Roadmap de Implementación - DentiCloud

Sistema SaaS para Gestión Dental Multi-Tenant
Stack: Next.js + NestJS + PostgreSQL + AWS
Fecha: 30 de Diciembre, 2025

Tabla de Contenidos

Visión General

Modelo de Negocio Correcto ✅

CLIENTES: TODOS los dentistas (cada uno paga suscripción)

Modelo Multi-Tenant:

- **Tenants = Dentistas:** Cada dentista es un tenant lógico
- **Single Database:** Row-level security con `tenant_id`
- **Pacientes N:M con Dentistas:** Un paciente puede tener múltiples dentistas
- **Staff Multi-Dentista:** Staff trabaja para múltiples dentistas
- **Clínicas:** Creadas por super admin, contienen consultorios compartidos

PROF

Características Diferenciadoras:

- 🌟 **WhatsApp Chatbot (Baileys + GPT-4)** - Feature crítico en MVP/Fase 2
- 📅 **Calendar Integrations** (Google, Outlook, Apple)
- 🦷 **Odontograma Digital** interactivo
- 🤖 **AI-powered** scheduling y recordatorios
- 💳 **Stripe Payments** integrado

Timeline General

Fase	Duración	Descripción
Discovery & Planning	2-4 semanas	Investigación, diseño, validación modelo
MVP (Fase 1)	12-16 semanas	Core features + WhatsApp básico

Fase	Duración	Descripción
Fase 2	8-12 semanas	WhatsApp Bot IA + Calendar integrations
Fase 3	12-16 semanas	Billing + Analytics + Odontograma
Fase 4	8-12 semanas	Mobile apps + Scale features
TOTAL	~12-14 meses	De inicio a producto completo

Equipo Recomendado

MVP (Fase 1):

- 1 Tech Lead / Arquitecto
- 2 Full-stack Developers (Next.js + NestJS)
- 1 UI/UX Designer
- 1 QA Engineer
- 1 DevOps Engineer (part-time)

Fases 2-4:

- Agregar 1-2 developers adicionales
- 1 Product Manager
- 1 Customer Success (para beta)

Stack Tecnológico

Frontend

```
Framework: Next.js 14+ (App Router)
Language: TypeScript 5+
UI Library:
  - Tailwind CSS 3+
  - shadcn/ui (componentes)
  - Radix UI (primitives)
State Management: Zustand
Forms: React Hook Form + Zod
Calendar UI: FullCalendar
Charts: Recharts
HTTP Client: TanStack Query (React Query)
Testing:
  - Vitest (unit tests)
  - Playwright (e2e)
```

PROF

Backend

```
Runtime: Node.js 20 LTS
Framework: NestJS 10+
Language: TypeScript 5+
API Style: REST + GraphQL (opcional Fase 2+)
ORM: Prisma (con row-level security)
Authentication: Passport.js + JWT
Authorization: CASL (attribute-based access control)
Queue: BullMQ (Redis-based)
WebSockets: Socket.io (WhatsApp, real-time)
Testing:
  - Jest (unit/integration)
  - Supertest (API testing)
Validation: class-validator + class-transformer
Documentation: Swagger/OpenAPI
```

Database

```
Primary DB: PostgreSQL 15+
  - ☆ SINGLE DATABASE para todos los tenants
  - Row-level security con tenant_id
  - Índices en tenant_id para performance
Cache: Redis 7+
  - Session storage
  - Rate limiting
  - Queue backend
  - Application cache
Search: PostgreSQL Full-Text Search (MVP)
File Storage: AWS S3
  - Encryption at rest (KMS)
  - Lifecycle policies
  - CloudFront CDN
```

PROF

Infrastructure

```
Cloud: AWS
  - Compute: ECS Fargate (MVP) → EKS (Scale)
  - Database: RDS PostgreSQL Multi-AZ
  - Cache: ElastiCache Redis
  - Storage: S3 + CloudFront
  - Load Balancer: Application Load Balancer
  - DNS: Route 53
  - Email: SES
  - Monitoring: CloudWatch
Container: Docker
Orchestration: ECS (MVP), Kubernetes/EKS (Fase 4)
IaC: Terraform
CI/CD: GitHub Actions
```

```
Monitoring:
  - CloudWatch (AWS metrics)
  - Sentry (error tracking)
  - DataDog (APM - producción)
```

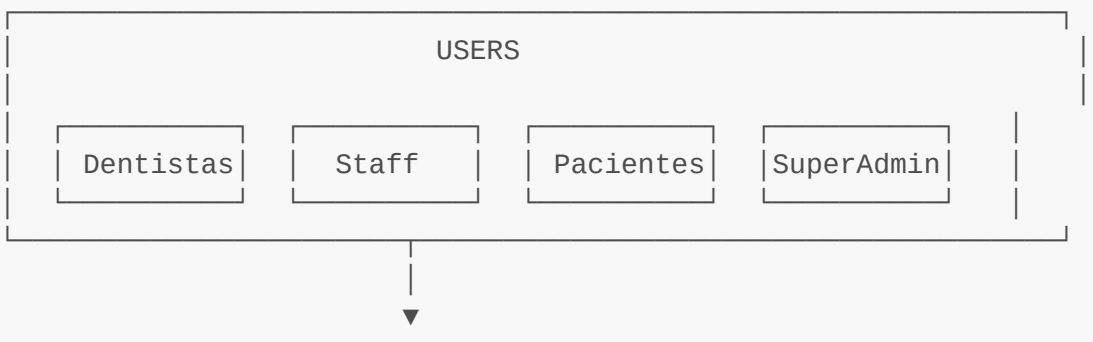
Integraciones

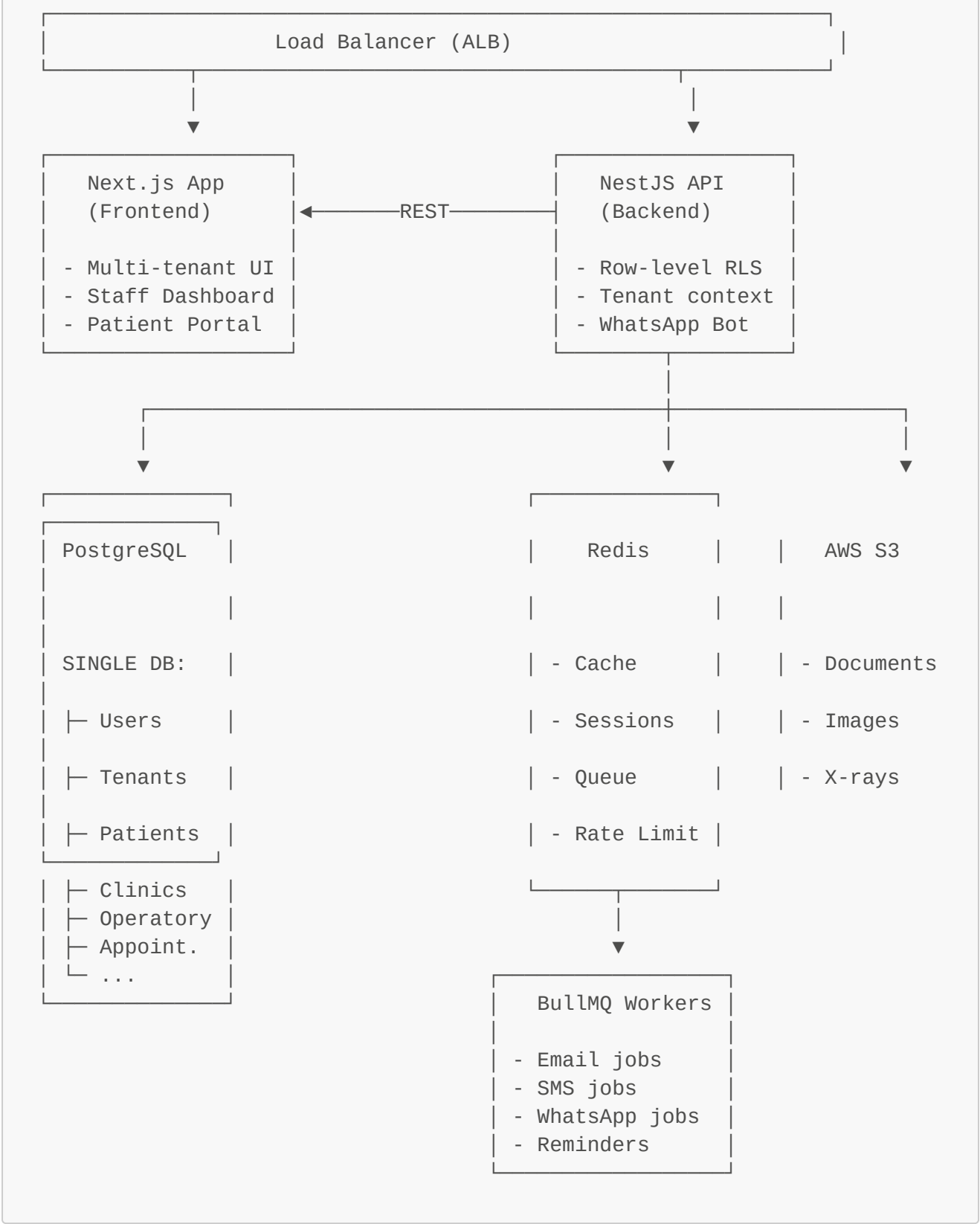
```
Authentication:
  - Google OAuth 2.0
  - Apple Sign In
  - Microsoft OAuth (Office 365)
  - NextAuth.js

Calendarios:
  - Google Calendar API
  - Microsoft Graph API (Outlook)
  - CalDAV (Apple Calendar)

Comunicación:
  - ☆ WhatsApp (Baileys) - CRÍTICO MVP/Fase 2
  - Twilio (SMS)
  - SendGrid (Email)

Pagos:
  - Stripe (payments + subscriptions)

AI/NLP:
  - OpenAI GPT-4 (chatbot, NLP)
  - Embeddings para RAG

Analytics:
  - Google Analytics
  - Mixpanel
```

PROF

Arquitectura del Sistema

Arquitectura de Alto Nivel





PROF

Decisión Arquitectural: Single Database

✅ **SELECCIONADO:** Single Database con Row-Level Security

Razones:

- Más simple de implementar y mantener
- Escalable hasta 10,000+ tenants

- Queries eficientes con índices en `tenant_id`
- Facilita features cross-tenant (si se necesitan)
- Backups y migrations más simples

× **RECHAZADO:** Database-per-Tenant

- Demasiado complejo para este caso de uso
- Difícil gestionar 1000+ databases
- Connection pooling problemático
- Dificulta reportes agregados

Row-Level Security (RLS)

```
// Ejemplo: Filtrado automático por tenant_id
const patients = await prisma.patient.findMany({
  where: {
    patientDentistRelations: {
      some: {
        dentistId: currentDentistId, // tenant_id del contexto
        isActive: true
      }
    }
  }
});
```

Modelo de Datos Core

Entidades Principales

```
// User - TODOS los usuarios (dentistas, staff, pacientes)
User {
  id: uuid
  email: string (unique)
  name: string
  password_hash: string (nullable si OAuth)
  phone: string
  role: enum (super_admin, dentist, staff_receptionist, staff_billing,
patient)
  avatar_url: string

  // Si es dentista
  licenseNumber: string
  npiNumber: string
  specialization: string

  created_at: timestamp
}
```

PROF

```
// Tenant - CADA DENTISTA es un tenant
Tenant {
  id: uuid
  ownerId: uuid // User (dentista que paga)
  name: string
  subdomain: string (unique)

  // Subscription
  subscriptionTier: enum (starter, professional, enterprise)
  subscriptionStatus: enum (trial, active, past_due, cancelled)
  stripeCustomerId: string
  trialEndsAt: timestamp

  // Limits
  maxPatients: int
  storageGB: int

  // WhatsApp
  whatsappConnected: boolean

  created_at: timestamp
}

// TenantMembership - Staff trabaja para múltiples dentistas
TenantMembership {
  id: uuid
  userId: uuid // Staff member
  tenantId: uuid // Dentista
  role: enum (staff_receptionist, staff_billing, staff_assistant)

  permissions: jsonb {
    canViewPatients: boolean
    canManageAppointments: boolean
    canProcessPayments: boolean
  }

  status: enum (active, inactive, pending_invitation)
  isActive: boolean
  created_at: timestamp
}

// ☆ PatientDentistRelation - N:M Pacientes ↔ Dentistas
PatientDentistRelation {
  id: uuid
  patientId: uuid
  dentistId: uuid // tenant_id

  isActive: boolean // Relación actual
  startedAt: timestamp
  endedAt: timestamp
  notes: string

  tenant_id: uuid // = dentistId (para RLS)
```

```
}

// Patient
Patient {
  id: uuid
  firstName: string
  lastName: string
  email: string
  phone: string
  dateOfBirth: date
  gender: enum

  // Medical info
  medicalHistory: jsonb
  allergies: text[]
  medications: text[]

  // Portal access
  portalEnabled: boolean
  portalPassword: string

  created_at: timestamp
}

// ☆ Clinic - Creada por SUPER ADMIN
Clinic {
  id: uuid
  name: string
  address: jsonb
  phone: string
  email: string

  createdBy: uuid // super admin
  isActive: boolean

  created_at: timestamp
}

// Operatory - Consultorio (pertenece a clínica)
Operatory {
  id: uuid
  clinicId: uuid
  name: string
  description: text
  isActive: boolean
  equipment: jsonb
}

// ☆ OperatoryAssignment - N:M Dentistas ↔ Consultorios
OperatoryAssignment {
  id: uuid
  operatoryId: uuid
  dentistId: uuid // tenant_id
```

```
  schedule: jsonb { // Horarios asignados
    monday: { start: "09:00", end: "17:00" }
    // ...
  }

  startDate: date
  endDate: date (nullable)
  isActive: boolean

  tenant_id: uuid // = dentistId (para RLS)
}

// Appointment
Appointment {
  id: uuid
  patientId: uuid
  dentistId: uuid // provider (tenant_id)
  operatoryId: uuid

  appointmentDate: timestamp
  duration: int
  status: enum (scheduled, completed, cancelled, no_show)
  procedureType: string
  notes: text

  // Reminders
  reminderSent: boolean
  confirmedVia: enum (email, sms, whatsapp)

  tenant_id: uuid // = dentistId (para RLS)
  created_at: timestamp
}

// ☆ WhatsAppConnection - Una por dentista
WhatsAppConnection {
  id: uuid
  dentistId: uuid (unique) // tenant_id
  phoneNumber: string
  connectionStatus: enum (disconnected, connecting, connected)

  // Baileys session
  sessionData: string (encrypted JSON)
  qrCode: string

  // Config
  welcomeMessage: text
  businessHours: jsonb
  aiPrompt: text

  tenant_id: uuid // = dentistId (para RLS)
  created_at: timestamp
}
```

```
// ChatSession - Conversaciones de WhatsApp
ChatSession {
  id: uuid
  dentistId: uuid // tenant_id
  patientPhone: string
  patientId: uuid (nullable, hasta identificarse)

  status: enum (active, completed, handed_off)
  context: jsonb

  tenant_id: uuid // = dentistId (para RLS)
  started_at: timestamp
  ended_at: timestamp
}

// ChatMessage
ChatMessage {
  id: uuid
  sessionId: uuid
  role: enum (user, assistant, system)
  content: text
  metadata: jsonb

  created_at: timestamp
}
```

Fases de Implementación

FASE 0: Discovery & Planning (2-4 semanas)

Objetivos:

PROF

- Validar modelo de negocio con dentistas reales
- Diseñar arquitectura completa
- Validar WhatsApp como feature diferenciador
- Preparar diseños UI/UX

Semana 1-2: Customer Development

- ☐ **Entrevistas con 10-15 dentistas**
  - ✅ Validar que TODOS pagarían suscripción
  - ✅ Validar necesidad de WhatsApp chatbot
  - ✅ Validar pacientes cambian de dentista (N:M)
  - ✅ Validar staff trabaja para múltiples dentistas
  - Pain points con software actual
  - Willingness to pay

- [ ] **Entrevistas con 3-5 administradores de clínicas**
  - Cómo gestionan consultorios compartidos
  - Necesidad de super admin para crear clínicas
  - Privacidad de datos clínicos
- [ ] **Análisis competitivo**
  - Trials de Dentrix, Open Dental, Curve Dental
  - **CRÍTICO:** ¿Alguno tiene WhatsApp chatbot con IA?
  - Identificar gaps de mercado
- [ ] **Validación de integraciones**
  - Baileys para WhatsApp (QR code auth)
  - OpenAI GPT-4 API
  - Google Calendar, Outlook, Apple Calendar APIs
  - Stripe

Semana 2-3: Technical Planning

- [ ] **Arquitectura detallada**
  - Diagramas C4 model
  - Database schema completo (ver 02-MODELO-DATOS.md)
  - **CRÍTICO:** Single DB con row-level security
  - **CRÍTICO:** PatientDentistRelations N:M
  - **CRÍTICO:** TenantMemberships (Staff multi-dentista)
  - **CRÍTICO:** OperatoryAssignments (Shared consultorios)
  - API design (REST + GraphQL opcional)
  - Security architecture (HIPAA)
- [ ] **Technical spikes (POCs)**
  - Row-level security con Prisma + PostgreSQL
  - **Baileys WhatsApp connection + QR generation**
  - **OpenAI GPT-4 function calling para chatbot**
  - Google Calendar bidirectional sync
  - Tenant context management (AsyncLocalStorage)

PROF

Semana 3-4: Design & Documentation

- [ ] **UI/UX Design**
  - User personas (ver 05-FLUJOS-PRINCIPALES.md)
  - User journey maps
  - **CRÍTICO:** WhatsApp QR setup flow
  - **CRÍTICO:** Patient portal (ver historial con múltiples dentistas)
  - **CRÍTICO:** Staff dashboard (selector de dentista)

    - Wireframes + High-fidelity mockups
    - Design system

- ☐ **Legal & Compliance**
    - HIPAA compliance checklist
    - BAA template
    - Privacy Policy
    - Terms of Service

- ☐ **Project Setup**
    - GitHub repos
    - Project board
    - Documentation wiki
    - Slack/Discord

Entregables:

- ✅ Customer development informe
- ✅ Arquitectura documentada
- ✅ Database schema Prisma
- ✅ API specification
- ✅ Diseños UI/UX
- ✅ PRD completo
- ✅ Repos configurados

FASE 1: MVP - Core Features (12-16 semanas)

Objetivo: Producto mínimo viable con features esenciales + WhatsApp básico

Features Incluidas:

PROF

- ✅ Multi-tenant authentication (single DB, tenant_id)
- ✅ Patient management (con N:M dentist relations)
- ✅ Appointment scheduling
- ✅ **WhatsApp connection (Baileys + QR)** ☆
- ✅ **WhatsApp mensajes básicos** ☆
- ✅ Email/SMS reminders
- ✅ Super admin dashboard (crear clínicas)
- ✅ Staff multi-dentista (invitaciones)

No Incluido en MVP:

- × WhatsApp AI Chatbot (Fase 2)
- × Odontograma digital (Fase 3)
- × Calendar integrations (Fase 2)
- × Billing/Payments (Fase 3)

Sprint 1-2: Infrastructure & Authentication (4 semanas)

Sprint 1 (Semana 1-2): ✅ COMPLETADO - 30 Diciembre 2025

Backend:

- ☑ Setup NestJS project ✅
  - Modular architecture
  - Config service (env vars)
  - Swagger documentation
  - Exception filters
- ☑ Database setup ✅
  - PostgreSQL Docker (local dev - puerto 5435)
  - Redis Docker (local dev - puerto 6381)
  - **Prisma schema inicial:**
    - User ✅
    - Tenant (cada dentista) ✅
    - TenantMembership (staff multi-dentista) ✅
    - Patient ✅
    - **PatientDentistRelation** ☆ ✅
    - Clinic (creada por super admin) ✅
    - Operatory ✅
    - **OperatoryAssignment** ☆ ✅
  - Migrations ✅
  - Seed scripts ✅
- ☑ **Row-level security foundation** ✅
  - Prisma queries con filtrado por tenant_id
  - Índices en `tenant_id`
  - PatientDentistRelation para multi-tenancy

PROF

- ☑ Authentication básica ✅
  - JWT strategy (access token)
  - Register endpoint ✅
  - Login endpoint ✅
  - Password hashing (bcrypt) ✅
  - **PROBADO CON CURL** ✅

Frontend:

- ☐ Setup Next.js project (PENDIENTE)
  - App router
  - TypeScript
  - Tailwind CSS + shadcn/ui

- [ ] Auth pages (PENDIENTE)
    - Login
    - Register (dentista)
    - Forgot password

DevOps:

- [x] Docker setup ✅
    - docker-compose.yml (PostgreSQL + Redis)
    - Configuración local completa
    - Scripts de setup automatizados
- [ ] GitHub Actions (PENDIENTE)
    - Lint & format
    - Unit tests
    - Build verification
- [ ] AWS (Terraform) (PENDIENTE - Fase posterior)
    - VPC
    - RDS PostgreSQL
    - ElastiCache Redis
    - S3
    - ECR

Sprint 2 (Semana 3-4):

Backend:

- [ ] OAuth integration
    - Google OAuth
    - Apple Sign In
    - Microsoft OAuth
- [ ] Authorization (RBAC + ABAC)
    - Roles: super_admin, dentist, staff_*, patient
    - Guards (role-based + tenant-scoped)
    - CASL integration
    - **Row-level filtering automático en queries**
- [ ] **TenantMembership management** ☆
    - Invite staff endpoint
    - Accept/reject invitation
    - List my workspaces (staff)
    - List staff members (dentista)

PROF

    - Update permissions
- ☐ Super Admin features
    - Create/edit/delete Clinic
    - Create/edit/delete Operatory
    - View all tenants
    - Suspend tenant

Frontend:

- ☐ OAuth buttons
- ☐ **Staff workspace selector** (dropdown en header)
- ☐ Super Admin Dashboard
    - Clinics management
    - Operatories management
    - Tenants list
- ☐ Protected routes
- ☐ State management (Zustand)
    - Auth store
    - Tenant context store

Milestone 1: ✅ Authentication & Multi-tenancy Working - **COMPLETADO 30/12/2025**

Implementación Local Completada:

- ✅ Backend NestJS corriendo en http://localhost:3000
- ✅ PostgreSQL en Docker (puerto 5435)
- ✅ Redis en Docker (puerto 6381)
- ✅ Base de datos con seed data
- ✅ Endpoints de autenticación probados con curl
- ✅ Endpoints de pacientes probados con curl
- ✅ Documentación Swagger en http://localhost:3000/api/docs

PROF

Credenciales de Prueba:

- Admin: admin@dentista.com / Admin123!
- Dentist: dentist@dentista.com / Dentist123!
- Patient: patient@dentista.com / Patient123!

Sprint 3-4: Patient Management (4 semanas)

Sprint 3 (Semana 5-6): 🚧 **EN PROGRESO - Iniciado 30/12/2025**

Backend:

- ☑ Patient module ✅
    - CRUD patients ✅

    - **PROBADO CON CURL** ✅
    - Upload documents (S3) - PENDIENTE
- [x] **PatientDentistRelation module** ☆ ✅
    - Create relation (when creating patient) ✅
    - Update relation (mark as inactive) ✅
    - List patients for dentist (active relations) ✅
    - **PROBADO CON CURL** ✅
    - Transfer patient between dentists - PENDIENTE
- [ ] Medical history
    - Medical history form
    - Allergies & medications

Frontend:

- [ ] Patients list page
    - Search (by name, phone, email)
    - Pagination
    - **Indicator:** Badge showing dentist relation
- [ ] **Add patient flow** ☆
    - Search if patient exists (by email/phone)
    - If exists: "Add to my patients" (create PatientDentistRelation)
    - If not: Create patient + relation
- [ ] Patient detail page
    - Tabs: Overview, Medical History, Appointments, Documents
    - Edit mode
    - **Show all dentist relations** (for patient portal later)

PROF

- [ ] Documents section
    - Upload documents
    - View/download documents

Sprint 4 (Semana 7-8):

Backend:

- [ ] Patient search optimization
    - PostgreSQL full-text search
    - Índices (name, email, phone)
- [ ] Bulk operations

    - Export patients CSV
    - Import patients CSV

- ☐ **Patient portal foundation**
    - Patient registration
    - Patient login
    - View own info

Frontend:

- ☐ Advanced search & filters
- ☐ Export/Import UI
- ☐ Patient analytics dashboard
- ☐ **Basic patient portal**
    - Login page
    - Dashboard
    - View personal info

Milestone 2: ✅ Complete Patient Management with N:M Relations

---

Sprint 5-6: Appointment Scheduling (4 semanas)

Sprint 5 (Semana 9-10): ✅ COMPLETADO - 30 Diciembre 2025

Backend:

- ☑ **OperatoryAssignment module** ⭐ ✅
    - Assign operatory to dentist ✅
    - Define schedule per assignment (JSON) ✅
    - Validate conflicts ✅
    - Check if dentist has operatory access at time ✅
    - **PROBADO CON CURL** ✅

PROF

- ☑ Appointment module ✅
    - Create appointment ✅
    - **Validation:** Dentist has operatory access (check OperatoryAssignment) ✅
    - **Validation:** Patient has relation with dentist (check PatientDentistRelation) ✅
    - Get appointments (by date, by dentist, by operatory) ✅
    - Update/cancel appointment ✅
    - Check availability (considers OperatoryAssignments) ✅
    - **PROBADO CON CURL** ✅

- ☑ **Clinic & Operatory modules** ⭐ ✅
    - Create/update/delete clinics (Super Admin) ✅
    - Create/update/delete operatories ✅

    - Assign operatories to dentists ✅
    - Manage schedules per assignment ✅
    - **PROBADO CON CURL** ✅

Frontend:

- [ ] Calendar view (FullCalendar)
    - Day/Week/Month/Agenda views
    - **Filter by dentist** (for staff managing multiple)
- [ ] Appointment creation modal
    - Select patient (autocomplete - shows only related patients)
    - **Select operatory** (filtered by dentist's assignments)
    - Select date/time (validates availability)
    - Select duration & type
    - Add notes
- [ ] Appointment detail modal
    - View/edit/cancel
    - Mark completed/no-show

Sprint 6 (Semana 11-12):

Backend:

- [ ] Recurring appointments
    - Recurrence patterns
    - Generate series
- [ ] Waitlist
    - Add to waitlist
    - Auto-offer when slot opens
- [ ] Appointment reminders
    - BullMQ job scheduler
    - Email reminder (SendGrid)
    - SMS reminder (Twilio)

Frontend:

- [ ] Recurring appointments UI
- [ ] Waitlist panel
- [ ] **Operatory schedule management** (super admin)
    - Assign consultorios to dentists

PROF

    - Define schedules (days, hours)
    - View utilization
    - Detect conflicts

- ☐ Appointment list view
    - Alternative to calendar
    - Filters

Milestone 3: ✅ Full Appointment Scheduling with Shared Operatories - **COMPLETADO 30/12/2025**

Implementación Completada:

- ✅ Appointments CRUD con validaciones de tenant y paciente
- ✅ Operatory Assignments con schedules configurables
- ✅ Clinics y Operatories management (Super Admin)
- ✅ Conflict detection en appointments
- ✅ Todos los endpoints probados con curl
- ✅ Documentación Swagger actualizada

Sprint 7-8: WhatsApp Integration (4 semanas) ⭐ CRÍTICO

Sprint 7 (Semana 13-14):

Backend:

- ☐ **Baileys setup** ⭐
    - WhatsApp connection manager service
    - QR code generation
    - Session persistence (encrypted in DB)
    - Multi-device support
    - Connection status tracking

PROF

- ☐ **WhatsAppConnection module**
    - Create connection (per dentist)
    - Generate QR endpoint
    - WebSocket for QR updates
    - Store session data (encrypted)
    - Handle connection events (connected, disconnected)

- ☐ **WhatsApp messaging (basic)**
    - Send text message
    - Receive message webhook
    - Message status tracking

Frontend:

- [ ] **WhatsApp connection page** ☆
  - Settings → WhatsApp
  - "Connect WhatsApp" button
  - Display QR code (WebSocket updates)
  - Connection status indicator
  - Disconnect button
- [ ] WhatsApp config
  - Phone number input
  - Welcome message
  - Business hours

Sprint 8 (Semana 15-16):

Backend:

- [ ] **ChatSession & ChatMessage modules**
  - Create session on incoming message
  - Store messages
  - Track session status
- [ ] **WhatsApp automation (basic)**
  - Appointment reminder via WhatsApp
  - Send confirmation messages
  - Handle simple responses ("confirm", "cancel")

Frontend:

- [ ] **WhatsApp templates**
  - Create message templates
  - Variables (patient name, appointment time)
  - Preview templates
- [ ] Send WhatsApp messages manually
  - From patient detail page
  - From communications log
- [ ] **WhatsApp chat history**
  - View conversations with patient
  - Filter by patient

Testing:

- [ ] WhatsApp connection flow E2E
- [ ] QR code generation & scanning

PROF

- ☐ Message send/receive
- ☐ Session persistence

Milestone 4: ✅ WhatsApp Conectado y Mensajes Básicos

Sprint Adicional: Polish & Beta Prep (2 semanas)

Semana 17-18:

- ☐ Bug fixes
- ☐ Performance optimization
    - Database query optimization
    - Índices en tenant_id, patientId, dentistId
    - Frontend bundle size
- ☐ Security audit
    - Dependency scan
    - OWASP Top 10 check
    - Basic penetration testing
- ☐ UX improvements
    - Loading states
    - Error states
    - Empty states
    - Success notifications
- ☐ Documentation
    - API docs (Swagger)
    - User guide (básico)
    - Deployment runbook
- ☐ **HIPAA compliance checklist**
    - Encryption at rest ✓
    - Encryption in transit ✓
    - Access controls ✓
    - Audit logs ✓
    - BAA con AWS ✓

Milestone 5: ✅ MVP Ready for Beta Testing

FASE 2: WhatsApp AI Chatbot & Integrations (8-12 semanas)

Objetivo: Chatbot inteligente con IA + Calendar integrations

PROF

Features:

- ✅ **WhatsApp AI Chatbot (GPT-4)** ☆☆☆
- ✅ Chatbot puede agendar citas
- ✅ Chatbot responde FAQs
- ✅ Google Calendar sync
- ✅ Outlook Calendar sync
- ✅ Apple Calendar sync
- ✅ Portal de pacientes mejorado
- ✅ Online booking

Sprint 9-11: AI Chatbot (6 semanas) ★★★

Sprint 9-10 (Semana 19-22):

Backend:

- ☐ **OpenAI GPT-4 integration** ☆
  - API client
  - Conversation context management
  - Prompt engineering
- ☐ **Chatbot service** ☆
  - Process incoming WhatsApp messages
  - Call GPT-4 with conversation history
  - **Function calling:**
    - `buscar_paciente(name, phone)`
    - `consultar_disponibilidad(date, procedureType)`
    - `agendar_cita(patientId, date, procedureType, notes)`
    - `consultar_proxima_cita(patientId)`
    - `cancelar_cita(appointmentId)`
  - Execute functions and return results
  - Generate response with GPT-4
- ☐ **Knowledge base (RAG)**
  - FAQ embeddings
  - Vector search (pgvector)
  - Retrieval-augmented generation
- ☐ **Tenant-specific prompts**
  - Custom AI instructions per dentist
  - Clinic services info
  - Pricing (optional)

Frontend:

- [ ] **Chatbot config page**
    - AI personality settings
    - Custom instructions textarea
    - Add services & pricing
    - Add FAQs
    - Test chatbot (live preview)
- [ ] **Chatbot analytics**
    - Conversations per day
    - Intents detected
    - Booking conversion rate
    - Common questions
    - Handoff rate

Sprint 11 (Semana 23-24):

Backend:

- [ ] **Advanced chatbot actions**
    - Book appointment (full flow)
    - Reschedule appointment
    - Cancel appointment
    - Send invoice/recipe via WhatsApp
    - Get directions
    - Check clinic hours
- [ ] **Handoff to human**
    - Detect when bot can't help
    - Notify staff
    - Staff can take over chat

PROF

Frontend:

- [ ] **Live chat dashboard** (for staff)
    - See active chats
    - Take over from bot
    - Respond manually
    - View chat history
- [ ] **Chatbot testing tools**
    - Test conversations
    - Debug function calls
    - View GPT-4 prompts & responses

Testing:

- ☐ Chatbot conversation flows
- ☐ Function calling (appointment booking)
- ☐ RAG accuracy
- ☐ E2E: Patient books via WhatsApp

Milestone 6: ✅ AI Chatbot Live ☆

Sprint 12-14: Calendar Integrations (6 semanas)

Sprint 12: Google Calendar

Backend:

- ☐ Google Calendar integration
    - OAuth flow
    - Store refresh tokens
    - Create event when appointment created
    - Update event when appointment updated
    - Delete event when appointment cancelled
    - **Webhook:** Sync from Google Calendar to our system
- ☐ Calendar sync service
    - Background job
    - Conflict detection
    - Conflict resolution (manual)

Frontend:

- ☐ Calendar settings page
    - Connect Google Calendar
    - OAuth flow
    - Select which calendar
    - Sync status
    - Disconnect
- ☐ Conflict resolution UI
    - Show conflicts
    - Allow manual resolution

Sprint 13: Outlook Calendar

Backend:

- ☐ Microsoft Graph integration
    - OAuth flow
    - Create/update/delete events

PROF

    - Webhook sync

Frontend:

- ☐ Connect Outlook Calendar
- ☐ Similar UI to Google Calendar

Sprint 14: Apple Calendar (CalDAV)

Backend:

- ☐ CalDAV integration
    - CalDAV client
    - Create/update/delete events
    - Polling-based sync (no webhooks)

Frontend:

- ☐ Connect Apple Calendar
    - CalDAV credentials input
    - Test connection

Testing:

- ☐ Integration tests (sandbox)
- ☐ E2E full sync cycle

Milestone 7: ✅ All Calendar Integrations Working

FASE 3: Clinical Module & Billing (8-10 semanas)

Objetivo: Odontograma digital, Treatment Plans, Facturación completa

Features:

PROF

- ✅ Odontograma digital interactivo
- ✅ Treatment plans con procedimientos
- ✅ Facturación a pacientes
- ✅ Pagos y planes de pago
- ✅ Recetas médicas
- ✅ Documentos y archivos

Sprint 15-16: Clinical Module - Odontograma & Treatment Plans (4 semanas)

Sprint 15 (Semana 25-26):

Backend:

- ☐ **DentalChart module** ☆

    - Create dental chart (per patient, per dentist)
    - Update tooth status (jsonb structure)
    - Tooth numbering systems (Universal, FDI, Palmer)
    - Conditions: healthy, cavity, filling, crown, missing, etc.
    - History tracking (changes over time)
    - **PROBADO CON CURL** después de implementar

- [ ] **DentalChart schema**

```
model DentalChart {
  id          String   @id @default(uuid())
  patientId   String
  dentistId   String   // tenant_id
  tenantId    String   // RLS
  chartData   Json     // Estructura del odontograma
  version     Int      @default(1)
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  patient     Patient  @relation(fields: [patientId], references:
[id])
  dentist     User     @relation(fields: [dentistId], references:
[id])

  @@index([patientId])
  @@index([dentistId])
  @@index([tenantId])
}
```

- [ ] **Procedure catalog**

    - Predefined procedures (cleaning, filling, extraction, etc.)
    - Custom procedures per dentist
    - Pricing per procedure
    - Duration estimates

PROF

Frontend:

- [ ] **Odontograma digital interactivo** ☆

    - SVG-based tooth diagram
    - Click tooth to mark conditions
    - Color coding (healthy=white, cavity=red, filling=blue, etc.)
    - Tooth numbering overlay
    - Zoom & pan
    - Mobile responsive

- [ ] **Dental chart history**

- Timeline view
- Compare versions
- Show changes over time

Sprint 16 (Semana 27-28):

Backend:

- ☐ **TreatmentPlan module** ☆
  - Create treatment plan
  - Add procedures to plan
  - Calculate total cost
  - Status: draft, proposed, accepted, in_progress, completed
  - Track procedure completion
  - **PROBADO CON CURL** después de implementar
- ☐ **TreatmentPlan schema**

```
model TreatmentPlan {
  id          String   @id @default(uuid())
  patientId   String
  dentistId   String   // tenant_id
  tenantId    String   // RLS
  title       String
  description String?
  status      TreatmentPlanStatus
  totalCost   Decimal  @db.Decimal(10, 2)
  procedures  Json     // Array of procedures
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  patient     Patient  @relation(fields: [patientId], references:
[id])
  dentist     User     @relation(fields: [dentistId], references:
[id])

  @@index([patientId])
  @@index([dentistId])
  @@index([tenantId])
  @@index([status])
}

enum TreatmentPlanStatus {
  draft
  proposed
  accepted
  in_progress
  completed
  cancelled
}
```

PROF

- [ ] **Procedure tracking**
  - Mark procedure as completed
  - Add notes per procedure
  - Upload before/after photos

Frontend:

- [ ] **Treatment plan builder** ☆
  - Add procedures from catalog
  - Drag & drop to reorder
  - Set priority (urgent, high, medium, low)
  - Calculate total automatically
  - Add notes per procedure
  - Save as draft or propose to patient
- [ ] **Treatment plan view (patient)**
  - See proposed plan
  - Accept/decline plan
  - View progress
  - See completed procedures
- [ ] **Clinical notes**
  - Rich text editor
  - Attach to appointment or procedure
  - Search notes

Testing:

- [ ] Odontograma CRUD con curl
- [ ] Treatment plan creation con curl
- [ ] Procedure completion workflow
- [ ] E2E: Create plan → Patient accepts → Complete procedures

PROF

Milestone 8: ✅ Clinical Module Complete

Sprint 17-18: Billing & Payments Module (4 semanas)

Sprint 17 (Semana 29-30):

Backend:

- [ ] **Invoice module** ☆
  - Create invoice (from treatment plan or manual)
  - Line items (procedures, products, services)

- Tax calculation
- Discounts
- Status: draft, sent, paid, overdue, cancelled
- PDF generation
- **PROBADO CON CURL** después de implementar

- [ ] **Invoice schema**

```
model Invoice {
  id          String   @id @default(uuid())
  invoiceNumber String @unique
  patientId   String
  dentistId   String   // tenant_id
  tenantId    String   // RLS
  treatmentPlanId String?

  items       Json     // Line items
  subtotal    Decimal  @db.Decimal(10, 2)
  tax         Decimal  @db.Decimal(10, 2)
  discount    Decimal  @db.Decimal(10, 2) @default(0)
  total       Decimal  @db.Decimal(10, 2)

  status      InvoiceStatus
  dueDate     DateTime?
  paidAt      DateTime?

  notes       String?

  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  patient     Patient  @relation(fields: [patientId], references:
[id])
  dentist     User     @relation(fields: [dentistId], references:
[id])
  treatmentPlan TreatmentPlan? @relation(fields: [treatmentPlanId],
references: [id])
  payments    Payment[]

  @@index([patientId])
  @@index([dentistId])
  @@index([tenantId])
  @@index([status])
  @@index([invoiceNumber])
}

enum InvoiceStatus {
  draft
  sent
  paid
  partial
```

```
  overdue
  cancelled
}
```

- [ ] **PDF generation**
  - Invoice template
  - Clinic branding (logo, colors)
  - QR code for payment
  - Email invoice to patient

Frontend:

- [ ] **Invoice creation** ☆
  - Create from treatment plan (auto-populate)
  - Create manual invoice
  - Add line items
  - Apply discount
  - Preview PDF
  - Send to patient (email + WhatsApp)
- [ ] **Invoice list**
  - Filter by status, patient, date
  - Quick actions (send, mark paid, download PDF)
  - Total revenue summary
- [ ] **Invoice detail**
  - View/edit invoice
  - Payment history
  - Send reminder
  - Download PDF

PROF

Sprint 18 (Semana 31-32):

Backend:

- [ ] **Payment module** ☆
  - Record payment
  - Payment methods: cash, card, transfer, insurance
  - Partial payments
  - Payment plans
  - Refunds
  - **PROBADO CON CURL** después de implementar
- [ ] **Payment schema**

```
```

```
model Payment {
  id          String   @id @default(uuid())
  invoiceId   String
  patientId   String
  dentistId   String   // tenant_id
  tenantId    String   // RLS

  amount      Decimal  @db.Decimal(10, 2)
  method      PaymentMethod
  reference   String?  // Transaction ID, check number, etc.

  notes       String?

  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  invoice     Invoice  @relation(fields: [invoiceId], references:
[id])
  patient     Patient  @relation(fields: [patientId], references:
[id])
  dentist     User     @relation(fields: [dentistId], references:
[id])

  @@index([invoiceId])
  @@index([patientId])
  @@index([dentistId])
  @@index([tenantId])
}

enum PaymentMethod {
  cash
  credit_card
  debit_card
  bank_transfer
  insurance
  check
  other
}
```

PROF

- [ ] **Payment plans**
  - Create installment plan
  - Track installments
  - Automatic reminders
  - Late fees (optional)
- [ ] **Stripe integration (optional)**
  - Online payments
  - Card on file

    - Recurring payments

Frontend:

- [ ] **Payment recording** ⭐
    - Record payment from invoice
    - Select payment method
    - Partial payment support
    - Receipt generation (PDF)
- [ ] **Payment plans**
    - Create plan with installments
    - Track payment schedule
    - Send reminders
- [ ] **Financial reports**
    - Revenue by period
    - Outstanding invoices
    - Payment methods breakdown
    - Top procedures by revenue
- [ ] **Patient billing portal**
    - View invoices
    - Pay online (if Stripe enabled)
    - Download receipts
    - Payment history

Testing:

- [ ] Invoice creation con curl
- [ ] Payment recording con curl
- [ ] Payment plan workflow
- [ ] PDF generation
- [ ] E2E: Create invoice → Send → Record payment

PROF

Milestone 9: ✅ Billing & Payments Complete

Sprint 19-20: Documents & Prescriptions (4 semanas)

Sprint 19 (Semana 33-34):

Backend:

- [ ] **Document module** ⭐
    - Upload documents (S3)

- Document types: x-ray, consent, insurance, prescription, other
- Associate with patient
- Associate with appointment/procedure
- File size limits per subscription tier
- **PROBADO CON CURL** después de implementar

- [ ] **Document schema**

```
model Document {
  id          String   @id @default(uuid())
  patientId   String
  dentistId   String   // tenant_id
  tenantId    String   // RLS

  fileName    String
  fileSize    Int      // bytes
  mimeType    String
  s3Key       String   // S3 object key
  s3Url       String?  // Presigned URL (temporary)

  documentType DocumentType
  category    String?  // Custom category

  appointmentId String?
  procedureId   String?

  description String?
  uploadedBy  String   // User ID

  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  patient     Patient  @relation(fields: [patientId], references:
[id])
  dentist     User     @relation(fields: [dentistId], references:
[id])
  appointment Appointment? @relation(fields: [appointmentId],
references: [id])

  @@index([patientId])
  @@index([dentistId])
  @@index([tenantId])
  @@index([documentType])
}

enum DocumentType {
  xray
  photo
  consent_form
  insurance_card
  prescription
```

```
    lab_report
    referral
    other
  }
```

- [ ] **S3 integration**
  - Upload to S3
  - Generate presigned URLs
  - Delete from S3
  - Virus scanning (ClamAV or AWS)
- [ ] **Storage quota enforcement**
  - Track storage usage per tenant
  - Enforce limits based on subscription
  - Alert when approaching limit

Frontend:

- [ ] **Document upload** ⭐
  - Drag & drop upload
  - Multiple files
  - Progress indicator
  - File type validation
  - Size validation
  - Associate with patient/appointment
- [ ] **Document viewer**
  - Image viewer (zoom, rotate)
  - PDF viewer
  - Download document
  - Delete document
- [ ] **Document library**
  - Filter by type, patient, date
  - Search by filename
  - Bulk actions
  - Storage usage indicator

PROF

Sprint 20 (Semana 35-36):

Backend:

- [ ] **Prescription module** ⭐
  - Create prescription

- Medication database (optional)
- Dosage, frequency, duration
- Digital signature
- PDF generation
- Send via email/WhatsApp
- **PROBADO CON CURL** después de implementar

- [ ] **Prescription schema**

```
model Prescription {
  id          String   @id @default(uuid())
  patientId   String
  dentistId   String   // tenant_id
  tenantId    String   // RLS

  medications Json     // Array of medications
  diagnosis   String?
  notes       String?

  issuedAt    DateTime @default(now())
  expiresAt   DateTime?

  digitalSignature String?

  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  patient     Patient  @relation(fields: [patientId], references:
[id])
  dentist     User     @relation(fields: [dentistId], references:
[id])

  @@index([patientId])
  @@index([dentistId])
  @@index([tenantId])
}
```

PROF

Frontend:

- [ ] **Prescription creator** ☆

  - Add medications
  - Medication search (autocomplete)
  - Dosage input
  - Frequency selector
  - Duration input
  - Diagnosis field
  - Preview PDF
  - Digital signature

- ☐ **Prescription list**
  - View all prescriptions
  - Filter by patient, date
  - Resend prescription
  - Download PDF
- ☐ **Patient prescription view**
  - View my prescriptions
  - Download PDF
  - Receive via WhatsApp

Testing:

- ☐ Document upload con curl
- ☐ S3 integration
- ☐ Prescription creation con curl
- ☐ PDF generation
- ☐ E2E: Upload document → View → Download

Milestone 10: ✅ Documents & Prescriptions Complete

Sprint 15-16: Patient Portal & Online Booking (4 semanas)

Backend:

- ☐ Patient portal API
  - **View all my dentists** (PatientDentistRelations)
  - **View history with each dentist separately**
  - View appointments
  - View treatment plans
  - View documents
  - Update personal info
- ☐ Online booking API
  - Public available slots
  - Book appointment
  - Cancel appointment

Frontend:

- ☐ **Enhanced patient portal** ☆
  - Modern, responsive design
  - **Dashboard:** "My Dentists" section
  - **View history per dentist**
  - My Appointments

PROF

    - My Treatment Plans
    - My Documents
    - Profile settings

- ☐ **Online booking widget**
    - Embeddable (iframe)
    - Select date/time
    - Fill patient info (if new)
    - Confirmation

- ☐ Portal branding
    - Customizable logo
    - Color scheme per tenant

Milestone 11: ✅ Patient Portal & Online Booking Complete

FASE 4: Communication, Notifications & Platform Billing (6-8 semanas)

Objetivo: Sistema de comunicación completo, notificaciones, y billing de plataforma

Features:

- ✅ Sistema de notificaciones (email, SMS, push)
- ✅ Templates de comunicación
- ✅ Platform billing (facturación a dentistas)
- ✅ Subscription management
- ✅ Audit logs & compliance

Sprint 21-22: Communication & Notifications Module (4 semanas)

PROF

Sprint 21 (Semana 37-38):

Backend:

- ☐ **Notification module** ☆
    - Multi-channel notifications (email, SMS, push, WhatsApp)
    - Notification preferences per user
    - Notification history
    - Mark as read/unread
    - **PROBADO CON CURL** después de implementar

- ☐ **Notification schema**

```
model Notification {
  id          String   @id @default(uuid())
```

```
  userId      String
  tenantId    String?  // Null for platform notifications

  type        NotificationType
  channel     NotificationChannel[]

  title       String
  message     String
  data        Json?    // Additional data

  isRead      Boolean  @default(false)
  readAt      DateTime?

  sentAt      DateTime?
  deliveredAt DateTime?
  failedAt    DateTime?
  error       String?

  createdAt   DateTime @default(now())

  user        User     @relation(fields: [userId], references:
[id])

  @@index([userId])
  @@index([tenantId])
  @@index([isRead])
  @@index([type])
}

enum NotificationType {
  appointment_reminder
  appointment_confirmation
  appointment_cancelled
  payment_received
  invoice_sent
  prescription_ready
  document_uploaded
  staff_invitation
  system_alert
}

enum NotificationChannel {
  email
  sms
  push
  whatsapp
  in_app
}
```

PROF

- [ ] **Email service (SendGrid/AWS SES)**
  - Email templates

    - Send transactional emails
    - Track delivery status
    - Bounce handling

- [ ] **SMS service (Twilio)**
    - Send SMS
    - Track delivery
    - Cost tracking per tenant

- [ ] **Push notifications (Firebase)**
    - Web push
    - Mobile push (future)
    - Device token management

Frontend:

- [ ] **Notification center** ☆
    - Bell icon with unread count
    - Dropdown with recent notifications
    - Mark as read
    - View all notifications page
    - Filter by type

- [ ] **Notification preferences**
    - Enable/disable per channel
    - Quiet hours
    - Notification frequency

Sprint 22 (Semana 39-40):

Backend:

PROF

- [ ] **Communication template module**
    - Create templates (email, SMS, WhatsApp)
    - Template variables (patient name, date, etc.)
    - Template categories
    - Default templates
    - **PROBADO CON CURL** después de implementar

- [ ] **Template schema**

```
model CommunicationTemplate {
  id          String   @id @default(uuid())
  tenantId    String?  // Null for system templates
```

```
  name        String
  category    TemplateCategory
  channel     NotificationChannel

  subject     String?  // For email
  body        String
  variables   Json     // Available variables

  isActive    Boolean  @default(true)
  isDefault   Boolean  @default(false)

  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  tenant      Tenant?  @relation(fields: [tenantId], references:
[id])

  @@index([tenantId])
  @@index([category])
}

enum TemplateCategory {
  appointment_reminder
  appointment_confirmation
  welcome
  invoice
  prescription
  marketing
  custom
}
```

- [ ] **Bulk messaging**
  - Send to multiple patients
  - Filter recipients
  - Schedule send time
  - Track delivery

PROF

Frontend:

- [ ] **Template manager** ☆
  - Create/edit templates
  - Template preview
  - Variable insertion
  - Test send
  - Template library
- [ ] **Bulk messaging UI**
  - Select recipients

- Choose template
- Preview message
- Schedule or send now
- Delivery report

Testing:

- ☐ Notification creation con curl
- ☐ Email sending
- ☐ SMS sending
- ☐ Template rendering
- ☐ E2E: Create template → Send notification → Receive

Milestone 12: ✅ Communication & Notifications Complete

Sprint 23-24: Platform Billing & Subscription Management (4 semanas)

Sprint 23 (Semana 41-42):

Backend:

- ☐ **Subscription module** ☆
  - Subscription plans (Starter, Professional, Enterprise)
  - Features per plan
  - Pricing tiers
  - Trial management
  - **PROBADO CON CURL** después de implementar
- ☐ **Subscription schema**

```
model SubscriptionPlan {
  id          String   @id @default(uuid())
  name        String
  description String?

  price       Decimal  @db.Decimal(10, 2)
  currency    String   @default("USD")
  interval    BillingInterval

  features    Json     // Feature flags
  limits      Json     // Usage limits

  maxPatients Int?
  maxStaff    Int?
  storageGB   Int

  isActive    Boolean  @default(true)
```

PROF

```
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  tenants     Tenant[]

  @@index([isActive])
}

enum BillingInterval {
  monthly
  yearly
}

model PlatformInvoice {
  id          String   @id @default(uuid())
  tenantId    String

  invoiceNumber String @unique

  subscriptionPlanId String
  billingPeriodStart DateTime
  billingPeriodEnd   DateTime

  subtotal    Decimal  @db.Decimal(10, 2)
  tax         Decimal  @db.Decimal(10, 2)
  total       Decimal  @db.Decimal(10, 2)

  status      PlatformInvoiceStatus
  dueDate     DateTime
  paidAt      DateTime?

  stripeInvoiceId String?

  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  tenant      Tenant   @relation(fields: [tenantId], references:
[id])
  plan        SubscriptionPlan @relation(fields:
[subscriptionPlanId], references: [id])

  @@index([tenantId])
  @@index([status])
  @@index([invoiceNumber])
}

enum PlatformInvoiceStatus {
  draft
  open
  paid
  void
  uncollectible
}
```

- [ ] **Stripe subscription integration**
  - Create Stripe customer
  - Create subscription
  - Handle webhooks (invoice.paid, subscription.updated, etc.)
  - Cancel subscription
  - Update payment method
- [ ] **Usage tracking**
  - Track patient count
  - Track storage usage
  - Track API calls (if applicable)
  - Enforce limits

Frontend:

- [ ] **Subscription management (dentist)** ☆
  - View current plan
  - Upgrade/downgrade plan
  - Update payment method
  - View billing history
  - Download invoices
- [ ] **Usage dashboard**
  - Current usage vs limits
  - Storage usage
  - Patient count
  - Upgrade prompts

PROF

Sprint 24 (Semana 43-44):

Backend:

- [ ] **Super admin platform management**
  - View all tenants
  - View revenue metrics
  - Suspend/activate tenant
  - Override limits (temporary)
  - **PROBADO CON CURL** después de implementar
- [ ] **Audit log module**
  - Log all critical actions
  - User actions
  - Data changes

- Login attempts
- HIPAA compliance

- [ ] **AuditLog schema**

```
model AuditLog {
  id          String   @id @default(uuid())
  tenantId    String?
  userId      String?

  action      String   // e.g., "patient.created", "invoice.paid"
  resource    String   // e.g., "Patient", "Invoice"
  resourceId  String?

  changes     Json?    // Before/after data
  metadata    Json?    // Additional context

  ipAddress   String?
  userAgent   String?

  createdAt   DateTime @default(now())

  user        User?    @relation(fields: [userId], references:
[id])

  @@index([tenantId])
  @@index([userId])
  @@index([action])
  @@index([createdAt])
}
```

Frontend:

PROF

- [ ] **Super admin dashboard** ☆

- Platform metrics
- Revenue charts
- Active subscriptions
- Churn rate
- Tenant list with status

- [ ] **Audit log viewer**

- Search logs
- Filter by user, action, date
- Export logs
- Compliance reports

Testing:

- ☐ Subscription creation con curl
- ☐ Stripe webhook handling
- ☐ Usage limit enforcement
- ☐ Audit log creation
- ☐ E2E: Sign up → Trial → Subscribe → Upgrade

Milestone 13: ✅ Platform Billing & Compliance Complete

FASE 5: Analytics, Reports & Advanced Features (6-8 semanas)

Objetivo: Analytics completo, reportes financieros, insurance management

Features:

- ✅ Advanced analytics dashboard
- ✅ Financial reports
- ✅ Patient reports
- ✅ Insurance management
- ✅ Inventory management (optional)

Sprint 25-26: Analytics & Reports Module (4 semanas)

Sprint 25 (Semana 45-46):

Backend:

- ☐ **Analytics module** ☆
  - Revenue analytics
  - Appointment analytics
  - Patient analytics
  - Operatory utilization
  - **PROBADO CON CURL** después de implementar
- ☐ **Analytics endpoints**
  - GET /analytics/revenue (by period, dentist)
  - GET /analytics/appointments (by status, type, dentist)
  - GET /analytics/patients (new, active, retention)
  - GET /analytics/procedures (most common, revenue by procedure)
  - GET /analytics/operatory-utilization
- ☐ **Data aggregation jobs**
  - Daily metrics calculation
  - Monthly reports
  - Cache frequently accessed metrics

Frontend:

PROF

- [ ] **Analytics dashboard** ⭐
  - Revenue charts (line, bar)
  - Appointment metrics
  - Patient growth
  - Top procedures
  - Date range selector
  - Export to PDF/Excel
- [ ] **Financial reports**
  - Revenue by period
  - Revenue by procedure type
  - Outstanding invoices
  - Payment methods breakdown
  - Tax reports

Sprint 26 (Semana 47-48):

Backend:

- [ ] **Report generation module**
  - Generate PDF reports
  - Generate Excel reports
  - Schedule automated reports
  - Email reports
  - **PROBADO CON CURL** después de implementar
- [ ] **Report types**
  - Financial summary
  - Patient list
  - Appointment history
  - Treatment plan status
  - Compliance reports (HIPAA)

PROF

Frontend:

- [ ] **Report builder** ⭐
  - Select report type
  - Configure parameters
  - Preview report
  - Download or email
  - Schedule recurring reports
- [ ] **Patient analytics**
  - Patient lifetime value
  - Retention rate

    - Referral sources
    - Demographics

Testing:

- [ ] Analytics endpoints con curl
- [ ] Report generation
- [ ] PDF/Excel export
- [ ] E2E: View analytics → Generate report → Download

Milestone 14: ✅ Analytics & Reports Complete

Sprint 27-28: Insurance & Advanced Features (4 semanas)

Sprint 27 (Semana 49-50):

Backend:

- [ ] **Insurance module**
    - Store patient insurance info
    - Insurance providers catalog
    - Coverage verification (manual)
    - Claims tracking
    - **PROBADO CON CURL** después de implementar
- [ ] **Insurance schema**

```
model Insurance {
  id          String   @id @default(uuid())
  patientId   String

  provider    String
  policyNumber String
  groupNumber String?

  subscriberName String
  subscriberDOB  DateTime?
  relationship   String   // self, spouse, child, other

  coverageType   String   // primary, secondary
  effectiveDate  DateTime
  expirationDate DateTime?

  isActive    Boolean  @default(true)

  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  patient     Patient  @relation(fields: [patientId], references:
```

PROF

```
[id])

  @@index([patientId])
  @@index([isActive])
}

model InsuranceClaim {
  id          String   @id @default(uuid())
  patientId   String
  dentistId   String   // tenant_id
  tenantId    String   // RLS
  insuranceId String
  invoiceId   String?

  claimNumber String   @unique

  dateOfService DateTime
  procedures    Json     // Procedures claimed

  amountBilled  Decimal  @db.Decimal(10, 2)
  amountApproved Decimal? @db.Decimal(10, 2)
  amountPaid    Decimal? @db.Decimal(10, 2)

  status      ClaimStatus

  submittedAt DateTime?
  processedAt DateTime?
  paidAt      DateTime?

  notes       String?

  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  patient     Patient  @relation(fields: [patientId], references:
[id])
  dentist     User     @relation(fields: [dentistId], references:
[id])
  insurance   Insurance @relation(fields: [insuranceId],
references: [id])
  invoice     Invoice? @relation(fields: [invoiceId], references:
[id])

  @@index([patientId])
  @@index([dentistId])
  @@index([tenantId])
  @@index([status])
}

enum ClaimStatus {
  draft
  submitted
  pending
```

```
  approved
  partially_approved
  denied
  paid
}
```

Frontend:

- [ ] **Insurance management** ☆
  - Add patient insurance
  - View coverage details
  - Verify eligibility (manual)
  - Insurance card upload
- [ ] **Claims management**
  - Create claim from invoice
  - Track claim status
  - Record payments
  - Denial management

Sprint 28 (Semana 51-52):

Backend:

- [ ] **Inventory module (optional)**
  - Product catalog
  - Stock tracking
  - Low stock alerts
  - Purchase orders
  - **PROBADO CON CURL** después de implementar

PROF

- [ ] **Inventory schema**

```
model Product {
  id          String   @id @default(uuid())
  tenantId    String   // RLS

  name        String
  description String?
  sku         String?
  category    String

  unitPrice   Decimal  @db.Decimal(10, 2)
  cost        Decimal? @db.Decimal(10, 2)

  currentStock Int     @default(0)
  minStock    Int      @default(0)
```

```
  isActive    Boolean  @default(true)

  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  tenant      Tenant   @relation(fields: [tenantId], references:
[id])

  @@index([tenantId])
  @@index([category])
}

model StockMovement {
  id          String   @id @default(uuid())
  productId   String
  tenantId    String   // RLS

  type        MovementType
  quantity    Int

  reference   String?  // Invoice ID, PO number, etc.
  notes       String?

  createdBy   String
  createdAt   DateTime @default(now())

  product     Product  @relation(fields: [productId], references:
[id])
  tenant      Tenant   @relation(fields: [tenantId], references:
[id])
  user        User     @relation(fields: [createdBy], references:
[id])

  @@index([productId])
  @@index([tenantId])
}

enum MovementType {
  purchase
  sale
  adjustment
  return
}
```

PROF

Frontend:

- ☐ **Inventory management** ☆
  - Product catalog
  - Add/edit products

    - Stock levels
    - Low stock alerts
    - Stock movements history

- ☐ **Purchase orders**
    - Create PO
    - Receive stock
    - Track vendor orders

Testing:

- ☐ Insurance CRUD con curl
- ☐ Claims tracking con curl
- ☐ Inventory management con curl
- ☐ E2E: Add insurance → Create claim → Track → Record payment

Milestone 15: ✅ Insurance & Inventory Complete

🎯 RESUMEN DE FASES Y MILESTONES

✅ FASE 1: MVP - Core Features (Completada 30/12/2025)

- **Milestone 1:** Authentication & Multi-tenancy ✅
- **Milestone 2:** Patient Management ✅
- **Milestone 3:** Appointment Scheduling ✅
- **Milestone 4:** WhatsApp Basic Integration ✅
- **Milestone 5:** MVP Ready for Beta ✅

🚧 FASE 2: WhatsApp AI Chatbot & Integrations (8-12 semanas)

- **Milestone 6:** AI Chatbot Live ☆
- **Milestone 7:** Calendar Integrations Complete

PROF

📋 FASE 3: Clinical Module & Billing (8-10 semanas)

- **Milestone 8:** Clinical Module Complete (Odontograma + Treatment Plans)
- **Milestone 9:** Billing & Payments Complete
- **Milestone 10:** Documents & Prescriptions Complete
- **Milestone 11:** Patient Portal & Online Booking Complete

📊 FASE 4: Communication & Platform Billing (6-8 semanas)

- **Milestone 12:** Communication & Notifications Complete
- **Milestone 13:** Platform Billing & Compliance Complete

📈 FASE 5: Analytics & Advanced Features (6-8 semanas)

- **Milestone 14:** Analytics & Reports Complete

- **Milestone 15:** Insurance & Inventory Complete

📊 ENTIDADES COMPLETAS DEL SISTEMA

Core Entities (Implementadas)

- ✅ User
- ✅ Tenant
- ✅ TenantMembership
- ✅ Patient
- ✅ PatientDentistRelation
- ✅ Clinic
- ✅ Operatory
- ✅ OperatoryAssignment
- ✅ Appointment

Clinical Entities (Por Implementar)

- ☐ DentalChart
- ☐ TreatmentPlan
- ☐ Procedure (catalog)
- ☐ ClinicalNote

Billing Entities (Por Implementar)

- ☐ Invoice
- ☐ Payment
- ☐ PaymentPlan

Document Entities (Por Implementar)

- ☐ Document
- ☐ Prescription

PROF

Communication Entities (Por Implementar)

- ☐ Notification
- ☐ CommunicationTemplate
- ☐ WhatsAppConnection (Implementada parcialmente)
- ☐ ChatSession
- ☐ ChatMessage

Platform Entities (Por Implementar)

- ☐ SubscriptionPlan
- ☐ PlatformInvoice
- ☐ AuditLog

Insurance & Inventory (Por Implementar)

- [ ] Insurance
- [ ] InsuranceClaim
- [ ] Product
- [ ] StockMovement

🔧 STACK TECNOLÓGICO COMPLETO

Backend

- **Framework:** NestJS ✅
- **ORM:** Prisma ✅
- **Database:** PostgreSQL ✅
- **Cache:** Redis ✅
- **Authentication:** Passport.js (JWT, OAuth) ✅
- **File Storage:** AWS S3
- **Email:** SendGrid / AWS SES
- **SMS:** Twilio
- **Push Notifications:** Firebase
- **PDF Generation:** PDFKit / Puppeteer
- **WhatsApp:** Baileys
- **AI:** OpenAI GPT-4
- **Payment:** Stripe
- **Queue:** BullMQ

Frontend

- **Framework:** Next.js 14 (App Router)
- **Language:** TypeScript
- **Styling:** Tailwind CSS
- **Components:** shadcn/ui
- **State:** Zustand
- **Forms:** React Hook Form + Zod
- **Charts:** Recharts / Chart.js
- **Calendar:** FullCalendar
- **Tables:** TanStack Table

PROF

DevOps

- **Containers:** Docker ✅
- **Orchestration:** Docker Compose (local) ✅
- **Cloud:** AWS
- **IaC:** Terraform
- **CI/CD:** GitHub Actions
- **Monitoring:** CloudWatch / Sentry
- **Logs:** Winston / CloudWatch Logs

📝 TESTING PROTOCOL

IMPORTANTE: Todos los endpoints deben ser probados con curl usando autenticación JWT.

Credenciales de Prueba

- **Admin:** admin@dentista.com / Admin123!
- **Dentist:** dentist@dentista.com / Dentist123!
- **Patient:** patient@dentista.com / Patient123!

Ejemplo de Testing

```
# 1. Login
TOKEN=$(curl -s -X POST http://localhost:3000/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"dentist@dentista.com","password":"Dentist123!"}' \
  | jq -r '.accessToken')

# 2. Test endpoint
curl -X GET http://localhost:3000/patients \
  -H "Authorization: Bearer $TOKEN"
```

🎯 PRÓXIMOS PASOS INMEDIATOS

1. **Continuar con Sprint 2:** OAuth integration y RBAC avanzado
2. **Implementar TenantMembership:** Invitaciones de staff
3. **Frontend Next.js:** Iniciar desarrollo del dashboard
4. **Deployment:** Preparar infraestructura AWS con Terraform

---

PROF

ROADMAP ACTUALIZADO: 30 de Diciembre, 2025
Versión: 2.0 - Completo con todos los módulos requeridos
Estado: Fase 1 completada, Fases 2-5 planificadas en detalle

- [ ] Invoice PDF generation
- [ ] Email invoice

Frontend:

- [ ] Invoice creation
  - From appointment
  - From treatment plan
  - Manual
  - Add/edit line items
- [ ] Invoice list & detail

- [ ] Invoice settings

Sprint 18: Stripe Integration

Backend:

- [ ] Stripe integration
    - Connect Stripe account
    - Payment intent
    - Webhooks (payment.succeeded, etc.)
    - Refunds
- [ ] Payment methods
    - Credit/debit card
    - ACH
    - Save payment method
- [ ] Payment plans
    - Installments
    - Recurring billing

Frontend:

- [ ] Stripe connect
- [ ] Payment processing UI
    - Stripe Elements (card input)
    - Payment confirmation
    - Receipt
- [ ] Payment plans UI

PROF

Sprint 19: Insurance (Basic)

Backend:

- [ ] Insurance verification
    - Clearinghouse integration (Availity)
    - Verify eligibility
    - Get coverage
- [ ] Claims (basic)
    - Generate claim
    - Submit claim
    - Track status

Frontend:

- ☐ Insurance verification UI
- ☐ Claims tracking

Milestone 9: ✅ Complete Billing System

Sprint 20-22: Odontograma & Clinical (6 semanas)

Sprint 20-21: Odontograma Digital

Backend:

- ☐ DentalChart schema
  - Tooth-by-tooth data
  - Periodontal charting
  - Versioning
- ☐ Dental chart API
  - Get chart
  - Update tooth status
  - Add findings
  - Snapshot comparisons

Frontend:

- ☐ **Odontograma component** ☆
  - SVG-based teeth
  - Interactive selection
  - Universal numbering system
  - Charting tools (caries, filling, crown, missing)
  - Surface selection
  - Notes per tooth

PROF

- ☐ Periodontal charting
  - Pocket depths
  - Bleeding on probing
  - Mobility
- ☐ Chart history
  - View previous
  - Compare side-by-side
- ☐ Print/Export PDF

Sprint 22: Treatment Planning

Backend:

- [ ] Procedure catalog
    - Seed ADA codes
    - CRUD procedures
- [ ] TreatmentPlan module
    - Create plan
    - Add procedures
    - Calculate cost
    - Update status
- [ ] Generate PDF

Frontend:

- [ ] Treatment plan builder
- [ ] Treatment plan view
- [ ] Treatment plan PDF

Milestone 10: ✅ Odontograma & Treatment Planning Complete

Sprint 23-24: Analytics & Inventory (4 semanas)

Sprint 23: Analytics

Backend:

- [ ] Analytics service
    - Aggregate data
    - Calculate KPIs
    - Cache results
- [ ] KPI endpoints
    - Financial (revenue, A/R)
    - Operational (appointments, no-shows)
    - Provider performance

PROF

Frontend:

- [ ] Analytics dashboard
    - KPI cards
    - Charts (line, bar, pie)
    - Date range selector
- [ ] Reports
    - Revenue report

    - Production report
    - Appointment report
    - Custom report builder

Sprint 24: Inventory

Backend:

- ☐ Inventory module
    - CRUD items
    - Track quantity
    - Low stock alerts

Frontend:

- ☐ Inventory list
- ☐ Add/edit item
- ☐ Usage tracking
- ☐ Reorder alerts

Milestone 11: ✅ Fase 3 Complete

FASE 4: Mobile & Scale (8-12 semanas)

Objetivo: Mobile apps + DSO features

Features:

- ✅ Multi-location (DSO)
- ✅ Mobile app pacientes (React Native)
- ✅ Mobile app providers (React Native)
- ✅ Telesalud

PROF

Sprint 25-26: Multi-Location (4 semanas)

Backend:

- ☐ Location entity
- ☐ Multi-location model
- ☐ Cross-location scheduling
- ☐ Consolidated reporting

Frontend:

- ☐ Location management
- ☐ Location switcher
- ☐ DSO dashboard

Sprint 27-29: Mobile Apps (6 semanas)

Patient App (React Native):

- [ ] Authentication
- [ ] Dashboard
- [ ] View/book appointments
- [ ] View treatment plans
- [ ] View documents
- [ ] Push notifications
- [ ] Chat with clinic
- [ ] Payments

Provider App (React Native):

- [ ] Today's schedule
- [ ] Patient charts
- [ ] Dental charting
- [ ] Treatment notes
- [ ] View documents

Sprint 30: Telesalud (2 semanas)

Backend:

- [ ] WebRTC signaling server
- [ ] Video appointment type

Frontend:

- [ ] Video call UI (Twilio Video)
- [ ] Screen share
- [ ] Chat during call

Milestone 12: ✅ Full Product - Scale Ready

PROF

Plan de Deployment

Ambientes

Development

- Local (docker-compose)
- PostgreSQL + Redis local
- LocalStack (S3)

Staging

- AWS ECS Fargate
- RDS PostgreSQL (db.t3.small)

- ElastiCache Redis (cache.t3.micro)
- S3 bucket
- Domain: staging.denticloud.com

Production

- AWS ECS Fargate (auto-scaling)
- RDS PostgreSQL Multi-AZ (db.r5.large+)
- ElastiCache Redis Multi-AZ (cache.r5.large)
- S3 + CloudFront
- Domain: app.denticloud.com

CI/CD Pipeline

```
# GitHub Actions

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main, develop]

jobs:
  lint:
    - ESLint
    - Prettier
    - TypeScript check

  test:
    - Unit tests (Jest/Vitest)
    - Integration tests
    - Coverage report (Codecov)

  build:
    - Build backend Docker image
    - Build frontend Docker image
    - Push to ECR

  deploy-staging:
    if: branch == develop
    - Terraform plan/apply
    - Deploy to ECS
    - Run smoke tests
    - Notify team (Slack)

  deploy-production:
    if: branch == main
    - Manual approval
    - Terraform plan/apply
    - Blue-green deployment
```

```
      - Smoke tests
      - Rollback if fail
      - Notify team
```

Database Migrations

Estrategia:

- Prisma migrations
- Migrations en Git
- Execute before deploy (CI/CD)

Single DB Advantage:

- ✅ Una sola migration para todos los tenants
- ✅ Más simple que database-per-tenant
- ✅ Backup antes de migration
- ✅ Rollback strategy

Monitoring & Alerting

Metrics:

- Request rate, error rate, latency
- Database query time, connections
- Cache hit rate
- Queue length
- Tenant-specific metrics

Tools:

- CloudWatch (infra)
- Sentry (errors)
- DataDog (APM)
- PagerDuty (on-call)

PROF

Alerts:

- Error rate > 1%
- P95 latency > 1000ms
- Database connections > 80%
- Failed jobs

Backup & DR

Database:

- Automated daily backups (RDS)
- Retention: 30 days
- Point-in-time recovery

- Cross-region replication

S3:

- Versioning enabled
- Lifecycle policies

DR Plan:

- RPO: < 1 hour
- RTO: < 4 hours
- Quarterly drills

Plan de Testing

Testing Pyramid

```
           E2E Tests (Playwright)
         Integration Tests (Jest)
       Unit Tests (Jest/Vitest)
```

Unit Tests

Backend:

- Services (business logic)
- Utilities
- Validators
- **Row-level security filtering**

Target: 80%+ coverage

PROF

Frontend:

- Components
- Hooks
- Utilities
- State management

Integration Tests

Backend:

- API endpoints
- Database operations
- **Multi-tenant queries (tenant_id filtering)**
- External services (mocked)

E2E Tests

Critical Flows:

- User registration & login
- **Create patient with dentist relation**
- **Staff switch between dentists**
- Book appointment
- **WhatsApp QR setup & connection**
- **Chatbot conversation & booking**
- Process payment

Performance Testing

Tools: k6

Scenarios:

- Load test: 100 concurrent users
- Stress test
- Spike test
- Soak test

Focus:

- Single DB query performance with tenant_id filters
- Index efficiency
- Connection pooling

Security Testing

OWASP Top 10:

- Injection
- Broken Authentication
- Sensitive Data Exposure
- XSS
- Broken Access Control
- **Tenant isolation (critical!)**

Tools:

- OWASP ZAP
- Snyk
- npm audit

Frequency:

- Automated: Every build
- OWASP ZAP: Weekly (staging)

- Professional pentest: Before launch, then annually

Estrategia de Go-to-Market

Pre-Launch (Durante Desarrollo)

Meses 1-3:

- [ ] Landing page
  - Value proposition
  - **Destacar WhatsApp chatbot como diferenciador**
  - Early access signup
  - Blog (SEO)
- [ ] Content marketing
  - Artículos sobre:
    - "Automatiza tu consultorio dental con WhatsApp"
    - "Cómo la IA puede ayudar a tu práctica dental"
    - HIPAA compliance
  - SEO optimization
- [ ] Social media
  - LinkedIn (B2B)
  - Facebook groups (dental professionals)
  - Instagram

Meses 4-6:

- [ ] **Beta program**
  - Recruit 5-10 dentistas
  - Free durante beta (3-6 meses)
  - Feedback sessions
  - **Testimonios del WhatsApp chatbot**
- [ ] Partnerships
  - Asociaciones de dentistas
  - Dental schools
  - Equipment suppliers
- [ ] Influencer outreach
  - Dentistas con presencia digital

Launch (Fin de Fase 2)

Estrategia:

PROF

- [ ] Soft launch (beta users)
- [ ] Refinar
- [ ] Public launch

Canales:

- [ ] Product Hunt
- [ ] Dental industry publications
- [ ] Email marketing
- [ ] LinkedIn ads
- [ ] Google Ads ("dental software with WhatsApp", "AI dental assistant")

Launch Content:

- [ ] Demo video (**mostrar WhatsApp chatbot**)
- [ ] Case studies
- [ ] Press release

Post-Launch Growth

Inbound:

- [ ] SEO content
    - Comparisons ("DentiCloud vs Dentrix")
    - "Best dental software with WhatsApp"
    - How-to guides
- [ ] Webinars
    - "Automate your practice with AI"
    - HIPAA compliance

Outbound:

PROF

- [ ] Target DSOs
- [ ] Large clinics (5+ dentists)
- [ ] Dental conferences
    - ADA annual session
    - Booth demos (**live WhatsApp chatbot demo**)

Referral Program:

- [ ] 1 month free for referrer
- [ ] Discount for referee

Customer Success:

- [ ] Onboarding
    - **WhatsApp setup assistance**

    - Data migration
    - Training
- [ ] Regular check-ins
- [ ] User community

Pricing Strategy

IMPORTANTE: Pricing por DENTISTA (tenant), NO por clínica

Tier 1: Starter - $99/mes

- 1 dentista
- 500 pacientes activos
- Features básicos
- **WhatsApp mensajes básicos**
- Email support

Tier 2: Professional - $299/mes ⭐ (Recommended)

- 1 dentista
- Pacientes ilimitados
- **WhatsApp AI Chatbot** ⭐
- Calendar integrations
- Odontograma digital
- Todos los features
- Priority support

Tier 3: Enterprise - Custom

- Multi-location (DSOs)
- Dentistas ilimitados
- White-label
- Dedicated account manager
- Custom integrations
- SLA

Add-ons:

- Advanced AI features: +$50/mes
- Telesalud: +$30/mes
- SMS adicionales: $0.01/SMS

Free Trial: 30 días (no credit card required)

Discounts:

- 10% annual billing
- 20% dental schools

PROF

Métricas de Éxito

Métricas de Desarrollo

Métrica	Target
Sprint velocity	30-40 story points
Code coverage	>80%
Build success rate	>95%
Deploy frequency	Daily (staging), Weekly (prod)
MTTR	<1 hour

Métricas de Producto (Post-Launch)

Métrica	3 Meses	6 Meses	12 Meses
Sign-ups	50	150	500
Paying customers	20	75	250
MRR	$5k	$20k	$70k
Churn rate	<10%	<5%	<5%
NPS	>40	>50	>60
WhatsApp chatbot adoption	80%	90%	95%

Métricas Técnicas

Métrica	Target
Uptime	99.9%
API response time (p95)	<500ms
Page load time	<2s
Error rate	<0.1%
Database query time (p95)	<100ms

PROF

Riesgos y Mitigaciones

Riesgos de Desarrollo

Riesgo	Probabilidad	Impacto	Mitigación
Row-level security bugs	Media	Alto	Extensive testing, code reviews, audits

Riesgo	Probabilidad	Impacto	Mitigación
WhatsApp Baileys instability	Media	Alto	Backup plan: WhatsApp Business API
Performance issues (single DB)	Baja	Medio	Índices, connection pooling, caching
Security vulnerability	Baja	Crítico	Regular audits, dependency scanning

Riesgos de Producto

Riesgo	Probabilidad	Impacto	Mitigación
No product-market fit	Media	Crítico	Customer development, beta program
WhatsApp chatbot no agrega valor	Baja	Alto	Validar en discovery, beta testing
Competencia lanza similar	Media	Medio	Velocity, UX diferenciada
Adopción lenta	Media	Alto	Free trial, content marketing

Riesgos de Compliance

Riesgo	Probabilidad	Impacto	Mitigación
Violación HIPAA	Baja	Crítico	Compliance desde día 1, audits
Data breach	Baja	Crítico	Defense-in-depth, encryption, monitoring
Tenant data leakage	Baja	Crítico	**Row-level security testing exhaustivo**

Presupuesto Estimado

PROF

Costos de Desarrollo (MVP - 4 meses)

Item	Costo Mensual	Total (4 meses)
Equipo		
Tech Lead	$12,000	$48,000
2x Full-stack Developers	$16,000	$64,000
UI/UX Designer	$8,000	$32,000
QA Engineer	$6,000	$24,000
DevOps (part-time)	$4,000	$16,000
Subtotal equipo	**$46,000**	**$184,000**
Infraestructura		

Item	Costo Mensual	Total (4 meses)
AWS (staging + dev)	$500	$2,000
Tools (GitHub, Figma)	$200	$800
Subtotal infra	**$700**	**$2,800**
Otros		
Legal (HIPAA)	-	$5,000
Security audit	-	$10,000
Contingency (10%)	-	$20,180
TOTAL MVP	**$46,700**	**$221,980**

Costos Operacionales (Post-Launch, mensual)

Item	Costo
Infraestructura	
AWS (production)	$2,000
Monitoring (DataDog)	$500
Software	
Twilio (SMS)	$200
SendGrid (Email)	$150
OpenAI API	**$500**
Stripe fees	Variable
Equipo	
2-3 developers	$24,000
1 customer success	$5,000
Marketing	
Content + Ads	$5,000
TOTAL MENSUAL	**~$37,350**

PROF

Break-Even Analysis

- Costo mensual: ~$37,000
- Precio promedio: $200
- **Break-even: ~185 paying customers**
- **Timeline:** 12-18 meses

Consideraciones Especiales del Modelo

Diferencias vs. Modelo Multi-DB

✅ NUESTRO MODELO (Single DB):

```
// Todas las queries automáticamente filtran por tenant_id
const patients = await prisma.patient.findMany({
  where: {
    patientDentistRelations: {
      some: {
        dentistId: currentTenantId,
        isActive: true
      }
    }
  }
});
```

Ventajas:

- ✅ Más simple (una sola DB, una migration)
- ✅ Queries cross-tenant posibles (analytics)
- ✅ Connection pooling más eficiente
- ✅ Backups más simples

Desafíos:

- ⚠ Row-level security CRÍTICO (testing exhaustivo)
- ⚠ Índices en tenant_id esenciales
- ⚠ Queries deben SIEMPRE incluir tenant_id

Tenant Isolation Testing

PROF

Escenarios Críticos:

- ✅ Dentista A no puede ver pacientes de Dentista B
- ✅ Staff de Dentista A no puede ver datos de Dentista B
- ✅ Paciente puede ver historial con TODOS sus dentistas
- ✅ Transfer de paciente actualiza relaciones correctamente
- ✅ WhatsApp chatbot solo accede a datos del dentista correcto

---

Conclusión

Este roadmap implementa DentiCloud con el **modelo de negocio correcto**:

Modelo Correcto ✅

- ✅ **TODOS los dentistas pagan suscripción**

- ✅ **Single database con row-level security**
- ✅ **Pacientes N:M con dentistas**
- ✅ **Staff trabaja para múltiples dentistas**
- ✅ **Clínicas creadas por super admin**
- ✅ **WhatsApp chatbot como diferenciador clave** ⭐

Próximos Pasos

1. ✅ **Aprobar roadmap**
2. ✅ **Formar equipo**
3. ✅ **Iniciar Fase 0** (Discovery)
   - **CRÍTICO:** Validar WhatsApp como feature must-have
   - **CRÍTICO:** Validar modelo N:M pacientes-dentistas
   - Technical spike: Row-level security + Baileys
4. ✅ **Setup infrastructure**
5. ✅ **Kick off Sprint 1**

Filosofía

- **Lean & Agile:** Entregar valor temprano
- **Customer-centric:** Beta users desde día 1
- **Quality-first:** No sacrificar calidad
- **Compliance-native:** HIPAA desde el inicio
- **AI-powered:** WhatsApp chatbot como diferenciador

Versión: 3.0 - **MODELO CORRECTO FINAL**
Fecha: 30 de Diciembre, 2025

Cambios principales en v3.0:

- ✅ **Single Database** con row-level security (NO multi-DB)
- ✅ **TODOS los dentistas pagan** (NO "dentista invitado gratis")
- ✅ **Pacientes N:M con dentistas** (PatientDentistRelations)
- ✅ **Staff multi-dentista** (TenantMemberships)
- ✅ **WhatsApp/Baileys en MVP/Fase 2** (NO Fase 3)
- ✅ **Clínicas creadas por super admin** (NO son tenants que pagan)
- ✅ **OperatoryAssignments** (Shared consultorios)
- ✅ Actualizado pricing (por dentista, no por clínica)
- ✅ Actualizado todos los sprints con modelo correcto

PROF

Referencias:

- 00-README.md - Modelo de negocio
- 01-ARQUITECTURA-SISTEMA.md - Arquitectura completa
- 02-MODELO-DATOS.md - Database schema
- 03-ROLES-PERMISOS.md - RBAC
- 04-WHATSAPP-CHATBOT.md - WhatsApp integration

- 05-FLUJOS-PRINCIPALES.md - User journeys