💻 LA RESERVA - DESARROLLO WEB

PARTE 3-2: Base de Datos (Parte 2)

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

📑 CONTENIDO

---

1. DATOS INICIALES (SEED)

Ejecuta estos scripts en **SQL Editor** para insertar datos iniciales:

1.1 Servicios

```
-- ============================================
-- SEED: services
-- ============================================
INSERT INTO services (name, slug, description, price_from, features, icor
(
  'Bartending para Eventos',
  'bartending-eventos',
  'Servicio completo de barra y bartenders profesionales para tu evento e
  1800.00,
  ARRAY[
    'Bartenders profesionales certificados',
    'Barra completa equipada',
    'Cristalería premium',
```

```
    'Ingredientes frescos y garnish',
    'Setup y decoración de barra',
    'Servicio durante todo el evento'
  ],
  'cocktail',
  true,
  1
),
(
  'Mixología Corporativa',
  'mixologia-corporativa',
  'Experiencia de coctelería personalizada para eventos empresariales y t
  2500.00,
  ARRAY[
    'Cócteles signature con branding',
    'Presentación profesional',
    'Team building de mixología',
    'Barra corporativa premium',
    'Material promocional incluido'
  ],
  'briefcase',
  true,
  2
),
(
  'Cócteles de Autor',
  'cocteles-autor',
  'Creación de cócteles exclusivos diseñados especialmente para tu eventc
  2200.00,
  ARRAY[
    'Consulta previa personalizada',
    'Receta exclusiva creada para ti',
    'Ingredientes premium seleccionados',
    'Técnicas artesanales',
    'Presentación impecable'
  ],
  'sparkles',
  true,
  3
),
(
  'Barra Móvil Premium',
  'barra-movil',
  'Barra móvil completamente equipada con todo lo necesario para tu event
```

```
  800.00,
  ARRAY[
    'Barra portátil elegante',
    'Equipo completo de bartending',
    'Decoración incluida',
    'Setup y desmontaje',
    'Variedad de diseños disponibles'
  ],
  'truck',
  true,
  4
);
```

1.2 Paquetes

```
-- ============================================
-- SEED: packages
-- ============================================
INSERT INTO packages (name, slug, description, price, guest_range, durati
(
  'Paquete Básico',
  'basico',
  'Ideal para eventos íntimos',
  1800.00,
  '25-50',
  4,
  ARRAY[
    '1 bartender profesional',
    'Barra básica equipada',
    '3 cócteles a elegir',
    'Cristalería y hielo',
    'Setup y limpieza'
  ],
  false,
  true,
  1
),
(
  'Paquete Completo',
  'completo',
  'Perfecto para bodas y eventos medianos',
  3500.00,
```

```
    '100-200',
    5,
    ARRAY[
      '2 bartenders profesionales',
      'Barra premium equipada',
      '5 cócteles de autor',
      'Cristalería premium',
      'Decoración de barra',
      'Garnish artístico',
      'Setup y limpieza completa'
    ],
    true,
    true,
    2
  ),
  (
    'Paquete Premium',
    'premium',
    'Experiencia exclusiva para eventos grandes',
    6500.00,
    '200-500',
    6,
    ARRAY[
      '3+ bartenders profesionales',
      'Doble barra premium',
      'Cócteles de autor ilimitados',
      'Cristalería de lujo',
      'Decoración personalizada',
      'Garnish gourmet',
      'Sommelier de cócteles',
      'Servicio de fotografía de bebidas',
      'Setup, limpieza y coordinación completa'
    ],
    false,
    true,
    3
  );
```

1.3 Configuraciones del Sitio

```
-- ============================================
-- SEED: site_settings
```

```
-- ============================================
INSERT INTO site_settings (key, value, description) VALUES
('contact_email', '"contacto@lareserva.pe"', 'Email principal de contacto
('contact_phone', '"+51999888777"', 'Teléfono de contacto'),
('whatsapp_number', '"+51999888777"', 'Número de WhatsApp Business'),
('business_hours',
  '{"weekdays": "Lunes - Viernes: 9:00 AM - 5:00 PM", "saturday": "Sábado
  'Horarios de atención'
),
('social_links',
  '{"instagram": "https://instagram.com/lareservabar", "facebook": "https
  'Links a redes sociales'
),
('site_name', '"La Reserva"', 'Nombre del sitio'),
('site_tagline', '"Mixología Exclusiva"', 'Tagline del sitio'),
('max_guests', '500', 'Máximo de invitados permitido'),
('min_guests', '25', 'Mínimo de invitados requerido');
```

1.4 Testimonios de Ejemplo

```
-- ============================================
-- SEED: testimonials (ejemplos)
-- ============================================
INSERT INTO testimonials (client_name, client_company, event_type, rating
(
  'María Fernández',
  NULL,
  'Boda',
  5,
  'El servicio de La Reserva superó todas nuestras expectativas. Los cóct
  true,
  true
),
(
  'Carlos Mendoza',
  'Empresa XYZ SAC',
  'Evento Corporativo',
  5,
  'Contratamos La Reserva para nuestro evento anual y fue un éxito rotund
  true,
  false
),
```

```
(
  'Andrea López',
  NULL,
  'Cumpleaños',
  5,
  'Mi cumpleaños número 30 fue inolvidable gracias a La Reserva. Los bart
  true,
  false
);
```

---

2. ROW LEVEL SECURITY (RLS)

RLS permite controlar qué usuarios pueden ver/modificar qué filas.

2.1 Habilitar RLS en todas las tablas

```
-- ============================================
-- ENABLE ROW LEVEL SECURITY
-- ============================================
ALTER TABLE admin_users ENABLE ROW LEVEL SECURITY;
ALTER TABLE clients ENABLE ROW LEVEL SECURITY;
ALTER TABLE quotes ENABLE ROW LEVEL SECURITY;
ALTER TABLE events ENABLE ROW LEVEL SECURITY;
ALTER TABLE event_images ENABLE ROW LEVEL SECURITY;
ALTER TABLE services ENABLE ROW LEVEL SECURITY;
ALTER TABLE packages ENABLE ROW LEVEL SECURITY;
ALTER TABLE testimonials ENABLE ROW LEVEL SECURITY;
ALTER TABLE blog_posts ENABLE ROW LEVEL SECURITY;
ALTER TABLE site_settings ENABLE ROW LEVEL SECURITY;
```

2.2 Policies para quotes (Público puede crear)

```
-- ============================================
-- POLICIES: quotes
-- ============================================

-- Cualquiera puede crear una cotización (formulario público)
CREATE POLICY "Anyone can insert quotes"
  ON quotes FOR INSERT
```

```
  TO anon, authenticated
  WITH CHECK (true);

-- Solo admins pueden ver todas las cotizaciones
CREATE POLICY "Admins can view all quotes"
  ON quotes FOR SELECT
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

-- Solo admins pueden actualizar cotizaciones
CREATE POLICY "Admins can update quotes"
  ON quotes FOR UPDATE
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  )
  WITH CHECK (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

-- Solo admins pueden eliminar cotizaciones
CREATE POLICY "Admins can delete quotes"
  ON quotes FOR DELETE
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );
```

2.3 Policies para services y packages (Público lectura)

```sql
-- ============================================
-- POLICIES: services
-- ============================================

-- Cualquiera puede leer servicios activos
CREATE POLICY "Anyone can view active services"
  ON services FOR SELECT
  TO anon, authenticated
  USING (active = true);

-- Admins pueden ver todos los servicios (activos e inactivos)
CREATE POLICY "Admins can view all services"
  ON services FOR SELECT
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

-- Solo admins pueden insertar servicios
CREATE POLICY "Admins can insert services"
  ON services FOR INSERT
  TO authenticated
  WITH CHECK (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

-- Solo admins pueden actualizar servicios
CREATE POLICY "Admins can update services"
  ON services FOR UPDATE
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

-- Solo admins pueden eliminar servicios
```

```sql
CREATE POLICY "Admins can delete services"
  ON services FOR DELETE
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

-- ============================================
-- POLICIES: packages (igual que services)
-- ============================================

CREATE POLICY "Anyone can view active packages"
  ON packages FOR SELECT
  TO anon, authenticated
  USING (active = true);

CREATE POLICY "Admins can view all packages"
  ON packages FOR SELECT
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

CREATE POLICY "Admins can manage packages"
  ON packages FOR ALL
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );
```

2.4 Policies para testimonials

```sql
-- ============================================
-- POLICIES: testimonials
-- ============================================

-- Cualquiera puede ver testimonios aprobados
CREATE POLICY "Anyone can view approved testimonials"
  ON testimonials FOR SELECT
  TO anon, authenticated
  USING (approved = true);

-- Admins pueden ver todos los testimonios
CREATE POLICY "Admins can view all testimonials"
  ON testimonials FOR SELECT
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

-- Solo admins pueden gestionar testimonios
CREATE POLICY "Admins can manage testimonials"
  ON testimonials FOR ALL
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );
```

2.5 Policies para blog_posts

```sql
-- ============================================
-- POLICIES: blog_posts
-- ============================================

-- Cualquiera puede ver posts publicados
CREATE POLICY "Anyone can view published posts"
  ON blog_posts FOR SELECT
  TO anon, authenticated
```

```
  USING (published = true);

-- Admins pueden ver todos los posts
CREATE POLICY "Admins can view all posts"
  ON blog_posts FOR SELECT
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

-- Solo admins pueden gestionar posts
CREATE POLICY "Admins can manage posts"
  ON blog_posts FOR ALL
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );
```

2.6 Policies para events y clients (Solo admins)

```
-- ============================================
-- POLICIES: events (Solo admins)
-- ============================================

CREATE POLICY "Admins only for events"
  ON events FOR ALL
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

-- ============================================
-- POLICIES: clients (Solo admins)
```

```sql
-- ============================================

CREATE POLICY "Admins only for clients"
  ON clients FOR ALL
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

-- ============================================
-- POLICIES: event_images (Solo admins)
-- ============================================

CREATE POLICY "Admins can manage event_images"
  ON event_images FOR ALL
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

-- Cualquiera puede ver imágenes de eventos completados
CREATE POLICY "Anyone can view completed event images"
  ON event_images FOR SELECT
  TO anon, authenticated
  USING (
    EXISTS (
      SELECT 1 FROM events
      WHERE events.id = event_images.event_id
      AND events.status = 'completed'
    )
  );
```

2.7 Policies para site_settings

```sql
-- ============================================
-- POLICIES: site_settings
```

```
-- ============================================

-- Cualquiera puede leer configuraciones
CREATE POLICY "Anyone can view settings"
  ON site_settings FOR SELECT
  TO anon, authenticated
  USING (true);

-- Solo super_admin puede modificar
CREATE POLICY "Super admins can manage settings"
  ON site_settings FOR ALL
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
      AND role = 'super_admin'
    )
  );
```

---

3. STORAGE (ALMACENAMIENTO)

3.1 Crear Buckets

En Supabase Dashboard → **Storage** → **New Bucket**:

```
-- Ejecutar en SQL Editor para crear buckets via SQL
INSERT INTO storage.buckets (id, name, public) VALUES
  ('events', 'events', true),
  ('blog', 'blog', true),
  ('testimonials', 'testimonials', true);
```

O crear manualmente desde la UI:

- **Bucket name:** events
- **Public bucket:** Yes (imágenes públicas)

3.2 Storage Policies

```sql
-- ============================================
-- STORAGE POLICIES: events bucket
-- ============================================

-- Cualquiera puede ver imágenes
CREATE POLICY "Public can view event images"
  ON storage.objects FOR SELECT
  TO public
  USING (bucket_id = 'events');

-- Solo admins pueden subir imágenes
CREATE POLICY "Admins can upload event images"
  ON storage.objects FOR INSERT
  TO authenticated
  WITH CHECK (
    bucket_id = 'events' AND
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

-- Solo admins pueden eliminar imágenes
CREATE POLICY "Admins can delete event images"
  ON storage.objects FOR DELETE
  TO authenticated
  USING (
    bucket_id = 'events' AND
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

-- ============================================
-- STORAGE POLICIES: blog bucket (igual)
-- ============================================

CREATE POLICY "Public can view blog images"
  ON storage.objects FOR SELECT
  TO public
  USING (bucket_id = 'blog');

CREATE POLICY "Admins can upload blog images"
```

```
  ON storage.objects FOR INSERT
  TO authenticated
  WITH CHECK (
    bucket_id = 'blog' AND
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );

CREATE POLICY "Admins can delete blog images"
  ON storage.objects FOR DELETE
  TO authenticated
  USING (
    bucket_id = 'blog' AND
    EXISTS (
      SELECT 1 FROM admin_users
      WHERE admin_users.id = auth.uid()
    )
  );
```

3.3 Subir imágenes desde código

```
// Ejemplo de uso en el frontend
import { supabase } from '@/lib/supabase';

async function uploadEventImage(file: File, eventId: string) {
  const fileExt = file.name.split('.').pop();
  const fileName = `${eventId}/${Date.now()}.${fileExt}`;

  const { data, error } = await supabase.storage
    .from('events')
    .upload(fileName, file, {
      cacheControl: '3600',
      upsert: false
    });

  if (error) throw error;

  // Obtener URL pública
  const { data: { publicUrl } } = supabase.storage
    .from('events')
```

```
    .getPublicUrl(fileName);

  return publicUrl;
}
```

---

4. QUERIES COMUNES

4.1 Obtener servicios activos

```
// src/lib/queries/services.ts
import { supabase } from '@/lib/supabase';

export async function getActiveServices() {
  const { data, error } = await supabase
    .from('services')
    .select('*')
    .eq('active', true)
    .order('order_index', { ascending: true });

  if (error) throw error;
  return data;
}
```

4.2 Crear nueva cotización

```
// src/lib/queries/quotes.ts
import { supabase } from '@/lib/supabase';
import type { QuoteFormData } from '@/utils/validators';

export async function createQuote(data: QuoteFormData) {
  const { data: quote, error } = await supabase
    .from('quotes')
    .insert({
      client_name: data.name,
      client_email: data.email,
      client_phone: data.phone,
      event_type: data.eventType,
      event_date: data.eventDate,
      guest_count: data.guestCount,
```

```
      message: data.message,
      status: 'new',
    })
    .select()
    .single();

  if (error) throw error;
  return quote;
}
```

4.3 Obtener cotizaciones pendientes (admin)

```
// src/lib/queries/admin/quotes.ts
import { supabaseAdmin } from '@/lib/supabase';

export async function getPendingQuotes() {
  const { data, error } = await supabaseAdmin
    .from('quotes')
    .select('*')
    .in('status', ['new', 'contacted'])
    .order('created_at', { ascending: false });

  if (error) throw error;
  return data;
}
```

4.4 Obtener eventos del mes (admin)

```
// src/lib/queries/admin/events.ts
import { supabaseAdmin } from '@/lib/supabase';
import { startOfMonth, endOfMonth } from 'date-fns';

export async function getEventsThisMonth() {
  const start = startOfMonth(new Date());
  const end = endOfMonth(new Date());

  const { data, error } = await supabaseAdmin
    .from('events')
    .select(`
      *,
      client:clients(name, email, phone)
```

```
    `)
    .gte('event_date', start.toISOString().split('T')[0])
    .lte('event_date', end.toISOString().split('T')[0])
    .order('event_date', { ascending: true });

  if (error) throw error;
  return data;
}
```

4.5 Obtener imágenes para portafolio

```
// src/lib/queries/portfolio.ts
import { supabase } from '@/lib/supabase';

export async function getPortfolioImages(limit: number = 12) {
  const { data, error } = await supabase
    .from('event_images')
    .select(`
      *,
      event:events(event_type, event_date, status)
    `)
    .eq('events.status', 'completed')
    .order('created_at', { ascending: false })
    .limit(limit);

  if (error) throw error;
  return data;
}
```

---

5. INTEGRACIÓN CON FRONTEND

5.1 Archivo de tipos generado

Supabase puede generar tipos TypeScript automáticamente:

```
# Instalar Supabase CLI
npm install -g supabase

# Generar tipos
supabase gen types typescript --project-id "tu-project-id" > src/types/da
```

5.2 Usar tipos en el código

```
// src/types/database.ts (generado automáticamente)
export type Database = {
  public: {
    Tables: {
      quotes: {
        Row: {
          id: string;
          client_name: string;
          client_email: string;
          // ...
        };
        Insert: {
          id?: string;
          client_name: string;
          // ...
        };
        Update: {
          client_name?: string;
          // ...
        };
      };
      // ... resto de tablas
    };
  };
};


// Usar tipos en queries
import type { Database } from '@/types/database';

type Quote = Database['public']['Tables']['quotes']['Row'];
type QuoteInsert = Database['public']['Tables']['quotes']['Insert'];

async function createQuote(data: QuoteInsert): Promise<Quote> {
  // ...
}
```

✅ CHECKLIST COMPLETO - BASE DE DATOS

Verifica que hayas completado:

Setup:

- [ ] Proyecto de Supabase creado
- [ ] Credenciales copiadas a `.env`
- [ ] Cliente configurado en `src/lib/supabase.ts`

Tablas:

- [ ] Todas las tablas creadas (10 tablas)
- [ ] Índices creados correctamente
- [ ] Triggers de `updated_at` funcionando

Datos:

- [ ] Servicios insertados (4 servicios)
- [ ] Paquetes insertados (3 paquetes)
- [ ] Configuraciones del sitio insertadas
- [ ] Testimonios de ejemplo insertados

Seguridad:

- [ ] RLS habilitado en todas las tablas
- [ ] Policies creadas para cada tabla
- [ ] Policies de storage configuradas

Storage:

- [ ] Buckets creados (events, blog, testimonials)
- [ ] Storage policies configuradas

Testing:

- [ ] Puedes obtener servicios desde el frontend
- [ ] Puedes crear una cotización sin auth
- [ ] No puedes ver cotizaciones sin auth (RLS funciona)

🎯 PRÓXIMOS PASOS

Has completado la configuración de base de datos. Continúa con:

→ Archivo 04: Componentes y UI

- Componentes base reutilizables
- Componentes específicos del sitio
- Ejemplos de uso
