💻 LA RESERVA - DESARROLLO WEB

PARTE A: Introducción y Stack Tecnológico

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

📑 CONTENIDO DE ESTE ARCHIVO

---

1. INTRODUCCIÓN AL PROYECTO

🎯 Objetivos del Sitio Web

Desarrollar un sitio web premium para **La Reserva**, empresa de bartending y mixología exclusiva en Lima, Perú.

Objetivos de Negocio:

✅ Generar leads cualificados (cotizaciones de eventos)
✅ Mostrar portafolio profesional de trabajos realizados
✅ Transmitir elegancia y profesionalismo de la marca
✅ Facilitar la gestión interna con panel administrativo
✅ Aumentar presencia online y visibilidad en buscadores

Objetivos Técnicos:

✅ Performance excepcional (< 2s carga inicial)
✅ SEO optimizado (posicionar en Google para "bartending Lima")
✅ Accesibilidad (WCAG 2.1 AA mínimo)
✅ Responsive design (móvil, tablet, desktop)
✅ Mantenibilidad y escalabilidad del código

🌟 Características Principales

Sitio Público (Para Clientes):

- 🏠 **Home:** Hero impactante + servicios + testimonios
- 🍸 **Servicios:** Detalle de servicios especializados
- 📦 **Paquetes:** Paquetes predefinidos con inforación e imagenes
- 📸 **Portafolio:** Mini-Galería de eventos realizados con filtros y Mini-Galeria de cocteles
- ℹ **Sobre Nosotros:** Historia, misión, visión, equipo
- 📝 **Blog:** Artículos sobre mixología y tendencias
- 📞 **Contacto:** Formulario + mapa + WhatsApp integrado

Panel Administrativo (Para el Equipo):

- 📊 **Dashboard:** Métricas, eventos próximos, cotizaciones pendientes
- 📅 **Calendario:** Vista de todos los eventos (mensual/semanal)
- 📋 **Cotizaciones:** Gestión completa (ver, responder, convertir)
- 🎉 **Eventos:** CRUD de eventos confirmados
- 👥 **Clientes:** Base de datos de clientes y historial
- ✏️ **Contenido:** Editar servicios, blog, testimonios, portafolio
- ⚙️ **Configuración:** Ajustes del sitio, usuarios, precios

🎨 Diseño y Experiencia

Paleta de Colores:

```
Primary:   #D4AF37 (Dorado elegante)
Secondary: #1A1A1A (Negro profundo)
```

```
Accent:    #F59E0B (Dorado brillante)
Neutrals:  Grises para texto y backgrounds
```

Tipografía:

```
Display: Playfair Display (títulos, logo)
Body:    Montserrat (texto general, UI)
```

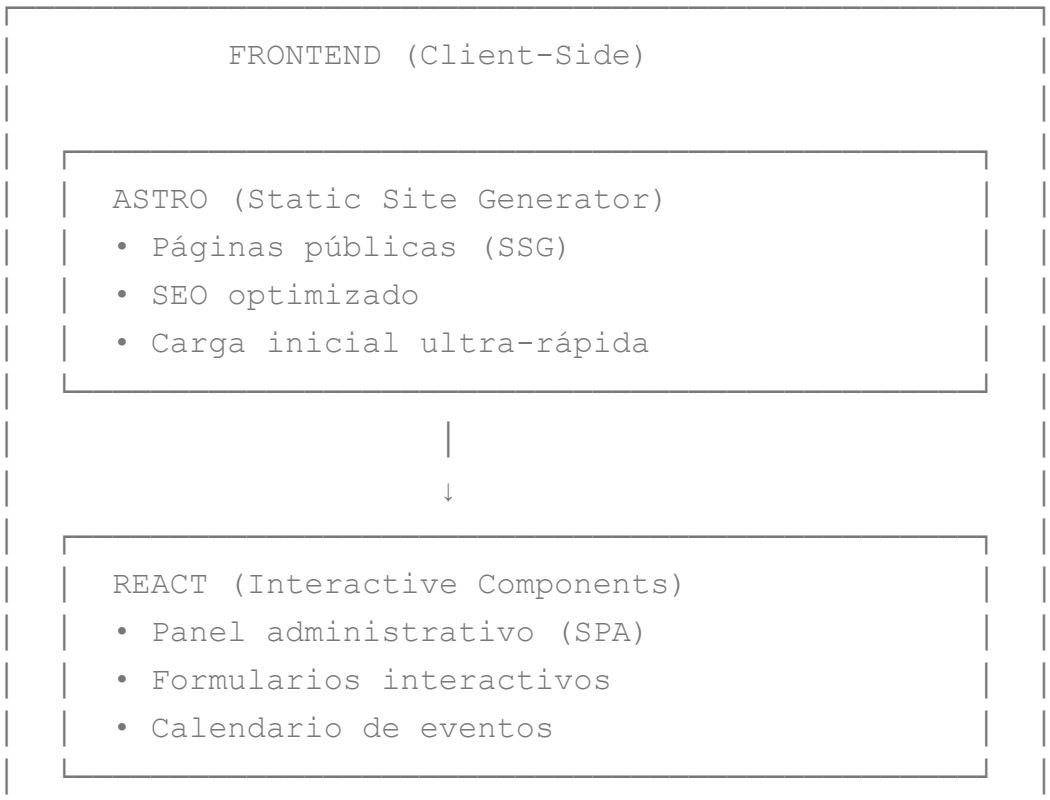
Estilo Visual:

- Elegancia premium sin ser pretencioso
- Fotografías de alta calidad con iluminación dramática
- Animaciones suaves y fluidas
- Espacios generosos (whitespace)
- Contraste alto para legibilidad

---

2. ARQUITECTURA GENERAL

🏗️ Diagrama de Arquitectura




TAILWIND CSS (Styling)
• Sistema de diseño consistente
• Responsive utilities
↓ API Calls
BACKEND (Server-Side)
SUPABASE
PostgreSQL Database
• Eventos, cotizaciones, usuarios
• Contenido del sitio
Auth (Autenticación)
• Login de administradores
• Roles y permisos (RLS)
Storage (Archivos)
• Imágenes de eventos
• Fotos del portafolio
Edge Functions (Serverless)
• Envío de emails
• Validaciones complejas
↓ Integrations

```
┌─────────────────────────────────────────────────────┐
│           SERVICIOS EXTERNOS                        │
│                                                     │
│  • Resend (Emails transaccionales)                  │
│  • WhatsApp Business API (Mensajería)               │
│  • Google Maps (Ubicación en Lima)                  │
│  • Vercel (Hosting y deployment)                    │
└─────────────────────────────────────────────────────┘
```

📊 Flujo de Datos Típico

Ejemplo: Usuario solicita una cotización

1. Usuario llena formulario en /cotizacion
   ↓
2. React Hook Form valida datos (Zod)
   ↓
3. POST a Supabase Database
   • Tabla: quotes
   • Datos: nombre, email, numero_celular, Tipo_de_evento, fecha, invitac
   ↓
4. Supabase Edge Function se dispara
   • Valida datos adicionales
   • Calcula precio estimado
   ↓
5. Resend envía email
   • Al cliente: Confirmación de recepción
   • Al admin: Nueva cotización recibida
   ↓
6. WhatsApp API (opcional)
   • Notificación al admin
   ↓
7. Admin ve en panel /admin/cotizaciones
   • Lista actualizada en tiempo real
   • Puede responder o convertir a evento

---

3. STACK TECNOLÓGICO FRONTEND

⚡ Astro 4.x (Framework Principal)

¿Qué es Astro?

Astro es un framework web moderno diseñado para construir sitios rápidos y centrados en el contenido.

¿Por qué Astro?

1. Performance Excepcional:

- Genera HTML estático (SSG) por defecto
- Cero JavaScript en cliente por defecto
- Carga solo el JS necesario (partial hydration)
- **Resultado:** Sitios 2-3x más rápidos que Next.js/Gatsby

2. SEO Perfecto:

- HTML estático = Google indexa perfecto
- Meta tags fáciles de configurar
- Sitemap automático
- Tiempos de carga ultra-rápidos (Core Web Vitals)

3. Flexibilidad:

- Usa React solo donde necesitas interactividad
- Soporta múltiples frameworks (React, Vue, Svelte)
- Componentes `.astro` para contenido estático

4. Developer Experience:

- Sintaxis simple e intuitiva
- Hot Module Replacement (HMR) rápido
- TypeScript integrado
- File-based routing (como Next.js)

Uso en La Reserva:

```
Sitio público (SSG):
├── / (home) → Astro SSG
├── /servicios → Astro SSG
├── /portafolio → Astro SSG + React (filtros)
```

```
├── /blog → Astro SSG
└── /cotizacion → Astro + React (formulario)

Panel admin (SPA):
└── /admin/* → React puro (client:only)
```

Ejemplo de Componente Astro:

```
---
// src/pages/servicios.astro
import BaseLayout from '@layouts/BaseLayout.astro';
import ServiceCard from '@components/services/ServiceCard.astro';

const services = await getServices(); // Fetch en build time
---

<BaseLayout title="Servicios - La Reserva">
  <section class="section">
    <h1>Nuestros Servicios</h1>
    {services.map(service => (
      <ServiceCard service={service} />
    ))}
  </section>
</BaseLayout>
```

⚛️ React 18

¿Por qué React?

1. Ecosistema Maduro:

- Miles de librerías disponibles
- Comunidad enorme
- Soluciones probadas para problemas comunes

2. Perfecto para Interfaces Complejas:

- Panel admin con muchos componentes interactivos
- Forms complejos con validación
- Estado de aplicación (cotizaciones, eventos)

3. Hooks Modernos:

- `useState`, `useEffect` para estado y efectos
- Custom hooks para lógica reutilizable
- Context API para estado global

Uso en La Reserva:

```
Componentes React:
├── QuoteForm.tsx → Formulario de cotización
├── PortfolioGrid.tsx → Galería con filtros
├── EventCalendar.tsx → Calendario admin
├── Dashboard.tsx → Panel administrativo
└── ContentEditor.tsx → Editor de contenido
```

Ejemplo de Componente React:

```
// src/components/quote/QuoteForm.tsx
import { useState } from 'react';
import { useForm } from 'react-hook-form';

export function QuoteForm() {
  const [isSubmitting, setIsSubmitting] = useState(false);
  const { register, handleSubmit, formState: { errors } } = useForm();

  const onSubmit = async (data) => {
    setIsSubmitting(true);
    // Enviar a Supabase
    await supabase.from('quotes').insert(data);
    setIsSubmitting(false);
  };

  return (
    <form onSubmit={handleSubmit(onSubmit)}>
      <input {...register('name', { required: true })} />
      {errors.name && <span>Este campo es requerido</span>}
      <button disabled={isSubmitting}>Enviar</button>
    </form>
  );
}
```

🎨 Tailwind CSS 3.4

¿Por qué Tailwind?

1. Velocidad de Desarrollo:

- No escribir CSS custom
- Clases utilitarias para todo
- Prototipado rápido

2. Consistencia:

- Sistema de diseño integrado
- Espaciados consistentes
- Paleta de colores predefinida

3. Performance:

- Purge automático (solo CSS usado)
- Tamaño final muy pequeño (~10kb)
- No hay CSS sin usar

4. Responsive Design:

```
<!-- Mobile first, luego breakpoints -->
<div class="text-sm md:text-base lg:text-lg">
  Texto responsive
</div>
```

Configuración Personalizada:

```
// Paleta de La Reserva
colors: {
  primary: {
    500: '#D4AF37', // Dorado
    600: '#B8941F',
  },
  secondary: {
    600: '#1A1A1A', // Negro
  }
}
```

📘 TypeScript

¿Por qué TypeScript?

1. Prevención de Errores:

```
// ❌ JavaScript - Error en runtime
function addEvent(event) {
  return event.date + 1; // Si date es string, error!
}

// ✅ TypeScript - Error en desarrollo
function addEvent(event: Event) {
  return event.date.getTime() + 1; // Type-safe
}
```

2. Autocompletado Inteligente:

- VS Code sugiere propiedades
- Detecta typos antes de ejecutar
- Refactoring seguro

3. Documentación Integrada:

```
interface Quote {
  client_name: string;
  event_date: Date;
  guest_count: number; // 25-500
  cocktails: string[]; // Mínimo 3, máximo 5
}
```

4. Mejor Mantenibilidad:

- Código auto-documentado
- Refactors seguros
- Menos bugs en producción

---

4. STACK TECNOLÓGICO BACKEND

🗄️ Supabase (Backend-as-a-Service)

¿Qué es Supabase?

Supabase es una alternativa open-source a Firebase. Provee backend completo: database, auth, storage, y más.

¿Por qué Supabase?

1. PostgreSQL Real:

- Base de datos relacional robusta
- SQL completo (joins, transactions, etc.)
- Mejor que NoSQL para datos estructurados

2. Auth Integrado:

- Login/registro out-of-the-box
- Roles y permisos (Row Level Security)
- Social logins (Google, Facebook, etc.)

3. Storage para Archivos:

- Subir imágenes de eventos
- Optimización automática
- CDN integrado

4. Real-time Subscriptions:

- Actualizar dashboard en tiempo real
- Notificaciones instantáneas

5. Edge Functions:

- Lógica de servidor serverless
- Envío de emails
- Cálculos complejos

6. Free Tier Generoso:

```
✅ 500MB database
✅ 1GB storage
✅ 50,000 usuarios activos/mes
```

✅ 2GB bandwidth
✅ Perfecto para empezar

Servicios de Supabase Usados:

1. Database (PostgreSQL)

```
-- Ejemplo de tablas
CREATE TABLE events (
  id UUID PRIMARY KEY,
  client_name TEXT,
  event_date DATE,
  guest_count INTEGER,
  status TEXT CHECK (status IN ('pendiente', 'confirmado', 'completado'))
  created_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE quotes (
  id UUID PRIMARY KEY,
  client_email TEXT,
  event_type TEXT,
  status TEXT,
  created_at TIMESTAMP DEFAULT NOW()
);
```

2. Auth (Autenticación)

```
// Login de administrador
const { data, error } = await supabase.auth.signInWithPassword({
  email: 'admin@lareserva.com',
  password: 'password123',
});

// Verificar rol
const { data: profile } = await supabase
  .from('admin_profiles')
  .select('role')
  .eq('user_id', user.id)
  .single();
```

3. Storage (Archivos)

```
// Subir imagen de evento
const { data, error } = await supabase.storage
  .from('events')
  .upload('boda-maria/foto1.jpg', file);

// URL pública
const { data: { publicUrl } } = supabase.storage
  .from('events')
  .getPublicUrl('boda-maria/foto1.jpg');
```

4. Edge Functions

```
// Función para enviar email al recibir cotización
Deno.serve(async (req) => {
  const { quote } = await req.json();

  // Enviar email con Resend
  await fetch('https://api.resend.com/emails', {
    method: 'POST',
    headers: { 'Authorization': `Bearer ${RESEND_API_KEY}` },
    body: JSON.stringify({
      to: quote.client_email,
      subject: 'Cotización recibida - La Reserva',
      html: '<h1>Gracias por tu interés...</h1>',
    }),
  });

  return new Response('OK');
});
```

---

5. LIBRERÍAS PRINCIPALES

📝 React Hook Form + Zod

React Hook Form

Para: Manejo eficiente de formularios

Ventajas:

- ⚡ Performance: Menos re-renders
- 🎯 API simple e intuitiva
- 🔌 Fácil integración con validadores

```
const { register, handleSubmit, formState: { errors } } = useForm();
```

Zod

Para: Validación de datos con TypeScript

Ventajas:

- 🔒 Type-safe: Tipos inferidos automáticamente
- 📝 Declarativo: Schemas claros
- 🎨 Mensajes customizables

```
const quoteSchema = z.object({
  name: z.string().min(2, 'Nombre muy corto'),
  email: z.string().email('Email inválido'),
  guestCount: z.number().min(25).max(500),
});
```

🎬 Framer Motion

Para: Animaciones fluidas en React

Características:

- Animaciones declarativas
- Gestos (drag, hover, tap)
- Layout animations
- Exit animations

```
<motion.div
  initial={{ opacity: 0, y: 20 }}
  animate={{ opacity: 1, y: 0 }}
  exit={{ opacity: 0 }}
```

```
>
  Contenido animado
</motion.div>
```

📅 React Big Calendar + date-fns

React Big Calendar

Para: Calendario de eventos en admin

Características:

- Vistas: mensual, semanal, diaria
- Drag & drop
- Personalizable

date-fns

Para: Manipulación de fechas

Ventajas sobre Moment.js:

- Modular (tree-shakeable)
- Inmutable
- Más pequeño

```
import { format, addDays } from 'date-fns';
import { es } from 'date-fns/locale';

format(new Date(), 'PPP', { locale: es });
// "17 de octubre de 2025"
```

🎨 Radix UI

Para: Componentes UI accesibles y sin estilos

Componentes usados:

- Dialog (modales)
- Dropdown Menu

- Select
- Toast (notificaciones)
- Tabs
- Label

Ventaja: Accesibilidad perfecta out-of-the-box (keyboard navigation, ARIA, etc.)

---

6. HERRAMIENTAS DE DESARROLLO

📦 pnpm (Package Manager)

Por qué pnpm en lugar de npm:

Feature	npm	pnpm
Velocidad	1x	3x ⚡
Espacio en disco	100%	20% 💾
Seguridad	Media	Alta 🔒
Monorepo	Complejo	Nativo 🚀

Comando de instalación:

```
npm install -g pnpm
```

🎨 Prettier

Para: Formateo automático de código

Beneficios:

- Código consistente en todo el equipo
- Ahorra tiempo en code reviews
- Soporte para Astro, TypeScript, React

Configuración:

```
{
  "semi": true,
  "singleQuote": true,
  "tabWidth": 2,
  "plugins": ["prettier-plugin-astro"]
}
```

🚀 Vercel (Deployment Platform)

¿Por qué Vercel?

1. Optimizado para Astro:

- Integración nativa con Astro
- Zero-config deployment
- Edge Functions soporte

2. Performance Excepcional:

- CDN Global (Edge Network)
- Automatic HTTPS
- Image Optimization
- Caching inteligente

3. Developer Experience:

- Deploy con `git push`
- Preview URLs para cada PR
- Rollback instantáneo
- Logs en tiempo real

4. Free Tier Generoso:

```
✅ 100GB bandwidth/mes
✅ Deployments ilimitados
✅ SSL automático
✅ Preview deployments
✅ Perfecto para La Reserva
```

5. Integración con Supabase:

- Variables de entorno fáciles
- Edge Functions compatibles
- Sin configuración adicional

Flujo de Deployment:

```
1. Push a GitHub
   ↓
2. Vercel detecta cambios
   ↓
3. Build automático (pnpm build)
   ↓
4. Deploy a edge network
   ↓
5. Sitio live en lareserva.pe
```

Comandos de Vercel:

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

# Login
vercel login

# Deploy a preview
vercel

# Deploy a producción
vercel --prod
```

---

7. REQUISITOS PREVIOS

💻 Software Necesario

1. Node.js

```
Versión mínima: 18.14.1
Versión recomendada: 20.x LTS (Long Term Support)

Verificar instalación:
node --version

Descargar:
https://nodejs.org/
```

¿Por qué Node.js 18+?

- Astro requiere Node 18 mínimo
- Mejor performance
- Soporte de ES modules moderno

---

2. pnpm

```
# Instalar globalmente
npm install -g pnpm

# Verificar
pnpm --version
# Debería mostrar: 8.x.x o superior
```

---

3. Git

```
# Verificar instalación
git --version

# Configurar (si es primera vez)
git config --global user.name "Tu Nombre"
git config --global user.email "tu@email.com"

# Descargar:
https://git-scm.com/
```

---

4. Editor de Código

Recomendado: Visual Studio Code

Extensiones NECESARIAS:

```
1. Astro (astro-build.astro-vscode)
   → Syntax highlighting para .astro
   → Autocompletado
   → Formateo

2. Tailwind CSS IntelliSense (bradlc.vscode-tailwindcss)
   → Autocompletado de clases
   → Preview de colores
   → Documentación inline

3. Prettier (esbenp.prettier-vscode)
   → Formateo automático
   → Consistencia de código

4. ESLint (dbaeumer.vscode-eslint)
   → Detectar errores
   → Mejores prácticas
```

Configuración recomendada de VS Code:

```
{
  "editor.defaultFormatter": "esbenp.prettier-vscode",
  "editor.formatOnSave": true,
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": true
  },
  "[astro]": {
    "editor.defaultFormatter": "astro-build.astro-vscode"
  }
}
```

🌐 Cuentas y Servicios

1. GitHub (Control de versiones)

```
URL: https://github.com/
Plan: Free (suficiente)

Necesario para:
- Repositorio del código
- Colaboración en equipo
- Integration con Vercel
```

---

2. Vercel ⭐ (Hosting y Deployment)

```
URL: https://vercel.com/
Plan: Free Hobby (100GB bandwidth/mes)

Conectar con:
- GitHub account (para auto-deploy)

Configuración:
1. Crear cuenta en Vercel
2. Conectar repositorio GitHub
3. Seleccionar "Astro" como framework
4. Deploy automático
```

Dominios en Vercel:

```
Incluido gratis:
- la-reserva.vercel.app

Custom domain (lareserva.pe):
- Agregar en Vercel dashboard
- Configurar DNS en registrador
- SSL automático
```

---

3. Supabase (Backend as a Service)

```
URL: https://supabase.com/
Plan: Free tier

Free tier incluye:
✅ 500MB database
```

✅ 1GB file storage
✅ 50,000 usuarios activos/mes
✅ 2GB bandwidth/mes
✅ Edge Functions

Obtener credenciales:
1. Crear proyecto en Supabase
2. Dashboard > Settings > API
3. Copiar:
   - Project URL
   - anon/public key
   - service_role key (¡secreto!)

---

4. Resend (Email Service)

URL: https://resend.com/
Plan: Free tier (3,000 emails/mes)

Necesario para:
- Confirmación de cotizaciones
- Notificaciones a admin
- Recordatorios de eventos

Setup:
1. Crear cuenta en Resend
2. Verificar dominio (lareserva.pe)
3. Obtener API Key
4. Configurar en variables de entorno

---

5. Google Maps Platform (Opcional)

URL: https://console.cloud.google.com/
Plan: Pay as you go (pero generoso free tier)

Para mostrar:
- Ubicación de eventos en Lima
- Mapa de contacto

APIs necesarias:

```
- Maps JavaScript API
- Places API
```

📝 Conocimientos Recomendados

Nivel Mínimo (Para empezar):

- ✅ HTML básico (estructura, elementos)
- ✅ CSS básico (selectores, propiedades)
- ✅ JavaScript básico (variables, funciones, arrays)
- ✅ Terminal/Línea de comandos (cd, ls, mkdir)
- ✅ Git básico (clone, commit, push)

Nivel Intermedio (Para desarrollo completo):

- ✅ JavaScript moderno (ES6+: arrow functions, destructuring, async/await)
- ✅ React básico (componentes, props, state, hooks)
- ✅ TypeScript básico (tipos, interfaces)
- ✅ Tailwind CSS (clases utilitarias)
- ✅ Git workflows (branches, pull requests)

Nivel Avanzado (Para arquitectura):

- ✅ Astro (islands architecture, SSG vs SSR)
- ✅ Supabase (PostgreSQL, RLS, Edge Functions)
- ✅ Optimización de performance
- ✅ SEO técnico
- ✅ Accesibilidad web

📚 Recursos de Aprendizaje

Documentación Oficial:

```
Astro:     https://docs.astro.build/
React:     https://react.dev/
Tailwind:  https://tailwindcss.com/docs
Supabase:  https://supabase.com/docs
```

```
Vercel:     https://vercel.com/docs
TypeScript: https://www.typescriptlang.org/docs/
```

Tutoriales Recomendados:

```
YouTube:
- "Astro Crash Course" - Traversy Media
- "Tailwind CSS Full Course" - Dave Gray
- "Supabase Tutorial" - Fireship
- "React Hooks" - Web Dev Simplified

Cursos (Gratuitos):
- FreeCodeCamp.org (HTML, CSS, JavaScript, React)
- Scrimba (React, TypeScript)
- Egghead.io (Astro, React)
```

Comunidades:

```
Discord:
- Astro: https://astro.build/chat
- Tailwind CSS: https://discord.gg/tailwindcss
- Supabase: https://discord.supabase.com

Reddit:
- r/astro
- r/reactjs
- r/webdev
```

✅ Checklist de Preparación

Antes de continuar con el siguiente archivo (02B - Instalación), verifica:

Software:

- [ ] Node.js 18+ instalado
- [ ] pnpm instalado y funcionando
- [ ] Git instalado y configurado
- [ ] VS Code instalado

- [ ] Extensiones de VS Code instaladas

Cuentas:

- [ ] Cuenta de GitHub creada
- [ ] Cuenta de Vercel creada (conectada con GitHub)
- [ ] Cuenta de Supabase creada
- [ ] Cuenta de Resend creada
- [ ] Credenciales guardadas de forma segura

Conocimientos:

- [ ] Familiarizado con terminal básico
- [ ] Conocimiento básico de Git
- [ ] HTML/CSS/JavaScript fundamentals
- [ ] (Opcional) React básico

Preparación:

- [ ] Espacio en disco: ~2GB libre
- [ ] Conexión a internet estable
- [ ] Tiempo disponible: 2-3 horas para setup inicial

🎯 Resumen del Stack

```
┌─────────────────────────────────────────┐
│ FRONTEND                                │
├─────────────────────────────────────────┤
│ • Astro 4.x (Framework)                 │
│ • React 18 (Interactividad)             │
│ • TypeScript (Type Safety)              │
│ • Tailwind CSS (Estilos)                │
└─────────────────────────────────────────┘

┌─────────────────────────────────────────┐
│ BACKEND                                 │
├─────────────────────────────────────────┤
│ • Supabase (BaaS)                       │
│   - PostgreSQL (Database)               │
│   - Auth (Autenticación)                │
```

```
|   - Storage (Archivos)                    |
|   - Edge Functions (Serverless)           |
└──────────────────────────────────────────┘

┌──────────────────────────────────────────┐
| LIBRERÍAS                                |
├──────────────────────────────────────────┤
| • React Hook Form (Formularios)          |
| • Zod (Validación)                       |
| • Framer Motion (Animaciones)            |
| • React Big Calendar (Calendario)        |
| • date-fns (Fechas)                      |
| • Lucide React (Iconos)                  |
| • Radix UI (Componentes accesibles)      |
└──────────────────────────────────────────┘

┌──────────────────────────────────────────┐
| HERRAMIENTAS                             |
├──────────────────────────────────────────┤
| • pnpm (Package Manager)                 |
| • Prettier (Code Formatter)              |
| • Git (Version Control)                  |
| • VS Code (Editor)                       |
└──────────────────────────────────────────┘

┌──────────────────────────────────────────┐
| SERVICIOS                                |
├──────────────────────────────────────────┤
| • Vercel (Hosting & Deployment) ⭐       |
| • Resend (Emails)                        |
| • WhatsApp Business API                  |
| • Google Maps (Opcional)                 |
└──────────────────────────────────────────┘
```

🚀 Ventajas de Este Stack

Performance:

- ⚡ Astro SSG = Sitio ultra-rápido
- 📦 Tailwind Purge = CSS minimalista
- 🖼️ Vercel CDN = Carga global rápida

- 🎯 Lazy loading automático

SEO:

- 🔍 HTML estático = Google indexa perfecto
- 📊 Core Web Vitals excelentes
- 🗺️ Sitemap automático
- 📱 Mobile-first por defecto

Developer Experience:

- 🛠️ TypeScript = Menos bugs
- 💅 Tailwind = Desarrollo rápido
- 🔄 pnpm = Instalaciones rápidas
- 🚀 Vercel = Deploy con git push

Escalabilidad:

- 📈 Supabase = Crece con tu negocio
- 🔐 RLS = Seguridad por defecto
- ⚙️ Edge Functions = Lógica serverless
- 💾 PostgreSQL = Base de datos robusta

Costo:

- 💰 Todo con free tier generoso
- 📊 Vercel: 100GB/mes gratis
- 🗄️ Supabase: 500MB DB gratis
- 📧 Resend: 3,000 emails/mes gratis

💡 Decisiones Arquitectónicas Clave

1. ¿Por qué Astro sobre Next.js?

```
Next.js:
✅ React everywhere
✅ Routing potente
❌ Más JavaScript en cliente
❌ Más complejo para sitios de contenido
```

```
Astro:
✅ Zero JS por defecto
✅ Mejor para sitios de contenido
✅ Mix de frameworks (React donde necesario)
✅ SSG perfecto para SEO
❌ Menos features avanzados de routing
```

Decisión: Astro es mejor para La Reserva porque:

- Mayoría del sitio es contenido estático
- SEO es crítico
- Performance es prioridad
- React solo en formularios y admin

---

2. ¿Por qué Supabase sobre Firebase?

```
Firebase:
✅ Muy popular
✅ Real-time excelente
❌ NoSQL (menos flexible para queries)
❌ Vendor lock-in

Supabase:
✅ PostgreSQL (SQL completo)
✅ Open source
✅ RLS (seguridad granular)
✅ Edge Functions más flexibles
❌ Menos maduro que Firebase
```

Decisión: Supabase porque:

- Datos relacionales (eventos → clientes)
- SQL queries complejas
- Open source (no vendor lock-in)
- RLS para seguridad perfecta

---

3. ¿Por qué Vercel sobre Netlify?

```
Netlify:
✅ Excelente para sitios estáticos
✅ Drag & drop deploy
✅ Split testing
❌ Edge Functions más limitadas
❌ Menos optimizado para frameworks modernos

Vercel:
✅ Creadores de Next.js (experiencia en frameworks)
✅ Edge Network superior
✅ Integración perfecta con Astro
✅ Analytics incluido
✅ Image Optimization
```

Decisión: Vercel porque:

- Mejor integración con Astro
- CDN global más rápido
- Edge Functions más potentes
- Analytics integrado
- Deploy experience superior

📊 Comparación con Alternativas

Stack Alternativo 1: WordPress

```
Ventajas:
✅ No-code/Low-code
✅ Muchos plugins
✅ Fácil para no-developers

Desventajas:
❌ Lento (PHP + MySQL)
❌ Vulnerabilidades de seguridad
❌ Plugins de calidad variable
❌ Hosting más caro
❌ Difícil de personalizar profundamente

Resultado: No elegido - Necesitamos performance y customización
```

Stack Alternativo 2: Next.js + MongoDB

Ventajas:
✅ React everywhere
✅ API routes integradas
✅ MongoDB flexible

Desventajas:
❌ Más JavaScript en cliente (peor performance)
❌ MongoDB menos ideal para datos relacionales
❌ Más complejo de mantener
❌ Costo de hosting más alto

Resultado: No elegido - Astro + Supabase son mejor match

Stack Alternativo 3: Gatsby + Contentful

Ventajas:
✅ Gatsby rápido
✅ Contentful es buen CMS

Desventajas:
❌ Gatsby tiene problemas de build time
❌ Contentful es caro ($300+/mes para features completas)
❌ GraphQL overhead
❌ Menos flexible que Supabase

Resultado: No elegido - Astro + Supabase más económico y flexible

---

🎓 Glosario de Términos

SSG: Static Site Generation - Generar HTML en build time
SSR: Server-Side Rendering - Generar HTML en request time
SPA: Single Page Application - App de una sola página (React puro)
CDN: Content Delivery Network - Red de servidores globales
Edge: Computación en servidores cerca del usuario
RLS: Row Level Security - Seguridad a nivel de fila en BD
BaaS: Backend as a Service - Backend listo para usar

```
API: Application Programming Interface - Interfaz de comunicación
CRUD: Create, Read, Update, Delete - Operaciones básicas de BD
```

Siguiente archivo: 02B - Instalación y Configuración
