

Guía de Implementación Paso a Paso - DHérmica Estética

Roadmap Completo de Implementación

Esta guía detalla cada paso necesario para implementar la aplicación DHérmica Estética desde cero hasta el deploy en producción.

FASE 1: SETUP Y CONFIGURACIÓN INICIAL

Paso 1: Crear el Proyecto Next.js (SIN TypeScript)

- Crear proyecto con `create-next-app@latest` SIN `--typescript` flag
- Usar JavaScript puro con validaciones Zod en runtime
- Verificar que el proyecto funciona con `npm run dev`

IMPORTANTE: Este proyecto usa **JavaScript** en lugar de TypeScript para simplicidad

Paso 2: Instalar Dependencias Principales

- **Firebase:** `firebase`
- **Estado y formularios:** `zustand react-hook-form @hookform/resolvers zod`
- **UI y utilidades:** `lucide-react date-fns clsx tailwind-merge class-variance-authority`
- **PWA:** `next-pwa`
- **Animaciones:** `framer-motion` (opcional)

Paso 3: Configurar shadcn/ui

- Inicializar con `npx shadcn-ui@latest init`
- Instalar componentes necesarios: button, input, form, card, dialog, select, calendar, badge, avatar, separator, scroll-area, skeleton, toast, textarea, label

Paso 4: Crear Estructura de Carpetas

- Organizar carpetas según la documentación técnica
- Crear rutas agrupadas: `(auth)`, `(client)`, `(admin)`
- Estructurar `components/`, `lib/`, `hooks/`, `store/`, `types/`

Paso 5: Configurar Variables de Entorno

- Crear `.env.local` con variables de Firebase y Cloudinary
- Configurar variables públicas y privadas

Paso 6: Configurar Firebase

- Crear archivo `lib/firebase.js` con configuración
- Inicializar Auth, Firestore y Storage

Paso 7: Definir Estructuras de Datos (JavaScript)

- Crear `lib/data-structures.js` con ejemplos de objetos
- Documentar estructura de cada entidad sin interfaces TypeScript
- Usar comentarios JSDoc para documentación

Paso 8: Configurar Tailwind CSS

- Actualizar `globals.css` con variables CSS personalizadas
 - Agregar animaciones custom y tema de colores
-



FASE 2: SISTEMA DE AUTENTICACIÓN (JavaScript)

Paso 9: Crear Store de Autenticación

- Configurar Zustand store para manejo de estado de usuario
- Incluir funciones de login, logout y manejo de perfil

Paso 10: Crear AuthProvider

- Componente provider para manejar estado de autenticación
- Listener para cambios en Firebase Auth
- Cargar perfil de usuario desde Firestore

Paso 11: Crear Formularios de Autenticación

- Formulario de login con validación Zod
- Formulario de registro para clientes
- Manejo de errores y feedback visual

Paso 12: Crear Middleware de Protección

- Configurar `middleware.js` para proteger rutas
- Verificar roles y redireccionar según corresponda
- Proteger rutas admin, client y públicas

Paso 13: Crear Layout Principal

- Layout base con providers necesarios
- Configurar Toaster para notificaciones
- Metadata y configuración SEO básica

- Archivo: `app/layout.js`
-

FASE 3: SERVICIOS Y VALIDACIONES

Paso 14: Crear Servicios de Firebase

- Crear `lib/firebase-services.js` con CRUD para todas las entidades
- Servicios para: treatments, professionals, clients, appointments
- Incluir consultas optimizadas con índices
- **Nota:** Sin TypeScript, usar JSDoc para documentar parámetros y retornos

Paso 15: Crear Utilidades de Validación

- Configurar schemas Zod para cada entidad
- Validaciones de formularios consistentes
- Reglas de negocio validadas

Paso 16: Crear Utilidades de Tiempo

- Funciones para manejo de horarios y fechas
 - Validación de conflictos de tiempo
 - Generación de slots disponibles
-

FASE 4: COMPONENTES PRINCIPALES

Paso 17: Componente de Búsqueda de Clientes

- Búsqueda con debounce y autocompletado
- Visualización de información médica
- Opción de crear nuevo cliente

Paso 18: Selector de Horarios

- Mostrar horarios disponibles/ocupados
- Validar horarios laborales del profesional
- Calcular slots según duración del tratamiento

Paso 19: Formulario Principal de Citas

- Wizard multi-paso para crear citas
- Validación en tiempo real de conflictos
- Integración con todos los componentes

Paso 20: Componente de Calendario

- Vista mensual y diaria de citas
- Filtros por profesional
- Navegación intuitiva

Paso 21: Alertas Médicas

- Componente para mostrar restricciones médicas
 - Validación automática cliente-tratamiento
 - Alertas visuales claras
-

FASE 5: INTERFAZ DE USUARIO

Paso 22: Dashboard Admin

- Vista principal con estadísticas básicas
- Accesos rápidos a funciones principales
- Resumen de citas del día

Paso 23: Gestión de Tratamientos

- CRUD completo de tratamientos
- Subida de imágenes a Cloudinary
- Configuración de restricciones médicas

Paso 24: Gestión de Profesionales

- CRUD de profesionales
- Configuración de horarios laborales
- Especialidades y disponibilidad

Paso 25: Gestión de Clientes

- Lista de clientes con búsqueda
- Formulario de datos médicos
- Historial de citas por cliente

Paso 26: Lista y Calendario de Citas

- Vista de lista con filtros
- Calendario visual interactivo
- Acciones rápidas (editar, eliminar)

FASE 6: PANEL DEL CLIENTE

Paso 27: Registro de Clientes

- Formulario de registro con datos personales
- Formulario de información médica
- Validación y creación de cuenta

Paso 28: Dashboard del Cliente

- Vista de próximas citas
- Acceso rápido a funciones
- Información personal

Paso 29: Mis Citas

- Lista de citas futuras y pasadas
- Detalles de cada cita
- Estado de las citas

Paso 30: Mi Perfil

- Edición de datos personales
- Actualización de información médica
- Configuración de notificaciones

Paso 31: Historial de Tratamientos

- Lista completa de tratamientos realizados
- Precios pagados
- Estadísticas personales

FASE 7: PÁGINAS PÚBLICAS

Paso 32: Landing Page

- Página de inicio atractiva
- Información sobre servicios
- Call-to-action para registro

Paso 33: Catálogo de Tratamientos

- Lista pública de tratamientos

- Páginas de detalle por tratamiento
- SEO optimizado

Paso 34: Páginas de Información

- Página "Acerca de"
 - Información de contacto
 - Políticas y términos
-

FASE 8: OPTIMIZACIÓN Y PWA

Paso 35: Configurar PWA

- Configurar `next-pwa`
- Crear manifest.json
- Service worker para offline

Paso 36: Optimización de Performance

- Lazy loading de componentes
- Optimización de imágenes
- Code splitting

Paso 37: SEO y Meta Tags

- Meta tags dinámicos
- Sitemap automático
- Open Graph tags

Paso 38: Analytics y Monitoreo

- Configurar Firebase Analytics
 - Tracking de eventos importantes
 - Monitoreo de errores
-

FASE 9: SEGURIDAD Y VALIDACIONES

Paso 39: Reglas de Firestore Security

- Configurar rules para cada colección
- Validación de permisos por rol
- Protección de datos sensibles

Paso 40: Validaciones del Lado del Servidor

- Firebase Functions para validaciones críticas
- Sanitización de datos
- Rate limiting

Paso 41: Manejo de Errores

- Error boundaries
 - Logging de errores
 - Fallbacks para componentes
-

FASE 10: TESTING

Paso 42: Setup de Testing

- Configurar Jest y React Testing Library
- Testing con Firebase Emulator
- Configurar Cypress para E2E

Paso 43: Unit Tests

- Tests para utilidades y hooks
- Tests para servicios de Firebase
- Validaciones y funciones críticas

Paso 44: Component Tests

- Tests para componentes principales
- Mocking de servicios
- Testing de formularios

Paso 45: E2E Tests

- Flujos críticos de usuario
 - Test de creación de citas
 - Test de registro y login
-

FASE 11: DEPLOYMENT Y PRODUCCIÓN

Paso 46: Configurar Vercel

- Conectar repositorio

- Configurar variables de entorno
- Setup de dominio personalizado

Paso 47: Configurar Firebase para Producción

- Proyecto separado para producción
- Índices de Firestore
- Reglas de seguridad finales

Paso 48: Configurar Cloudinary

- Setup de transformaciones automáticas
- Configurar folders por entorno
- Optimización de imágenes

Paso 49: Monitoreo Post-Deploy

- Configurar alertas
- Monitoreo de performance
- Analytics de usuarios



FASE 12: DOCUMENTACIÓN Y TRAINING

Paso 50: Documentación de Usuario

- Manual para administradores
- Guía para clientes
- Troubleshooting común

Paso 51: Training del Personal

- Capacitación para profesionales
- Casos de uso comunes
- Soporte técnico básico

Paso 52: Documentación Técnica

- README del proyecto
- Deployment guide
- Maintenance procedures



CHECKLIST DE FUNCIONALIDADES CORE

Autenticación y Usuarios

- ☐ Login/Logout para admin y clientes
- ☐ Registro de clientes con datos médicos
- ☐ Protección de rutas por rol
- ☐ Recuperación de contraseña

Gestión de Citas

- ☐ Crear cita con validación de horarios
- ☐ Editar citas existentes
- ☐ Ver calendario de citas
- ☐ Búsqueda inteligente de clientes
- ☐ Validación de restricciones médicas

Gestión de Datos

- ☐ CRUD de tratamientos
- ☐ CRUD de profesionales
- ☐ CRUD de clientes
- ☐ Subida de imágenes

Panel del Cliente

- ☐ Ver mis citas
- ☐ Actualizar perfil
- ☐ Historial de tratamientos
- ☐ Datos médicos

Optimizaciones

- ☐ PWA funcional
- ☐ Performance optimizado
- ☐ SEO implementado
- ☐ Analytics configurado

Seguridad

- ☐ Reglas de Firestore
- ☐ Validación de entrada
- ☐ Manejo de errores
- ☐ Rate limiting

Hoy (2-3 horas)

1. Crear proyecto Next.js
2. Instalar dependencias
3. Configurar estructura de carpetas
4. Setup de Firebase básico

Esta Semana

1. Sistema de autenticación completo
2. Servicios de Firebase
3. Formularios básicos
4. Componentes UI principales

Próximas 2 Semanas





1. Funcionalidad completa de citas
2. Panel de administración
3. Panel del cliente
4. Validaciones médicas

Primer Mes



1. PWA funcional
2. Deploy en producción
3. Testing básico
4. Documentación de usuario

CRITERIOS DE ÉXITO

Funcionalidad

-  Admin puede crear citas en menos de 2 minutos
-  No hay conflictos de horarios
-  Clientes pueden ver su historial
-  Validaciones médicas funcionan

Performance

-  Carga inicial < 3 segundos
-  PWA instalable

- ☒ Funciona offline básico
- ☒ Responsive en todos los dispositivos

UX

- ☒ Interfaz intuitiva para no técnicos
 - ☒ Feedback visual en todas las acciones
 - ☒ Errores claros y útiles
 - ☒ Flujo de trabajo optimizado
-

Esta guía debe seguirse secuencialmente para asegurar una implementación exitosa. Cada paso está diseñado para construir sobre el anterior y minimizar problemas de integración.