# Arquitetura Técnica e Plano de Implementação

**Projeto:** Gestão de Saúde Ocupacional (SaaS) **Stack:** Next.js + Supabase + TypeScript

## 1. Stack Tecnológica Definida

| Camada | Tecnologia | Justificativa |
|---|---|---|
| **Framework** | **Next.js 15 (App Router)** | Padrão de mercado, Server Actions facilitam o backend. |
| **Linguagem** | **TypeScript** | Segurança de tipos crítica para dados médicos/financeiros. |
| **Estilização** | **Tailwind CSS** | Desenvolvimento rápido e responsivo. |
| **Componentes** | **shadcn/ui** | Componentes acessíveis e bonitos (Radix UI) copiados para o código. |
| **Banco de Dados** | **Supabase (PostgreSQL)** | Relacional robusto, Auth integrado, Realtime. |
| **Validação** | **Zod** | Validação de esquemas (Forms e API) rigorosa. |
| **Formulários** | **React Hook Form** | Gerenciamento de estado de formulários complexos. |
| **PDF** | **@react-pdf/renderer** | Geração de Faturas e Relatórios direto no React. |
| **Datas** | **date-fns** | Manipulação segura de datas (cálculo de validade de ASO). |

## 2. Estrutura de Diretórios (Padrão Profissional)

Não usaremos a estrutura padrão simples. Organizaremos por **Funcionalidades (Features)** para escalar melhor.

```
/src
  /app                  # Rotas (App Router)
    /(auth)             # Rotas de Login/Recuperação (Layout isolado)
    /(dashboard)        # Área logada (Layout com Sidebar)
      /admin            # Módulo Administrativo
      /comercial        # Módulo Comercial (Clientes, Preços)
      /operacional      # Módulo Operacional (Atendimentos)
      /financeiro       # Módulo Financeiro (Faturas)
    /api                # Webhooks (se necessário)
    globals.css         # Tailwind imports
    layout.tsx          # Root Layout

  /components
```

```
    /ui                 # Componentes genéricos (Botão, Input - shadcn)
    /shared             # Componentes reutilizáveis (Header, Sidebar)
    /features           # Componentes específicos de negócio
      /aso              # Ex: AsoForm.tsx, ProtocolList.tsx
      /finance          # Ex: InvoiceCard.tsx

  /lib                  # Configurações e Utilitários
    supabase.ts         # Cliente Supabase
    utils.ts            # Helpers (formatação de moeda, datas)
    validations         # Schemas do Zod (ex: employeeSchema.ts)

  /types                # Tipagem Global
    database.types.ts   # Gerado automaticamente pelo Supabase CLI
    index.ts            # Tipos auxiliares da aplicação

  /services             # Camada de Dados (Server Actions)
    authService.ts
    clientService.ts
    asoService.ts       # Toda lógica de banco de dados fica aqui
```

## 3. Padrões de Código (Guidelines)

### 3.1. Server Actions vs API Routes

- **Regra:** Priorizar **Server Actions** para todas as mutações (criar, editar, deletar).
- **Motivo:** Menos boilerplate, tipagem direta e segurança integrada com o Next.js.
- **Exemplo:** Em vez de fazer um `POST /api/cliente`, criaremos uma função `createClientAction(formData)` em `/services`.

### 3.2. Validação de Dados (Zod)

- NENHUM dado entra no banco sem passar pelo Zod.
- Os Schemas do Zod devem ficar em `/lib/validations` para serem usados tanto no Frontend (React Hook Form) quanto no Backend (Server Action).

### 3.3. Estilização

- Usar classes utilitárias do Tailwind.
- Para cores condicionais (ex: Risco 3 = Vermelho), criar variantes usando a função `cn()` do shadcn/ui.

## 4. Roadmap de Desenvolvimento (Fases)

Para você não se perder, vamos "fatiar o elefante".

### Fase 0: Setup (Dia 1)

1. Criar projeto Next.js.
2. Configurar Supabase (criar projeto, rodar script SQL gerado na documentação anterior).
3. Instalar shadcn/ui e componentes base (Button, Input, Table, Card, Form).

4. Configurar autenticação (Login screen).

**Fase 1: O "Admin" (Dias 2-3)**

*Foco: Módulo Administrativo*

1. Criar tela de Configuração da Consultoria (Upload Logo, Dados).
2. Criar gestão de Usuários (Convidar Operador).
3. **Entrega:** Sistema onde você loga e vê os dados da sua empresa.

**Fase 2: O "Comercial" (Dias 4-6)**

*Foco: Cadastros Base*

1. CRUD de Clientes (Empresas).
2. CRUD de Procedimentos (Catálogo).
3. Tela de **Tabela de Preços** (A mais complexa desta fase).
4. **Entrega:** Cadastro completo da empresa cliente e suas regras de preço.

**Fase 3: O "Operacional" (Dias 7-10)**

*Foco: Fluxo de Trabalho*

1. Cadastro de Funcionários (Vínculo com Cargo).
2. **O Grande Desafio:** Tela de Novo Atendimento.
   - Lógica de puxar protocolo automático.
   - Lógica de snapshot de preço.
3. **Entrega:** Conseguir lançar um exame e vê-lo no banco com o preço correto.

**Fase 4: O "Financeiro" (Dias 11-13)**

*Foco: Relatórios*

1. Tela de Fechamento Mensal (Agrupamento).
2. Geração de PDF da Fatura.
3. Baixa de pagamentos.
4. **Entrega:** O ciclo completo do software.

## 5. Próximo Passo Imediato (Setup)

Abra seu terminal no Windows e rode:

```
npx create-next-app@latest sso-consultoria --typescript --tailwind --eslint
```

- **Would you like to use `src/` directory?** Yes
- **Would you like to use App Router?** Yes

- **Would you like to customize the default import alias?** No (@/*)

Após criar, instale o core do UI:

```
npx shadcn-ui@latest init
```