Master Plan Operativo: Dental SaaS V1 (Zero to Hero)

Obiettivo: Rilascio in Produzione su Google Cloud Platform in 9 Settimane.
Stack: VS Code + Gemini Code Assist | Turborepo | Next.js | NestJS | GCP.

🛠️ FASE 0: Setup Ambiente & "Google-Native" Workflow (Giorni 1-3)

Obiettivo: Configurare la macchina di sviluppo per operare come un'estensione del Cloud.

Passo 1: Configurazione VS Code "Agentica"

1. **Installa VS Code.**
2. **Estensioni Obbligatorie:**
   - Google Cloud Code (ID: googlecloudtools.cloudcode) -> **CRITICO**: Fornisce Gemini, Cloud Run emulator, e log viewer.
   - ESLint + Prettier -> Per la pulizia automatica.
   - Prisma -> Per il database.
   - Docker -> Per gestire i container locali.
   - Dev Containers (Opzionale ma consigliato per isolare l'ambiente).
3. **Configurazione Gemini:**
   - Clicca sull'icona Cloud Code nella barra di stato.
   - Loggati con l'account GCP.
   - Seleziona il progetto dental-saas-production.
   - **Pro Tip:** Assegna una shortcut (es. Cmd+Shift+G) a "Cloud Code: Focus on Gemini Code Assist" per invocare l'AI istantaneamente.

Passo 2: Inizializzazione Monorepo (Turborepo)

Non creare repo sparsi. Centralizza tutto.

1. Terminale: npx create-turbo@latest dental-saas.
2. Scegli pnpm come package manager (più veloce di npm).
3. Struttura target da creare:

```
/apps
  /web (Next.js 14 App Router)
  /api (NestJS)
/packages
  /db (Prisma Schema, Seed scripts)
  /dto (Tipi condivisi Zod/TypeScript)
  /config (ESLint presets, Tailwind config)
```

/infra (File Terraform)

4. **Pro Tip:** Nel package.json root, aggiungi script scorciatoia:
    - "dev": "turbo run dev"
    - "db:up": "docker-compose up -d"

Passo 3: Infrastruttura Locale (Docker)

Crea docker-compose.yml nella root.

```
version: '3.8'
services:
  postgres:
    image: postgres:15-alpine
    ports: ["5432:5432"]
    environment:
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: password
      POSTGRES_DB: dental_saas
  minio: # Simula Google Cloud Storage
    image: minio/minio
    ports: ["9000:9000", "9001:9001"]
    command: server /data --console-address ":9001"
```

- Avvia: docker-compose up -d.

Passo 4: Infrastruttura Cloud (Terraform)

1. Spostati in /infra.
2. Crea il file main.tf (usando il codice fornito nella documentazione "Architettura Master").
3. Esegui: terraform init -> terraform plan -> terraform apply.
4. **Output:** Segnati l'IP del Cloud SQL, il nome del Bucket e l'URL del Registry.

🔐 FASE 1: Backend Core & Sicurezza Dati (Settimane 1-2)

Obiettivo: Creare un backend "blindato" dove i dati dei tenant non si mischiano mai.

Passo 5: Database Schema & Prisma

1. In packages/db, edita schema.prisma.
2. Copia lo schema fornito (Tenant, User, Patient, etc.).
3. Esegui: npx prisma db push (sincronizza il DB locale Docker).
4. **Genialata RLS (Row Level Security):**
    - Crea un file client.ts in packages/db.
    - Usa $extends di Prisma per intercettare ogni query.
    - Logica: *Se non c'è tenantId nel contesto, blocca la query.*

Passo 6: NestJS Setup & Auth

1. In apps/api: nest new . (se vuoto).
2. Installa: @nestjs/passport, passport-jwt, @nestjs/config.
3. **Auth Module:**
    - Implementa Login (email/password).
    - Il JWT deve contenere: { sub: userId, tenantId: string }.
4. **Global Guard:**
    - Crea TenantGuard. Legge il JWT dall'header.
    - Salva il tenantId in ClsService (o AsyncLocalStorage) per renderlo accessibile a Prisma.

Passo 7: API Base (CRUD Pazienti)

1. Genera risorsa: nest g resource patients.
2. Implementa Controller e Service.
3. **Pro Tip:** Usa i DTO condivisi da packages/dto.
    - Se definisci CreatePatientDto con Zod nel pacchetto condiviso, lo usi sia nel Backend (validazione) che nel Frontend (form type-safe).

💻 FASE 2: Frontend & Clinica (Settimane 3-4)

Obiettivo: Un'interfaccia veloce che il medico voglia usare.

Passo 8: Next.js Shell & Shadcn

1. In apps/web: npx create-next-app@latest ..
2. Installa Shadcn: npx shadcn-ui@latest init.
3. Componenti base: button, input, table, dialog, sheet, form, toast.
4. **Layout:** Crea app/(dashboard)/layout.tsx con la Sidebar fissa e l'Header utente.

Passo 9: State Management (React Query)

1. Installa @tanstack/react-query.
2. Avvolgi l'app in un QueryClientProvider.
3. Crea un custom hook usePatients che chiama la tua API NestJS.
4. **Pro Tip:** Attiva staleTime: 5 minuti. Evita che il browser "bombardi" il server ogni volta che il medico cambia tab.

Passo 10: Modulo Pazienti

1. **Lista:** Usa TanStack Table (integrata in Shadcn). Filtri server-side per nome e CF.
2. **Dettaglio:** Route dinamica /patients/[id].
3. **Tabs:** "Anagrafica", "Piano di Cura", "Files".

Passo 11: Odontogramma V1

1. Non reinventare la ruota. Cerca su GitHub un SVG "Dental Chart" con i path separati per dente.
2. Crea componente <DentalArch />.
3. Logica:
    - Stato locale: const [teethStatus, setTeethStatus] = useState({}).
    - Click su dente -> Apre modale -> Seleziona "Carie" -> Aggiorna stato.
    - Salva su DB come JSONB.

🩻 FASE 3: Imaging & Storage (Settimane 5-6)

Obiettivo: Caricamento TAC (200MB+) senza passare dal server API.

Passo 12: Backend Signing Service

1. In apps/api, crea StorageModule.
2. Usa @google-cloud/storage.
3. Endpoint: POST /storage/sign-upload.
    - Input: { filename: "tac.dcm", contentType: "application/dicom" }.
    - Output: { uploadUrl: "https://storage.googleapis.com/..." }.

Passo 13: Frontend Uploader

1. Crea componente Dropzone.
2. Logica al drop:
    1. Chiedi URL firmato all'API.
    2. Usa axios.put(uploadUrl, file) direttamente verso Google.
    3. Notifica il backend a upload finito.

Passo 14: Healthcare API & Viewer

1. **Google Cloud:** Attiva Healthcare API -> Crea Dataset -> Crea DICOM Store.
2. **Frontend:** Installa cornerstone-core, cornerstone-tools, cornerstone-wado-image-loader.
3. Configura il loader per puntare all'endpoint WADO delle Healthcare API (tramite proxy backend o token pubblico temporaneo).
4. **Risultato:** Streaming progressivo delle immagini.

⚖️ FASE 4: Compliance & Admin (Settimane 7-8)

Obiettivo: Rendere il software legale (MDR/GDPR).

Passo 15: Audit Logging

1. Crea un Interceptor globale in NestJS.
2. Intercetta ogni metodo POST, PUT, DELETE.
3. Scrivi asincronamente su tabella AuditLog: "Chi, Cosa, Quando, Vecchio Valore, Nuovo Valore".

Passo 16: Modulo Laboratorio (MDR)

1. Nuova tabella MaterialBatch (Lotti).
2. Nuova tabella LabOrder.
3. **Logica di Blocco:** Nel backend, impedisci la transizione di stato a "COMPLETED" se LabOrder.materials è vuoto.

Passo 17: Consensi Informati

1. Carica i template HTML/Markdown (dal tuo Drive) su un bucket GCS templates.
2. Backend:
    - Scarica template.
    - Replace {{PAZIENTE}} con dati reali.
    - Usa puppeteer o pdf-lib per generare PDF.
    - Salva PDF su Storage e restituisci link.

🚀 FASE 5: Deploy & Rilascio (Settimana 9)

Obiettivo: Go Live.

Passo 18: Containerizzazione

1. Crea Dockerfile nella root (multistage build per ridurre dimensioni).
2. **Pro Tip:** Usa node:alpine e il sistema output: 'standalone' di Next.js per avere immagini di 100MB invece di 1GB.

Passo 19: CI/CD (Cloud Build)

1. Crea cloudbuild.yaml.
2. Configura trigger su push al branch main.
3. Passaggi:
    - npm install
    - npm test (Bloccante!)
    - docker build -> Push su Artifact Registry.
    - gcloud run deploy (Frontend e Backend).

Passo 20: Configurazione Finale

1. **Domain Mapping:** Mappa app.tuodominio.it su Cloud Run.
2. **Secret Manager:** Carica DATABASE_URL, JWT_SECRET su GCP Secret Manager.
3. **Smoke Test:** Esegui un ciclo completo in produzione.

💡 Consigli Pratici ("Pro Tips")

1. **Non ottimizzare prematuramente:** Non ti serve Kubernetes. Non ti serve Microservices (ancora). Un monolite modulare su Cloud Run ti porta fino a 10.000 utenti senza problemi.
2. **Gestione Errori:** Non esporre mai stack trace al frontend. Crea un filtro eccezioni globale che restituisce messaggi "sanitizzati".
3. **Timezone:** Salva sempre in UTC nel DB. Converti in "Europe/Rome" solo nel componente React al momento del render.
4. **Backup:** Verifica che Cloud SQL abbia i backup automatici attivi e la protezione da cancellazione. È la tua assicurazione sulla vita.