implementar:
- [ ] Notificaciones en tiempo real
- [ ] Centro de notificaciones
- [ ] Configuración de alertas
- [ ] Notificaciones por email
- [ ] Webhooks para integraciones externas

🏗️ Arquitectura y Patrones Utilizados

Stack Tecnológico

```
{
  "Frontend": "Next.js 14 + React 18 + TypeScript",
  "Backend": "Next.js API Routes",
  "Base de datos": "PostgreSQL + Prisma ORM",
  "Autenticación": "NextAuth.js",
  "UI": "Tailwind CSS + Shadcn/ui + Radix UI",
  "Estado": "Zustand + React Query",
  "Formularios": "React Hook Form + Zod",
  "Notificaciones": "React Hot Toast"
}
```

Estructura de Archivos Establecida

```
/app
  /dashboard
    /[modulo]/
      page.tsx                    # Página principal del módulo
      /new/page.tsx               # Página de creación
      /[id]/page.tsx              # Página de detalle
      /[id]/edit/page.tsx         # Página de edición
      /_components/               # Componentes específicos del módulo
        [modulo]-table.tsx        # Tabla principal
        [modulo]-form.tsx         # Formulario de creación/edición
        [modulo]-stats.tsx        # Componente de estadísticas
  /api
    /dashboard
      /[modulo]/
        route.ts                  # GET/POST para listar y crear
        /[id]/route.ts            # GET/PUT/DELETE para operaciones específicas
        /[id]/[accion]/route.ts # Acciones específicas (send, stats, etc.)
```

Patrones de Diseño Implementados

1. Patrón de Componentes Reutilizables

- `components/ui/` - Componentes base de Shadcn/ui
- `lib/utils.ts` - Utilidades compartidas
- Cada módulo tiene sus propios componentes en `_components/`

2. Patrón de API Consistente

```
// Estructura estándar de respuesta API
{
  success: boolean,
  data?: any,
  error?: string,
  meta?: {
    total: number,
    page: number,
    limit: number
  }
}
```

3. Patrón de Validación

- Zod schemas para validación de formularios
- Validación tanto en frontend como backend
- Mensajes de error consistentes

Convenciones de Código

Nomenclatura:

- Archivos: `kebab-case.tsx`
- Componentes: `PascalCase`
- Variables: `camelCase`
- Constantes: `UPPER_SNAKE_CASE`

Estructura de Componentes:

```
// Imports
import { ... } from 'react'
import { ... } from 'next'
// UI imports
import { Button } from '@/components/ui/button'
// Local imports
import { ... } from './components'

// Types
interface ComponentProps {
  // props definition
}

// Component
export default function ComponentName({ props }: ComponentProps) {
  // hooks
  // state
  // effects
  // handlers
  // render
}
```

🔧 Herramientas de Desarrollo

Scripts Disponibles

```
{
  "dev": "next dev",
  "build": "next build",
  "start": "next start",
  "lint": "next lint",
  "prisma:studio": "prisma studio",
  "prisma:generate": "prisma generate",
  "prisma:push": "prisma db push",
  "prisma:seed": "tsx --require dotenv/config scripts/seed.ts"
}
```

Base de Datos

- **ORM:** Prisma
- **Archivo principal:** `/prisma/schema.prisma`
- **Migraciones:** Se usa `prisma db push` para desarrollo
- **Seed:** Script en `/scripts/seed.ts` (pendiente implementar)

🚀 Guía para Continuar el Desarrollo

Para implementar un nuevo módulo:

1. **Crear estructura básica:**
   `bash`

```
   mkdir -p app/dashboard/[modulo]/_components
   mkdir -p app/api/dashboard/[modulo]
```

2. **Actualizar base de datos:**
   - Añadir modelo en `prisma/schema.prisma`
   - Ejecutar `npx prisma db push`

3. **Crear páginas base:**
   - `page.tsx` - Lista principal
   - `new/page.tsx` - Formulario creación
   - `[id]/page.tsx` - Vista detalle

4. **Implementar APIs:**
   - `route.ts` - GET/POST base
   - `[id]/route.ts` - GET/PUT/DELETE

5. **Crear componentes:**
   - `[modulo]-table.tsx` - Tabla con datos
   - `[modulo]-form.tsx` - Formularios
   - `[modulo]-stats.tsx` - Estadísticas

6. **Actualizar navegación:**
   - Añadir enlaces en `/components/sidebar.tsx`

Patrones a seguir:

API Response Pattern:

```
// Success response
return NextResponse.json({
  success: true,
  data: result,
  meta: { total, page, limit } // para listas paginadas
});

// Error response
return NextResponse.json(
  { success: false, error: "Error message" },
  { status: 400 }
);
```

Form Validation Pattern:

```
const formSchema = z.object({
  // field validations
});

const form = useForm<z.infer<typeof formSchema>>({
  resolver: zodResolver(formSchema),
  defaultValues: { /* defaults */ }
});
```

Table Component Pattern:

Seguir estructura de `lists-table.tsx`:
- Filtros en la parte superior

- Tabla con columnas ordenables
- Acciones por fila (ver, editar, eliminar)
- Paginación en la parte inferior

📚 Recursos y Referencias

Documentación Técnica:

- Next.js 14 Documentation (https://nextjs.org/docs)
- Prisma Documentation (https://www.prisma.io/docs)
- Shadcn/ui Components (https://ui.shadcn.com)
- NextAuth.js Guide (https://next-auth.js.org)

APIs y Servicios:

- **SMS Providers:** Twilio, MessageBird, AWS SNS
- **Email:** Resend, SendGrid, AWS SES
- **Storage:** AWS S3, Cloudinary
- **Analytics:** PostHog, Google Analytics

Herramientas Recomendadas:

- **Desarrollo:** VS Code + Prisma extension
- **Base de datos:** Prisma Studio, pgAdmin
- **Testing:** Jest + Testing Library (pendiente configurar)
- **Monitoring:** Sentry (pendiente integrar)

🎯 Próximos Pasos Recomendados

Prioridad Alta (1-2 semanas):

1. **Completar módulo de Contactos individuales**
2. **Implementar sistema de plantillas básico**
3. **Mejorar sistema de importación de contactos**
4. **Añadir tests unitarios básicos**

Prioridad Media (3-4 semanas):

1. **Módulo de Analytics básico**
2. **Sistema de notificaciones**
3. **Configuración avanzada de cuenta**
4. **Integración con proveedores SMS reales**

Prioridad Baja (1-2 meses):

1. **Sistema de facturación y créditos**
2. **API REST pública para integraciones**
3. **Webhooks y automatizaciones**
4. **Aplicación móvil**

🆘 Puntos de Atención

Limitaciones Actuales:

- ❗ No hay integración real con proveedores SMS (solo simulación)
- ❗ Sistema de créditos es básico (no hay facturación real)
- ❗ Falta sistema de logs y auditoría
- ❗ No hay rate limiting en las APIs
- ❗ Falta validación avanzada de números de teléfono

Consideraciones de Seguridad:

- ✅ Autenticación implementada
- ❗ Falta autorización granular por recursos
- ❗ No hay encriptación de datos sensibles
- ❗ Falta protección contra ataques de fuerza bruta

Consideraciones de Performance:

- ❗ Falta optimización de consultas Prisma
- ❗ No hay cache implementado
- ❗ Falta paginación en algunas listas
- ❗ No hay optimización de imágenes

📞 Contacto para Desarrollo

Si tienes preguntas sobre el código o arquitectura:

- 📧 **Email técnico:** dev@smscloudmx.com
- 📝 **Issues GitHub:** Crear issue en el repositorio
- 📖 **Wiki del proyecto:** [Pendiente crear]

Última actualización: $(date)
Versión del proyecto: v1.0.0
Desarrollado por: DeepAgent - Abacus.AI

¡Happy Coding! 🚀