

# MAPEADOR DE COMUNIDADE - DOCUMENTAÇÃO COMPLETA

## ÍNDICE

1. [Visão Geral do Projeto](#)
  2. [Stack Tecnológica](#)
  3. [Estrutura do Banco de Dados](#)
  4. [Arquitetura do Sistema](#)
  5. [Componentes Principais](#)
  6. [APIs e Rotas](#)
  7. [Funcionalidades Implementadas](#)
  8. [Guia de Instalação](#)
  9. [Configuração do Ambiente](#)
  10. [Fluxos de Dados](#)
  11. [Sistema de Autenticação](#)
  12. [Problemas Conhecidos e Soluções](#)
  13. [Roadmap de Desenvolvimento](#)
- 

## VISÃO GERAL DO PROJETO



### Descrição





Sistema web para mapeamento político e social de relacionamentos comunitários, permitindo organizar e analisar redes de contatos com foco em influência política e profissional.

### Objetivos Principais

- Mapear pessoas e seus relacionamentos em diferentes contextos
- Categorizar contatos por proximidade, influência e importância
- Visualizar redes de relacionamento através de diferentes interfaces
- Analisar potencial de mobilização política
- Gerenciar informações detalhadas sobre cada pessoa

### Status Atual

-  Sistema funcional com CRUD completo
-  Autenticação implementada

-  Visualizações múltiplas (tabela, círculos, grafo, dashboard)
  -  Sistema de tags
  -  Dashboard político
  -  Sistema de relacionamentos (parcialmente implementado)
- 

## STACK TECNOLÓGICA

### Frontend

```
json

{
  "framework": "Next.js 15.3.3",
  "ui": {
    "react": "18",
    "typescript": "5",
    "styling": "Tailwind CSS 3",
    "icons": "Lucide Icons",
    "charts": "Recharts",
    "graphs": "D3.js (via NetworkGraph)"
  }
}
```

### Backend

```
json

{
  "runtime": "Node.js",
  "framework": "Next.js API Routes",
  "authentication": "NextAuth.js 4",
  "database": {
    "type": "MySQL 8.0",
    "driver": "mysql2",
    "host": "localhost",
    "user": "rodrigo",
    "password": "884422",
    "database": "community_mapper"
  }
}
```

### Ferramentas de Desenvolvimento

- TypeScript para type safety
- ESLint para linting

- Git para controle de versão

## ESTRUTURA DO BANCO DE DADOS

### Diagrama ER Simplificado

```
users (1) ----< (N) people
|
|
+----- (1) ---< (N) tags
|
|
+----- (1) ---< (N) groups
|
people (N) >---< (N) groups (person_groups)
people (N) >---< (N) tags (person_tags)
people (N) >---< (N) people (person_relationships)
```

### Tabelas Detalhadas

#### 1. users

sql

```
CREATE TABLE users (
  id INT PRIMARY KEY AUTO_INCREMENT,
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  name VARCHAR(255),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

#### 2. people (Tabela Principal)



```

CREATE TABLE people (
  id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT NOT NULL,
  -- Informações Básicas
  name VARCHAR(255) NOT NULL,
  nickname VARCHAR(100),
  birth_date DATE,
  gender ENUM('M', 'F', 'N') DEFAULT 'N',

  -- Categorização
  context VARCHAR(50) NOT NULL, -- residencial, profissional, social, etc
  proximity VARCHAR(50) NOT NULL, -- nucleo, primeiro, segundo, terceiro, periferia

  -- Níveis de Avaliação
  importance INT DEFAULT 3, -- 1-5
  trust_level INT DEFAULT 3, -- 1-5
  influence_level INT DEFAULT 3, -- 1-5

  -- Informações Profissionais
  occupation VARCHAR(255),
  company VARCHAR(255),
  position VARCHAR(255),
  professional_class VARCHAR(255),
  education_level VARCHAR(50),
  income_range VARCHAR(50),

  -- Informações Políticas
  political_party VARCHAR(100),
  political_position VARCHAR(100),
  is_candidate BOOLEAN DEFAULT FALSE,
  is_elected BOOLEAN DEFAULT FALSE,
  political_role VARCHAR(255),

  -- Contatos
  phone VARCHAR(20),
  mobile VARCHAR(20),
  email VARCHAR(255),
  whatsapp VARCHAR(20),

  -- Endereço
  address TEXT,
  city VARCHAR(100),
  state VARCHAR(2),
  zip_code VARCHAR(10),

  -- Redes Sociais (armazenadas como campos separados)

```

```

facebook VARCHAR(255),
instagram VARCHAR(255),
twitter VARCHAR(255),
linkedin VARCHAR(255),

-- Outros
notes TEXT,
last_contact DATE,
contact_frequency VARCHAR(50), -- daily, weekly, monthly, quarterly, yearly

-- Timestamps
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
INDEX idx_user_id (user_id),
INDEX idx_context (context),
INDEX idx_proximity (proximity),
INDEX idx_political_party (political_party)
);

```

### 3. tags

sql

```

CREATE TABLE tags (
  id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT NOT NULL,
  name VARCHAR(50) NOT NULL,
  color VARCHAR(7) DEFAULT '#3B82F6',
  description TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
  UNIQUE KEY unique_tag_per_user (user_id, name)
);

```

### 4. person\_tags (Associação N:N)

sql

```
CREATE TABLE person_tags (  
  person_id INT NOT NULL,  
  tag_id INT NOT NULL,  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  PRIMARY KEY (person_id, tag_id),  
  FOREIGN KEY (person_id) REFERENCES people(id) ON DELETE CASCADE,  
  FOREIGN KEY (tag_id) REFERENCES tags(id) ON DELETE CASCADE  
);
```

## 5. groups

sql

```
CREATE TABLE groups (  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  user_id INT NOT NULL,  
  name VARCHAR(255) NOT NULL,  
  type VARCHAR(50),  
  description TEXT,  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE  
);
```

## 6. person\_relationships

sql

```
CREATE TABLE person_relationships (  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  person_a_id INT NOT NULL,  
  person_b_id INT NOT NULL,  
  relationship_type VARCHAR(50),  
  strength INT DEFAULT 3,  
  notes TEXT,  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  FOREIGN KEY (person_a_id) REFERENCES people(id) ON DELETE CASCADE,  
  FOREIGN KEY (person_b_id) REFERENCES people(id) ON DELETE CASCADE,  
  UNIQUE KEY unique_relationship (person_a_id, person_b_id)  
);
```

---

# ARQUITETURA DO SISTEMA

## Estrutura de Diretórios

mapa-comunidade/

```
├─ app/                                # App Router do Next.js
│   ├─ api/                            # API Routes
│   │   ├─ auth/[...nextauth]/        # Autenticação
│   │   │   └─ route.ts
│   │   ├─ people/                    # CRUD de pessoas
│   │   │   └─ route.ts
│   │   ├─ tags/                      # Gerenciamento de tags
│   │   │   └─ route.ts
│   │   └─ person-tags/               # Associação pessoa-tag
│   │       └─ route.ts
│   │
│   └─ components/                    # Componentes React
│       ├─ AnalyticsDashboard.tsx
│       ├─ BulkActions.tsx
│       ├─ ExportModal.tsx
│       ├─ MobileMenu.tsx
│       ├─ NetworkGraph.tsx
│       ├─ NextAuthProvider.tsx
│       ├─ PersonTags.tsx
│       ├─ PhotoUploadModal.tsx
│       ├─ PoliticalDashboard.tsx
│       ├─ RelationshipManager.tsx
│       ├─ TagManager.tsx
│       └─ ThemeProvider.tsx
│   │
│   └─ hooks/                          # Custom React Hooks
│       ├─ usePeople.ts
│       └─ useTags.ts
│   │
│   └─ login/                          # Página de login
│       └─ page.tsx
│   │
│   └─ globals.css                    # Estilos globais
│   └─ layout.tsx                     # Layout principal
│   └─ page.tsx                       # Dashboard principal
│
├─ lib/                                # Utilitários e configurações
│   └─ db.ts                          # Conexão e queries MySQL
│
├─ public/                             # Assets públicos
├─ .env.local                          # Variáveis de ambiente
├─ next.config.mjs                     # Configuração Next.js
└─ package.json                        # Dependências
```



— tailwind.config.ts	# Configuração Tailwind
— tsconfig.json	# Configuração TypeScript

## Fluxo de Dados

```
Frontend (React)
  ↓↑ (HTTP/JSON)
API Routes (Next.js)
  ↓↑ (SQL)
MySQL Database
```

---

## COMPONENTES PRINCIPAIS

### 1. page.tsx (Dashboard Principal)

#### Responsabilidades:

- Gerenciamento de estado global da aplicação
- Controle de visualizações (tabela, círculos, grafo, dashboard)
- Formulários de adição/edição de pessoas
- Integração com todos os outros componentes

#### Estados principais:

```
typescript

- people: Person[]           // Lista de pessoas
- searchTerm: string          // Busca
- filterContext: string       // Filtro por contexto
- filterProximity: string     // Filtro por proximidade
- viewMode: string            // Modo de visualização
- editingPerson: Person | null // Pessoa sendo editada
- selectedPeople: Set<string> // Seleção múltipla
```

### 2. NetworkGraph.tsx

**Descrição:** Visualização em grafo de rede usando D3.js **Features:**

- Nós coloridos por contexto
- Tamanho dos nós por importância
- Animações e interatividade
- Zoom e pan

### 3. AnalyticsDashboard.tsx

**Descrição:** Dashboard analítico com gráficos **Métricas:**

- Distribuição por contexto
- Distribuição por proximidade
- Influência média por grupo
- Frequência de contatos

#### 4. PoliticalDashboard.tsx

**Descrição:** Dashboard específico para análise política **Features:**

- Distribuição por partido
- Top influenciadores políticos
- Potencial de mobilização
- Taxa de eleitos

#### 5. TagManager.tsx

**Descrição:** Gerenciador de tags/etiquetas **Features:**

- CRUD de tags
- Seletor de cores
- Contador de pessoas por tag

#### 6. RelationshipManager.tsx

**Descrição:** Gerenciador de relacionamentos entre pessoas **Status:** Parcialmente implementado

---

### APIs E ROTAS

#### 1. /api/auth/[...nextauth]

**Método:** Todos **Descrição:** Autenticação via NextAuth **Configuração:**

```
typescript
```

- Provider: Credentials
- Session: JWT
- Callbacks: session, jwt

#### 2. /api/people

**GET** - Lista todas as pessoas do usuário

typescript

Response: Person[]

## POST - Cria nova pessoa

typescript

Body: Omit<Person, 'id'>

Response: Person

## PUT - Atualiza pessoa

typescript

Body: Person

Response: Person

## DELETE - Remove pessoa

typescript

Query: ?id=123

Response: { success: boolean }

## 3. /api/tags

**GET** - Lista tags do usuário **POST** - Cria nova tag **PUT** - Atualiza tag **DELETE** - Remove tag

## 4. /api/person-tags

**GET** - Lista tags de uma pessoa **POST** - Adiciona tag a pessoa **DELETE** - Remove tag de pessoa

---

## ✅ FUNCIONALIDADES IMPLEMENTADAS

### 1. Autenticação e Segurança

- ✅ Login com email/senha
- ✅ Sessões JWT seguras
- ✅ Proteção de rotas
- ✅ Criptografia bcrypt

### 2. Gestão de Pessoas

- ✅ CRUD completo
- ✅ Formulário detalhado com 30+ campos
- ✅ Validação de dados

- ✓ Busca e filtros
- ✓ Seleção múltipla
- ✓ Ações em massa

### 3. Categorização

- ✓ 6 contextos pré-definidos
- ✓ 5 níveis de proximidade
- ✓ Níveis de importância/confiança/influência
- ✓ Tags personalizadas

### 4. Visualizações

- ✓ Tabela interativa
- ✓ Círculos concêntricos
- ✓ Grafo de rede (D3.js)
- ✓ Dashboard analítico
- ✓ Dashboard político

### 5. Features Adicionais

- ✓ Modo escuro/claro
- ✓ Interface responsiva
- ✓ Exportação CSV/JSON
- ✓ Upload de fotos (localStorage)
- ✓ Mapeamento guiado (wizard)



## GUIA DE INSTALAÇÃO

### Pré-requisitos

- Node.js 18+
- MySQL 8.0+
- Git

### Passo a Passo

#### 1. Clone o repositório

bash

```
git clone [URL_DO_REPOSITORIO]
cd mapa-comunidade
```

#### 2. Instale as dependências

```
bash
```

```
npm install
```

### 3. Configure o banco de dados

```
bash
```

```
mysql -u root -p  
CREATE DATABASE community_mapper;  
USE community_mapper;  
# Execute o SQL do arquivo database-schema.sql
```

### 4. Configure as variáveis de ambiente

```
bash
```

```
cp .env.example .env.local  
# Edite o arquivo com suas configurações
```

### 5. Execute as migrações (se houver)

```
bash
```

```
# Criar usuário de teste  
INSERT INTO users (email, password_hash, name)  
VALUES ('rodrigo@example.com', '$2a$10$...', 'Rodrigo');
```

### 6. Inicie o servidor

```
bash
```

```
npm run dev
```

### 7. Acesse o sistema

```
http://localhost:3000  
Email: rodrigo@example.com  
Senha: admin123
```

---

## CONFIGURAÇÃO DO AMBIENTE

### Arquivo .env.local

env

```
# Banco de Dados
DATABASE_HOST=localhost
DATABASE_USER=rodrigo
DATABASE_PASSWORD=884422
DATABASE_NAME=community_mapper

# MySQL alternativo (se usar variáveis diferentes)
MYSQL_HOST=localhost
MYSQL_USER=rodrigo
MYSQL_PASSWORD=884422
MYSQL_DATABASE=community_mapper

# NextAuth
NEXTAUTH_URL=http://localhost:3000
NEXTAUTH_SECRET=[gerar com: openssl rand -base64 32]
```

## Configurações Importantes

### next.config.mjs

javascript

```
const nextConfig = {
  reactStrictMode: true,
  swcMinify: true,
}
```

### tailwind.config.ts

typescript

- Dark mode: 'class'
- Custom animations
- Extended colors



## FLUXOS DE DADOS

### 1. Fluxo de Autenticação

Login Form → NextAuth → MySQL (users) → JWT → Session → Dashboard

### 2. Fluxo CRUD de Pessoas

Form → usePeople Hook → API Route → MySQL → Response → State Update → UI Update

### 3. Fluxo de Tags

TagManager → useTags Hook → API → MySQL (tags + person\_tags) → UI Update

---

## SISTEMA DE AUTENTICAÇÃO

### NextAuth Configuration

```
typescript

providers: [
  CredentialsProvider({
    credentials: {
      email: { label: "Email", type: "email" },
      password: { label: "Password", type: "password" }
    },
    authorize: async (credentials) => {
      // Valida contra MySQL
      // Retorna user object ou null
    }
  })
]
```

### Proteção de Rotas

- Middleware do NextAuth
  - Verificação de sessão em cada API route
  - Redirect automático para /login
- 

## PROBLEMAS CONHECIDOS E SOLUÇÕES

### 1. Erro "Failed to create"

**Causa:** Incompatibilidade entre tipos de ID (string vs number) **Solução:**

- db.ts usa IDs numéricos (AUTO\_INCREMENT)
- API converte para string antes de enviar ao frontend

### 2. Dark Mode Flash

**Causa:** Tema aplicado após renderização **Solução:** Script inline no head que aplica tema antes da renderização

### 3. Fotos em localStorage

**Limitação:** ~5MB por domínio **Alternativa futura:** Upload para servidor ou cloud storage

---



## ROADMAP DE DESENVOLVIMENTO

### Fase 1 - Correções (Atual)

- ☒ Corrigir erro de criação de pessoas
- ☒ Estabilizar sistema de tags
- ☐ Implementar relacionamentos completos
- ☐ Corrigir bugs menores

### Fase 2 - Melhorias

- ☐ Sistema de grupos funcional
- ☐ Upload de fotos para servidor
- ☐ Histórico de interações
- ☐ Notificações e lembretes
- ☐ Busca avançada

### Fase 3 - Features Avançadas

- ☐ API REST completa
- ☐ App mobile (React Native)
- ☐ Integração WhatsApp
- ☐ Relatórios PDF
- ☐ Backup automático
- ☐ Multi-tenancy

### Fase 4 - Análises e IA

- ☐ Análise de sentimento
- ☐ Predição de influência
- ☐ Sugestões de conexões
- ☐ Mapas geográficos
- ☐ Timeline de relacionamentos



## COMANDOS ÚTEIS

### Desenvolvimento



bash

```
npm run dev          # Inicia servidor de desenvolvimento
npm run build        # Build de produção
npm start            # Inicia servidor de produção
npm run lint         # Verifica código
```

## Banco de Dados

bash

```
mysql -u rodrigo -p884422
USE community_mapper;
SELECT COUNT(*) FROM people;
SELECT * FROM people WHERE user_id = 1 LIMIT 10;
```

## Git

bash

```
git status
git add .
git commit -m "feat: descrição"
git push origin main
```

---

## CONVENÇÕES DO PROJETO

### Código

- TypeScript strict mode
- Componentes funcionais React
- Hooks customizados para lógica
- Async/await para promises

### Nomenclatura

- Componentes: PascalCase
- Funções: camelCase
- Constantes: UPPER\_CASE
- Arquivos: kebab-case ou PascalCase

### Git Commits

- feat: nova funcionalidade

- fix: correção de bug
  - docs: documentação
  - style: formatação
  - refactor: refatoração
  - test: testes
  - chore: tarefas gerais
- 

## SUPORTE E CONTATO

### Problemas Comuns

1. **Erro de conexão MySQL:** Verificar credenciais no .env.local
2. **Erro 401:** Fazer login novamente
3. **Tela branca:** Verificar console do navegador
4. **Dados não aparecem:** Verificar se há pessoas cadastradas

### Debug

- Console do navegador (F12)
  - Logs do servidor (terminal)
  - MySQL logs
  - Network tab para requisições
- 

## INÍCIO RÁPIDO PARA NOVO CHAT

Copie e cole no início do chat:

Estou desenvolvendo o sistema Mapeador de Comunidade, um sistema web para mapeamento político e social.

Stack: Next.js 15, React 18, TypeScript, Tailwind CSS, MySQL 8, NextAuth

Estrutura do banco: users → people (com 30+ campos incluindo contexto, proximidade, dados políticos)

Status: Sistema funcional com CRUD completo, autenticação, visualizações (tabela, círculos, grafo), tags e dashboard político.

Arquivos principais:

- app/page.tsx (dashboard principal)
- lib/db.ts (conexão MySQL com IDs numéricos)
- app/api/people/route.ts (CRUD API)
- components/\* (15+ componentes)

Credenciais MySQL: user=rodrigo, pass=884422, db=community\_mapper

Problemas resolvidos: IDs numéricos no banco, strings no frontend.

Como posso ajudar hoje?

---

**Última atualização:** Janeiro 2025 **Versão:** 1.0.0