# Technical Architecture

*(Detailed architecture specifications documented separately in docs/architecture.md)*

## Tech Stack

**Frontend:** - Next.js 14+ (App Router) - React 18+ - TypeScript - TailwindCSS for styling - Recharts for data visualization - React Hook Form for form management - Zod for validation

**Backend:** - Supabase (PostgreSQL + PostgREST API) - Supabase Auth - Supabase Realtime (Phase 2) - pgvector extension (Phase 2)

**Infrastructure:** - Vercel (Next.js hosting) - Supabase Cloud (database, auth, storage)

**Development Tools:** - Git + GitHub - Supabase CLI - VS Code - Postman / REST Client

## Database Schema

*(Full schema documented in docs/database-schema-overview.md)*

**Core Tables:** - municipalities - departments - fiscal\_years - users

**Planning Tables:** - strategic\_plans - strategic\_goals - initiatives - initiative\_budgets - initiative\_kpis - quarterly\_milestones

**Supporting Tables:** - initiative\_dependencies - initiative\_collaborators - comments - audit\_logs - document\_embeddings (Phase 2)

**Data Model:** - Hybrid approach: Relational + JSONB - Row-level security (RLS) enforced - Comprehensive audit trail

## Application Architecture

**Folder Structure:**

/app  
 /(auth)  
 /login  
 /signup  
 /(dashboard)  
 /plans  
 /[planId]  
 /goals  
 /initiatives  
 /initiatives  
 /budgets  
 /kpis  
 /api  
/components  
 /ui (shadcn/ui components)  
 /plans  
 /initiatives  
 /dashboards  
/lib  
 /supabase (client, types)  
 /utils  
/types  
 /supabase.ts (generated types)

**State Management:** - Server Components for data fetching - React Context for global state (user, preferences) - React Query for client-side caching (if needed)

**Forms:** - React Hook Form + Zod validation - JSONB fields rendered as dynamic forms