💻 LA RESERVA - DESARROLLO WEB

PARTE 3-1: Base de Datos (Parte 1)

Versión: 1.0
Fecha: Octubre 2025
Tiempo de lectura: 15 minutos

📑 CONTENIDO

---

1. INTRODUCCIÓN A SUPABASE

¿Qué es Supabase?

Supabase es una alternativa open-source a Firebase que proporciona:

- **PostgreSQL Database** - Base de datos relacional completa
- **Authentication** - Sistema de auth out-of-the-box
- **Storage** - Almacenamiento de archivos
- **Edge Functions** - Funciones serverless
- **Real-time** - Subscripciones en tiempo real

¿Por qué Supabase para La Reserva?

✅ **PostgreSQL real** - Perfecto para datos relacionales (eventos → clientes)
✅ **Row Level Security (RLS)** - Seguridad granular por usuario
✅ **Free tier generoso** - 500MB DB, 1GB storage gratis
✅ **Auto-generación de APIs** - REST y GraphQL automáticos
✅ **Open source** - No vendor lock-in

---

2. CONFIGURACIÓN INICIAL

2.1 Crear Proyecto en Supabase

1. Ve a https://supabase.com
2. Click en "Start your project"
3. Crear nueva organización: "La Reserva"
4. Crear proyecto:
    - **Name:** la-reserva-prod
    - **Database Password:** [Generar password fuerte y guardarlo]
    - **Region:** São Paulo (más cercano a Lima)
    - **Pricing Plan:** Free

2.2 Obtener Credenciales

Una vez creado el proyecto:

1. Ve a **Settings** → **API**
2. Copia las credenciales:

```
# .env
SUPABASE_URL=https://tu-proyecto-id.supabase.co
SUPABASE_ANON_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
SUPABASE_SERVICE_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

⚠️ **IMPORTANTE:**

- `ANON_KEY` - Puede ser pública (cliente)
- `SERVICE_KEY` - NUNCA exponer al cliente (solo servidor)

2.3 Configurar Cliente en el Proyecto

```
// src/lib/supabase.ts
import { createClient } from '@supabase/supabase-js';
import type { Database } from '@/types/database';

const supabaseUrl = import.meta.env.SUPABASE_URL;
const supabaseAnonKey = import.meta.env.SUPABASE_ANON_KEY;

if (!supabaseUrl || !supabaseAnonKey) {
  throw new Error('Missing Supabase environment variables');
}

// Cliente para uso general (cliente y servidor)
export const supabase = createClient<Database>(supabaseUrl, supabaseAnonK

// Cliente con service key (SOLO para servidor)
export const supabaseAdmin = createClient<Database>(
  supabaseUrl,
  import.meta.env.SUPABASE_SERVICE_KEY || supabaseAnonKey,
  {
    auth: {
      autoRefreshToken: false,
      persistSession: false,
    },
  }
);
```

---

3. DISEÑO DE BASE DE DATOS

3.1 Diagrama ER Simplificado

```
┌─────────────┐       ┌─────────────┐       ┌─────────────┐
│   quotes    │       │   events    │       │   clients   │
├─────────────┤       ├─────────────┤       ├─────────────┤
│ id (PK)     │       │ id (PK)     │       │ id (PK)     │
│ client_name │       │ client_id   ──────▶ │ name        │
│ email       │       │ event_date  │       │ email       │
│ phone       │       │ guest_count │       │ phone       │
│ event_type  │       │ status      │       │ created_at  │
│ event_date  │       │ total_price │       └─────────────┘
│ guest_count │       │ created_at  │
│ status      │       └─────────────┘
│ created_at  │              │
└─────────────┘              │
                             ▼
                      ┌─────────────┐
                      │event_images │
                      ├─────────────┤
                      │ id (PK)     │
                      │ event_id(FK)│
```

```
                        │ image_url   │
                        │ caption     │
                        └─────────────┘


    ┌─────────────┐     ┌─────────────┐     ┌─────────────┐
    │  services   │     │  packages   │     │testimonials │
    ├─────────────┤     ├─────────────┤     ├─────────────┤
    │ id (PK)     │     │ id (PK)     │     │ id (PK)     │
    │ name        │     │ name        │     │ client_name │
    │ slug        │     │ slug        │     │ rating      │
    │ description │     │ price       │     │ comment     │
    │ price_from  │     │ guest_range │     │ approved    │
    │ features[]  │     │ features[]  │     │ created_at  │
    └─────────────┘     └─────────────┘     └─────────────┘
```

3.2 Tablas Principales

quotes - Cotizaciones solicitadas

- Formulario público del sitio
- Estado: new → contacted → quoted → converted/declined

events - Eventos confirmados

- Creados desde el panel admin
- Estado: pending → confirmed → completed/cancelled

clients - Base de datos de clientes

- Centraliza información de contacto
- Historial de eventos

services - Catálogo de servicios

- Público en el sitio
- Editable desde admin

packages - Paquetes predefinidos

- Combos de servicios
- Precios especiales

---

4. SCRIPTS SQL - CREACIÓN DE TABLAS

Ve a **SQL Editor** en Supabase Dashboard y ejecuta estos scripts:

4.1 Extensiones y Setup Inicial

```sql
-- ============================================
-- LA RESERVA - DATABASE SCHEMA
-- Versión: 1.0
-- ============================================

-- Enable UUID extension
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
```

4.2 Tabla: admin_users

```sql
-- ============================================
-- TABLE: admin_users
-- Usuarios administradores del sistema
-- ============================================
CREATE TABLE admin_users (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  email TEXT UNIQUE NOT NULL,
  role TEXT NOT NULL CHECK (role IN ('super_admin', 'admin')),
  full_name TEXT NOT NULL,
  avatar_url TEXT,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

COMMENT ON TABLE admin_users IS 'Usuarios administradores del panel';
COMMENT ON COLUMN admin_users.role IS 'super_admin: acceso completo, admi
```

4.3 Tabla: clients

```sql
-- ============================================
-- TABLE: clients
-- Base de datos de clientes
-- ============================================
CREATE TABLE clients (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  name TEXT NOT NULL,
  email TEXT NOT NULL,
  phone TEXT NOT NULL,
  company TEXT,
  notes TEXT,
  total_events INTEGER DEFAULT 0,
  total_spent DECIMAL(10,2) DEFAULT 0,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Indexes para búsqueda rápida
CREATE INDEX idx_clients_email ON clients(email);
CREATE INDEX idx_clients_phone ON clients(phone);
CREATE INDEX idx_clients_name ON clients(name);

COMMENT ON TABLE clients IS 'Base de datos centralizada de clientes';
```

4.4 Tabla: quotes

```
  -- ===============================================
  -- TABLE: quotes
  -- Cotizaciones solicitadas por clientes
  -- ===============================================
  CREATE TABLE quotes (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    client_id UUID REFERENCES clients(id) ON DELETE SET NULL,

    -- Información del cliente (duplicada para histórico)
    client_name TEXT NOT NULL,
    client_email TEXT NOT NULL,
    client_phone TEXT NOT NULL,

    -- Información del evento
    event_type TEXT NOT NULL,
    event_date DATE NOT NULL,
    guest_count INTEGER NOT NULL CHECK (guest_count >= 25 AND guest_count <
    message TEXT,

    -- Estado y seguimiento
    status TEXT NOT NULL DEFAULT 'new'
      CHECK (status IN ('new', 'contacted', 'quoted', 'converted', 'decline
    estimated_price DECIMAL(10,2),
    admin_notes TEXT,

    -- Timestamps
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    contacted_at TIMESTAMPTZ,
    converted_at TIMESTAMPTZ
  );

  -- Indexes para performance
  CREATE INDEX idx_quotes_status ON quotes(status);
  CREATE INDEX idx_quotes_date ON quotes(event_date);
  CREATE INDEX idx_quotes_created ON quotes(created_at DESC);
  CREATE INDEX idx_quotes_client ON quotes(client_id);

  COMMENT ON TABLE quotes IS 'Cotizaciones solicitadas desde el formulario
  COMMENT ON COLUMN quotes.status IS 'new: recién creada, contacted: admin
```

4.5 Tabla: events

```
-- ============================================
-- TABLE: events
-- Eventos confirmados
-- ============================================
CREATE TABLE events (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  client_id UUID REFERENCES clients(id) ON DELETE SET NULL,
  quote_id UUID REFERENCES quotes(id) ON DELETE SET NULL,

  -- Información del evento
  event_type TEXT NOT NULL,
  event_date DATE NOT NULL,
  event_time TIME,
  guest_count INTEGER NOT NULL CHECK (guest_count >= 25 AND guest_count <

  -- Ubicación
  venue TEXT,
  venue_address TEXT,
  venue_district TEXT,

  -- Servicios y pricing
  package_id TEXT,
```

```sql
  service_ids TEXT[], -- Array de IDs de servicios
  total_price DECIMAL(10,2) NOT NULL,
  deposit_paid DECIMAL(10,2) DEFAULT 0,
  balance_due DECIMAL(10,2),

  -- Estado
  status TEXT NOT NULL DEFAULT 'pending'
    CHECK (status IN ('pending', 'confirmed', 'completed', 'cancelled')),

  -- Notas y detalles
  notes TEXT,
  special_requests TEXT,
  cocktails_selected TEXT[],

  -- Timestamps
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  confirmed_at TIMESTAMPTZ,
  completed_at TIMESTAMPTZ
);

-- Indexes
CREATE INDEX idx_events_date ON events(event_date);
CREATE INDEX idx_events_status ON events(status);
CREATE INDEX idx_events_client ON events(client_id);

COMMENT ON TABLE events IS 'Eventos confirmados y programados';
COMMENT ON COLUMN events.status IS 'pending: esperando confirmación, conf
```

4.6 Tabla: event_images

```sql
-- ============================================
-- TABLE: event_images
-- Fotos de eventos realizados (portafolio)
-- ============================================
CREATE TABLE event_images (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  event_id UUID REFERENCES events(id) ON DELETE CASCADE,
  image_url TEXT NOT NULL,
  thumbnail_url TEXT,
  caption TEXT,
  order_index INTEGER DEFAULT 0,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_event_images_event ON event_images(event_id);
CREATE INDEX idx_event_images_order ON event_images(order_index);

COMMENT ON TABLE event_images IS 'Galería de fotos de eventos para el por
```

4.7 Tabla: services

```
-- ===============================================
-- TABLE: services
-- Catálogo de servicios ofrecidos
-- ===============================================
CREATE TABLE services (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  name TEXT NOT NULL,
  slug TEXT UNIQUE NOT NULL,
  description TEXT NOT NULL,
  long_description TEXT,
  price_from DECIMAL(10,2) NOT NULL,
  features TEXT[] NOT NULL,
  icon TEXT,
  image_url TEXT,
  active BOOLEAN DEFAULT true,
  order_index INTEGER DEFAULT 0,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_services_slug ON services(slug);
CREATE INDEX idx_services_active ON services(active);
CREATE INDEX idx_services_order ON services(order_index);

COMMENT ON TABLE services IS 'Catálogo de servicios mostrados en el sitic
```

4.8 Tabla: packages

```
-- ===============================================
-- TABLE: packages
-- Paquetes predefinidos
-- ============================================
CREATE TABLE packages (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  name TEXT NOT NULL,
  slug TEXT UNIQUE NOT NULL,
  description TEXT NOT NULL,
  price DECIMAL(10,2) NOT NULL,
  guest_range TEXT NOT NULL,
  duration INTEGER NOT NULL, -- Horas de servicio
  features TEXT[] NOT NULL,
  popular BOOLEAN DEFAULT false,
  active BOOLEAN DEFAULT true,
  order_index INTEGER DEFAULT 0,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_packages_slug ON packages(slug);
CREATE INDEX idx_packages_active ON packages(active);
CREATE INDEX idx_packages_popular ON packages(popular);

COMMENT ON TABLE packages IS 'Paquetes predefinidos con precios especiale
COMMENT ON COLUMN packages.popular IS 'Marcar como destacado/recomendado
```

4.9 Tabla: testimonials

```
-- ============================================
-- TABLE: testimonials
-- Testimonios de clientes
-- ============================================
CREATE TABLE testimonials (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  client_name TEXT NOT NULL,
  client_company TEXT,
  event_type TEXT,
  rating INTEGER NOT NULL CHECK (rating >= 1 AND rating <= 5),
  comment TEXT NOT NULL,
  image_url TEXT,
  approved BOOLEAN DEFAULT false,
  featured BOOLEAN DEFAULT false,
  created_at TIMESTAMPTZ DEFAULT NOW()
);
```

```
CREATE INDEX idx_testimonials_approved ON testimonials(approved);
CREATE INDEX idx_testimonials_featured ON testimonials(featured);
CREATE INDEX idx_testimonials_rating ON testimonials(rating);

COMMENT ON TABLE testimonials IS 'Testimonios de clientes satisfechos';
COMMENT ON COLUMN testimonials.approved IS 'Solo testimonios aprobados se
COMMENT ON COLUMN testimonials.featured IS 'Destacar en home o landing pa
```

4.10 Tabla: blog_posts

```
-- ============================================
-- TABLE: blog_posts
-- Posts del blog
-- ============================================
CREATE TABLE blog_posts (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  title TEXT NOT NULL,
  slug TEXT UNIQUE NOT NULL,
  excerpt TEXT NOT NULL,
  content TEXT NOT NULL,
  image_url TEXT,
  author_id UUID REFERENCES admin_users(id),
  published BOOLEAN DEFAULT false,
  published_at TIMESTAMPTZ,
  views INTEGER DEFAULT 0,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_blog_slug ON blog_posts(slug);
CREATE INDEX idx_blog_published ON blog_posts(published);
CREATE INDEX idx_blog_published_at ON blog_posts(published_at DESC);

COMMENT ON TABLE blog_posts IS 'Artículos del blog sobre mixología y ever
```

4.11 Tabla: site_settings

```
-- ============================================
-- TABLE: site_settings
-- Configuraciones del sitio
-- ============================================
CREATE TABLE site_settings (
  key TEXT PRIMARY KEY,
  value JSONB NOT NULL,
  description TEXT,
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

COMMENT ON TABLE site_settings IS 'Configuraciones editables desde el par

-- Ejemplo de valores:
-- key: 'contact_email', value: '"contacto@lareserva.pe"'
-- key: 'social_links', value: '{"instagram": "...", "facebook": "..."}'
```

4.12 Triggers para updated_at

```
-- ============================================
-- FUNCTION: Actualizar updated_at automáticamente
-- ============================================
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
  NEW.updated_at = NOW();
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Aplicar trigger a todas las tablas con updated_at
CREATE TRIGGER update_admin_users_updated_at
  BEFORE UPDATE ON admin_users
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_clients_updated_at
  BEFORE UPDATE ON clients
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_quotes_updated_at
  BEFORE UPDATE ON quotes
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_events_updated_at
  BEFORE UPDATE ON events
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_services_updated_at
  BEFORE UPDATE ON services
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_packages_updated_at
  BEFORE UPDATE ON packages
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_blog_posts_updated_at
  BEFORE UPDATE ON blog_posts
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_site_settings_updated_at
  BEFORE UPDATE ON site_settings
  FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();
```

✅ VERIFICACIÓN

Después de ejecutar los scripts:

1. Ve a **Table Editor** en Supabase
2. Deberías ver todas las tablas creadas:
   - admin_users
   - clients
   - quotes
   - events
   - event_images
   - services
   - packages
   - testimonials
   - blog_posts
   - site_settings
3. Verifica que los índices se hayan creado correctamente

🔜 CONTINÚA CON

→ Archivo 03-2: Base de Datos (Parte 2)

- Datos iniciales (seed)
- Row Level Security (RLS)
- Storage y buckets
- Queries comunes
