Arquitetura do Sistema Fertili - Documentação Técnica

Visão Geral da Arquitetura

O sistema Fertili é uma plataforma completa de gestão pecuária construída em Django, projetada para atender desde pequenos produtores até grandes fazendas. A arquitetura segue os princípios de modularidade, escalabilidade e integração, permitindo que diferentes componentes trabalhem de forma sinérgica.

Componentes Principais

1. **Fertili 360** - Aplicação web principal (Django + Frontend)
2. **Fertili Campo** - Aplicativo móvel (API REST + App nativo/híbrido)
3. **API REST** - Django Rest Framework para integração
4. **Banco de Dados** - PostgreSQL com suporte a dados geoespaciais
5. **Sistema de Sincronização** - Para funcionamento offline

Módulos e Relacionamentos

1. Módulo de Usuários e Propriedades

Entidades Principais:
- `Usuario` (herda de AbstractUser)
- `Propriedade`
- `Area`

Relacionamentos:
- Um usuário pode gerenciar múltiplas propriedades (1:N)
- Uma propriedade possui múltiplas áreas (1:N)
- Controle de permissões granular por propriedade

Funcionalidades:
- Autenticação e autorização
- Gestão multi-propriedade
- Controle de acesso por perfil (proprietário, gerente, funcionário, veterinário)

2. Módulo de Rebanho

Entidades Principais:
- `Animal` - Entidade central do sistema
- `Lote` - Agrupamento de animais
- `HistoricoLoteAnimal` - Rastreamento de movimentações
- `HistoricoOcupacaoArea` - Controle de pastejo

Relacionamentos Complexos:

```
Animal (self-referencing)
├── pai_id → Animal (M)
├── mae_id → Animal (F)
├── lote_atual_id → Lote
└── propriedade_id → Propriedade

Lote
├── area_atual_id → Area
├── propriedade_id → Propriedade
└── animais ← Animal (1:N)
```

Funcionalidades:
- Genealogia completa com árvore familiar
- Gestão de lotes dinâmicos
- Rastreamento de movimentações
- Cálculo automático de UA (Unidades Animais)
- Histórico completo por animal

3. Módulo de Manejos

Arquitetura Hierárquica:

```
Manejo (classe base)
├── AnimalManejo (M:N com Animal)
├── Pesagem (1:1 com Manejo)
├── Vacinacao (1:1 com Manejo)
├── AdministracaoMedicamento (1:1 com Manejo)
├── Inseminacao (1:1 com Manejo)
├── DiagnosticoGestacao (1:1 com Manejo)
└── Parto (1:1 com Manejo)
```

Padrão de Design:
- Classe base `Manejo` com atributos comuns
- Especializações através de relacionamentos 1:1
- Tabela de junção `AnimalManejo` para relacionamento M:N
- Integração automática com módulo financeiro (custos)

4. Módulo de Reprodução

Fluxo de Dados:

```
EstacaoMonta
└── Inseminacao
    ├── DiagnosticoGestacao
    └── Parto → Animal (novo)
```

Funcionalidades Avançadas:
- Protocolos IATF configuráveis
- Cálculo automático de taxas reprodutivas
- Agendamento automático de diagnósticos
- Controle de estações de monta

5. Módulo de Sanidade

Entidades Especializadas:
- `Vacina` e `Medicamento` - Cadastros base
- `Vacinacao` e `AdministracaoMedicamento` - Aplicações
- `CalendarioSanitario` - Agendamentos futuros

Funcionalidades:
- Calendário sanitário com notificações
- Controle de carência de medicamentos
- Histórico sanitário completo por animal
- Protocolos sanitários personalizáveis

6. Módulo Financeiro

Estrutura Contábil:

```
ContaFinanceira
└── LancamentoFinanceiro
    ├── CategoriaFinanceira
    ├── manejo_relacionado → Manejo
    └── animal_relacionado → Animal
```

Integração com Operações:
- Custos de manejos automaticamente registrados
- Receitas de vendas de animais
- Relatórios de custo por animal/lote
- Análise de rentabilidade

Padrões de Design Implementados

1. Repository Pattern

- Métodos customizados nos models para consultas complexas
- Separação clara entre lógica de negócio e acesso a dados

2. Observer Pattern

- Signals do Django para ações automáticas
- Exemplo: Atualização de custos ao criar manejos

3. Strategy Pattern

- Diferentes tipos de manejo com comportamentos específicos
- Cálculos de GMD, UA, taxas reprodutivas

4. Factory Pattern

- Criação de animais por diferentes eventos (nascimento, compra)
- Geração de relatórios personalizados

Integrações e APIs

API REST (Django Rest Framework)

```
/api/v1/
├── propriedades/
├── animais/
├── lotes/
├── manejos/
├── pesagens/
├── reproducao/
├── sanidade/
└── financeiro/
```

Sincronização Offline

- Banco local SQLite no app móvel
- Fila de sincronização com resolução de conflitos
- Timestamps para controle de versão

Integrações Externas

- Bluetooth para equipamentos de pesagem
- GPS para localização de áreas
- APIs de cotação para preços de commodities

Cálculos e Algoritmos Principais

1. Ganho Médio Diário (GMD)

```python
def get_gmd_periodo(self, dias=30):
    # Busca pesagens no período
    # Calcula: (peso_final - peso_inicial) / dias
```

2. Taxa de Ocupação (UA/ha)

```python
def get_taxa_ocupacao_atual(self):
    # Soma UA dos animais / área em hectares
```

3. Índices Reprodutivos

```python
def get_taxa_prenhez(self):
    # (diagnósticos positivos / total fêmeas) * 100
```

4. Análise Financeira

```python
def get_custo_por_arroba(self):
    # Total custos / total arrobas produzidas
```

Segurança e Permissões

Níveis de Acesso

1. **Proprietário** - Acesso total à propriedade
2. **Gerente** - Acesso a relatórios e gestão operacional
3. **Funcionário** - Acesso limitado a coleta de dados
4. **Veterinário** - Foco em sanidade e reprodução

Auditoria

- Todos os models possuem campos de auditoria
- Rastreamento de usuário em operações críticas
- Logs de sincronização e alterações

Escalabilidade e Performance

Otimizações de Banco

- Índices em campos de busca frequente
- Particionamento por propriedade
- Queries otimizadas com select_related/prefetch_related

Cache

- Cache de cálculos complexos (GMD, taxas)
- Cache de relatórios frequentes
- Redis para sessões e cache distribuído

Monitoramento

- Métricas de performance por módulo
- Alertas para operações críticas
- Dashboard de saúde do sistema

Considerações de Implementação

Banco de Dados

- PostgreSQL com extensão PostGIS para dados geoespaciais
- Backup automático e replicação
- Particionamento por data para tabelas de histórico

Deploy

- Containerização com Docker
- CI/CD com testes automatizados
- Deploy em nuvem com auto-scaling

Monitoramento

- Logs estruturados
- Métricas de negócio (animais cadastrados, manejos realizados)
- Alertas para falhas de sincronização

Roadmap Técnico

Fase 1 - MVP

- Módulos básicos funcionais
- API REST completa
- App móvel com sincronização

Fase 2 - Inteligência

- Machine Learning para predições
- Análise preditiva de desempenho
- Otimização automática de manejos

Fase 3 - IoT

- Integração com sensores
- Monitoramento automático de pastagens
- Alertas em tempo real

Esta arquitetura garante que o sistema Fertili seja robusto, escalável e capaz de evoluir com as necessidades do agronegócio moderno, mantendo sempre o foco na usabilidade e na geração de valor para o produtor rural.