ESTRUTURA COMPLETA DO PROJETO

Sistema de Agendamento - Fotografia Imobiliária

Versão: 1.0
Framework: Next.js 14 + TypeScript + Tailwind CSS
Banco: Prisma + SQLite (dev) / Supabase (prod)

📁 ESTRUTURA DE PASTAS COMPLETA

```
agendamento-fotos/
│
├── .env                        # Variáveis de ambiente
├── .env.example                # Exemplo de variáveis
├── .gitignore                  # Arquivos ignorados pelo Git
├── next.config.js              # Configuração Next.js
├── package.json                # Dependências
├── tsconfig.json               # Configuração TypeScript
├── tailwind.config.ts          # Configuração Tailwind
├── postcss.config.js           # Configuração PostCSS
├── README.md                   # Documentação do projeto
│
├── prisma/
│   ├── schema.prisma           # Schema do banco de dados
│   ├── seed.ts                 # Dados iniciais (fotógrafos, bairros)
│   └── migrations/             # Histórico de migrations
│       └── [timestamp]_init/
│           └── migration.sql
│
├── public/
│   ├── logo.svg                # Logo da empresa
│   └── favicon.ico             # Favicon
│
├── app/
│   ├── layout.tsx              # Layout global
│   ├── page.tsx                # Landing page (/)
│   ├── globals.css             # Estilos globais
│   │
│   ├── agendar/                # FLUXO DE AGENDAMENTO
```

```
│   │   ├── page.tsx              # Página principal de agendamento
│   │   └── [token]/              # Agendamento com link único
│   │       └── page.tsx
│   │
│   ├── meus-agendamentos/        # DASHBOARD DO CLIENTE
│   │   └── page.tsx              # Lista de agendamentos do cliente
│   │
│   ├── admin/                    # PAINEL ADMINISTRATIVO
│   │   ├── layout.tsx            # Layout do admin
│   │   ├── page.tsx              # Dashboard admin (métricas)
│   │   │
│   │   ├── fotografos/           # Gestão de Fotógrafos
│   │   │   ├── page.tsx          # Lista de fotógrafos
│   │   │   ├── novo/
│   │   │   │   └── page.tsx      # Adicionar fotógrafo
│   │   │   └── [id]/
│   │   │       ├── page.tsx      # Editar fotógrafo
│   │   │       └── areas/
│   │   │           └── page.tsx  # Gerenciar áreas do fotógrafo
│   │   │
│   │   ├── agendamentos/         # Gestão de Agendamentos
│   │   │   ├── page.tsx          # Lista todos agendamentos
│   │   │   └── [id]/
│   │   │       └── page.tsx      # Detalhes do agendamento
│   │   │
│   │   └── configuracoes/        # Configurações Gerais
│   │       ├── page.tsx          # Configurações
│   │       ├── bairros/
│   │       │   └── page.tsx      # Gerenciar bairros atendidos
│   │       └── horarios/
│   │           └── page.tsx      # Configurar horários de funcionamento
│   │
│   └── api/                      # APIS BACKEND
│       │
│       ├── photographers/        # FOTÓGRAFOS
│       │   ├── route.ts          # GET (listar), POST (criar)
│       │   └── [id]/
│       │       ├── route.ts      # GET, PUT, DELETE
│       │       └── coverage-areas/
│       │           └── route.ts  # GET, POST, DELETE áreas
│       │
│       ├── coverage-areas/       # ÁREAS DE COBERTURA
│       │   ├── route.ts          # GET (todas), POST (criar)
```

```
│   │   │   └── [id]/
│   │   │       └── route.ts          # PUT, DELETE
│   │   │
│   │   ├── address/                  # VALIDAÇÃO DE ENDEREÇO
│   │   │   ├── validate/
│   │   │   │   └── route.ts          # POST - validar endereço
│   │   │   └── search/
│   │   │       └── route.ts          # GET - autocomplete (Google Places)
│   │   │
│   │   ├── availability/             # DISPONIBILIDADE
│   │   │   └── route.ts              # GET - horários disponíveis
│   │   │
│   │   ├── bookings/                 # AGENDAMENTOS
│   │   │   ├── route.ts              # GET (listar), POST (criar)
│   │   │   ├── [id]/
│   │   │   │   └── route.ts          # GET, PUT, DELETE
│   │   │   └── [token]/
│   │   │       └── cancel/
│   │   │           └── route.ts      # DELETE - cancelar via token
│   │   │
│   │   ├── webhooks/                 # WEBHOOKS
│   │   │   ├── tadabase/
│   │   │   │   └── route.ts          # POST - receber do Tadabase
│   │   │   └── send-to-tadabase/
│   │   │       └── route.ts          # POST - enviar para Tadabase
│   │   │
│   │   └── emails/                   # EMAILS
│   │       ├── confirmation/
│   │       │   └── route.ts          # POST - enviar confirmação
│   │       └── cancellation/
│   │           └── route.ts          # POST - enviar cancelamento
│   │
├── components/                       # COMPONENTES REACT
│   │
│   ├── booking/                      # Componentes de Agendamento
│   │   ├── StepIndicator.tsx         # Indicador de progresso (1/6)
│   │   ├── AddressStep.tsx           # Passo 1: Endereço
│   │   ├── ServicesStep.tsx          # Passo 2: Serviços
│   │   ├── DateStep.tsx              # Passo 3: Data
│   │   ├── TimeStep.tsx              # Passo 4: Horário
│   │   ├── ClientDataStep.tsx        # Passo 5: Dados do cliente
│   │   ├── ConfirmationStep.tsx      # Passo 6: Confirmação
│   │   └── SuccessStep.tsx           # Passo 7: Sucesso
```

```
│   │
│   ├── dashboard/                # Dashboard do Cliente
│   │   ├── StatsCards.tsx        # Cards de estatísticas
│   │   ├── UpcomingBookings.tsx  # Próximos agendamentos
│   │   ├── BookingCard.tsx       # Card de agendamento
│   │   ├── HistoryList.tsx       # Lista de histórico
│   │   └── CancelModal.tsx       # Modal de cancelamento
│   │
│   ├── admin/                    # Componentes Admin
│   │   ├── Sidebar.tsx           # Menu lateral
│   │   ├── Header.tsx            # Cabeçalho
│   │   ├── PhotographerForm.tsx  # Formulário de fotógrafo
│   │   ├── PhotographerList.tsx  # Lista de fotógrafos
│   │   ├── CoverageAreaForm.tsx  # Formulário de área
│   │   ├── BookingsList.tsx      # Lista de agendamentos
│   │   └── MetricsCard.tsx       # Card de métrica
│   │
│   └── ui/                       # Componentes Genéricos UI
│       ├── Button.tsx            # Botão
│       ├── Input.tsx             # Input
│       ├── Select.tsx            # Select
│       ├── Calendar.tsx          # Calendário
│       ├── Modal.tsx             # Modal
│       ├── Card.tsx              # Card
│       ├── Badge.tsx             # Badge
│       ├── Alert.tsx             # Alert/Notificação
│       ├── Loading.tsx           # Loading spinner
│       └── ErrorBoundary.tsx     # Error boundary
│
├── lib/                          # BIBLIOTECAS E UTILITÁRIOS
│   ├── prisma.ts                 # Cliente Prisma singleton
│   ├── utils.ts                  # Funções utilitárias gerais
│   ├── validations.ts            # Schemas Zod de validação
│   ├── constants.ts              # Constantes (serviços, durações)
│   │
│   ├── services/                 # Lógica de Negócio
│   │   ├── photographerService.ts  # Lógica de fotógrafos
│   │   ├── bookingService.ts       # Lógica de agendamentos
│   │   ├── availabilityService.ts  # Cálculo de disponibilidade
│   │   ├── addressService.ts       # Validação de endereço
│   │   ├── emailService.ts         # Envio de emails
│   │   └── webhookService.ts       # Processamento de webhooks
│   │
```

```
│   └── integrations/              # Integrações Externas
│       ├── googleMaps.ts          # Google Maps API
│       ├── resend.ts              # Resend (email)
│       └── tadabase.ts            # Tadabase
│
├── types/                         # TIPOS TYPESCRIPT
│   ├── index.ts                   # Tipos principais
│   ├── photographer.ts            # Tipos de fotógrafo
│   ├── booking.ts                 # Tipos de agendamento
│   ├── coverageArea.ts            # Tipos de área
│   └── api.ts                     # Tipos de respostas API
│
├── hooks/                         # REACT HOOKS CUSTOMIZADOS
│   ├── useBooking.ts              # Hook para agendamento
│   ├── useAvailability.ts         # Hook para disponibilidade
│   ├── usePhotographers.ts        # Hook para fotógrafos
│   └── useDebounce.ts             # Hook de debounce
│
├── emails/                        # TEMPLATES DE EMAIL
│   ├── ConfirmationEmail.tsx      # Template: confirmação
│   ├── CancellationEmail.tsx      # Template: cancelamento
│   ├── ReminderEmail.tsx          # Template: lembrete
│   └── PhotographerNotification.tsx  # Template: notificação fotógrafo
│
└── __tests__/                     # TESTES (opcional para MVP)
    ├── api/
    ├── components/
    └── services/
```

DETALHAMENTO DE CADA ARQUIVO

ROOT (Raiz do Projeto)

.env

```
env
```

```
# Database
DATABASE_URL="file:./dev.db"                  # SQLite local
# DATABASE_URL="postgresql://..."             # Supabase (produção)

# Google Maps
NEXT_PUBLIC_GOOGLE_MAPS_API_KEY="AIza..."     # API Key pública

# Email
RESEND_API_KEY="re_..."                       # Resend API

# Tadabase
TADABASE_WEBHOOK_SECRET="secret_..."          # Secret para validação
TADABASE_API_URL="https://..."                # URL da API

# App
NEXT_PUBLIC_APP_URL="http://localhost:3000"    # URL do app
```

.env.example

```
env

DATABASE_URL="file:./dev.db"
NEXT_PUBLIC_GOOGLE_MAPS_API_KEY="sua_key_aqui"
RESEND_API_KEY="sua_key_aqui"
TADABASE_WEBHOOK_SECRET="seu_secret_aqui"
TADABASE_API_URL="https://api.tadabase.com"
NEXT_PUBLIC_APP_URL="http://localhost:3000"
```

.gitignore

```
# Dependencies
node_modules/
.pnp/

# Next.js
.next/
out/
build/
dist/

# Environment
.env
.env.local
.env*.local

# Database
*.db
*.db-journal

# Logs
npm-debug.log*
yarn-debug.log*

# OS
.DS_Store
Thumbs.db

# IDE
.vscode/
.idea/
*.swp
*.swo
```

README.md

```
markdown
```

```
# Sistema de Agendamento - Fotografia Imobiliária

## Requisitos
- Node.js 18+
- npm ou yarn

## Setup
1. Clone o repositório
2. Instale dependências: `npm install`
3. Configure .env
4. Execute migrations: `npx prisma migrate dev`
5. Popule banco: `npx prisma db seed`
6. Inicie servidor: `npm run dev`

## Comandos
- `npm run dev` - Servidor desenvolvimento
- `npx prisma studio` - Interface banco de dados
- `npm run build` - Build produção
- `npm start` - Servidor produção
```

PRISMA

prisma/schema.prisma

✅ Já criado (schema completo com Photographer, CoverageArea, Booking, WebhookLog)

prisma/seed.ts

✅ Já criado (4 fotógrafos + bairros + exclusões)

APP - PÁGINAS

app/layout.tsx

```typescript
```

```
// Layout global
// - Metadata (SEO)
// - Providers
// - Fontes
// - Analytics
```

app/page.tsx

```typescript
// Landing page
// - Hero section
// - Como funciona
// - Serviços oferecidos
// - CTA para agendar
```

app/agendar/page.tsx

```typescript
// Fluxo de agendamento (6 passos)
// - Estado global do formulário
// - Navegação entre passos
// - Validações
```

app/meus-agendamentos/page.tsx

```typescript
// Dashboard do cliente
// - Listar agendamentos futuros
// - Histórico
// - Cancelar agendamento
```

app/admin/layout.tsx

```typescript
```

```
// Layout admin
// - Sidebar
// - Header
// - Proteção de rota (auth futura)
```

app/admin/page.tsx

```typescript
// Dashboard admin
// - Métricas gerais
// - Agendamentos hoje
// - Fotógrafos ativos
// - Gráficos
```

app/admin/fotografos/page.tsx

```typescript
// Gestão de fotógrafos
// - Lista de fotógrafos
// - Ativar/desativar
// - Editar
// - Adicionar novo
```

app/admin/fotografos/novo/page.tsx

```typescript
// Adicionar fotógrafo
// - Formulário
// - Validação
// - Criar fotógrafo + áreas padrão
```

app/admin/fotografos/[id]/page.tsx

```typescript
```

```
// Editar fotógrafo
// - Dados básicos
// - Serviços oferecidos
// - Contato
```

app/admin/fotografos/[id]/areas/page.tsx

```typescript
// Gerenciar áreas do fotógrafo
// - Lista de bairros atendidos
// - Adicionar bairro
// - Remover bairro
// - Adicionar exclusão
```

app/admin/agendamentos/page.tsx

```typescript
// Lista todos agendamentos
// - Filtros (data, fotógrafo, status)
// - Paginação
// - Exportar
```

app/admin/configuracoes/bairros/page.tsx

```typescript
// Gerenciar bairros globais
// - Adicionar bairro master
// - Aplicar a todos fotógrafos
```

APP - APIs

app/api/photographers/route.ts

```typescript
```

```
// GET  - Listar todos fotógrafos (com filtros)
// POST - Criar novo fotógrafo
```

app/api/photographers/[id]/route.ts

```typescript
// GET    - Buscar fotógrafo por ID
// PUT    - Atualizar fotógrafo
// DELETE - Deletar fotógrafo (soft delete)
```

app/api/photographers/[id]/coverage-areas/route.ts

```typescript
// GET    - Listar áreas do fotógrafo
// POST   - Adicionar área
// DELETE - Remover área
```

app/api/coverage-areas/route.ts

```typescript
// GET  - Listar todas áreas (com filtros)
// POST - Criar área
```

app/api/address/validate/route.ts

```typescript
// POST - Validar endereço
// Input: { address: string }
// Output: { valid, neighborhood, lat, lng, inCoverage }
// Integração Google Maps Geocoding
```

app/api/address/search/route.ts

```typescript
```

```
// GET - Autocomplete de endereço
// Query: ?q=Rua XV
// Output: { suggestions: Address[] }
// Integração Google Places Autocomplete
```

app/api/availability/route.ts

```typescript
// GET - Consultar disponibilidade
// Query: ?date=2026-01-30&services=photo,video
// Output: { slots: TimeSlot[] }
// Lógica complexa de disponibilidade
```

app/api/bookings/route.ts

```typescript
// GET  - Listar agendamentos (filtros)
// POST - Criar agendamento
```

app/api/bookings/[id]/route.ts

```typescript
// GET    - Buscar agendamento
// PUT    - Atualizar agendamento
// DELETE - Cancelar agendamento
```

app/api/bookings/[token]/cancel/route.ts

```typescript
// DELETE - Cancelar via token único
// Validar prazo (24h)
// Calcular taxa
// Enviar emails
```

app/api/webhooks/tadabase/route.ts

```typescript
```

```
// POST - Receber webhook do Tadabase
// Validar assinatura HMAC
// Processar agendamento
// Verificar conflitos
// Salvar ou rejeitar
```

app/api/webhooks/send-to-tadabase/route.ts

```typescript
// POST - Enviar para Tadabase
// Retry automático
// Log de tentativas
```

app/api/emails/confirmation/route.ts

```typescript
// POST - Enviar email de confirmação
// Template React
// Integração Resend
```

COMPONENTS

components/booking/StepIndicator.tsx

```typescript
// Barra de progresso (1/6, 2/6, etc)
// Props: currentStep, totalSteps
```

components/booking/AddressStep.tsx

```typescript
// Passo 1: Endereço
// - Input com autocomplete
// - Validação em tempo real
// - Campo complemento
```

components/booking/ServicesStep.tsx

typescript

```
// Passo 2: Serviços
// - Checkbox múltiplo
// - Cálculo duração total
// - Mostrar fotógrafos disponíveis
```

components/booking/DateStep.tsx

typescript

```
// Passo 3: Data
// - Calendário 7 dias/semana
// - Separadores de mês
// - Domingos desabilitados
```

components/booking/TimeStep.tsx

typescript

```
// Passo 4: Horário
// - Grid de horários
// - Mostrar início e fim
// - Indicar disponibilidade
```

components/booking/ClientDataStep.tsx

typescript

```
// Passo 5: Dados
// - Nome, email, telefone
// - Observações
// - Aceite LGPD
```

components/booking/ConfirmationStep.tsx

typescript

```
// Passo 6: Confirmação
// - Resumo completo
// - Regras importantes
// - Botão confirmar
```

components/booking/SuccessStep.tsx

```typescript
// Passo 7: Sucesso
// - Protocolo
// - Instruções
// - Próximos passos
```

components/dashboard/CancelModal.tsx

```typescript
// Modal de cancelamento
// - Resumo agendamento
// - Relógio regressivo
// - Taxa aplicável
// - Confirmação
```

components/admin/PhotographerForm.tsx

```typescript
// Formulário de fotógrafo
// - Validação Zod
// - Submit handler
// - Loading state
```

components/ui/Button.tsx

```typescript
// Botão genérico
// Variants: primary, secondary, danger
// Sizes: sm, md, lg
```

components/ui/Calendar.tsx

```typescript
// Componente de calendário
// Reutilizável
// Suporta range selection
```

LIB - Serviços

lib/services/photographerService.ts

```typescript
// CRUD de fotógrafos
// Lógica de ativação/desativação
// Validações de negócio
```

lib/services/bookingService.ts

```typescript
// CRUD de agendamentos
// Gerar protocolo único
// Gerar token de cancelamento
// Validar regras de cancelamento
```

lib/services/availabilityService.ts

```typescript
// Calcular disponibilidade
// Filtrar por serviço
// Calcular distância
// Bloquear slots consecutivos
// Retornar horários disponíveis
```

lib/services/addressService.ts

```typescript
```

```
// Validar endereço (Google Maps)
// Autocomplete
// Geocodificação
// Validar cobertura (whitelist/blacklist)
```

lib/services/emailService.ts

```
typescript

// Enviar emails via Resend
// Templates React
// Queue de envio
// Retry automático
```

lib/services/webhookService.ts

```
typescript

// Processar webhooks Tadabase
// Validar assinatura HMAC
// Criar agendamento
// Verificar conflitos
```

LIB - Integrações

lib/integrations/googleMaps.ts

```
typescript

// Cliente Google Maps API
// Funções:
// - geocode(address)
// - autocomplete(query)
// - calculateDistance(origin, destination)
// Cache de resultados
```

lib/integrations/resend.ts

```
typescript
```

```
// Cliente Resend
// Funções:
// - sendEmail(to, subject, html)
// - sendBulk(emails[])
```

lib/integrations/tadabase.ts

```typescript
// Cliente Tadabase
// Funções:
// - createBooking(data)
// - updateBooking(id, data)
// - validateWebhook(signature, payload)
```

LIB - Utilitários

lib/validations.ts

```typescript
// Schemas Zod
// - PhotographerSchema
// - BookingSchema
// - AddressSchema
// - CoverageAreaSchema
```

lib/constants.ts

```typescript
// Constantes
// - SERVICES (foto, vídeo, drone)
// - DURATIONS (40min, 50min, etc)
// - OPERATING_HOURS
// - CANCELLATION_RULES
```

lib/utils.ts

typescript

```
// Funções utilitárias
// - formatPhone()
// - formatCurrency()
// - calculateEndTime()
// - generateProtocol()
// - cn() (className merge)
```

TYPES

types/index.ts

typescript

```
export * from './photographer';
export * from './booking';
export * from './coverageArea';
export * from './api';
```

types/photographer.ts

typescript

```
export type Photographer = {
  id: string;
  name: string;
  email: string;
  phone: string | null;
  services: ServiceType[];
  active: boolean;
  createdAt: Date;
  updatedAt: Date;
};

export type ServiceType = 'photo' | 'video' | 'drone';
```

types/booking.ts

typescript

```typescript
export type Booking = {
  id: string;
  protocol: string;
  photographerId: string;
  clientName: string;
  clientEmail: string;
  // ... outros campos
};

export type BookingStatus = 'confirmed' | 'cancelled' | 'completed';
```

EMAILS - Templates

emails/ConfirmationEmail.tsx

```typescript
// Template React de confirmação
// Props: booking, photographer
// Estilos inline (email-friendly)
```

emails/CancellationEmail.tsx

```typescript
// Template de cancelamento
// Props: booking, fee
```

📋 CHECKLIST DE ARQUIVOS

Configuração (10 arquivos)

- [ ] .env
- [ ] .env.example
- [ ] .gitignore
- [ ] README.md
- [ ] package.json

- [ ] tsconfig.json
- [ ] tailwind.config.ts
- [ ] next.config.js
- [ ] postcss.config.js
- [ ] prisma/schema.prisma

Prisma (2 arquivos)

- [ ] prisma/seed.ts
- [ ] lib/prisma.ts

Páginas App Router (14 arquivos)

- [ ] app/layout.tsx
- [ ] app/page.tsx
- [ ] app/globals.css
- [ ] app/agendar/page.tsx
- [ ] app/meus-agendamentos/page.tsx
- [ ] app/admin/layout.tsx
- [ ] app/admin/page.tsx
- [ ] app/admin/fotografos/page.tsx
- [ ] app/admin/fotografos/novo/page.tsx
- [ ] app/admin/fotografos/[id]/page.tsx
- [ ] app/admin/fotografos/[id]/areas/page.tsx
- [ ] app/admin/agendamentos/page.tsx
- [ ] app/admin/agendamentos/[id]/page.tsx
- [ ] app/admin/configuracoes/bairros/page.tsx

APIs (14 rotas)

- [ ] app/api/photographers/route.ts
- [ ] app/api/photographers/[id]/route.ts
- [ ] app/api/photographers/[id]/coverage-areas/route.ts
- [ ] app/api/coverage-areas/route.ts
- [ ] app/api/address/validate/route.ts
- [ ] app/api/address/search/route.ts
- [ ] app/api/availability/route.ts
- [ ] app/api/bookings/route.ts
- [ ] app/api/bookings/[id]/route.ts

- [ ] app/api/bookings/[token]/cancel/route.ts
- [ ] app/api/webhooks/tadabase/route.ts
- [ ] app/api/webhooks/send-to-tadabase/route.ts
- [ ] app/api/emails/confirmation/route.ts
- [ ] app/api/emails/cancellation/route.ts

Componentes Booking (7 arquivos)

- [ ] components/booking/StepIndicator.tsx
- [ ] components/booking/AddressStep.tsx
- [ ] components/booking/ServicesStep.tsx
- [ ] components/booking/DateStep.tsx
- [ ] components/booking/TimeStep.tsx
- [ ] components/booking/ClientDataStep.tsx
- [ ] components/booking/ConfirmationStep.tsx
- [ ] components/booking/SuccessStep.tsx

Componentes Dashboard (5 arquivos)

- [ ] components/dashboard/StatsCards.tsx
- [ ] components/dashboard/UpcomingBookings.tsx
- [ ] components/dashboard/BookingCard.tsx
- [ ] components/dashboard/HistoryList.tsx
- [ ] components/dashboard/CancelModal.tsx

Componentes Admin (7 arquivos)

- [ ] components/admin/Sidebar.tsx
- [ ] components/admin/Header.tsx
- [ ] components/admin/PhotographerForm.tsx
- [ ] components/admin/PhotographerList.tsx
- [ ] components/admin/CoverageAreaForm.tsx
- [ ] components/admin/BookingsList.tsx
- [ ] components/admin/MetricsCard.tsx

Componentes UI (10 arquivos)

- [ ] components/ui/Button.tsx
- [ ] components/ui/Input.tsx
- [ ] components/ui/Select.tsx

- [ ] components/ui/Calendar.tsx
- [ ] components/ui/Modal.tsx
- [ ] components/ui/Card.tsx
- [ ] components/ui/Badge.tsx
- [ ] components/ui/Alert.tsx
- [ ] components/ui/Loading.tsx
- [ ] components/ui/ErrorBoundary.tsx

Serviços (6 arquivos)

- [ ] lib/services/photographerService.ts
- [ ] lib/services/bookingService.ts
- [ ] lib/services/availabilityService.ts
- [ ] lib/services/addressService.ts
- [ ] lib/services/emailService.ts
- [ ] lib/services/webhookService.ts

Integrações (3 arquivos)

- [ ] lib/integrations/googleMaps.ts
- [ ] lib/integrations/resend.ts
- [ ] lib/integrations/tadabase.ts

Utilitários (3 arquivos)

- [ ] lib/utils.ts
- [ ] lib/validations.ts
- [ ] lib/constants.ts

Types (5 arquivos)

- [ ] types/index.ts
- [ ] types/photographer.ts
- [ ] types/booking.ts
- [ ] `types/coverageArea.ts