**# Esquema de Componentes para Sistema de Busca de Locais de Adoção e Veterinários**

## Visão Geral do Sistema

O sistema consiste em um aplicativo móvel (React Native), um servidor backend (PHP) e um banco de dados (PostgreSQL) para fornecer funcionalidades de busca de locais de adoção de animais e clínicas veterinárias com geolocalização.

## 1. Componentes do Frontend (React Native)

### 1.1 Módulos Principais

- \*\*AuthModule\*\*: Autenticação de usuários

- \*\*SearchModule\*\*: Busca de locais

- \*\*MapModule\*\*: Visualização em mapa

- \*\*ProfileModule\*\*: Perfil do usuário

- \*\*FavoritesModule\*\*: Locais favoritos

### 1.2 Componentes UI

- \*\*LocationSearchBar\*\*: Barra de busca com filtros

- \*\*MapViewComponent\*\*: Mapa interativo com marcadores

- \*\*LocationCard\*\*: Cartão de exibição de local

- \*\*FilterModal\*\*: Modal de filtros avançados

- \*\*ReviewComponent\*\*: Avaliações de locais

### 1.3 Serviços

- \*\*LocationService\*\*: Comunicação com backend para busca

- \*\*AuthService\*\*: Gerenciamento de autenticação

- \*\*GeoLocationService\*\*: Obtenção da localização do usuário

- \*\*StorageService\*\*: Armazenamento local (favoritos, histórico)

## 2. Componentes do Backend (PHP)

### 2.1 Controladores

- \*\*AuthController\*\*: Login/registro/recuperação

- \*\*LocationController\*\*: CRUD de locais

- \*\*SearchController\*\*: Lógica de busca

- \*\*ReviewController\*\*: Gerenciamento de avaliações

### 2.2 Serviços

- \*\*GeocodingService\*\*: Conversão de endereços em coordenadas

- \*\*SearchService\*\*: Algoritmos de busca e filtragem

- \*\*NotificationService\*\*: Envio de notificações

### 2.3 Middlewares

- \*\*AuthMiddleware\*\*: Verificação de autenticação

- \*\*CorsMiddleware\*\*: Gerenciamento de CORS

- \*\*LoggingMiddleware\*\*: Registro de requisições

## 3. Banco de Dados (PostgreSQL)

### 3.1 Esquema Principal

```sql

-- Tabela de usuários

CREATE TABLE users (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

phone VARCHAR(20),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Tabela de locais (adoção/veterinário)

CREATE TABLE locations (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

description TEXT,

address TEXT NOT NULL,

latitude DECIMAL(10, 8) NOT NULL,

longitude DECIMAL(11, 8) NOT NULL,

location\_type VARCHAR(20) CHECK (location\_type IN ('adoption', 'veterinary')),

phone VARCHAR(20),

email VARCHAR(100),

website VARCHAR(255),

opening\_hours JSONB,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Tabela de avaliações

CREATE TABLE reviews (

id SERIAL PRIMARY KEY,

user\_id INTEGER REFERENCES users(id),

location\_id INTEGER REFERENCES locations(id),

rating INTEGER NOT NULL CHECK (rating BETWEEN 1 AND 5),

comment TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Tabela de favoritos

CREATE TABLE favorites (

user\_id INTEGER REFERENCES users(id),

location\_id INTEGER REFERENCES locations(id),

PRIMARY KEY (user\_id, location\_id),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Tabela de animais para adoção (se aplicável)

CREATE TABLE adoption\_animals (

id SERIAL PRIMARY KEY,

location\_id INTEGER REFERENCES locations(id),

name VARCHAR(100),

animal\_type VARCHAR(50),

breed VARCHAR(100),

age INTEGER,

gender VARCHAR(10),

description TEXT,

photos TEXT[],

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

```

### 3.2 Índices

```sql

CREATE INDEX idx\_locations\_type ON locations(location\_type);

CREATE INDEX idx\_locations\_geo ON locations(latitude, longitude);

CREATE INDEX idx\_reviews\_location ON reviews(location\_id);

CREATE INDEX idx\_adoption\_location ON adoption\_animals(location\_id);

```

## 4. API Endpoints (PHP)

### 4.1 Autenticação

- `POST /api/auth/register` - Registrar novo usuário

- `POST /api/auth/login` - Login

- `POST /api/auth/refresh` - Refresh token

### 4.2 Busca

- `GET /api/locations/search` - Busca por locais (parâmetros: lat, lng, radius, type, etc.)

- `GET /api/locations/{id}` - Detalhes de um local específico

### 4.3 Favoritos

- `POST /api/favorites` - Adicionar aos favoritos

- `DELETE /api/favorites/{locationId}` - Remover dos favoritos

- `GET /api/favorites` - Listar favoritos do usuário

### 4.4 Avaliações

- `POST /api/reviews` - Criar avaliação

- `GET /api/locations/{id}/reviews` - Listar avaliações de um local

## 5. Fluxo de Geolocalização

1. App solicita permissão de localização

2. Obtém coordenadas do dispositivo

3. Envia coordenadas para `/api/locations/search` com raio de busca

4. Backend calcula locais mais próximos usando PostGIS (PostgreSQL)

5. Retorna resultados ordenados por proximidade

6. App exibe marcadores no mapa e lista de resultados

## 6. Bibliotecas Recomendadas

### React Native

- react-native-maps (mapas)

- react-native-geolocation (geolocalização)

- axios (chamadas API)

- react-navigation (navegação)

- redux (gerenciamento de estado)

### PHP

- Slim Framework ou Laravel (API)

- PHP-JWT (autenticação)

- Guzzle (para chamadas a APIs de geocoding)

### PostgreSQL

- PostGIS (extensão para geolocalização)

Este esquema fornece uma base sólida para o desenvolvimento do sistema com todas as funcionalidades essenciais para busca de locais de adoção e veterinários com geolocalização.

**# Ordem Recomendada para Desenvolvimento dos Componentes**

Baseado no esquema anterior, aqui está uma ordem lógica de desenvolvimento e indicação de quais componentes podem ser feitos em paralelo:

## 1. Fase Inicial (Fundações)

\*\*1.1 Banco de Dados (Prioridade Máxima)\*\*

- Esquema do PostgreSQL com tabelas básicas (users, locations)

- Extensão PostGIS para geolocalização

- \*Pode ser feito em paralelo com o backend básico\*

\*\*1.2 Backend Básico (PHP)\*\*

- Configuração inicial do projeto PHP (Slim/Laravel)

- Conexão com o banco de dados

- Rotas básicas de autenticação (register/login)

- \*Pode ser feito em paralelo com o banco de dados\*

\*\*1.3 Frontend Básico (React Native)\*\*

- Configuração do projeto React Native

- Tela de login/registro básica

- Navegação básica

- \*Depende do backend de autenticação estar pronto para testes\*

## 2. Fase Principal (Funcionalidades Essenciais)

\*\*2.1 Backend (Pode ser feito em paralelo com frontend)\*\*

- [ ] Geolocalização (SearchController)

- [ ] CRUD de locais (LocationController)

- [ ] Sistema de avaliações (ReviewController)

- [ ] Endpoints de favoritos

\*\*2.2 Frontend (Componentes independentes podem ser paralelizados)\*\*

- [ ] Módulo de mapa (MapModule)

- [ ] Barra de busca (LocationSearchBar)

- [ ] Listagem de resultados

- [ ] Tela de detalhes do local

- [ ] Sistema de favoritos

## 3. Fase de Aprimoramento

\*\*3.1 Backend\*\*

- [ ] Filtros avançados

- [ ] Paginação de resultados

- [ ] Cache de consultas

- [ ] Sistema de notificações

\*\*3.2 Frontend\*\*

- [ ] Filtros avançados (FilterModal)

- [ ] Avaliações (ReviewComponent)

- [ ] Perfil do usuário

- [ ] Histórico de buscas

## Paralelização Recomendada

### Grupos que podem ser desenvolvidos em paralelo:

\*\*Grupo 1: Autenticação\*\*

- Backend: AuthController

- Frontend: AuthModule, AuthService

- Banco: tabela users

\*\*Grupo 2: Núcleo de Busca\*\*

- Backend: SearchController, GeocodingService

- Frontend: SearchModule, GeoLocationService

- Banco: índices espaciais

\*\*Grupo 3: Gestão de Locais\*\*

- Backend: LocationController

- Frontend: LocationCard, MapViewComponent

- Banco: tabela locations

\*\*Grupo 4: Interações do Usuário\*\*

- Backend: ReviewController

- Frontend: ReviewComponent, FavoritesModule

- Banco: tabelas reviews e favorites

## Ordem de Prioridade Recomendada

1. \*\*Banco de Dados + Autenticação\*\* (fundação para tudo)

2. \*\*Geolocalização básica\*\* (mostrar locais no mapa)

3. \*\*Sistema de busca\*\* (filtros básicos)

4. \*\*Detalhes do local\*\* (informações completas)

5. \*\*Avaliações e favoritos\*\* (engajamento)

6. \*\*Funcionalidades avançadas\*\* (notificações, histórico)

## Dependências Críticas

1. O banco de dados deve estar estável antes do desenvolvimento intensivo do backend

2. Os endpoints básicos de API devem estar prontos antes do desenvolvimento do frontend que os consome

3. O módulo de mapa no frontend depende da geolocalização estar funcionando no backend

## Sugestão de Sprint Inicial

\*\*Sprint 0 (1-2 semanas):\*\*

- Configuração de ambientes

- Banco de dados básico

- Autenticação (backend + frontend)

- Tela de mapa básica com marcadores estáticos

\*\*Sprint 1:\*\*

- Geolocalização funcional

- Busca por proximidade

- Listagem de resultados

- Tela de detalhes do local

\*\*Sprint 2:\*\*

- Sistema de avaliações

- Favoritos

- Filtros básicos

Esta abordagem permite progresso contínuo com entregas de valor a cada sprint, mantendo as dependências sob controle e maximizando o trabalho em paralelo onde possível.