💻 LA RESERVA - DESARROLLO WEB

PARTE D: Utilidades y Mejores Prácticas

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

📑 CONTENIDO

---

1. ARCHIVO utils.ts

```
// src/utils/utils.ts

import { type ClassValue, clsx } from 'clsx';
import { twMerge } from 'tailwind-merge';

/**
 * Combina clases de Tailwind sin conflictos
 * Usa clsx para lógica condicional y twMerge para resolver conflictos
 *
 * @example
 * cn('px-4', 'px-6') → 'px-6'
 * cn('btn', isActive && 'btn-active') → 'btn btn-active'
 */
export function cn(...inputs: ClassValue[]) {
```

```
  return twMerge(clsx(inputs));
}

/**
 * Espera un tiempo determinado (útil para debugging o delays)
 *
 * @param ms - Milisegundos a esperar
 * @example
 * await sleep(1000); // Espera 1 segundo
 */
export function sleep(ms: number): Promise<void> {
  return new Promise(resolve => setTimeout(resolve, ms));
}

/**
 * Genera un slug URL-friendly desde un string
 *
 * @param text - Texto a convertir
 * @example
 * slugify('Boda en Lima 2025') → 'boda-en-lima-2025'
 */
export function slugify(text: string): string {
  return text
    .toString()
    .toLowerCase()
    .normalize('NFD')                   // Normaliza caracteres especiale
    .replace(/[\u0300-\u036f]/g, '')    // Elimina diacríticos
    .replace(/[^a-z0-9\s-]/g, '')       // Solo letras, números, espacios
    .trim()
    .replace(/\s+/g, '-')               // Reemplaza espacios con guiones
    .replace(/-+/g, '-');               // Elimina guiones múltiples
}

/**
 * Capitaliza la primera letra de cada palabra
 *
 * @param text - Texto a capitalizar
 * @example
 * capitalize('juan pérez') → 'Juan Pérez'
 */
export function capitalize(text: string): string {
  return text
    .toLowerCase()
    .split(' ')
```

```
    .map(word => word.charAt(0).toUpperCase() + word.slice(1))
    .join(' ');
}

/**
 * Trunca un texto a cierta longitud agregando ellipsis
 *
 * @param text - Texto a truncar
 * @param length - Longitud máxima
 * @example
 * truncate('Este es un texto muy largo', 10) → 'Este es un...'
 */
export function truncate(text: string, length: number): string {
  if (text.length <= length) return text;
  return text.slice(0, length).trim() + '...';
}

/**
 * Valida si un email es válido (básico)
 * Para validación más robusta, usar Zod en validators.ts
 *
 * @param email - Email a validar
 */
export function isValidEmail(email: string): boolean {
  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  return emailRegex.test(email);
}

/**
 * Valida si un teléfono peruano es válido
 * Formato: +51 seguido de 9 dígitos (912345678)
 *
 * @param phone - Teléfono a validar
 */
export function isValidPeruvianPhone(phone: string): boolean {
  // Formato: +51999888777 o 999888777 o 51999888777
  const phoneRegex = /^(\+?51)?9\d{8}$/;
  return phoneRegex.test(phone.replace(/\s/g, ''));
}

/**
 * Formatea un número de teléfono peruano
 *
 * @param phone - Teléfono a formatear
```

```
 * @example
 * formatPhone('999888777') → '+51 999 888 777'
 */
export function formatPhone(phone: string): string {
  const cleaned = phone.replace(/\D/g, '');

  // Si empieza con 51, eliminarlo
  const withoutCountryCode = cleaned.startsWith('51')
    ? cleaned.slice(2)
    : cleaned;

  // Formatear: +51 999 888 777
  if (withoutCountryCode.length === 9) {
    return `+51 ${withoutCountryCode.slice(0, 3)} ${withoutCountryCode.sl
  }

  return phone; // Si no es válido, retornar original
}

/**
 * Genera un ID único (nanoid alternativo simple)
 * Para producción, usar nanoid o UUID
 */
export function generateId(): string {
  return `${Date.now()}-${Math.random().toString(36).substr(2, 9)}`;
}

/**
 * Debounce una función (útil para búsquedas)
 *
 * @param func - Función a ejecutar
 * @param wait - Milisegundos a esperar
 */
export function debounce<T extends (...args: any[]) => any>(
  func: T,
  wait: number
): (...args: Parameters<T>) => void {
  let timeout: NodeJS.Timeout;

  return function executedFunction(...args: Parameters<T>) {
    const later = () => {
      clearTimeout(timeout);
      func(...args);
    };
```

```
    clearTimeout(timeout);
    timeout = setTimeout(later, wait);
  };
}

/**
 * Obtiene el saludo apropiado según la hora
 */
export function getGreeting(): string {
  const hour = new Date().getHours();

  if (hour < 12) return 'Buenos días';
  if (hour < 19) return 'Buenas tardes';
  return 'Buenas noches';
}

/**
 * Valida si una fecha es futura
 */
export function isFutureDate(date: Date | string): boolean {
  const dateObj = typeof date === 'string' ? new Date(date) : date;
  return dateObj > new Date();
}

/**
 * Obtiene días entre dos fechas
 */
export function getDaysBetween(date1: Date, date2: Date): number {
  const oneDay = 24 * 60 * 60 * 1000; // milisegundos en un día
  return Math.round(Math.abs((date1.getTime() - date2.getTime()) / oneDay
}

/**
 * Scroll suave a un elemento
 */
export function scrollToElement(elementId: string, offset: number = 80) {
  const element = document.getElementById(elementId);
  if (element) {
    const elementPosition = element.getBoundingClientRect().top;
    const offsetPosition = elementPosition + window.pageYOffset - offset;

    window.scrollTo({
      top: offsetPosition,
```

```
      behavior: 'smooth'
    });
  }
}

/**
 * Copia texto al clipboard
 */
export async function copyToClipboard(text: string): Promise<boolean> {
  try {
    await navigator.clipboard.writeText(text);
    return true;
  } catch (err) {
    console.error('Error al copiar:', err);
    return false;
  }
}

/**
 * Obtiene parámetros de URL
 */
export function getUrlParams<T extends Record<string, string>>(): T {
  const params = new URLSearchParams(window.location.search);
  const result: Record<string, string> = {};

  params.forEach((value, key) => {
    result[key] = value;
  });

  return result as T;
}

/**
 * Detecta si está en mobile
 */
export function isMobile(): boolean {
  return /Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|Opera Mini/i
    navigator.userAgent
  );
}
```

---

2. ARCHIVO constants.ts

```ts
// src/utils/constants.ts

/**
 * URLs y endpoints
 */
export const SITE_URL = import.meta.env.PUBLIC_SITE_URL || 'https://lares
export const API_URL = `${SITE_URL}/api`;

/**
 * Información de contacto
 */
export const CONTACT_INFO = {
  phone: '+51999888777',
  phoneFormatted: '+51 999 888 777',
  email: 'contacto@lareserva.pe',
  whatsapp: '+51999888777',
  whatsappUrl: 'https://wa.me/51999888777',
  address: 'Lima, Perú',
  addressFull: 'Lima Metropolitana, Perú',
} as const;

/**
 * Horarios de atención
 */
export const BUSINESS_HOURS = {
  weekdays: 'Lunes - Viernes: 9:00 AM - 5:00 PM',
  saturday: 'Sábado: 9:00 AM - 1:00 PM',
  sunday: 'Domingo: Cerrado',
  responseTime: 'Respuestas dentro de 1 hora',
} as const;

/**
 * Redes sociales
 */
export const SOCIAL_LINKS = {
  instagram: 'https://instagram.com/lareservabar',
  facebook: 'https://facebook.com/lareservabar',
  tiktok: 'https://tiktok.com/@lareserva',
} as const;

/**
 * Tipos de eventos disponibles
 */
export const EVENT_TYPES = [
```

```
  { value: 'boda', label: 'Boda' },
  { value: 'corporativo', label: 'Evento Corporativo' },
  { value: 'cumpleanos', label: 'Cumpleaños' },
  { value: 'aniversario', label: 'Aniversario' },
  { value: 'graduacion', label: 'Graduación' },
  { value: 'baby-shower', label: 'Baby Shower' },
  { value: 'otro', label: 'Otro' },
] as const;

/**
 * Rangos de invitados
 */
export const GUEST_RANGES = [
  { value: '25-50', label: '25 - 50 invitados' },
  { value: '51-100', label: '51 - 100 invitados' },
  { value: '101-200', label: '101 - 200 invitados' },
  { value: '201-300', label: '201 - 300 invitados' },
  { value: '301-500', label: '301 - 500 invitados' },
  { value: '500+', label: 'Más de 500 invitados' },
] as const;

/**
 * Límites de invitados
 */
export const GUEST_LIMITS = {
  min: 25,
  max: 500,
  recommended: 100,
} as const;

/**
 * Estados de cotizaciones
 */
export const QUOTE_STATUSES = {
  new: { label: 'Nueva', color: 'blue' },
  contacted: { label: 'Contactada', color: 'yellow' },
  quoted: { label: 'Cotizada', color: 'purple' },
  converted: { label: 'Convertida', color: 'green' },
  declined: { label: 'Declinada', color: 'red' },
} as const;

/**
 * Estados de eventos
 */
```

```
export const EVENT_STATUSES = {
  pending: { label: 'Pendiente', color: 'yellow' },
  confirmed: { label: 'Confirmado', color: 'green' },
  completed: { label: 'Completado', color: 'blue' },
  cancelled: { label: 'Cancelado', color: 'red' },
} as const;

/**
 * Servicios principales
 */
export const SERVICES = [
  {
    id: 'bartending-eventos',
    name: 'Bartending para Eventos',
    slug: 'bartending-eventos',
    description: 'Servicio completo de barra y bartenders profesionales p
    priceFrom: 1800,
    icon: 'cocktail',
    features: [
      'Bartenders profesionales certificados',
      'Barra completa equipada',
      'Cristalería premium',
      'Ingredientes frescos y garnish',
      'Setup y decoración de barra',
      'Servicio durante todo el evento',
    ],
  },
  {
    id: 'mixologia-corporativa',
    name: 'Mixología Corporativa',
    slug: 'mixologia-corporativa',
    description: 'Experiencia de coctelería personalizada para eventos em
    priceFrom: 2500,
    icon: 'briefcase',
    features: [
      'Cócteles signature con branding',
      'Presentación profesional',
      'Team building de mixología',
      'Barra corporativa premium',
      'Material promocional incluido',
    ],
  },
  {
    id: 'cocteles-autor',
```

```
    name: 'Cócteles de Autor',
    slug: 'cocteles-autor',
    description: 'Creación de cócteles exclusivos diseñados especialmente
    priceFrom: 2200,
    icon: 'sparkles',
    features: [
      'Consulta previa personalizada',
      'Receta exclusiva creada para ti',
      'Ingredientes premium seleccionados',
      'Técnicas artesanales',
      'Presentación impecable',
    ],
  },
  {
    id: 'barra-movil',
    name: 'Barra Móvil Premium',
    slug: 'barra-movil',
    description: 'Barra móvil completamente equipada con todo lo necesari
    priceFrom: 800,
    icon: 'truck',
    features: [
      'Barra portátil elegante',
      'Equipo completo de bartending',
      'Decoración incluida',
      'Setup y desmontaje',
      'Variedad de diseños disponibles',
    ],
  },
] as const;

/**
 * Paquetes predefinidos
 */
export const PACKAGES = [
  {
    id: 'basico',
    name: 'Paquete Básico',
    slug: 'basico',
    description: 'Ideal para eventos íntimos',
    price: 1800,
    guestRange: '25-50',
    duration: 4,
    features: [
      '1 bartender profesional',
```

```
      'Barra básica equipada',
      '3 cócteles a elegir',
      'Cristalería y hielo',
      'Setup y limpieza',
    ],
    popular: false,
  },
  {
    id: 'completo',
    name: 'Paquete Completo',
    slug: 'completo',
    description: 'Perfecto para bodas y eventos medianos',
    price: 3500,
    guestRange: '100-200',
    duration: 5,
    features: [
      '2 bartenders profesionales',
      'Barra premium equipada',
      '5 cócteles de autor',
      'Cristalería premium',
      'Decoración de barra',
      'Garnish artístico',
      'Setup y limpieza completa',
    ],
    popular: true,
  },
  {
    id: 'premium',
    name: 'Paquete Premium',
    slug: 'premium',
    description: 'Experiencia exclusiva para eventos grandes',
    price: 6500,
    guestRange: '200-500',
    duration: 6,
    features: [
      '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',
```

```
    ],
    popular: false,
  },
] as const;

/**
 * Cócteles destacados (para portafolio)
 */
export const FEATURED_COCKTAILS = [
  {
    name: 'Pisco Sour Reserva',
    description: 'Nuestro clásico peruano con un toque especial',
    category: 'Clásicos',
  },
  {
    name: 'Old Fashioned Ahumado',
    description: 'Bourbon premium con ahumado artesanal',
    category: 'Clásicos',
  },
  {
    name: 'Margarita de Maracuyá',
    description: 'Fusión tropical con maracuyá fresco',
    category: 'Tropicales',
  },
] as const;

/**
 * Metadata por defecto para SEO
 */
export const DEFAULT_SEO = {
  title: 'La Reserva - Mixología Exclusiva',
  description: 'Bartending premium para eventos exclusivos en Lima, Perú.
  keywords: [
    'bartending lima',
    'mixología perú',
    'eventos premium',
    'cócteles de autor',
    'bartender para bodas',
    'servicio de bar',
    'coctelería lima',
  ],
  ogImage: '/images/og-image.jpg',
} as const;
```

```
/**
 * Configuración de validación
 */
export const VALIDATION = {
  name: {
    min: 2,
    max: 100,
  },
  email: {
    max: 255,
  },
  phone: {
    min: 9,
    max: 15,
  },
  message: {
    min: 10,
    max: 1000,
  },
  guests: {
    min: GUEST_LIMITS.min,
    max: GUEST_LIMITS.max,
  },
} as const;

/**
 * Mensajes de error comunes
 */
export const ERROR_MESSAGES = {
  required: 'Este campo es obligatorio',
  invalidEmail: 'Email inválido',
  invalidPhone: 'Teléfono inválido',
  minLength: (min: number) => `Mínimo ${min} caracteres`,
  maxLength: (max: number) => `Máximo ${max} caracteres`,
  minValue: (min: number) => `Valor mínimo: ${min}`,
  maxValue: (max: number) => `Valor máximo: ${max}`,
  invalidDate: 'Fecha inválida',
  pastDate: 'La fecha debe ser futura',
  generic: 'Ocurrió un error. Por favor intenta de nuevo.',
} as const;

/**
 * Mensajes de éxito
 */
```

```
export const SUCCESS_MESSAGES = {
  quoteSubmitted: '¡Gracias! Tu cotización ha sido enviada. Te contactare
  contactSubmitted: '¡Mensaje enviado! Te responderemos a la brevedad.',
  subscribed: '¡Suscripción exitosa! Recibirás nuestras novedades.',
  copied: 'Copiado al portapapeles',
} as const;
```

---

3. ARCHIVO validators.ts

```
// src/utils/validators.ts
import { z } from 'zod';
import { VALIDATION, ERROR_MESSAGES } from './constants';

/**
 * Schema para formulario de cotización
 */
export const quoteSchema = z.object({
  name: z
    .string()
    .min(VALIDATION.name.min, ERROR_MESSAGES.minLength(VALIDATION.name.mi
    .max(VALIDATION.name.max, ERROR_MESSAGES.maxLength(VALIDATION.name.ma

  email: z
    .string()
    .email(ERROR_MESSAGES.invalidEmail)
    .max(VALIDATION.email.max),

  phone: z
    .string()
    .min(VALIDATION.phone.min, ERROR_MESSAGES.minLength(VALIDATION.phone.
    .max(VALIDATION.phone.max, ERROR_MESSAGES.maxLength(VALIDATION.phone.
    .regex(/^(\+?51)?9\d{8}$/, ERROR_MESSAGES.invalidPhone),

  eventType: z
    .string()
    .min(1, ERROR_MESSAGES.required),

  eventDate: z
    .string()
    .min(1, ERROR_MESSAGES.required)
    .refine((date) => {
```

```
      const eventDate = new Date(date);
      const today = new Date();
      today.setHours(0, 0, 0, 0);
      return eventDate >= today;
    }, ERROR_MESSAGES.pastDate),

  guestCount: z
    .number()
    .min(VALIDATION.guests.min, ERROR_MESSAGES.minValue(VALIDATION.guests
    .max(VALIDATION.guests.max, ERROR_MESSAGES.maxValue(VALIDATION.guests

  message: z
    .string()
    .min(VALIDATION.message.min, ERROR_MESSAGES.minLength(VALIDATION.mess
    .max(VALIDATION.message.max, ERROR_MESSAGES.maxLength(VALIDATION.mess
    .optional(),
});

export type QuoteFormData = z.infer<typeof quoteSchema>;

/**
 * Schema para formulario de contacto
 */
export const contactSchema = z.object({
  name: z
    .string()
    .min(VALIDATION.name.min, ERROR_MESSAGES.minLength(VALIDATION.name.mi
    .max(VALIDATION.name.max, ERROR_MESSAGES.maxLength(VALIDATION.name.ma

  email: z
    .string()
    .email(ERROR_MESSAGES.invalidEmail),

  phone: z
    .string()
    .optional(),

  subject: z
    .string()
    .min(3, ERROR_MESSAGES.minLength(3))
    .max(200, ERROR_MESSAGES.maxLength(200)),

  message: z
    .string()
```

```
    .min(VALIDATION.message.min, ERROR_MESSAGES.minLength(VALIDATION.mess
    .max(VALIDATION.message.max, ERROR_MESSAGES.maxLength(VALIDATION.mess
});

export type ContactFormData = z.infer<typeof contactSchema>;

/**
 * Schema para newsletter
 */
export const newsletterSchema = z.object({
  email: z
    .string()
    .email(ERROR_MESSAGES.invalidEmail),
});

export type NewsletterFormData = z.infer<typeof newsletterSchema>;

/**
 * Schema para login de admin
 */
export const loginSchema = z.object({
  email: z
    .string()
    .email(ERROR_MESSAGES.invalidEmail),

  password: z
    .string()
    .min(6, ERROR_MESSAGES.minLength(6)),
});

export type LoginFormData = z.infer<typeof loginSchema>;

/**
 * Schema para crear/editar evento (admin)
 */
export const eventSchema = z.object({
  clientName: z
    .string()
    .min(VALIDATION.name.min),

  clientEmail: z
    .string()
    .email(),
```

```
  clientPhone: z
    .string()
    .min(VALIDATION.phone.min),

  eventType: z
    .string()
    .min(1),

  eventDate: z
    .string()
    .min(1),

  eventTime: z
    .string()
    .optional(),

  guestCount: z
    .number()
    .min(VALIDATION.guests.min)
    .max(VALIDATION.guests.max),

  venue: z
    .string()
    .optional(),

  packageId: z
    .string()
    .optional(),

  totalPrice: z
    .number()
    .min(0)
    .optional(),

  deposit: z
    .number()
    .min(0)
    .optional(),

  status: z
    .enum(['pending', 'confirmed', 'completed', 'cancelled']),

  notes: z
    .string()
```

```
    .optional(),
});

export type EventFormData = z.infer<typeof eventSchema>;
```

---

4. ARCHIVO formatters.ts

```
// src/utils/formatters.ts
import { format, formatDistanceToNow, isToday, isYesterday, isTomorrow }
import { es } from 'date-fns/locale';

/**
 * Formatea una fecha a formato legible en español
 *
 * @example
 * formatDate(new Date()) → '22 de octubre de 2025'
 */
export function formatDate(date: Date | string, formatStr: string = 'PPP'
  const dateObj = typeof date === 'string' ? new Date(date) : date;
  return format(dateObj, formatStr, { locale: es });
}

/**
 * Formatea una fecha de manera relativa (hace X tiempo)
 *
 * @example
 * formatRelativeDate(yesterday) → 'hace 1 día'
 */
export function formatRelativeDate(date: Date | string): string {
  const dateObj = typeof date === 'string' ? new Date(date) : date;

  if (isToday(dateObj)) return 'Hoy';
  if (isYesterday(dateObj)) return 'Ayer';
  if (isTomorrow(dateObj)) return 'Mañana';

  return formatDistanceToNow(dateObj, { addSuffix: true, locale: es });
}

/**
 * Formatea moneda en soles peruanos
 *
```

```
 * @example
 * formatCurrency(1500) → 'S/ 1,500'
 */
export function formatCurrency(amount: number): string {
  return new Intl.NumberFormat('es-PE', {
    style: 'currency',
    currency: 'PEN',
    minimumFractionDigits: 0,
    maximumFractionDigits: 0,
  }).format(amount);
}

/**
 * Formatea número con separadores de miles
 *
 * @example
 * formatNumber(1500) → '1,500'
 */
export function formatNumber(num: number): string {
  return new Intl.NumberFormat('es-PE').format(num);
}

/**
 * Formatea un nombre completo (capitaliza)
 *
 * @example
 * formatName('juan pérez') → 'Juan Pérez'
 */
export function formatName(name: string): string {
  return name
    .toLowerCase()
    .split(' ')
    .map(word => word.charAt(0).toUpperCase() + word.slice(1))
    .join(' ');
}

/**
 * Formatea hora a formato 12h
 *
 * @example
 * formatTime('14:30') → '2:30 PM'
 */
export function formatTime(time: string): string {
  const [hours, minutes] = time.split(':').map(Number);
```

```
  const period = hours >= 12 ? 'PM' : 'AM';
  const displayHours = hours % 12 || 12;
  const displayMinutes = minutes.toString().padStart(2, '0');
  return `${displayHours}:${displayMinutes} ${period}`;
}

/**
 * Formatea duración en horas
 *
 * @example
 * formatDuration(5) → '5 horas'
 */
export function formatDuration(hours: number): string {
  if (hours === 1) return '1 hora';
  return `${hours} horas`;
}

/**
 * Formatea rango de invitados
 *
 * @example
 * formatGuestRange('100-200') → '100 - 200 invitados'
 */
export function formatGuestRange(range: string): string {
  if (range.includes('+')) {
    return `Más de ${range.replace('+', '')} invitados`;
  }
  return `${range.replace('-', ' - ')} invitados`;
}

/**
 * Extrae iniciales de un nombre
 *
 * @example
 * getInitials('Juan Pérez') → 'JP'
 */
export function getInitials(name: string): string {
  return name
    .split(' ')
    .map(word => word[0])
    .join('')
    .toUpperCase()
    .slice(0, 2);
}
```

5. ARCHIVO whatsapp.ts

```
// src/utils/whatsapp.ts
import { CONTACT_INFO } from './constants';

/**
 * Genera URL de WhatsApp con mensaje predefinido
 *
 * @param message - Mensaje a enviar
 * @param phone - Número de teléfono (opcional, usa el de contacto por de
 */
export function getWhatsAppUrl(message: string, phone?: string): string {
  const phoneNumber = phone || CONTACT_INFO.whatsapp.replace(/\D/g, '');
  const encodedMessage = encodeURIComponent(message);
  return `https://wa.me/${phoneNumber}?text=${encodedMessage}`;
}

/**
 * Genera mensaje de WhatsApp para cotización
 */
export function getQuoteWhatsAppMessage(data: {
  name: string;
  eventType: string;
  eventDate: string;
  guestCount: number;
}): string {
  return `Hola! Me gustaría solicitar una cotización:

👤 Nombre: ${data.name}
🎉 Tipo de evento: ${data.eventType}
📅 Fecha: ${data.eventDate}
👥 Cantidad de invitados: ${data.guestCount}

¿Podrían enviarme más información?`;
}

/**
 * Genera mensaje de WhatsApp para contacto general
 */
export function getContactWhatsAppMessage(name?: string): string {
  const greeting = name ? `Hola! Soy ${name}. ` : 'Hola! ';
```

```
  return `${greeting}Me gustaría obtener más información sobre sus servic
}

/**
 * Genera mensaje de WhatsApp para consulta de disponibilidad
 */
export function getAvailabilityWhatsAppMessage(date: string): string {
  return `Hola! Me gustaría consultar disponibilidad para el ${date}.`;
}

/**
 * Abre WhatsApp en nueva ventana/tab
 */
export function openWhatsApp(message: string, phone?: string): void {
  const url = getWhatsAppUrl(message, phone);
  window.open(url, '_blank', 'noopener,noreferrer');
}
```

---

6. MEJORES PRÁCTICAS DE CÓDIGO

6.1 Estructura de Componentes

✅ Componente bien estructurado:

```
---
// src/components/ServiceCard.astro

// 1. Imports
import { Image } from 'astro:assets';
import Button from './ui/Button.astro';

// 2. Props interface
interface Props {
  title: string;
  description: string;
  image?: string;
  price?: number;
  features?: string[];
}

// 3. Destructure props
```

```
const { title, description, image, price, features = [] } = Astro.props;

// 4. Logic/computations
const hasFeatures = features.length > 0;
---

<!-- 5. Template -->
<div class="card">
  {image && (
    <img src={image} alt={title} class="w-full h-48 object-cover" />
  )}

  <div class="p-6">
    <h3 class="text-2xl font-display mb-2">{title}</h3>
    <p class="text-secondary-400 mb-4">{description}</p>

    {hasFeatures && (
      <ul class="space-y-2 mb-6">
        {features.map(feature => (
          <li class="flex items-start gap-2">
            <span class="text-primary-500">✓</span>
            <span>{feature}</span>
          </li>
        ))}
      </ul>
    )}

    {price && (
      <p class="text-xl font-semibold text-primary-500 mb-4">
        Desde S/ {price}
      </p>
    )}

    <Button variant="primary">Ver más</Button>
  </div>
</div>
```

6.2 Manejo de Estado en React

✅ Custom Hooks para lógica reutilizable:

```
// src/hooks/useQuoteForm.ts
import { useState } from 'react';
```

```
import { useForm } from 'react-hook-form';
import { zodResolver } from '@hookform/resolvers/zod';
import { quoteSchema, type QuoteFormData } from '@/utils/validators';
import { supabase } from '@/lib/supabase';

export function useQuoteForm() {
  const [isSubmitting, setIsSubmitting] = useState(false);
  const [error, setError] = useState<string | null>(null);
  const [success, setSuccess] = useState(false);

  const form = useForm<QuoteFormData>({
    resolver: zodResolver(quoteSchema),
    defaultValues: {
      name: '',
      email: '',
      phone: '',
      eventType: '',
      eventDate: '',
      guestCount: 100,
      message: '',
    },
  });

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

      const { error: supabaseError } = 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 (supabaseError) throw supabaseError;

      setSuccess(true);
```

```
      form.reset();
    } catch (err) {
      setError('Error al enviar cotización. Por favor intenta de nuevo.')
      console.error(err);
    } finally {
      setIsSubmitting(false);
    }
  };

  return {
    form,
    isSubmitting,
    error,
    success,
    onSubmit: form.handleSubmit(onSubmit),
  };
}
```

6.3 Manejo de Errores

✅ Try-catch en operaciones async:

```
// API endpoint
export const POST: APIRoute = async ({ request }) => {
  try {
    const body = await request.json();
    const validatedData = quoteSchema.parse(body);

    const { data, error } = await supabase
      .from('quotes')
      .insert(validatedData);

    if (error) {
      return new Response(
        JSON.stringify({ error: error.message }),
        { status: 400, headers: { 'Content-Type': 'application/json' } }
      );
    }

    return new Response(
      JSON.stringify({ success: true, data }),
      { status: 201, headers: { 'Content-Type': 'application/json' } }
    );
```

```
  } catch (error) {
    console.error('Error in POST /api/quotes:', error);

    return new Response(
      JSON.stringify({ error: 'Invalid request data' }),
      { status: 400, headers: { 'Content-Type': 'application/json' } }
    );
  }
};
```

6.4 Optimización de Imágenes

✅ Usar componente Image de Astro:

```
---
import { Image } from 'astro:assets';
import heroImage from '@/assets/images/hero.jpg';
---

<!-- Optimizado automáticamente -->
<Image
  src={heroImage}
  alt="Evento de La Reserva"
  width={1200}
  height={600}
  format="webp"
  quality={80}
  loading="lazy"
/>

<!-- Para imágenes de public/ -->
<img
  src="/images/logo.svg"
  alt="La Reserva"
  width="200"
  height="100"
  loading="lazy"
/>
```

6.5 SEO en cada página

✅ **Metadata completo:**

```
---
// src/pages/servicios.astro
import PageLayout from '@/layouts/PageLayout.astro';

const seo = {
  title: 'Servicios de Bartending Premium',
  description: 'Descubre nuestros servicios de bartending y mixología exc
  image: '/images/servicios-og.jpg',
};
---

<PageLayout {...seo}>
  <!-- Contenido -->
</PageLayout>
```

6.6 Accesibilidad

✅ **Siempre incluir:**

```
<!-- Alt text descriptivo -->
<img src="..." alt="Bartender preparando un Pisco Sour con garnish de lim

<!-- ARIA labels en botones de iconos -->
<button aria-label="Abrir menú de navegación">
  <svg>...</svg>
</button>

<!-- Labels en formularios -->
<label for="email" class="label">Email</label>
<input id="email" type="email" class="input" />

<!-- Contraste adecuado -->
<div class="bg-secondary-600 text-white">
  <!-- Contraste 21:1 ✅ -->
</div>

<!-- Keyboard navigation -->
<button
  @click="handleClick"
  @keydown.enter="handleClick"
```

```
  @keydown.space="handleClick"
>
  Enviar
</button>
```

---

7. CONVENCIONES Y ESTÁNDARES

7.1 Git Commits

✅ Formato recomendado:

```
# Estructura: <tipo>: <descripción breve>

# Tipos:
feat: Nueva funcionalidad
fix: Corrección de bug
docs: Documentación
style: Formato, no afecta lógica
refactor: Refactorización de código
test: Tests
chore: Tareas de mantenimiento

# Ejemplos:
git commit -m "feat: agregar formulario de cotización"
git commit -m "fix: corregir validación de teléfono"
git commit -m "docs: actualizar README con instrucciones"
git commit -m "style: formatear componente Header"
git commit -m "refactor: extraer lógica de validación a utils"
```

7.2 Comentarios en Código

✅ Cuándo comentar:

```
// ✅ BUENO: Explica el "por qué"
// Usamos setTimeout porque el DOM necesita renderizar primero
setTimeout(() => scrollToElement('contact'), 100);

// ✅ BUENO: Documenta funciones complejas
/**
 * Calcula el precio estimado basado en invitados y duración
```

```
 * Incluye descuento por volumen y recargo por servicio nocturno
 */
function calculatePrice(guests: number, hours: number): number {
  // ...
}

// ❌ MALO: Explica el "qué" (obvio)
// Incrementar contador
count++;

// ❌ MALO: Código comentado (usar Git)
// const oldFunction = () => { ... }
```

7.3 Organización de Imports

✅ Orden recomendado:

```
// 1. React/Astro core
import { useState, useEffect } from 'react';

// 2. Librerías externas
import { useForm } from 'react-hook-form';
import { format } from 'date-fns';

// 3. Alias paths internos (@/)
import { supabase } from '@/lib/supabase';
import { Button } from '@/components/ui/Button';
import { formatCurrency } from '@/utils/formatters';

// 4. Tipos
import type { Event } from '@/types';

// 5. Estilos (si aplica)
import '@/styles/custom.css';
```

7.4 Constantes vs Magic Numbers

❌ MALO: Magic numbers

```
if (guests > 500) { // ¿Por qué 500?
  return 'Evento muy grande';
```

```
}

setTimeout(callback, 3000); // ¿Por qué 3000?
```

✅ **BUENO: Constantes con nombre**

```
const MAX_GUESTS = 500;
const TOAST_DURATION = 3000;

if (guests > MAX_GUESTS) {
  return 'Evento muy grande';
}

setTimeout(callback, TOAST_DURATION);
```

---

8. COMANDOS ÚTILES

8.1 Desarrollo

```
# Iniciar servidor de desarrollo
pnpm dev

# Abrir en puerto específico
pnpm dev --port 3001

# Abrir en red local
pnpm dev --host

# Limpiar cache y reiniciar
rm -rf node_modules/.astro && pnpm dev
```

8.2 Build y Preview

```
# Build para producción
pnpm build

# Preview del build localmente
pnpm preview
```

```
# Build y preview en un comando
pnpm build && pnpm preview
```

8.3 Type Checking

```
# Verificar tipos TypeScript
pnpm type-check

# Verificar en modo watch
pnpm type-check --watch

# Verificar solo un archivo
npx tsc --noEmit src/components/QuoteForm.tsx
```

8.4 Formateo y Linting

```
# Formatear todo el proyecto
pnpm format

# Solo verificar formato (no modificar)
pnpm format:check

# Formatear archivo específico
npx prettier --write src/pages/index.astro

# Ver qué archivos necesitan formato
npx prettier --check .
```

8.5 Gestión de Dependencias

```
# Agregar dependencia
pnpm add <package>

# Agregar dev dependency
pnpm add -D <package>

# Eliminar dependencia
pnpm remove <package>
```

```
# Actualizar dependencias
pnpm update

# Actualizar dependencia específica
pnpm update <package>

# Ver dependencias desactualizadas
pnpm outdated

# Limpiar node_modules y reinstalar
rm -rf node_modules pnpm-lock.yaml && pnpm install
```

8.6 Git Workflow

```
# Crear rama para nueva feature
git checkout -b feat/quote-form

# Hacer commits atómicos
git add src/components/forms/QuoteForm.tsx
git commit -m "feat: crear componente QuoteForm"

# Push a GitHub
git push origin feat/quote-form

# Actualizar desde main
git checkout main
git pull origin main
git checkout feat/quote-form
git merge main

# Squash commits antes de merge (opcional)
git rebase -i HEAD~3
```

8.7 Vercel Deployment

```
# Instalar Vercel CLI
pnpm add -g vercel

# Login
vercel login
```

```
# Deploy preview
vercel

# Deploy a producción
vercel --prod

# Ver logs de deployment
vercel logs

# Ver información del proyecto
vercel inspect
```

✅ CHECKLIST COMPLETO - 02D

Antes de continuar con el siguiente archivo, verifica:

Archivos de utilidades creados:

- [ ] `src/utils/utils.ts` - Funciones generales
- [ ] `src/utils/constants.ts` - Constantes del proyecto
- [ ] `src/utils/validators.ts` - Schemas de Zod
- [ ] `src/utils/formatters.ts` - Formateo de datos
- [ ] `src/utils/whatsapp.ts` - Helpers de WhatsApp

TypeScript:

- [ ] Tipos definidos en cada archivo
- [ ] Exports e imports correctos
- [ ] No hay errores de tipo (`pnpm type-check`)

Testing de utilidades:

- [ ] `cn()` combina clases correctamente
- [ ] Validadores de Zod funcionan
- [ ] Formateo de fechas en español
- [ ] Formateo de moneda correcto (S/)
- [ ] URLs de WhatsApp se generan bien

Mejores prácticas implementadas:

- [ ] Comentarios JSDoc en funciones
- [ ] Manejo de errores con try-catch
- [ ] Constantes en lugar de magic numbers
- [ ] Nombres descriptivos de variables
- [ ] Código formateado con Prettier

🎯 PRÓXIMOS PASOS

Has completado la configuración de utilidades y mejores prácticas. Ahora continúa con:

→ Archivo 03: Base de Datos y Backend (Supabase)

- Diseño de tablas
- Row Level Security (RLS)
- Edge Functions
- Seeds iniciales
