# Cronograma UBSF App - API Backend

## 📋 Visão Geral

API REST para gerenciamento de cronogramas de atividades para Unidades Básicas de Saúde da Família (UBSF). Desenvolvida com Vercel Functions, Prisma ORM e PostgreSQL.

## 🚀 Tecnologias

- **Runtime:** Node.js 18+
- **Framework:** Vercel Functions (Serverless)
- **Banco de Dados:** PostgreSQL (Neon)
- **ORM:** Prisma
- **Validação:** Joi
- **Deploy:** Vercel

## 📂 Estrutura do Projeto

```
Cronograma-UBSF-App/
├── api/                      # Vercel Functions (Backend)
│   ├── cronogramas/
│   │   ├── index.js          # GET/POST /api/cronogramas
│   │   ├── [id].js           # GET/PUT/DELETE /api/cronogramas/[id]
│   │   └── [id]/
│   │       └── atividades.js # GET/POST /api/cronogramas/[id]/atividades
│   ├── atividades/
│   │   └── [id].js           # GET/PUT/DELETE /api/atividades/[id]
│   └── health.js             # Health check
├── lib/                       # Utilitários compartilhados
│   ├── database.js           # Configuração Prisma
│   └── utils.js              # Funções utilitárias
├── prisma/
│   ├── schema.prisma         # Schema do banco
│   └── seed.js               # Dados de exemplo
├── vercel.json               # Configuração Vercel
├── package.json
└── .env                      # Variáveis de ambiente
```

## 🛠️ Configuração Local

1. Instalar Dependências

```
npm install
```

### 2. Configurar Banco de Dados

1. **Criar conta no Neon:** https://neon.tech
2. **Criar novo projeto PostgreSQL**
3. **Copiar connection string**
4. **Atualizar .env:**

```
DATABASE_URL="postgresql://username:password@host/database?sslmode=require"
```

### 3. Configurar Prisma

```
# Gerar cliente Prisma
npm run db:generate

# Executar migrações
npm run db:migrate

# Popular banco com dados de exemplo
npm run db:seed
```

### 4. Desenvolvimento Local

```
# Instalar Vercel CLI (se não tiver)
npm i -g vercel

# Iniciar servidor de desenvolvimento
npm run dev
```

A API estará disponível em: `http://localhost:3000`

## 🗄 Endpoints da API

### Health Check

- `GET /api/health` - Verificar status da API

### Cronogramas

- `GET /api/cronogramas` - Listar cronogramas
- `POST /api/cronogramas` - Criar cronograma
- `GET /api/cronogramas/{id}` - Buscar cronograma por ID
- `PUT /api/cronogramas/{id}` - Atualizar cronograma
- `DELETE /api/cronogramas/{id}` - Deletar cronograma

### Atividades

- `GET /api/cronogramas/{id}/atividades` - Listar atividades do cronograma
- `POST /api/cronogramas/{id}/atividades` - Criar atividade
- `GET /api/atividades/{id}` - Buscar atividade por ID
- `PUT /api/atividades/{id}` - Atualizar atividade
- `DELETE /api/atividades/{id}` - Deletar atividade

## 📝 Exemplos de Uso

### Criar Cronograma

```
curl -X POST http://localhost:3000/api/cronogramas \
  -H "Content-Type: application/json" \
  -d '{
    "mes": 3,
    "ano": 2024,
    "nomeUBSF": "UBSF Centro",
    "enfermeiro": "Maria Silva",
    "medico": "Dr. João Santos"
  }'
```

### Listar Cronogramas

```
curl http://localhost:3000/api/cronogramas
```

### Criar Atividade

```
curl -X POST http://localhost:3000/api/cronogramas/{cronograma_id}/atividades \
  -H "Content-Type: application/json" \
  -d '{
    "data": "2024-03-01",
    "diaSemana": "SEXTA-MANHÃ",
    "descricao": "Consultas de rotina"
  }'
```

## 🚀 Deploy no Vercel

### 1. Conectar Repositório

1. **Fazer push do código para GitHub/GitLab**
2. **Conectar repositório no Vercel**
3. **Configurar variáveis de ambiente**

### 2. Configurar Environment Variables

No painel do Vercel, adicionar:

```
DATABASE_URL=postgresql://...
NODE_ENV=production
```

### 3. Deploy

```
# Deploy manual
vercel --prod

# Ou via Git (automático)
git push origin main
```

## 🗄️ Schema do Banco

### Cronograma

```
id           String   @id @default(cuid())
mes          Int      // 1-12
ano          Int
nomeUBSF     String?
enfermeiro   String?
medico       String?
criadoEm     DateTime @default(now())
atualizadoEm DateTime @updatedAt
```

### Atividade

```
id           String   @id @default(cuid())
cronogramaId String
data         DateTime
diaSemana    String   // SEGUNDA-MANHÃ, TERÇA-TARDE, etc.
descricao    String
criadoEm     DateTime @default(now())
```

## 🔗 Scripts Disponíveis

```
npm run dev          # Desenvolvimento local
npm run build        # Build para produção
npm run deploy       # Deploy para Vercel
```

```
npm run db:migrate   # Executar migrações
npm run db:generate  # Gerar cliente Prisma
npm run db:studio    # Abrir Prisma Studio
npm run db:push      # Push schema para DB
npm run db:seed      # Popular banco com dados
```

## 📊 Monitoramento

### Logs

- **Local:** Console do terminal
- **Produção:** Vercel Dashboard > Functions > Logs

### Métricas

- **Performance:** Vercel Analytics
- **Erros:** Vercel Functions logs
- **Banco:** Neon Dashboard

## 🛡️ Validações

### Cronograma

- `mes`: 1-12 (obrigatório)
- `ano`: 2020-2030 (obrigatório)
- `nomeUBSF`: máximo 255 caracteres
- `enfermeiro`: máximo 255 caracteres
- `medico`: máximo 255 caracteres

### Atividade

- `data`: data válida (obrigatório)
- `diaSemana`: valores específicos (obrigatório)
- `descricao`: máximo 500 caracteres (obrigatório)
- Única por cronograma + data + período

## 📥 Tratamento de Erros

### Códigos de Status

- `200` - Sucesso
- `201` - Criado
- `400` - Dados inválidos
- `404` - Não encontrado
- `405` - Método não permitido
- `500` - Erro interno

### Formato de Resposta

```
{
  "success": true/false,
  "message": "Mensagem descritiva",
  "data": {...},
  "timestamp": "2024-01-01T00:00:00.000Z"
}
```

## 🤝 Contribuição

1. Fork o projeto
2. Crie uma branch (`git checkout -b feature/nova-funcionalidade`)
3. Commit suas mudanças (`git commit -m 'Adiciona nova funcionalidade'`)
4. Push para a branch (`git push origin feature/nova-funcionalidade`)
5. Abra um Pull Request

## 📄 Licença

MIT License - veja o arquivo LICENSE para detalhes.

---

**Desenvolvido para facilitar o gerenciamento de cronogramas em UBSFs** 🏥💙