Piano Operativo di Sviluppo: Dental SaaS V1 (9 Settimane)

Questo documento traccia il percorso dalla riga di comando vuota al rilascio in produzione su Google Cloud Platform. È ottimizzato per un team agile che utilizza lo stack **Next.js + NestJS + GCP**.

🗓️ Overview Timeline

Fase	Durata	Focus Principale	Output
0. Setup	Settimana 1	Repo, Docker, Terraform	Ambiente di Sviluppo & Prod pronti
1. Core	Settimana 2	DB, Auth, Multi-tenancy	Backend sicuro e isolato
2. UI/UX	Settimane 3-4	Dashboard, Pazienti	Interfaccia Clinica navigabile
3. Imaging	Settimane 5-6	DICOM, STL, Storage	Viewer funzionanti con file reali
4. Legal	Settimane 7-8	Audit, MDR, Consensi	Compliance normativa completa
5. Deploy	Settimana 9	CI/CD, SSL, Launch	Rilascio V1.0

🚀 Fase 0: Le Fondamenta (Settimana 1)

Obiettivo: Avere un ambiente locale che replica la produzione e l'infrastruttura cloud pronta.

0.1 Setup Monorepo (Giorno 1-2)

Usiamo **Turborepo** per gestire Frontend e Backend nello stesso repository.

1. **Init:** npx create-turbo@latest dental-saas.
2. **Pulizia:** Rimuovi i pacchetti di esempio.
3. **Struttura Cartelle:**
    - apps/web: Next.js 14 (App Router).
    - apps/api: NestJS (nuovo progetto: nest new api).
    - packages/db: Libreria condivisa per Prisma Schema.
    - packages/dto: Tipi TypeScript condivisi (Zod schemas).
4. **Tooling:** Installa husky e lint-staged per forzare il controllo qualità al commit.

💡 **Pro Tip:** Configura subito prettier-plugin-tailwindcss nel frontend. Risparmia ore di discussioni sullo stile del codice.

0.2 Ambiente Locale Docker (Giorno 3)

Non installare Postgres sul tuo Mac/PC. Usa Docker.

1. Crea docker-compose.yml nella root.
2. **Servizi:**
    - postgres:15-alpine (Porta 5432).
    - minio/minio (Porta 9000, simula Google Cloud Storage).
3. **Seed Script:** Scrivi uno script (packages/db/seed.ts) che popola il DB con:
    - 1 Tenant "Demo Studio".
    - 1 Admin, 1 Dottore.
    - 5 Pazienti fake.

0.3 Infrastruttura Cloud (Giorno 4-5)

1. Installa Terraform e GCloud CLI.
2. Crea un progetto su Google Cloud Console (es. dental-saas-prod).
3. Applica il file main.tf (già fornito nella documentazione).
    - Questo creerà: Cloud SQL, Bucket Storage, Artifact Registry.
4. **Verifica:** Controlla che il bucket abbia CORS abilitato per localhost:3000 (per i test) e il dominio di prod.

🔐 Fase 1: Backend Core & Sicurezza (Settimana 2)

Obiettivo: Un sistema sicuro dove i dati sono isolati per Tenant.

1.1 Prisma & RLS (Giorno 1-2)

1. Definisci lo schema in packages/db/schema.prisma.
2. **Estensione Client:** Non usare il client Prisma base. Crea un wrapper che inietta tenantId.

```
// Esempio concettuale
return prisma.$extends({
  query: {
    $allModels: {
      async $allOperations({ args, query }) {
        if (!currentTenantId) throw new Error("Tenant context missing");
        args.where = { ...args.where, tenantId: currentTenantId };
        return query(args);
      },
    },
  },
});
```

1.2 Autenticazione NestJS (Giorno 3-4)

1. Installa @nestjs/passport, passport-jwt.
2. Implementa AuthModule:

    - Endpoint /auth/login: Valida email/password (Argon2), restituisce JWT.
    - **Payload JWT:** { sub: userId, email, role, tenantId }.
3. Crea JwtStrategy e GqlAuthGuard (se usi GraphQL) o AuthGuard standard.

1.3 Tenant Context Middleware (Giorno 5)

1. Crea un Middleware NestJS globale.
2. Estrae il token Bearer, lo decodifica.
3. Imposta il tenantId in un **AsyncLocalStorage** (o nel request scope) in modo che il Service Prisma possa leggerlo senza passarlo come parametro in ogni funzione.

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

Obiettivo: Interfaccia usabile per la segreteria e i medici.

2.1 UI Shell & Shadcn (Settimana 3, Giorno 1-2)

1. In apps/web, installa Shadcn UI: npx shadcn-ui@latest init.
2. Installa componenti base: button, input, table, dialog, sheet (sidebar), card.
3. Implementa il Layout Principale (app/layout.tsx) con Sidebar responsive.

2.2 Gestione Stato (Giorno 3)

1. Installa @tanstack/react-query.
2. Crea un QueryProvider.
3. **Pro Tip:** Imposta staleTime: 5 minuti per le liste pazienti. Non fare refetch ogni volta che l'utente cambia tab.

2.3 Modulo Pazienti (Settimana 3, Giorno 4-5)

1. **Lista:** Tabella filtrabile (Nome, CF, Ultima Visita).
2. **Dettaglio:** Pagina dinamica /patients/[id].
3. **Tabs:** Anagrafica, Piani di Cura, File.

2.4 Odontogramma V1 (Settimana 4)

1. Cerca un SVG di un'arcata dentale (ISO 3950) open source.
2. Convertilo in componente React (<DentalArch />).
3. Rendilo interattivo:
    - Click su dente -> Apre Popover -> Seleziona Stato (Sano, Carie, Impianto).
    - Salva stato in un array JSON nel DB.

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

Obiettivo: Gestione dei file pesanti (TAC, STL) senza passare dal backend.

3.1 Backend Storage Service (Settimana 5, Giorno 1-2)

1. Implementa gcp_storage.service.ts in NestJS.

2. Endpoint: POST /files/sign-upload
    - Input: fileName, contentType.
    - Output: uploadUrl (V4 Signed URL), fileKey.

3.2 Frontend Uploader (Giorno 3-4)

1. Crea componente Dropzone.
2. Al drop del file:
    - Chiedi URL firmato al backend.
    - Usa axios.put(uploadUrl, file, { headers: { 'Content-Type': file.type } }).
    - Mostra progress bar reale.

3.3 Healthcare API Integration (Settimana 6)

1. Attiva **Cloud Healthcare API** su GCP. Crea un DICOM Store.
2. **Cloud Function (Trigger):**
    - Quando un file .dcm arriva su GCS -> Triggera funzione.
    - La funzione invia il file al DICOM Store tramite API Google (projects/.../dicomWeb/.../storeInstances).
    - Aggiorna lo stato nel DB a READY.

3.4 Viewer Integration (Giorno 4-5)

1. **Cornerstone.js (RX):**
    - Installa @cornerstonejs/core, @cornerstonejs/tools.
    - Configura WADO Image Loader per puntare al DICOM Store di Google.
2. **React-Three-Fiber (STL):**
    - Usa useLoader(STLLoader, url).
    - Aggiungi luci e OrbitControls.

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

Obiettivo: Rendere il software a norma di legge.

4.1 Audit Logging (Settimana 7)

1. Crea un Interceptor globale in NestJS.
2. Per ogni mutazione (POST/PUT/DELETE):
    - Salva in AuditLog: userId, action, resourceId, oldValue, newValue.
3. Questo è il salvavita in caso di ispezione GDPR.

4.2 Modulo Laboratorio & MDR (Settimana 8, Giorno 1-3)

1. **Tabella Lotti:** CRUD per i materiali del laboratorio (Zirconia, Ceramica) con lotto e scadenza.
2. **Blocco Logico:** Nel service LabOrder, impedisci status = CLOSED se non ci sono materiali collegati.

4.3 Consensi Informati (Giorno 4-5)

1. Crea un bucket templates su GCS. Carica i file HTML/Markdown.
2. Backend:
    - Scarica template.
    - Sostituisce {{NOME_PAZIENTE}} con i dati reali.
    - Genera PDF con puppeteer o libreria PDF.
    - Restituisce URL per il download/firma.

🚀 Fase 5: Deploy & Rilascio (Settimana 9)

Obiettivo: Go Live.

5.1 Dockerizzazione

1. Crea Dockerfile ottimizzati per Next.js (standalone output) e NestJS.
2. Assicurati che siano leggeri (usa alpine node).

5.2 CI/CD (Cloud Build)

1. Configura cloudbuild.yaml:
    - Step 1: Installa dipendenze & Test.
    - Step 2: Build Docker Images.
    - Step 3: Push su Artifact Registry.
    - Step 4: Deploy su Cloud Run (gcloud run deploy).

5.3 Configurazione Finale

1. **Domini:** Punta i DNS (es. app.tuostudio.it) su Cloud Run Mapping.
2. **Environment:** Setta le variabili di produzione in Secret Manager (DB password, API Keys).
3. **Smoke Test:** Esegui un ciclo completo (Login -> Carica Paziente -> Carica TAC -> Visualizza) in produzione.

💡 Pro Tips per l'Efficienza

- **Evita l'Over-Engineering:** Non ti serve Kubernetes. Cloud Run è sufficiente e scala a zero.
- **Gestione Errori:** Non esporre mai stack trace al frontend. Usa un HttpExceptionFilter che logga l'errore su Cloud Logging e restituisce un messaggio generico all'utente.
- **Timezones:** I dentisti lavorano con orari locali. Salva sempre UTC nel DB, ma usa date-fns-tz nel frontend per convertire in "Europe/Rome".
- **Backup:** Verifica che i backup automatici di Cloud SQL siano attivi e testa un restore su un'istanza di prova *prima* di andare live.