💻 LA RESERVA - DESARROLLO WEB

PARTE 4-2: Componentes React Interactivos

Versión: 1.0
Fecha: Octubre 2025
Tiempo de lectura: 25 minutos

📑 CONTENIDO

---

1. MODAL COMPONENT

1.1 Modal.tsx

```tsx
// src/components/ui/Modal.tsx
import { useEffect, useState } from 'react';
import { createPortal } from 'react-dom';
import { motion, AnimatePresence } from 'framer-motion';
import { cn } from '@/utils/utils';

interface ModalProps {
  isOpen: boolean;
  onClose: () => void;
  title?: string;
  size?: 'sm' | 'md' | 'lg' | 'xl' | 'full';
  children: React.ReactNode;
  showCloseButton?: boolean;
  closeOnOverlayClick?: boolean;
}
```

```
export function Modal({
  isOpen,
  onClose,
  title,
  size = 'md',
  children,
  showCloseButton = true,
  closeOnOverlayClick = true,
}: ModalProps) {
  const [mounted, setMounted] = useState(false);

  useEffect(() => {
    setMounted(true);
    return () => setMounted(false);
  }, []);

  useEffect(() => {
    if (isOpen) {
      document.body.style.overflow = 'hidden';
    } else {
      document.body.style.overflow = 'unset';
    }

    return () => {
      document.body.style.overflow = 'unset';
    };
  }, [isOpen]);

  const sizeClasses = {
    sm: 'max-w-md',
    md: 'max-w-lg',
    lg: 'max-w-2xl',
    xl: 'max-w-4xl',
    full: 'max-w-full mx-4',
  };

  if (!mounted) return null;

  return createPortal(
    <AnimatePresence>
      {isOpen && (
        <>
          {/* Overlay */}
```

```
<motion.div
  initial={{ opacity: 0 }}
  animate={{ opacity: 1 }}
  exit={{ opacity: 0 }}
  onClick={closeOnOverlayClick ? onClose : undefined}
  className="fixed inset-0 bg-black/60 backdrop-blur-sm z-50"
/>

{/* Modal */}
<div className="fixed inset-0 z-50 flex items-center justify-ce
  <motion.div
    initial={{ opacity: 0, scale: 0.95, y: 20 }}
    animate={{ opacity: 1, scale: 1, y: 0 }}
    exit={{ opacity: 0, scale: 0.95, y: 20 }}
    className={cn(
      'relative bg-white rounded-lg shadow-2xl w-full',
      sizeClasses[size]
    )}
  >
    {/* Header */}
    {(title || showCloseButton) && (
      <div className="flex items-center justify-between px-6 py
        {title && (
          <h3 className="text-xl font-display font-semibold tex
            {title}
          </h3>
        )}
        {showCloseButton && (
          <button
            onClick={onClose}
            className="p-2 hover:bg-secondary-100 rounded-full
            aria-label="Cerrar modal"
          >
            <svg
              className="w-5 h-5 text-secondary-600"
              fill="none"
              stroke="currentColor"
              viewBox="0 0 24 24"
            >
              <path
                strokeLinecap="round"
                strokeLinejoin="round"
                strokeWidth={2}
                d="M6 18L18 6M6 6l12 12"
```

```
                        />
                      </svg>
                    </button>
                  )}
                </div>
              )}

              {/* Content */}
              <div className="px-6 py-4 max-h-[calc(100vh-200px)] overflo
                {children}
              </div>
            </motion.div>
          </div>
        </>
      )}
    </AnimatePresence>,
    document.body
  );
}
```

1.2 Ejemplo de uso

```
// En cualquier componente React
import { useState } from 'react';
import { Modal } from '@/components/ui/Modal';

export function MyComponent() {
  const [isModalOpen, setIsModalOpen] = useState(false);

  return (
    <>
      <button onClick={() => setIsModalOpen(true)}>
        Abrir Modal
      </button>

      <Modal
        isOpen={isModalOpen}
        onClose={() => setIsModalOpen(false)}
        title="Título del Modal"
        size="md"
      >
        <p>Contenido del modal aquí</p>
```

```
        <button onClick={() => setIsModalOpen(false)}>
          Cerrar
        </button>
      </Modal>
    </>
  );
}
```

---

2. TOAST NOTIFICATIONS

2.1 Toast Context y Provider

```
// src/components/ui/Toast.tsx
import { createContext, useContext, useState, useCallback } from 'react';
import { motion, AnimatePresence } from 'framer-motion';
import { cn } from '@/utils/utils';

type ToastType = 'success' | 'error' | 'warning' | 'info';

interface Toast {
  id: string;
  message: string;
  type: ToastType;
}

interface ToastContextType {
  showToast: (message: string, type?: ToastType) => void;
}

const ToastContext = createContext<ToastContextType | undefined>(undefine

export function ToastProvider({ children }: { children: React.ReactNode }
  const [toasts, setToasts] = useState<Toast[]>([]);

  const showToast = useCallback((message: string, type: ToastType = 'info
    const id = Math.random().toString(36).substr(2, 9);
    const newToast = { id, message, type };

    setToasts((prev) => [...prev, newToast]);

    // Auto-remove after 5 seconds
```

```
    setTimeout(() => {
      setToasts((prev) => prev.filter((toast) => toast.id !== id));
    }, 5000);
  }, []);

  const removeToast = useCallback((id: string) => {
    setToasts((prev) => prev.filter((toast) => toast.id !== id));
  }, []);

  return (
    <ToastContext.Provider value={{ showToast }}>
      {children}

      {/* Toast Container */}
      <div className="fixed bottom-4 right-4 z-50 space-y-2">
        <AnimatePresence>
          {toasts.map((toast) => (
            <ToastItem
              key={toast.id}
              toast={toast}
              onClose={() => removeToast(toast.id)}
            />
          ))}
        </AnimatePresence>
      </div>
    </ToastContext.Provider>
  );
}

function ToastItem({ toast, onClose }: { toast: Toast; onClose: () => voi
  const typeConfig = {
    success: {
      bg: 'bg-green-50',
      border: 'border-green-500',
      text: 'text-green-800',
      icon: '✓',
    },
    error: {
      bg: 'bg-red-50',
      border: 'border-red-500',
      text: 'text-red-800',
      icon: '✕',
    },
    warning: {
```

```
      bg: 'bg-yellow-50',
      border: 'border-yellow-500',
      text: 'text-yellow-800',
      icon: '  ',
    },
    info: {
      bg: 'bg-blue-50',
      border: 'border-blue-500',
      text: 'text-blue-800',
      icon: 'i',
    },
  };

  const config = typeConfig[toast.type];

  return (
    <motion.div
      initial={{ opacity: 0, x: 100 }}
      animate={{ opacity: 1, x: 0 }}
      exit={{ opacity: 0, x: 100 }}
      className={cn(
        'flex items-center gap-3 px-4 py-3 rounded-lg shadow-lg border-l-
        config.bg,
        config.border
      )}
    >
      <span className="text-2xl">{config.icon}</span>
      <p className={cn('flex-1 font-medium', config.text)}>{toast.message
      <button
        onClick={onClose}
        className={cn('p-1 hover:opacity-70 transition-opacity', config.t
      >
        <svg className="w-4 h-4" fill="none" stroke="currentColor" viewBc
          <path strokeLinecap="round" strokeLinejoin="round" strokeWidth=
        </svg>
      </button>
    </motion.div>
  );
}

export function useToast() {
  const context = useContext(ToastContext);
  if (!context) {
    throw new Error('useToast must be used within ToastProvider');
```

```
  }
  return context;
}
```

2.2 Agregar Provider en Layout

```
---
// src/layouts/BaseLayout.astro (modificar)
import { ToastProvider } from '@/components/ui/Toast';
---

<!doctype html>
<html>
  <head>...</head>
  <body>
    <ToastProvider client:load>
      <slot />
    </ToastProvider>
  </body>
</html>
```

2.3 Uso del Toast

```
// En cualquier componente
import { useToast } from '@/components/ui/Toast';

export function MyForm() {
  const { showToast } = useToast();

  const handleSubmit = async () => {
    try {
      // ... lógica
      showToast('¡Formulario enviado con éxito!', 'success');
    } catch (error) {
      showToast('Error al enviar el formulario', 'error');
    }
  };

  return <form onSubmit={handleSubmit}>...</form>;
}
```

3. QUOTEFORM COMPONENT

3.1 QuoteForm.tsx (Formulario de Cotización)

```
// src/components/forms/QuoteForm.tsx
import { useState } from 'react';
import { useForm } from 'react-hook-form';
import { zodResolver } from '@hookform/resolvers/zod';
import { quoteSchema, type QuoteFormData } from '@/utils/validators';
import { EVENT_TYPES } from '@/utils/constants';
import { supabase } from '@/lib/supabase';
import { useToast } from '@/components/ui/Toast';
import { cn } from '@/utils/utils';

export function QuoteForm() {
  const [isSubmitting, setIsSubmitting] = useState(false);
  const { showToast } = useToast();

  const {
    register,
    handleSubmit,
    reset,
    formState: { errors },
  } = useForm<QuoteFormData>({
    resolver: zodResolver(quoteSchema),
    defaultValues: {
      guestCount: 100,
    },
  });

  const onSubmit = async (data: QuoteFormData) => {
    try {
      setIsSubmitting(true);

      const { 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',
    });

    if (error) throw error;

    showToast('¡Cotización enviada! Te contactaremos pronto.', 'success
    reset();

    // Opcional: Trigger email via Edge Function
    // await fetch('/api/send-quote-email', { method: 'POST', body: JSO
  } catch (error) {
    console.error('Error:', error);
    showToast('Error al enviar cotización. Por favor intenta de nuevo.'
  } finally {
    setIsSubmitting(false);
  }
};

return (
  <form onSubmit={handleSubmit(onSubmit)} className="space-y-6">
    {/* Nombre */}
    <div>
      <label htmlFor="name" className="label">
        Nombre completo <span className="text-red-500">*</span>
      </label>
      <input
        {...register('name')}
        id="name"
        type="text"
        className={cn('input', errors.name && 'border-red-500')}
        placeholder="Juan Pérez"
      />
      {errors.name && (
        <p className="text-sm text-red-500 mt-1">{errors.name.message}<
      )}
    </div>

    {/* Email */}
    <div>
      <label htmlFor="email" className="label">
        Email <span className="text-red-500">*</span>
      </label>
      <input
        {...register('email')}
```

```
    id="email"
    type="email"
    className={cn('input', errors.email && 'border-red-500')}
    placeholder="juan@ejemplo.com"
  />
  {errors.email && (
    <p className="text-sm text-red-500 mt-1">{errors.email.message}
  )}
</div>

{/* Teléfono */}
<div>
  <label htmlFor="phone" className="label">
    Teléfono / WhatsApp <span className="text-red-500">*</span>
  </label>
  <input
    {...register('phone')}
    id="phone"
    type="tel"
    className={cn('input', errors.phone && 'border-red-500')}
    placeholder="999 888 777"
  />
  {errors.phone && (
    <p className="text-sm text-red-500 mt-1">{errors.phone.message}
  )}
</div>

{/* Tipo de Evento */}
<div>
  <label htmlFor="eventType" className="label">
    Tipo de evento <span className="text-red-500">*</span>
  </label>
  <select
    {...register('eventType')}
    id="eventType"
    className={cn('input appearance-none', errors.eventType && 'bor
  >
    <option value="">Selecciona un tipo</option>
    {EVENT_TYPES.map((type) => (
      <option key={type.value} value={type.value}>
        {type.label}
      </option>
    ))}
  </select>
```

```
  {errors.eventType && (
    <p className="text-sm text-red-500 mt-1">{errors.eventType.mess
  )}
</div>

{/* Fecha del Evento */}
<div>
  <label htmlFor="eventDate" className="label">
    Fecha del evento <span className="text-red-500">*</span>
  </label>
  <input
    {...register('eventDate')}
    id="eventDate"
    type="date"
    min={new Date().toISOString().split('T')[0]}
    className={cn('input', errors.eventDate && 'border-red-500')}
  />
  {errors.eventDate && (
    <p className="text-sm text-red-500 mt-1">{errors.eventDate.mess
  )}
</div>

{/* Número de Invitados */}
<div>
  <label htmlFor="guestCount" className="label">
    Número de invitados <span className="text-red-500">*</span>
  </label>
  <input
    {...register('guestCount', { valueAsNumber: true })}
    id="guestCount"
    type="number"
    min={25}
    max={500}
    className={cn('input', errors.guestCount && 'border-red-500')}
  />
  {errors.guestCount && (
    <p className="text-sm text-red-500 mt-1">{errors.guestCount.mes
  )}
  <p className="text-sm text-secondary-400 mt-1">Mínimo 25, máximo
</div>

{/* Mensaje Opcional */}
<div>
  <label htmlFor="message" className="label">
```

```
          Mensaje adicional (opcional)
        </label>
        <textarea
          {...register('message')}
          id="message"
          rows={4}
          className="input resize-none"
          placeholder="Cuéntanos más sobre tu evento..."
        />
      </div>

      {/* Submit Button */}
      <button
        type="submit"
        disabled={isSubmitting}
        className="btn-primary w-full"
      >
        {isSubmitting ? 'Enviando...' : 'Solicitar Cotización'}
      </button>

      <p className="text-sm text-secondary-400 text-center">
        Te contactaremos dentro de las próximas 24 horas
      </p>
    </form>
  );
}
```

3.2 Uso en página Astro

```
---
// src/pages/cotizacion.astro
import PageLayout from '@/layouts/PageLayout.astro';
import { QuoteForm } from '@/components/forms/QuoteForm';
---

<PageLayout title="Solicitar Cotización">
  <section class="section">
    <div class="container-custom max-w-2xl">
      <div class="text-center mb-12">
        <h1 class="mb-4">Solicita tu Cotización</h1>
        <p class="text-lg text-secondary-400">
          Completa el formulario y te contactaremos pronto
```

```
        </p>
      </div>

      <div class="card">
        <QuoteForm client:load />
      </div>
    </div>
  </section>
</PageLayout>
```

---

4. HERO SECTION COMPONENT

4.1 Hero.tsx (Hero animado)

```tsx
// src/components/sections/Hero.tsx
import { motion } from 'framer-motion';
import { getWhatsAppUrl } from '@/utils/whatsapp';

interface HeroProps {
  title: string;
  subtitle: string;
  ctaPrimary?: { text: string; href: string };
  ctaSecondary?: { text: string; href: string };
  backgroundImage?: string;
}

export function Hero({
  title,
  subtitle,
  ctaPrimary = { text: 'Cotizar Evento', href: '/cotizacion' },
  ctaSecondary = { text: 'Ver Portafolio', href: '/portafolio' },
  backgroundImage = '/images/hero-bg.jpg',
}: HeroProps) {
  const handleWhatsApp = () => {
    const message = '¡Hola! Me gustaría obtener más información sobre sus
    window.open(getWhatsAppUrl(message), '_blank');
  };

  return (
    <section className="relative min-h-screen flex items-center justify-c
      {/* Background Image */}
```

```
<div
  className="absolute inset-0 bg-cover bg-center"
  style={{ backgroundImage: `url(${backgroundImage})` }}
>
  <div className="absolute inset-0 bg-black/60" />
</div>

{/* Content */}
<div className="relative z-10 container-custom text-center text-whi
  <motion.div
    initial={{ opacity: 0, y: 30 }}
    animate={{ opacity: 1, y: 0 }}
    transition={{ duration: 0.8 }}
  >
    <h1 className="text-5xl md:text-6xl lg:text-7xl font-display fc
      {title}
    </h1>

    <motion.p
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.8, delay: 0.2 }}
      className="text-xl md:text-2xl mb-12 max-w-3xl mx-auto"
    >
      {subtitle}
    </motion.p>

    <motion.div
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.8, delay: 0.4 }}
      className="flex flex-col sm:flex-row gap-4 justify-center"
    >
      <a href={ctaPrimary.href} className="btn-primary">
        {ctaPrimary.text}
      </a>
      <a href={ctaSecondary.href} className="btn-outline">
        {ctaSecondary.text}
      </a>
    </motion.div>

    {/* WhatsApp Float Button */}
    <motion.button
      initial={{ opacity: 0, scale: 0 }}
```

```
            animate={{ opacity: 1, scale: 1 }}
            transition={{ duration: 0.5, delay: 0.6 }}
            onClick={handleWhatsApp}
            className="fixed bottom-6 right-6 z-50 bg-green-500 hover:bg-
            aria-label="Contactar por WhatsApp"
          >
            <svg className="w-8 h-8" fill="currentColor" viewBox="0 0 24
              <path d="M17.472 14.382c-.297-.149-1.758-.867-2.03-.967-.27
            </svg>
          </motion.button>
        </motion.div>
      </div>

      {/* Scroll Indicator */}
      <motion.div
        initial={{ opacity: 0 }}
        animate={{ opacity: 1 }}
        transition={{ duration: 1, delay: 1 }}
        className="absolute bottom-8 left-1/2 -translate-x-1/2"
      >
        <motion.div
          animate={{ y: [0, 10, 0] }}
          transition={{ duration: 2, repeat: Infinity }}
          className="text-white"
        >
          <svg className="w-6 h-6" fill="none" stroke="currentColor" viewBox
            <path strokeLinecap="round" strokeLinejoin="round" strokeWidth
          </svg>
        </motion.div>
      </motion.div>
    </section>
  );
}
```

4.2 Uso del Hero

```
---
// src/pages/index.astro
import PageLayout from '@/layouts/PageLayout.astro';
import { Hero } from '@/components/sections/Hero';
---
```

```
<PageLayout title="Inicio">
  <Hero
    client:load
    title="Mixología Exclusiva para Momentos Únicos"
    subtitle="Transformamos tu celebración en una experiencia memorable c
    backgroundImage="/images/hero-cocktail.jpg"
  />

  <!-- Resto del contenido -->
</PageLayout>
```

---

5. TESTIMONIALS SLIDER

5.1 TestimonialsSlider.tsx

```
// src/components/sections/TestimonialsSlider.tsx
import { useState, useEffect } from 'react';
import { motion, AnimatePresence } from 'framer-motion';

interface Testimonial {
  id: string;
  client_name: string;
  event_type: string;
  rating: number;
  comment: string;
  image_url?: string;
}

interface TestimonialsSliderProps {
  testimonials: Testimonial[];
}

export function TestimonialsSlider({ testimonials }: TestimonialsSliderPr
  const [currentIndex, setCurrentIndex] = useState(0);
  const [direction, setDirection] = useState(0);

  useEffect(() => {
    const timer = setInterval(() => {
      handleNext();
    }, 5000);
```

```
  return () => clearInterval(timer);
}, [currentIndex]);

const handleNext = () => {
  setDirection(1);
  setCurrentIndex((prev) => (prev + 1) % testimonials.length);
};

const handlePrev = () => {
  setDirection(-1);
  setCurrentIndex((prev) => (prev - 1 + testimonials.length) % testimor
};

const current = testimonials[currentIndex];

const variants = {
  enter: (direction: number) => ({
    x: direction > 0 ? 1000 : -1000,
    opacity: 0,
  }),
  center: {
    x: 0,
    opacity: 1,
  },
  exit: (direction: number) => ({
    x: direction < 0 ? 1000 : -1000,
    opacity: 0,
  }),
};

return (
  <div className="relative max-w-4xl mx-auto">
    <AnimatePresence initial={false} custom={direction}>
      <motion.div
        key={currentIndex}
        custom={direction}
        variants={variants}
        initial="enter"
        animate="center"
        exit="exit"
        transition={{
          x: { type: 'spring', stiffness: 300, damping: 30 },
          opacity: { duration: 0.2 },
        }}
```

```
      className="bg-white p-8 md:p-12 rounded-lg shadow-xl"
    >
      {/* Stars */}
      <div className="flex justify-center mb-6">
        {Array.from({ length: 5 }).map((_, i) => (
          <svg
            key={i}
            className={`w-6 h-6 ${
              i < current.rating ? 'text-primary-500' : 'text-seconda
            }`}
            fill="currentColor"
            viewBox="0 0 20 20"
          >
            <path d="M9.049 2.927c.3-.921 1.603-.921 1.902 0l1.07 3.2
          </svg>
        ))}
      </div>

      {/* Quote */}
      <blockquote className="text-xl md:text-2xl text-center text-sec
        "{current.comment}"
      </blockquote>

      {/* Author */}
      <div className="flex flex-col items-center">
        {current.image_url && (
          <img
            src={current.image_url}
            alt={current.client_name}
            className="w-16 h-16 rounded-full mb-4 object-cover"
          />
        )}
        <p className="font-semibold text-lg text-secondary-600">
          {current.client_name}
        </p>
        <p className="text-secondary-400 text-sm">
          {current.event_type}
        </p>
      </div>
    </motion.div>
  </AnimatePresence>

  {/* Navigation Buttons */}
  <button
```

```
        onClick={handlePrev}
        className="absolute left-0 top-1/2 -translate-y-1/2 -translate-x-
        aria-label="Testimonio anterior"
      >
        <svg className="w-6 h-6 text-secondary-600" fill="none" stroke="c
          <path strokeLinecap="round" strokeLinejoin="round" strokeWidth=
        </svg>
      </button>

      <button
        onClick={handleNext}
        className="absolute right-0 top-1/2 -translate-y-1/2 translate-x-
        aria-label="Siguiente testimonio"
      >
        <svg className="w-6 h-6 text-secondary-600" fill="none" stroke="c
          <path strokeLinecap="round" strokeLinejoin="round" strokeWidth=
        </svg>
      </button>

      {/* Dots Indicator */}
      <div className="flex justify-center gap-2 mt-8">
        {testimonials.map((_, index) => (
          <button
            key={index}
            onClick={() => {
              setDirection(index > currentIndex ? 1 : -1);
              setCurrentIndex(index);
            }}
            className={`w-3 h-3 rounded-full transition-all ${
              index === currentIndex
                ? 'bg-primary-500 w-8'
                : 'bg-secondary-300 hover:bg-secondary-400'
            }`}
            aria-label={`Ir a testimonio ${index + 1}`}
          />
        ))}
      </div>
    </div>
  );
}
```

5.2 Uso del Testimonials Slider

```
---
// src/pages/index.astro (continuación)
import { TestimonialsSlider } from '@/components/sections/TestimonialsSli
import { supabase } from '@/lib/supabase';

const { data: testimonials } = await supabase
  .from('testimonials')
  .select('*')
  .eq('approved', true)
  .eq('featured', true)
  .limit(5);
---

<section class="section bg-secondary-50">
  <div class="container-custom">
    <div class="text-center mb-12">
      <h2 class="mb-4">Lo que dicen nuestros clientes</h2>
      <p class="text-lg text-secondary-400">
        Experiencias reales de eventos memorables
      </p>
    </div>

    {testimonials && testimonials.length > 0 && (
      <TestimonialsSlider client:load testimonials={testimonials} />
    )}
  </div>
</section>
```

---

6. CONTACTFORM COMPONENT (Simple)

6.1 ContactForm.tsx

```
// src/components/forms/ContactForm.tsx
import { useState } from 'react';
import { useForm } from 'react-hook-form';
import { zodResolver } from '@hookform/resolvers/zod';
import { contactSchema, type ContactFormData } from '@/utils/validators';
import { useToast } from '@/components/ui/Toast';
import { cn } from '@/utils/utils';
```

```tsx
export function ContactForm() {
  const [isSubmitting, setIsSubmitting] = useState(false);
  const { showToast } = useToast();

  const {
    register,
    handleSubmit,
    reset,
    formState: { errors },
  } = useForm<ContactFormData>({
    resolver: zodResolver(contactSchema),
  });

  const onSubmit = async (data: ContactFormData) => {
    try {
      setIsSubmitting(true);

      // Enviar a API endpoint o directamente
      const response = await fetch('/api/contact', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(data),
      });

      if (!response.ok) throw new Error('Error al enviar');

      showToast('¡Mensaje enviado! Te responderemos pronto.', 'success');
      reset();
    } catch (error) {
      console.error('Error:', error);
      showToast('Error al enviar mensaje. Intenta de nuevo.', 'error');
    } finally {
      setIsSubmitting(false);
    }
  };

  return (
    <form onSubmit={handleSubmit(onSubmit)} className="space-y-6">
      <div>
        <label htmlFor="name" className="label">
          Nombre <span className="text-red-500">*</span>
        </label>
        <input
          {...register('name')}
```

```
    id="name"
    type="text"
    className={cn('input', errors.name && 'border-red-500')}
    placeholder="Tu nombre"
  />
  {errors.name && (
    <p className="text-sm text-red-500 mt-1">{errors.name.message}<
  )}
</div>

<div>
  <label htmlFor="email" className="label">
    Email <span className="text-red-500">*</span>
  </label>
  <input
    {...register('email')}
    id="email"
    type="email"
    className={cn('input', errors.email && 'border-red-500')}
    placeholder="tu@email.com"
  />
  {errors.email && (
    <p className="text-sm text-red-500 mt-1">{errors.email.message}
  )}
</div>

<div>
  <label htmlFor="phone" className="label">
    Teléfono (opcional)
  </label>
  <input
    {...register('phone')}
    id="phone"
    type="tel"
    className="input"
    placeholder="999 888 777"
  />
</div>

<div>
  <label htmlFor="subject" className="label">
    Asunto <span className="text-red-500">*</span>
  </label>
  <input
```

```
            {...register('subject')}
            id="subject"
            type="text"
            className={cn('input', errors.subject && 'border-red-500')}
            placeholder="¿En qué podemos ayudarte?"
          />
          {errors.subject && (
            <p className="text-sm text-red-500 mt-1">{errors.subject.messag
          )}
        </div>

        <div>
          <label htmlFor="message" className="label">
            Mensaje <span className="text-red-500">*</span>
          </label>
          <textarea
            {...register('message')}
            id="message"
            rows={6}
            className={cn('input resize-none', errors.message && 'border-re
            placeholder="Escribe tu mensaje aquí..."
          />
          {errors.message && (
            <p className="text-sm text-red-500 mt-1">{errors.message.messag
          )}
        </div>

        <button
          type="submit"
          disabled={isSubmitting}
          className="btn-primary w-full"
        >
          {isSubmitting ? 'Enviando...' : 'Enviar Mensaje'}
        </button>
      </form>
    );
}
```

✅ CHECKLIST - Componentes React

Verifica que hayas creado:

Componentes interactivos:

- [ ] Modal.tsx con animaciones
- [ ] Toast.tsx con ToastProvider y useToast hook
- [ ] QuoteForm.tsx con validación completa
- [ ] Hero.tsx con animaciones y WhatsApp button
- [ ] TestimonialsSlider.tsx con navegación
- [ ] ContactForm.tsx

Integración:

- [ ] ToastProvider agregado a BaseLayout
- [ ] Componentes funcionan con `client:load`
- [ ] Formularios envían datos a Supabase
- [ ] Toast muestra mensajes correctamente

Testing:

- [ ] Modal abre y cierra correctamente
- [ ] Toast aparece y desaparece automáticamente
- [ ] QuoteForm valida campos
- [ ] Hero anima al cargar
- [ ] Testimonials slider navega correctamente

📝 NOTAS IMPORTANTES

Directiva client: en Astro

Todos los componentes React deben usar directivas client:

```
<!-- client:load - Carga inmediatamente (para componentes críticos) -->
<QuoteForm client:load />

<!-- client:idle - Carga cuando el navegador está idle (recomendado) -->
<TestimonialsSlider client:idle testimonials={data} />

<!-- client:visible - Carga cuando es visible (lazy loading) -->
<ContactForm client:visible />
```

```
<!-- client:only - Solo en cliente (no SSR) -->
<ComplexComponent client:only="react" />
```

Props en componentes Astro → React

```
---
// Obtener datos en Astro (servidor)
const testimonials = await getTestimonials();
---

<!-- Pasar a React -->
<TestimonialsSlider
  client:load
  testimonials={testimonials}
/>
```

➡️ CONTINÚA CON

→ Archivo 05: Páginas Principales del Sitio

- Página Home completa
- Página Servicios
- Página Portafolio
- Página Nosotros
- Página Contacto
