# Relatório de Funções por Módulo

## admin.py
- Classe `PropriedadeAdmin`:
  - Método `display_ciclos_pecuarios(self, obj)`: Sem descrição registrada.

## analise_financeira.py
- Classe `FluxoCaixa`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `analisar_fluxo_periodo(self, propriedade, data_inicio, data_fim)`: Analisa fluxo de caixa de um período específ
  - Método `projetar_fluxo_futuro(self, propriedade, meses_projecao)`: Projeta fluxo de caixa futuro baseado em histórico e
  - Método `_calcular_entradas_periodo(self, propriedade, inicio, fim)`: Calcula entradas do período
  - Método `_calcular_saidas_periodo(self, propriedade, inicio, fim)`: Calcula saídas do período
  - Método `_calcular_fluxo_diario(self, propriedade, inicio, fim)`: Calcula fluxo de caixa diário
  - Método `_calcular_indicadores_fluxo(self, total_entradas, total_saidas, saldo_periodo, fluxo_diario)`: Calcula indicadore
  - Método `_gerar_dados_grafico_waterfall(self, entradas, saidas)`: Gera dados para gráfico waterfall
  - Método `_projetar_entradas_mes(self, propriedade, data)`: Projeta entradas para um mês futuro
  - Método `_projetar_saidas_mes(self, propriedade, data)`: Projeta saídas para um mês futuro
  - Método `_identificar_meses_deficit(self, projecoes)`: Identifica meses com déficit projetado
- Classe `DRE`:
  - Método `gerar_dre_periodo(self, propriedade, data_inicio, data_fim)`: Gera DRE completo para o período
  - Método `_calcular_receita_vendas(self, propriedade, inicio, fim)`: Calcula receita de vendas de animais
  - Método `_calcular_outras_receitas(self, propriedade, inicio, fim)`: Calcula outras receitas
  - Método `_calcular_custos_variaveis(self, propriedade, inicio, fim)`: Calcula custos variáveis (variam com produção)
  - Método `_calcular_custos_fixos(self, propriedade, inicio, fim)`: Calcula custos fixos (não variam)
  - Método `_calcular_despesas_nao_operacionais(self, propriedade, inicio, fim)`: Calcula despesas não operacionais
  - Método `_calcular_indicadores_dre(self, receita_total, custos_variaveis, custos_fixos, lucro_liquido)`: Calcula indicador
  - Método `_analisar_estrutura_custos(self, custos_variaveis, custos_fixos)`: Analisa estrutura de custos
- Classe `AnaliseCustos`:
  - Método `calcular_custo_por_animal(self, propriedade, categoria, periodo_dias)`: Calcula custo total por animal em um p
  - Método `comparar_custos_categorias(self, propriedade, categorias)`: Compara custos entre categorias
  - Método `_calcular_custo_alimentacao(self, categoria, dias)`: Calcula custo de alimentação
  - Método `_calcular_custo_sanidade(self, categoria, dias)`: Calcula custos com saúde animal
  - Método `_calcular_custo_reproducao(self, categoria, dias)`: Calcula custos de reprodução
  - Método `_calcular_custo_manejo(self, categoria, dias)`: Calcula custos de manejo
  - Método `_analisar_custos_animal(self, diretos, indiretos, total)`: Analisa estrutura de custos
- Classe `IndicadoresFinanceiros`:
  - Método `calcular_indicadores_completos(self, propriedade, periodo_meses)`: Calcula todos os indicadores financeiros
  - Método `_analisar_tendencias_financeiras(self, propriedade)`: Analisa tendências dos indicadores
  - Método `_calcular_score_financeiro(self, rentabilidade, liquidez, endividamento)`: Calcula score geral de saúde finance
- Classe `ProjecaoFinanceira`:
  - Método `projetar_financeiro_5anos(self, propriedade, cenario)`: Projeta situação financeira para 5 anos
- Classe `AnalisadorFinanceiro`:

  - Método `__init__(self)`: Sem descrição registrada.
  - Método `gerar_relatorio_financeiro_completo(self, propriedade, data_inicio, data_fim)`: Gera relatório financeiro comple

## apis_integracao/api_agrofit.py
- Classe `AgrofitAPI`:
  - Método `__init__(self, api_key)`: Inicializa a API Agrofit
  - Método `_get_headers(self)`: Retorna headers para requisições
  - Método `buscar_produtos_por_cultura(self, cultura)`: Busca produtos fitossanitários permitidos para uma cultura
  - Método `buscar_produto_por_nome(self, nome)`: Busca produto fitossanitário por nome
  - Método `validar_produto_para_cultura(self, produto, cultura)`: Valida se produto pode ser usado em determinada cultur

## apis_integracao/api_bovtrace.py
- Classe `BovTraceAPI`:
  - Método `__init__(self, api_key)`: Inicializa a API BovTrace
  - Método `_get_headers(self)`: Retorna headers para requisições
  - Método `enviar_animal(self, animal_data)`: Envia dados de animal para BovTrace
  - Método `consultar_animal(self, numero_brinco)`: Consulta dados de animal no BovTrace
  - Método `validar_brinco(self, numero_brinco)`: Valida se brinco existe no sistema BovTrace
  - Método `registrar_movimentacao(self, movimentacao_data)`: Registra movimentação de animal
  - Método `obter_historico_animal(self, numero_brinco)`: Obtém histórico completo de movimentações de um animal

## apis_integracao/api_infodap.py
- Classe `InfoDAPAPI`:
  - Método `__init__(self, api_key)`: Inicializa a API InfoDAP
  - Método `_get_headers(self)`: Retorna headers para requisições
  - Método `consultar_dap(self, cpf_cnpj)`: Consulta DAP por CPF/CNPJ
  - Método `validar_propriedade_familiar(self, cpf_cnpj)`: Valida se propriedade é familiar (possui DAP)

## apps.py
- Classe `GestaoRuralConfig`:
  - Método `ready(self)`: Executado quando a aplicação está pronta
  - Método `criar_categorias_padrao(self)`: Cria as categorias padrão do sistema se não existirem

## forms.py
- Classe `PropriedadeForm`:
  - Método `__init__(self)`: Sem descrição registrada.
- Classe `ParametrosProjecaoForm`:
  - Método `clean_taxa_natalidade_anual(self)`: Sem descrição registrada.
  - Método `clean_taxa_mortalidade_bezerros_anual(self)`: Sem descrição registrada.
  - Método `clean_taxa_mortalidade_adultos_anual(self)`: Sem descrição registrada.
  - Método `clean_percentual_venda_machos_anual(self)`: Sem descrição registrada.
  - Método `clean_percentual_venda_femeas_anual(self)`: Sem descrição registrada.
  - Método `clean(self)`: Sem descrição registrada.

- Classe `CicloProducaoForm`:
  - Método `clean_area_plantada_ha(self)`: Sem descrição registrada.
  - Método `clean_produtividade_esperada_sc_ha(self)`: Sem descrição registrada.
  - Método `clean_custo_producao_por_ha(self)`: Sem descrição registrada.
  - Método `clean_preco_venda_por_sc(self)`: Sem descrição registrada.
  - Método `clean(self)`: Sem descrição registrada.
- Classe `TransferenciaPropriedadeForm`:
  - Método `__init__(self)`: Sem descrição registrada.
- Classe `CategoriaAnimalForm`:
  - Método `clean(self)`: Sem descrição registrada.

## forms_analise.py
- Classe `IndicadorFinanceiroForm`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `clean_valor(self)`: Sem descrição registrada.
  - Método `clean_data_referencia(self)`: Sem descrição registrada.

## forms_completos.py
- Classe `SetorPropriedadeForm`:
  - Método `__init__(self)`: Sem descrição registrada.
- Classe `ConviteCotacaoFornecedorForm`:
  - Método `__init__(self)`: Sem descrição registrada.
- Classe `RequisicaoCompraForm`:
  - Método `__init__(self)`: Sem descrição registrada.
- Classe `OrdemCompraForm`:
  - Método `__init__(self)`: Sem descrição registrada.
- Classe `OrcamentoCompraMensalForm`:
  - Método `__init__(self)`: Sem descrição registrada.
- Classe `CurralEventoForm`:
  - Método `__init__(self)`: Sem descrição registrada.

## forms_endividamento.py
- Classe `FinanciamentoForm`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `clean_valor_principal(self)`: Sem descrição registrada.
  - Método `clean_taxa_juros_anual(self)`: Sem descrição registrada.
  - Método `clean_numero_parcelas(self)`: Sem descrição registrada.
  - Método `clean_valor_parcela(self)`: Sem descrição registrada.
  - Método `clean(self)`: Sem descrição registrada.

## forms_financeiro.py
- Classe `CategoriaFinanceiraForm`:
  - Método `__init__(self)`: Sem descrição registrada.

- Classe `CentroCustoFinanceiroForm`:
  - Método `__init__(self)`: Sem descrição registrada.
- Classe `ContaFinanceiraForm`:
  - Método `__init__(self)`: Sem descrição registrada.
- Classe `LancamentoFinanceiroForm`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `clean(self)`: Sem descrição registrada.

## forms_imobilizado.py
- Classe `TipoBemForm`:
  - Método `clean_taxa_depreciacao(self)`: Sem descrição registrada.
- Classe `BemPatrimonialForm`:
  - Método `clean_valor_aquisicao(self)`: Sem descrição registrada.
  - Método `clean_valor_residual(self)`: Sem descrição registrada.

## forms_pesagem.py
- Classe `AnimalPesagemForm`:
  - Método `__init__(self)`: Sem descrição registrada.

## forms_vendas.py
- Classe `ParametrosVendaPorCategoriaForm`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `clean_percentual_venda_anual(self)`: Sem descrição registrada.
- Classe `BulkVendaPorCategoriaForm`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `clean(self)`: Sem descrição registrada.

## gestao_projetos.py
- Classe `GestorProjetos`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `criar_projeto(self, nome, tipo, propriedade, investimento_total, data_inicio, prazo_meses, objetivos, responsav
  - Método `acompanhar_projeto(self, projeto_id, percentual_concluido, investimento_realizado, observacoes)`: Atualiza e
  - Método `dashboard_projetos(self, propriedade)`: Gera dashboard completo de todos os projetos
  - Método `_gerar_etapas_padrao(self, tipo, prazo_total_meses)`: Gera etapas padrão baseadas no tipo de projeto
  - Método `_identificar_riscos_potenciais(self, tipo)`: Identifica riscos potenciais por tipo de projeto
  - Método `_definir_kpis_projeto(self, tipo)`: Define KPIs para acompanhamento do projeto
  - Método `_buscar_projeto(self, projeto_id)`: Busca projeto (simulado - implementar query real)
  - Método `_analisar_saude_projeto(self, projeto)`: Analisa saúde geral do projeto
  - Método `_calcular_desvios_projeto(self, projeto)`: Calcula desvios de orçamento e prazo
  - Método `_projetar_conclusao_projeto(self, projeto)`: Projeta data e custo de conclusão
  - Método `_gerar_alertas_projeto(self, analise, desvios)`: Gera alertas sobre o projeto
  - Método `_listar_projetos(self, propriedade)`: Lista todos os projetos (simulado)
  - Método `_listar_proximos_vencimentos(self, projetos)`: Lista projetos próximos do vencimento

## ia_analise_avancada.py
- Classe `IAAnalisePecuaria`:
  - Método `__init__(self, propriedade, inventario_atual)`: Sem descrição registrada.
  - Método `analisar_perfil_propriedade(self, respostas_questionario)`: Análise completa do perfil da propriedade
  - Método `_detectar_tipo_propriedade(self)`: Detecta o tipo de propriedade baseado no inventário
  - Método `_avaliar_nivel_tecnico(self)`: Avalia o nível técnico da propriedade
  - Método `_calcular_potencial_crescimento(self)`: Calcula o potencial de crescimento do rebanho
  - Método `_avaliar_viabilidade_economica(self)`: Avalia a viabilidade econômica da propriedade
  - Método `_identificar_riscos(self)`: Identifica riscos específicos da propriedade
  - Método `_identificar_oportunidades(self)`: Identifica oportunidades de melhoria
  - Método `gerar_estrategia_otimizada(self, perfil, respostas_questionario)`: Gera estratégia otimizada baseada na anális
  - Método `_calcular_proporcao_vacas(self)`: Calcula a proporção de vacas no rebanho
  - Método `_calcular_valor_medio_cabeca(self)`: Calcula o valor médio por cabeça
  - Método `_calcular_produtividade(self)`: Calcula a produtividade em UA/ha
  - Método `_calcular_receita_projetada(self)`: Calcula receita projetada anual
  - Método `_calcular_custos_projetados(self)`: Calcula custos projetados anuais
  - Método `_calcular_preco_medio_atual(self)`: Calcula preço médio atual do rebanho
  - Método `_classificar_crescimento(self, crescimento)`: Classifica o potencial de crescimento
  - Método `_calcular_score_geral(self, perfil, respostas)`: Calcula score geral da propriedade (0-100)
  - Método `_gerar_nome_estrategia(self, perfil)`: Gera nome da estratégia baseado no perfil
  - Método `_definir_objetivos(self, perfil)`: Define objetivos baseados no perfil
  - Método `_definir_acoes_imediata(self, perfil)`: Define ações imediatas (0-3 meses)
  - Método `_definir_acoes_curto_prazo(self, perfil)`: Define ações de curto prazo (3-12 meses)
  - Método `_definir_acoes_longo_prazo(self, perfil)`: Define ações de longo prazo (1-5 anos)
  - Método `_calcular_parametros_otimizados(self, perfil)`: Calcula parâmetros otimizados para a projeção
  - Método `_gerar_projecao_5_anos(self, perfil)`: Gera projeção para 5 anos
  - Método `_definir_indicadores_monitoramento(self, perfil)`: Define indicadores para monitoramento

## ia_avancada.py
- Classe `IAPecuariaAvancada`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `analisar_fazenda(self, inventario, parametros_usuario)`: Analisa a fazenda e retorna projeção inteligente
  - Método `obter_configuracao_otimizada(self, perfil_tipo)`: Retorna configuração otimizada baseada no perfil
  - Método `calcular_viabilidade_economica(self, inventario, perfil)`: Calcula viabilidade econômica da fazenda
  - Método `gerar_cenarios(self, inventario, perfil)`: Gera cenários otimista, realista e pessimista
  - Método `obter_benchmarking(self, perfil)`: Retorna benchmarking do setor para o perfil
  - Método `gerar_relatorio_completo(self, inventario, parametros_usuario)`: Gera relatório completo em HTML

## ia_cenarios_risco.py
- Classe `IACenariosRisco`:
  - Método `__init__(self, propriedade, inventario_atual, dados_regiao)`: Sem descrição registrada.
  - Método `analisar_cenarios_multiplos(self, estrategia_base)`: Analisa múltiplos cenários de risco

- Método `_simular_cenario(self, estrategia_base, dados_cenario, nome_cenario)`: Simula um cenário específico
- Método `_calcular_ano_cenario(self, estrategia_base, ano, fator_preco, fator_produtividade, fator_custos)`: Calcula dad
- Método `_calcular_metricas_cenario(self, projecao_anos)`: Calcula métricas consolidadas do cenário
- Método `_calcular_crescimento_anual(self, valor_inicial, valor_final, anos)`: Calcula crescimento anual composto
- Método `_calcular_volatilidade(self, valores)`: Calcula volatilidade (desvio padrão)
- Método `_calcular_var(self, valores, percentil)`: Calcula Value at Risk (VaR)
- Método `_calcular_score_risco(self, volatilidade, var, margem_lucro)`: Calcula score de risco (0-100, onde 100 = sem ri
- Método `_gerar_recomendacoes_cenario(self, nome_cenario, metricas)`: Gera recomendações específicas para o cena
- Método `gerar_plano_contingencia(self, cenarios)`: Gera plano de contingência baseado nos cenários
- Método `_identificar_alertas_risco(self, cenarios)`: Identifica alertas de risco baseados nos cenários
- Método `_definir_acoes_preventivas(self, cenarios)`: Define ações preventivas baseadas nos cenários
- Método `_definir_acoes_corretivas(self, cenarios)`: Define ações corretivas para situações de crise
- Método `_definir_indicadores_monitoramento(self)`: Define indicadores para monitoramento de risco
- Método `_definir_niveis_alerta(self)`: Define níveis de alerta e ações correspondentes
- Método `calcular_risco_portfolio(self, cenarios)`: Calcula risco do portfólio considerando todos os cenários
- Método `_classificar_risco_portfolio(self, sharpe_ratio, var_esperado)`: Classifica o risco do portfólio

## ia_compras_inteligentes.py
- Classe `IAComprasInteligentes`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `analisar_necessidade_compras(self, inventario_atual, perfil_fazenda, mes_atual)`: Analisa inventário e retorna
  - Método `detectar_oportunidades_mercado(self, preco_atual_categoria, mes_atual)`: Detecta oportunidades de compra
  - Método `calcular_investimento_necessario(self, sugestoes_compra)`: Calcula investimento total necessário para as co
  - Método `_calcular_prioridade_compra(self, categoria, deficit, quantidade_minima, mes_atual, perfil_fazenda)`: Calcula
  - Método `_calcular_melhor_momento_compra(self, categoria, mes_atual, dados_mercado)`: Calcula o melhor momento
  - Método `_calcular_meses_ate_proximo(self, mes_atual, meses_alvo)`: Calcula quantos meses faltam até o próximo mê
  - Método `_calcular_roi_esperado(self, categoria, preco_compra, perfil_fazenda)`: Calcula ROI esperado em 12 meses
  - Método `_avaliar_momento_sazonal(self, categoria, mes_atual)`: Avalia o momento sazonal para compra
  - Método `_calcular_score_oportunidade(self, desconto, momento_score, mes_atual)`: Calcula score final da oportunidad
  - Método `_gerar_justificativa_compra(self, categoria, deficit, prioridade, momento_compra)`: Gera justificativa para a co
  - Método `_gerar_recomendacao_timing(self, momento, meses_ate_melhor, economia_esperando)`: Gera recomendaçã
  - Método `_gerar_recomendacao_oportunidade(self, categoria, desconto, momento)`: Gera recomendação para oportuni

## ia_configuracao_automatica.py
- Classe `IAConfiguracaoAutomatica`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `analisar_inventario_e_configurar(self, inventario_data)`: Analisa o inventário e retorna configuração automática
  - Método `_calcular_configuracao_otimizada(self, inventario, perfil)`: Calcula configuração otimizada para rentabilidade e
  - Método `_gerar_vendas_automaticas(self, inventario, perfil)`: Gera configurações automáticas de vendas baseadas no
  - Método `_gerar_compras_automaticas(self, inventario, perfil)`: Gera configurações automáticas de compras baseadas
  - Método `_calcular_valor_unitario_venda(self, categoria, perfil)`: Calcula valor unitário de venda baseado na categoria e
  - Método `_calcular_valor_unitario_compra(self, categoria, perfil)`: Calcula valor unitário de compra baseado na categoria
  - Método `_calcular_projecao_rentabilidade(self, inventario, perfil)`: Calcula projeção de rentabilidade para 5 anos

  - Método `gerar_relatorio_configuracao(self, resultado_analise)`: Gera relatório HTML da configuração automática

## ia_evolucao_projecoes.py
- Classe `IAEvolucaoProjecoes`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `projetar_evolucao_completa(self, inventario_atual, parametros_atuais, anos_projecao, regiao, considerar_mel
  - Método `calcular_producao_estimada(self, inventario, parametros, tipo_producao)`: Calcula produção estimada (carne
  - Método `comparar_com_benchmark(self, metricas_propriedade, regiao)`: Compara métricas da propriedade com bench
  - Método `_analisar_situacao_atual(self, inventario, parametros, benchmark)`: Analisa situação atual do rebanho
  - Método `_projetar_ano(self, inventario_inicial, parametros, benchmark, ano)`: Projeta um ano completo
  - Método `_aplicar_melhorias_graduais(self, parametros, benchmark, ano)`: Aplica melhorias graduais aos parâmetros a
  - Método `_projetar_inventario_detalhado(self, inventario_inicial, parametros)`: Projeta inventário detalhado por categoria
  - Método `_calcular_producao_carne(self, inventario, parametros)`: Calcula produção estimada de carne (kg e @)
  - Método `_calcular_producao_leite(self, inventario, parametros)`: Calcula produção estimada de leite
  - Método `_consolidar_projecoes(self, analise_inicial, projecoes_anuais, benchmark)`: Consolida projeções de todos os a
  - Método `_gerar_recomendacoes_estrategicas(self, analise_inicial, projecoes, benchmark)`: Gera recomendações estra
  - Método `_calcular_potencial_melhoria(self, gap_natalidade, gap_mortalidade, total_animais)`: Calcula potencial de mel
  - Método `_avaliar_desempenho_ano(self, parametros, benchmark)`: Avalia desempenho comparado ao benchmark
  - Método `_calcular_score_geral(self, comparacoes)`: Calcula score geral de desempenho (0-100)
  - Método `_identificar_pontos_melhoria(self, comparacoes)`: Identifica principais pontos de melhoria

## ia_identificacao_fazendas.py
- Classe `SistemaIdentificacaoFazendas`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `identificar_perfil_fazenda(self, inventario, parametros)`: Identifica automaticamente o perfil da fazenda basead
  - Método `_analisar_composicao_inventario(self, inventario)`: Analisa a composição do inventário para identificar o perfil
  - Método `_analisar_parametros(self, parametros)`: Analisa os parâmetros de vendas e compras
  - Método `_detectar_perfil(self, analise_inventario, analise_parametros)`: Detecta o perfil da fazenda baseado na análise
  - Método `_gerar_estrategias_movimentacao(self, perfil, analise_inventario)`: Gera estratégias de movimentação basead
  - Método `_gerar_movimentacoes_automaticas(self, perfil, analise_inventario)`: Gera lista de movimentações automática
  - Método `calcular_valores_por_categoria(self, inventario)`: Calcula valores por categoria baseado no inventário atual
  - Método `_obter_valor_padrao_categoria(self, categoria)`: Retorna valor padrão por categoria
  - Método `gerar_relatorio_identificacao(self, resultado)`: Gera relatório HTML da identificação da fazenda

## ia_movimentacoes_automaticas.py

## ia_nascimentos_aprimorado.py
- Classe `IANascimentosAprimorada`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `gerar_nascimentos_inteligentes(self, propriedade, data_referencia, saldos_iniciais, parametros, perfil_fazenda
  - Método `_calcular_matrizes_disponiveis(self, saldos_iniciais)`: Calcula o número de matrizes disponíveis para reproduç
  - Método `_calcular_taxa_natalidade_mes(self, mes, parametros)`: Calcula taxa de natalidade baseada na sazonalidade
  - Método `_calcular_total_nascimentos(self, matrizes, taxa_natalidade, data_referencia)`: Calcula total de nascimentos e

  - Método `_calcular_fator_ambiental(self, mes)`: Calcula fator de ajuste ambiental por mês
  - Método `_distribuir_por_sexo(self, total_nascimentos)`: Distribui nascimentos entre machos e fêmeas
  - Método `_aplicar_mortalidade_neonatal(self, bezerros, bezerras)`: Aplica mortalidade neonatal (primeiros 30 dias)
  - Método `_gerar_observacao_nascimento(self, quantidade_sobrevivente, quantidade_morta, sexo, taxa_natalidade, me
  - Método `_registrar_mortalidade_neonatal(self, propriedade, data_referencia, categoria_bezerros, categoria_bezerras, r
  - Método `prever_nascimentos_proximo_ano(self, matrizes_atuais, parametros)`: Prevê nascimentos mês a mês para o
  - Método `calcular_capacidade_reproducao(self, inventario_atual)`: Calcula capacidade reprodutiva do rebanho

## ia_pecuaria_data.py
- Função `obter_dados_regiao(estado)`: Retorna dados da região baseado no estado
- Função `calcular_preco_sazonal(preco_base, mes)`: Calcula preço considerando sazonalidade
- Função `obter_benchmark_industria(metrica)`: Retorna benchmarks da indústria para uma métrica
- Função `obter_cenario_risco(cenario)`: Retorna dados de um cenário de risco

## ia_perfis_fazendas.py
- Função `detectar_perfil_fazenda(inventario, parametros_usuario)`: Detecta o perfil da fazenda baseado no inventário e p
- Função `calcular_projecao_inteligente(perfil, inventario, parametros_usuario, anos)`: Calcula projeção inteligente basead
- Função `gerar_recomendacoes_perfil(perfil, projecao)`: Gera recomendações específicas para o perfil da fazenda

## ia_transferencias_inteligentes.py
- Classe `IATransferenciasInteligentes`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `analisar_balanceamento_propriedades(self, produtor, incluir_recomendacoes)`: Analisa balanceamento entre
  - Método `calcular_transferencia_otimizada(self, propriedade_origem, propriedade_destino, categoria, quantidade, distan
  - Método `sugerir_transferencias_automaticas(self, produtor, mes_atual)`: Sugere transferências automáticas para balan
  - Método `_analisar_capacidade_propriedade(self, propriedade)`: Analisa capacidade de suporte e utilização de uma pro
  - Método `_identificar_desequilibrios(self, analise_propriedades)`: Identifica desequilíbrios que requerem ação
  - Método `_gerar_recomendacoes_transferencia(self, analise_propriedades, desequilibrios)`: Gera recomendações conc
  - Método `_verificar_viabilidade_transferencia(self, analise_origem, analise_destino, ua_transferencia)`: Verifica se a tran
  - Método `_calcular_custos_transferencia(self, quantidade, ua, distancia_km, categoria)`: Calcula todos os custos envolv
  - Método `_calcular_beneficios_transferencia(self, analise_origem, analise_destino, ua_transferencia, custo_transferenci
  - Método `_calcular_roi_transferencia(self, beneficios, custos)`: Calcula ROI da transferência
  - Método `_classificar_roi(self, roi_percentual)`: Classifica o ROI da transferência
  - Método `_gerar_recomendacao_transferencia(self, viabilidade, roi)`: Gera recomendação sobre a transferência
  - Método `_classificar_status_sistema(self, utilizacao_media)`: Classifica status geral do sistema
  - Método `_gerar_alertas_sistema(self, analise_propriedades, utilizacao_media)`: Gera alertas importantes sobre o siste
  - Método `_determinar_categoria_transferencia(self, inventario_origem, perfil_destino)`: Determina qual categoria é melh
  - Método `_selecionar_melhor_categoria_transferencia(self, inventario_origem, perfil_destino, ua_alvo)`: Seleciona melh
  - Método `_verificar_compatibilidade_categoria_perfil(self, categoria, perfil)`: Verifica se a categoria é compatível com o p
  - Método `_estimar_beneficio_transferencia(self, utilizacao_origem, utilizacao_destino, ua_transferidas)`: Estima benefíc
  - Método `_calcular_prioridade_transferencia(self, utilizacao_origem, ua_transferidas)`: Calcula prioridade da transferênc

## ia_vendas_otimizadas.py

- Classe `IAVendasOtimizadas`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `analisar_oportunidades_venda(self, inventario_atual, idade_media_categoria, peso_medio_categoria, mes_at
  - Método `calcular_receita_estimada(self, oportunidades_venda, percentual_venda)`: Calcula receita estimada das vend
  - Método `_calcular_ponto_ideal_venda(self, categoria, idade_atual, peso_atual, dados_venda)`: Calcula se o animal est
  - Método `_avaliar_momento_venda(self, categoria, mes_atual, dados_venda)`: Avalia se é bom momento sazonal para
  - Método `_prever_preco_futuro(self, categoria, mes_atual, dados_venda)`: Prevê preço para os próximos 3 meses
  - Método `_calcular_margem_lucro(self, categoria, peso_kg, preco_venda, perfil_fazenda)`: Calcula margem de lucro es
  - Método `_simular_cenarios_venda(self, quantidade, peso_kg, preco_atual, previsao_futura)`: Simula cenários: vender a
  - Método `_calcular_meses_ate_proximo(self, mes_atual, meses_alvo)`: Calcula quantos meses faltam até o próximo mê
  - Método `_gerar_recomendacao_ponto(self, score, idade_atual, idade_ideal, peso_atual, peso_ideal)`: Gera recomenda
  - Método `_gerar_recomendacao_timing_venda(self, momento, meses_ate_melhor, ganho_esperando)`: Gera recomend
  - Método `_gerar_recomendacao_venda(self, ponto_ideal, momento_venda, margem_lucro)`: Gera recomendação final
  - Método `_calcular_score_venda(self, score_ponto, score_momento, margem)`: Calcula score final de oportunidade de

## management/commands/carregar_categorias.py
- Classe `Command`:
  - Método `handle(self)`: Sem descrição registrada.

## management/commands/carregar_categorias_completo.py
- Classe `Command`:
  - Método `handle(self)`: Carrega as categorias padrão

## management/commands/carregar_categorias_padrao.py
- Classe `Command`:
  - Método `handle(self)`: Carrega as categorias padrão

## management/commands/criar_categorias_padrao_sistema.py
- Classe `Command`:
  - Método `add_arguments(self, parser)`: Sem descrição registrada.
  - Método `handle(self)`: Cria as categorias padrão do sistema

## management/commands/criar_dados_exemplo.py
- Classe `Command`:
  - Método `add_arguments(self, parser)`: Sem descrição registrada.
  - Método `handle(self)`: Sem descrição registrada.

## management/commands/gerar_animais_massivos.py
- Classe `Command`:
  - Método `add_arguments(self, parser)`: Sem descrição registrada.
  - Método `handle(self)`: Sem descrição registrada.

## management/commands/gerar_relatorios_pnib.py

- Classe `Command`:
  - Método `add_arguments(self, parser)`: Sem descrição registrada.
  - Método `handle(self)`: Sem descrição registrada.
  - Método `_gerar_identificacao(self, propriedade, destino, exec_time)`: Sem descrição registrada.
  - Método `_gerar_movimentacao(self, propriedade, destino, exec_time)`: Sem descrição registrada.
  - Método `_gerar_sanitario(self, propriedade, destino, exec_time)`: Sem descrição registrada.

## management/commands/popular_categorias.py
- Classe `Command`:
  - Método `handle(self)`: Sem descrição registrada.

## management/commands/popular_pesos_categorias.py
- Classe `Command`:
  - Método `handle(self)`: Sem descrição registrada.

## management/commands/populate_data.py
- Classe `Command`:
  - Método `handle(self)`: Sem descrição registrada.

## management/commands/remover_categorias_duplicadas.py
- Classe `Command`:
  - Método `add_arguments(self, parser)`: Sem descrição registrada.
  - Método `handle(self)`: Remove categorias duplicadas ou incorretas

## management/commands/seed_planejamento.py
- Função `_decimal(valor)`: Sem descrição registrada.
- Classe `Command`:
  - Método `add_arguments(self, parser)`: Sem descrição registrada.
  - Método `handle(self)`: Sem descrição registrada.
  - Método `_garantir_usuario(self, username)`: Sem descrição registrada.
  - Método `_garantir_produtor(self, usuario)`: Sem descrição registrada.
  - Método `_garantir_propriedade(self, produtor)`: Sem descrição registrada.
  - Método `_garantir_categorias(self)`: Sem descrição registrada.
  - Método `_garantir_inventario(self, propriedade, categorias, ano)`: Sem descrição registrada.
  - Método `_garantir_parametros_projecao(self, propriedade)`: Sem descrição registrada.
  - Método `_garantir_planejamento(self, propriedade, ano)`: Sem descrição registrada.
  - Método `_garantir_cenarios(self, planejamento)`: Sem descrição registrada.
  - Método `_garantir_atividades(self, planejamento, categorias, ano)`: Sem descrição registrada.
  - Método `_garantir_metas_comerciais(self, planejamento, categorias)`: Sem descrição registrada.
  - Método `_garantir_metas_financeiras(self, planejamento)`: Sem descrição registrada.
  - Método `_garantir_indicadores(self, planejamento)`: Sem descrição registrada.
  - Método `_garantir_movimentacoes(self, propriedade, planejamento, cenario_baseline)`: Sem descrição registrada.
  - Método `_garantir_financeiro(self, propriedade, ano)`: Sem descrição registrada.

## management/commands/testar_inventario.py
- Classe `Command`:
  - Método `handle(self)`: Sem descrição registrada.

## management/commands/testar_promocao.py
- Classe `Command`:
  - Método `handle(self)`: Sem descrição registrada.

## management/commands/testar_vendas_corretas.py
- Classe `Command`:
  - Método `handle(self)`: Sem descrição registrada.

## management/commands/verificar_categorias.py
- Classe `Command`:
  - Método `handle(self)`: Sem descrição registrada.
  - Método `sugerir_peso(self, nome_categoria)`: Sugere peso médio baseado no nome da categoria

## management/commands/verificar_promocoes.py
- Classe `Command`:
  - Método `handle(self)`: Sem descrição registrada.

## migrations/0022_financeiro_dashboard_personalizacao.py
- Função `create_default_widgets(apps, schema_editor)`: Sem descrição registrada.
- Função `delete_default_widgets(apps, schema_editor)`: Sem descrição registrada.

## migrations/0025_expand_rastreabilidade_fields.py
- Função `preencher_campos_animais(apps, schema_editor)`: Sem descrição registrada.
- Função `desfazer_preenchimento(apps, schema_editor)`: Sem descrição registrada.

## migrations/0026_propriedade_ciclos_multiplos.py
- Função `preparar_ciclos_para_json(apps, schema_editor)`: Sem descrição registrada.
- Função `normalizar_ciclos_json(apps, schema_editor)`: Sem descrição registrada.

## migrations/0039_adiciona_numero_requisicao.py
- Função `preencher_numeros(apps, schema_editor)`: Sem descrição registrada.
- Função `desfazer_numeros(apps, schema_editor)`: Sem descrição registrada.

## models.py
- Classe `ProdutorRural`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `idade(self)`: Calcula a idade do produtor baseada na data de nascimento
- Classe `PlanoAssinatura`:

  - Método `__str__(self)`: Sem descrição registrada.
- Classe `AssinaturaCliente`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `ativa(self)`: Sem descrição registrada.
  - Método `atualizar_status(self, status)`: Sem descrição registrada.
  - Método `alias_tenant(self)`: Sem descrição registrada.
- Classe `TenantWorkspace`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `caminho_path(self)`: Sem descrição registrada.
  - Método `marcar_erro(self, mensagem)`: Sem descrição registrada.
- Classe `Propriedade`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `ciclos_pecuarios_list(self)`: Retorna a lista normalizada de códigos de ciclo pecuário.
  - Método `get_ciclos_pecuarios_display(self)`: Retorna a descrição legível dos ciclos pecuários selecionados.
  - Método `get_tipo_ciclo_pecuario_display(self)`: Compatibilidade com templates que utilizam o método padrão do Djang
  - Método `valor_total_propriedade(self)`: Calcula o valor total da propriedade se for própria
  - Método `valor_mensal_total_arrendamento(self)`: Calcula o valor mensal total do arrendamento
  - Método `save(self)`: Sem descrição registrada.
- Classe `CategoriaAnimal`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `InventarioRebanho`:
  - Método `valor_total(self)`: Calcula o valor total da categoria
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `PlanejamentoAnual`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `AtividadePlanejada`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `MetaComercialPlanejada`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `MetaFinanceiraPlanejada`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `IndicadorPlanejado`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `CenarioPlanejamento`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `PoliticaVendasCategoria`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ParametrosProjecaoRebanho`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ParametrosVendaPorCategoria`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `MovimentacaoProjetada`:
  - Método `__str__(self)`: Sem descrição registrada.

- Classe `Cultura`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `RegraPromocaoCategoria`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `CicloProducaoAgricola`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `producao_total_esperada_sc(self)`: Calcula a produção total esperada em sacas
  - Método `receita_esperada_total(self)`: Calcula a receita esperada total
  - Método `custo_total_producao(self)`: Calcula o custo total de produção
  - Método `lucro_esperado(self)`: Calcula o lucro esperado
- Classe `TransferenciaPropriedade`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `propriedade_relacionada(self)`: Retorna a propriedade relacionada baseada no tipo de transferência
- Classe `ConfiguracaoVenda`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `CustoFixo`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `custo_anual(self)`: Sem descrição registrada.
  - Método `get_meses_nomes(self)`: Retorna os nomes dos meses aplicáveis
  - Método `get_custo_por_mes(self, mes)`: Retorna o custo para um mês específico
- Classe `CustoVariavel`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `impacto_total(self)`: Sem descrição registrada.
  - Método `custo_anual_por_cabeca(self)`: Calcula o custo anual por cabeça baseado no período
  - Método `get_meses_nomes(self)`: Retorna os nomes dos meses aplicáveis
  - Método `get_custo_por_mes(self, mes)`: Retorna o custo para um mês específico
- Classe `CategoriaImobilizado`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `BemImobilizado`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `valor_depreciavel(self)`: Valor depreciável (aquisição - residual)
  - Método `depreciacao_mensal(self)`: Depreciação mensal
  - Método `depreciacao_acumulada(self)`: Depreciação acumulada até hoje
  - Método `valor_atual(self)`: Valor atual (aquisição - depreciacao_acumulada)
- Classe `TipoFinanciamento`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `Financiamento`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `IndicadorFinanceiro`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `FluxoCaixa`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `calcular_margem_lucro(self)`: Calcula a margem de lucro baseada na receita total

- Classe `SCRBancoCentral`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `DividaBanco`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ContratoDivida`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `AmortizacaoContrato`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ProjetoBancario`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `DocumentoProjeto`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `AnimalIndividual`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `idade_meses(self)`: Calcula a idade do animal em meses
  - Método `idade_anos(self)`: Calcula a idade do animal em anos
- Classe `MovimentacaoIndividual`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `AnimalPesagem`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `AnimalVacinaAplicada`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `AnimalTratamento`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `AnimalReproducaoEvento`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `AnimalHistoricoEvento`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `AnimalDocumento`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `BrincoAnimal`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `CurralSessao`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `eventos_total(self)`: Sem descrição registrada.
  - Método `animais_manejados(self)`: Sem descrição registrada.
- Classe `CurralLote`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `CurralEvento`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.

## models_compras.py

- Classe `Fornecedor`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `CategoriaInsumo`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `Insumo`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `EstoqueInsumo`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `estoque_baixo(self)`: Verifica se estoque está abaixo do mínimo
  - Método `percentual_estoque(self)`: Calcula percentual de estoque em relação ao máximo
- Classe `OrdemCompra`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ItemOrdemCompra`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `MovimentacaoEstoque`:
  - Método `__str__(self)`: Sem descrição registrada.

## models_compras_financeiro.py
- Classe `Fornecedor`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `SetorPropriedade`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `OrcamentoCompraMensal`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `total_limite(self)`: Sem descrição registrada.
  - Método `valor_utilizado(self, ignorar_ordem)`: Sem descrição registrada.
  - Método `saldo_disponivel(self, ignorar_ordem)`: Sem descrição registrada.
  - Método `percentual_utilizado(self, ignorar_ordem)`: Sem descrição registrada.
  - Método `excede_limite(self, valor, ignorar_ordem)`: Sem descrição registrada.
- Classe `AjusteOrcamentoCompra`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ConviteCotacaoFornecedor`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
  - Método `expirado(self)`: Sem descrição registrada.
  - Método `pode_responder(self)`: Sem descrição registrada.
  - Método `marcar_enviado(self, usuario)`: Sem descrição registrada.
  - Método `marcar_respondido(self, observacao)`: Sem descrição registrada.
  - Método `cancelar(self)`: Sem descrição registrada.
- Classe `NotaFiscal`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `ItemNotaFiscal`:

  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `RequisicaoCompra`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `total_estimado(self)`: Sem descrição registrada.
  - Método `gerar_proximo_numero(cls, propriedade)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
  - Método `centro_custo_display(self)`: Sem descrição registrada.
  - Método `plano_conta_display(self)`: Sem descrição registrada.
  - Método `setor_display(self)`: Sem descrição registrada.
- Classe `ItemRequisicaoCompra`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `valor_estimado_total(self)`: Sem descrição registrada.
- Classe `AprovacaoRequisicaoCompra`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `CotacaoFornecedor`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ItemCotacaoFornecedor`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `RecebimentoCompra`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ItemRecebimentoCompra`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `EventoFluxoCompra`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `OrdemCompra`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
  - Método `setor_autorizado(self)`: Sem descrição registrada.
  - Método `centro_custo_display(self)`: Sem descrição registrada.
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
  - Método `setor_autorizado(self)`: Sem descrição registrada.
  - Método `centro_custo_display(self)`: Sem descrição registrada.
- Classe `ItemOrdemCompra`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `ContaPagar`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `ContaReceber`:
  - Método `__str__(self)`: Sem descrição registrada.

  - Método `save(self)`: Sem descrição registrada.

## models_controles_operacionais.py
- Classe `TipoDistribuicao`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `DistribuicaoPasto`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `Cocho`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ControleCocho`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `ArquivoKML`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `Pastagem`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `calcular_area_do_kml(self)`: Calcula área do polígono KML em hectares
- Classe `RotacaoPastagem`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `MonitoramentoPastagem`:
  - Método `__str__(self)`: Sem descrição registrada.

## models_financeiro.py
- Função `default_layout_config()`: Retorna layout padrão do dashboard financeiro.
- Função `default_filters_config()`: Retorna as seleções padrão de filtros do dashboard financeiro.
- Função `default_comparativo_config()`: Configuração inicial para o modo comparativo do dashboard.
- Classe `CategoriaFinanceira`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `CentroCusto`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `PlanoConta`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ContaFinanceira`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `LancamentoFinanceiroQuerySet`:
  - Método `receitas(self)`: Sem descrição registrada.
  - Método `despesas(self)`: Sem descrição registrada.
  - Método `transferencias(self)`: Sem descrição registrada.
  - Método `pendentes(self)`: Sem descrição registrada.
  - Método `quitados(self)`: Sem descrição registrada.
  - Método `atrasados(self)`: Sem descrição registrada.

- Classe `LancamentoFinanceiro`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `marcar_como_quitado(self, data)`: Atualiza o status e a data de quitação.
  - Método `cancelar(self, motivo)`: Cancela o lançamento registrando motivo (opcional).
  - Método `atualizar_tipo_por_categoria(self)`: Garante coerência do tipo com a categoria selecionada.
  - Método `clean(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `AnexoLancamentoFinanceiro`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `MovimentoFinanceiro`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `clean(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.

## models_funcionarios.py
- Classe `Funcionario`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `ativo(self)`: Sem descrição registrada.
- Classe `PontoFuncionario`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `FolhaPagamento`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `Holerite`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `DescontoFuncionario`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `CalculadoraImpostos`:
  - Método `calcular_inss(salario_base)`: Calcula desconto INSS
  - Método `calcular_irrf(salario_base, dependentes)`: Calcula desconto IRRF
  - Método `calcular_fgts(salario_base)`: Calcula FGTS (8% sobre salário)

## models_iatf_completo.py
- Classe `ProtocoloIATF`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `duracao_dias(self)`: Duração total do protocolo em dias
- Classe `TouroSemen`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `LoteSemen`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `LoteIATF`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.

  - Método `gerar_etapas_padrao(self, user_padrao)`: Gera etapas padrão (D0, D8, D10) caso ainda não existam registros
- Classe `EtapaLoteIATF`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `esta_atrasada(self)`: Sem descrição registrada.
- Classe `IATFIndividual`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
  - Método `dias_ate_diagnostico(self)`: Dias até o diagnóstico de prenhez
  - Método `custo_por_prenhez(self)`: Custo por prenhez (se confirmada)
- Classe `AplicacaoMedicamentoIATF`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `CalendarioIATF`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `calcular_numero_lotes(self)`: Calcula número de lotes possíveis no período
  - Método `save(self)`: Sem descrição registrada.

## models_manejo.py
- Classe `ManejoTipo`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `Manejo`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `atrasado(self)`: Sem descrição registrada.
  - Método `registrar_transicao(self, novo_status, usuario, observacao)`: Atualiza o status e cria histórico da transição.
- Classe `ManejoHistorico`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ManejoChecklistItem`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ManejoChecklistExecucao`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `concluir(self, usuario, observacao)`: Sem descrição registrada.

## models_operacional.py
- Classe `TanqueCombustivel`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `AbastecimentoCombustivel`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `ConsumoCombustivel`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `EstoqueSuplementacao`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `CompraSuplementacao`:

  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `DistribuicaoSuplementacao`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `Empreiteiro`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ServicoEmpreiteiro`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `TipoEquipamento`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `Equipamento`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `ManutencaoEquipamento`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.

## models_patrimonio.py
- Classe `TipoBem`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `BemPatrimonial`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `valor_atual(self)`: Calcula valor atual com depreciação
  - Método `depreciacao_acumulada(self)`: Calcula depreciação acumulada
  - Método `percentual_depreciacao(self)`: Percentual de depreciação

## models_projetos.py
- Classe `Projeto`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `percentual_gasto(self)`: Percentual do orçamento já gasto
  - Método `saldo_orcamento(self)`: Saldo restante do orçamento
  - Método `dias_restantes(self)`: Dias até a conclusão prevista
- Classe `EtapaProjeto`:
  - Método `__str__(self)`: Sem descrição registrada.

## models_reproducao.py
- Classe `Touro`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `calcular_taxa_prenhez(self)`: Calcula taxa de prenhez
  - Método `save(self)`: Sem descrição registrada.
- Classe `EstacaoMonta`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `calcular_duracao(self)`: Calcula duração em dias

  - Método `calcular_taxa_prenhez_real(self)`: Calcula taxa de prenhez real
  - Método `save(self)`: Sem descrição registrada.
- Classe `IATF`:
  - Método `__str__(self)`: Sem descrição registrada.
  - Método `save(self)`: Sem descrição registrada.
- Classe `MontaNatural`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `Nascimento`:
  - Método `__str__(self)`: Sem descrição registrada.
- Classe `CalendarioReprodutivo`:
  - Método `__str__(self)`: Sem descrição registrada.

## relatorios_avancados.py
- Classe `SistemaRelatoriosAvancados`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `gerar_relatorio_mensal_pdf(self, propriedade, mes, ano, dados_rebanho, dados_financeiros, dados_ia)`: Gera
  - Método `gerar_relatorio_anual_excel(self, propriedade, ano, dados_anuais)`: Gera relatório anual completo em Excel c
  - Método `_criar_aba_resumo(self, ws, propriedade, ano, dados)`: Cria aba de resumo executivo
  - Método `_criar_aba_evolucao(self, ws, evolucao_mensal)`: Cria aba de evolução mensal com gráfico
  - Método `_criar_aba_inventario(self, ws, inventario)`: Cria aba de inventário detalhado
  - Método `_criar_aba_movimentacoes(self, ws, movimentacoes)`: Cria aba de movimentações
  - Método `_criar_aba_financeira(self, ws, financeiro)`: Cria aba de análise financeira
  - Método `gerar_relatorio_projecao_5anos_pdf(self, propriedade, projecoes)`: Gera relatório de projeção para 5 anos

## relatorios_iatf.py
- Função `_parse_date(value)`: Sem descrição registrada.
- Função `_to_decimal(value)`: Sem descrição registrada.
- Função `_taxa(parte, total)`: Sem descrição registrada.
- Função `_media(valor, quantidade)`: Sem descrição registrada.
- Função `_timedelta_para_dias(valor)`: Sem descrição registrada.
- Função `_resolver_rotulo_usuario(item)`: Sem descrição registrada.
- Função `_resolver_rotulo_touro(item)`: Sem descrição registrada.
- Função `_resolver_rotulo_generico(item, campo, padrao)`: Sem descrição registrada.
- Função `_agrupar_por(qs, campos, rotulo_callback)`: Sem descrição registrada.
- Função `coletar_dados_iatf(propriedade, filtros)`: Retorna uma estrutura consolidada com dados e indicadores da IATF.

## scr_parser.py
- Classe `SCRParser`:
  - Método `__init__(self)`: Sem descrição registrada.
  - Método `extrair_dados_pdf(self, arquivo_pdf)`: Extrai dados do PDF do SCR
  - Método `_extrair_texto_pdfplumber(self, arquivo_pdf)`: Extrai texto usando pdfplumber (mais preciso para tabelas)
  - Método `_extrair_texto_pypdf2(self, arquivo_pdf)`: Fallback: extrai texto usando PyPDF2
  - Método `_processar_texto(self, texto)`: Processa o texto extraído para identificar dados

  - Método `_extrair_informacoes_basicas(self, linhas)`: Extrai informações básicas do SCR
  - Método `_extrair_dividas_por_banco(self, linhas)`: Extrai dívidas organizadas por banco
  - Método `_extrair_info_divida(self, linhas, inicio_idx, banco_nome)`: Extrai informações de uma dívida específica
  - Método `_converter_valor(self, valor_str)`: Converte string de valor para Decimal
  - Método `_calcular_resumo_total(self)`: Calcula resumo total das dívidas
  - Método `gerar_relatorio_extracao(self)`: Gera relatório da extração realizada
- Classe `SCRProcessor`:
  - Método `__init__(self, scr_obj, dados_extraidos)`: Sem descrição registrada.
  - Método `processar_e_salvar(self)`: Processa os dados extraídos e salva no banco de dados
  - Método `gerar_relatorio_extracao(self)`: Gera relatório da extração

## services/notificacoes.py
- Função `_remetente_padrao()`: Sem descrição registrada.
- Função `enviar_notificacao_compra(assunto, mensagem, destinatarios)`: Envia e-mail de notificação para eventos do m
- Função `_destinatarios_alerta_assinatura()`: Sem descrição registrada.
- Função `notificar_evento_assinatura(assinatura, assunto, mensagem)`: Notifica o time interno sobre eventos críticos de

## services/planejamento.py
- Função `_categoria_tem_keywords(nome_categoria, keywords)`: Sem descrição registrada.
- Função `_estimar_peso_categoria(categoria)`: Sem descrição registrada.
- Função `_calcular_valor_movimentacao_planejada(mov)`: Sem descrição registrada.
- Função `_obter_categoria_movimentacao(mov)`: Sem descrição registrada.
- Função `_decimal_or(valor)`: Sem descrição registrada.
- Classe `FinanceiroResumo`:
  - Método `planejado_resultado(self)`: Sem descrição registrada.
  - Método `realizado_resultado(self)`: Sem descrição registrada.
- Classe `PlanejamentoAnalyzer`:
  - Método `__init__(self, propriedade, planejamento, cenario, ano)`: Sem descrição registrada.
  - Método `inventario_resumo(self)`: Sem descrição registrada.
  - Método `_init_slot_mes(self, chave_mes)`: Sem descrição registrada.
  - Método `movimentacoes_planejadas(self)`: Sem descrição registrada.
  - Método `_movimento_relevante_para_propriedade(self, mov)`: Sem descrição registrada.
  - Método `movimentacoes_realizadas(self)`: Sem descrição registrada.
  - Método `financeiro_resumo(self, metas_financeiras, metas_comerciais)`: Sem descrição registrada.
  - Método `indicadores_realizados(self, indicadores)`: Sem descrição registrada.
  - Método `obter_meses_ordenados(self)`: Retorna a lista de meses consolidados (planejado + realizado) ordenados.

## services/provisionamento.py
- Função `_registrar_database(alias, caminho)`: Sem descrição registrada.
- Função `_obter_ou_criar_workspace(assinatura)`: Sem descrição registrada.
- Função `provisionar_workspace(assinatura)`: Sem descrição registrada.
- Função `registrar_workspaces_existentes()`: Recarrega workspaces ativos na inicialização do Django.

## services/stripe_client.py
- Função `_configurar_stripe()`: Define a chave secreta da Stripe se ainda não estiver configurada.
- Função `criar_checkout_session(assinatura, plano, success_url, cancel_url)`: Cria uma sessão de checkout na Stripe pa
- Função `anexar_customer_a_assinatura(assinatura, customer_id)`: Atualiza a assinatura com o identificador do cliente r
- Função `atualizar_assinatura_por_evento(assinatura, subscription)`: Atualiza os campos da assinatura a partir de um ob
- Função `confirmar_checkout_session(session)`: Confirma uma sessão de checkout concluída.
- Função `construir_evento_webhook(payload, assinatura_header)`: Constrói e valida o evento recebido via webhook.

## services_financeiro.py
- Classe `PeriodoFinanceiro`:
  - Método `range_lookup(self)`: Sem descrição registrada.
- Função `periodo_mes_atual()`: Sem descrição registrada.
- Função `calcular_totais_lancamentos(propriedade, periodo)`: Retorna totais de receitas, despesas e transferências no p
- Função `listar_pendencias(propriedade, limite)`: Retorna listas de lançamentos pendentes e atrasados.
- Função `calcular_saldos_contas(propriedade)`: Calcula o saldo atual de cada conta financeira considerando lançamento
- Função `gerar_series_temporais(propriedade, periodo)`: Gera dados simplificados para gráficos (entradas x saídas por d

## templatetags/formatacao_br.py
- Função `moeda_br(valor)`: Formata valor como moeda brasileira: R$ 1.000,00
- Função `numero_br(valor, casas_decimais)`: Formata número no padrão brasileiro: 1.000 ou 1.152,38
- Função `percentual_br(valor, casas_decimais)`: Formata percentual no padrão brasileiro: 23,5%
- Função `numero_abreviado(valor)`: Abrevia números grandes: 1.5k, 2.3M
- Função `moeda_com_classe(valor, mostrar_positivo)`: Formata moeda com classe CSS para positivo/negativo
- Função `variacao_percentual(valor_atual, valor_anterior)`: Calcula e formata variação percentual com cor
- Função `dict_get(mapping, key)`: Permite acessar itens de dicionários nos templates.

## templatetags/formato_numeros.py
- Função `formato_br(numero)`: Formata número no padrão brasileiro: 1.234.567,89
- Função `formato_monetario(numero)`: Formata valor monetário: R$ 1.234.567,89
- Função `formato_numero_inteiro(numero)`: Formata número inteiro: 1.234
- Função `formato_decimal(numero)`: Formata decimal: 1.234,56

## utils_forms.py
- Função `formatar_mensagem_erro_form(form)`: Formata os erros de um formulário Django em uma string legível.

## utils_kml.py
- Função `parse_kml_file(kml_file)`: Parse de arquivo KML e extrai informações
- Função `calcular_area_poligono_kml(coordenadas)`: Calcula área de um polígono KML em hectares
- Função `extrair_coordenadas_centro(pastagem_data)`: Extrai coordenadas do centro do polígono
- Função `validar_kml(kml_file)`: Valida se arquivo é um KML válido

## utils_pecuaria.py
- Função `obter_presets_parametros(tipo_ciclo)`: Retorna parâmetros padrão baseado no tipo de ciclo da propriedade

- Função `aplicar_presets_parametros(parametros, tipo_ciclo)`: Aplica presets de parâmetros baseado no tipo de ciclo
- Função `calcular_resumo_projecao(propriedade_id)`: Calcula resumo da projeção para visualização
- Função `gerar_series_tempo(movimentacoes, anos)`: Gera séries temporais para gráficos

## views.py
- Função `landing_page(request)`: Página pública do sistema antes do login.
- Função `login_view(request)`: View para login do usuário
- Função `logout_view(request)`: View para logout do usuário
- Função `dashboard(request)`: Dashboard principal - lista de produtores
- Função `produtor_novo(request)`: Cadastro de novo produtor rural
- Função `produtor_editar(request, produtor_id)`: Edição de produtor rural
- Função `produtor_excluir(request, produtor_id)`: Exclusão de produtor rural
- Função `propriedades_lista(request, produtor_id)`: Lista de propriedades de um produtor
- Função `propriedade_nova(request, produtor_id)`: Cadastro de nova propriedade
- Função `propriedade_editar(request, propriedade_id)`: Edição de propriedade
- Função `propriedade_excluir(request, propriedade_id)`: Exclusão de propriedade
- Função `pecuaria_dashboard(request, propriedade_id)`: Dashboard do módulo pecuária
- Função `pecuaria_inventario(request, propriedade_id)`: Gerenciamento do inventário inicial
- Função `pecuaria_parametros_avancados(request, propriedade_id)`: Configurações avançadas de vendas e reposição
- Função `pecuaria_parametros(request, propriedade_id)`: Configuração dos parâmetros de projeção com IA Avançada
- Função `pecuaria_projecao(request, propriedade_id)`: Visualização e geração da projeção
- Função `pecuaria_inventario_dados(request, propriedade_id)`: View para retornar dados do inventário em JSON para a
- Função `gerar_projecao(propriedade, anos)`: Função para gerar a projeção do rebanho com IA Inteligente
- Função `agricultura_dashboard(request, propriedade_id)`: Dashboard do módulo agricultura
- Função `agricultura_ciclo_novo(request, propriedade_id)`: Cadastro de novo ciclo de produção agrícola
- Função `relatorio_final(request, propriedade_id)`: Relatório final para análise bancária
- Função `gerar_resumo_projecao_tabela(movimentacoes, periodicidade)`: Gera resumo da projeção em formato de tabe
- Função `gerar_evolucao_categorias_tabela(movimentacoes, inventario_inicial)`: Gera evolução das categorias em forma
- Função `gerar_evolucao_detalhada_rebanho(movimentacoes, inventario_inicial)`: Gera evolução detalhada do rebanho
- Função `obter_parametros_padrao_ciclo(tipo_ciclo)`: Retorna parâmetros padrão baseados no tipo de ciclo pecuário
- Função `aplicar_parametros_ciclo(propriedade, parametros)`: Aplica parâmetros específicos baseados no tipo de ciclo d
- Função `transferencias_lista(request)`: Lista todas as transferências do usuário
- Função `transferencia_nova(request)`: Criar nova transferência entre propriedades
- Função `transferencia_editar(request, transferencia_id)`: Editar transferência existente
- Função `transferencia_excluir(request, transferencia_id)`: Excluir transferência
- Função `gerar_resumo_projecao_por_ano(movimentacoes, inventario_inicial)`: Gera resumo da projeção organizado po
- Função `categorias_lista(request)`: Lista todas as categorias de animais
- Função `categoria_nova(request)`: Cria uma nova categoria de animal
- Função `categoria_editar(request, categoria_id)`: Edita uma categoria existente
- Função `categoria_excluir(request, categoria_id)`: Exclui uma categoria
- Função `obter_saldo_atual_propriedade(propriedade, data_referencia)`: Obtém o saldo atual de uma propriedade em um
- Função `obter_valor_padrao_por_categoria(categoria)`: Retorna valores padrão por categoria de animal
- Função `processar_compras_configuradas(propriedade, data_referencia, fator_inflacao)`: Processa compras configurad

- Função `verificar_momento_compra(config, data_referencia)`: Verifica se é o momento correto para realizar uma compra
- Função `processar_transferencias_configuradas(propriedade_destino, data_referencia)`: Processa transferências config
- Função `verificar_momento_transferencia(config, data_referencia)`: Verifica se é o momento de processar uma transferê
- Função `testar_transferencias(request, propriedade_id)`: View para testar o sistema de transferências
- Função `obter_saldo_fazenda_ajax(request, fazenda_id, categoria_id)`: AJAX endpoint para obter saldo atual de uma fa
- Função `buscar_saldo_inventario(request, propriedade_id, categoria_id)`: View para buscar saldo do inventário de uma
- Função `preparar_dados_graficos(movimentacoes, resumo_por_ano)`: Prepara dados formatados para gráficos Chart.js
- Função `importar_scr(request, propriedade_id)`: Importar SCR do Banco Central
- Função `reprocessar_scr(request, propriedade_id, scr_id)`: Reprocessar SCR que falhou
- Função `distribuir_dividas_por_fazenda(request, propriedade_id, scr_id)`: Distribuir dívidas do SCR para fazendas espec
- Função `gerar_amortizacao_contrato(contrato)`: Gera tabela de amortização para um contrato
- Função `dividas_amortizacao(request, propriedade_id)`: Amortização de contratos
- Função `projeto_bancario_dashboard(request, propriedade_id)`: Dashboard do módulo Projeto Bancário
- Função `projeto_bancario_novo(request, propriedade_id)`: Criar novo projeto bancário
- Função `projeto_bancario_detalhes(request, propriedade_id, projeto_id)`: Detalhes do projeto bancário
- Função `projeto_bancario_editar(request, propriedade_id, projeto_id)`: Editar projeto bancário
- Função `dividas_contratos(request, propriedade_id)`: Lista todos os contratos de dívida de uma propriedade
- Função `api_valor_inventario(request, propriedade_id, categoria_id)`: API para buscar valor por cabeça do inventário de
- Função `dividas_dashboard(request, propriedade_id)`: Dashboard de dívidas financeiras
- Função `projeto_bancario_dashboard(request, propriedade_id)`: Dashboard de projetos bancários
- Função `propriedade_modulos(request, propriedade_id)`: Exibe os módulos disponíveis para uma propriedade

## views_agricultura.py
- Função `agricultura_dashboard(request, propriedade_id)`: Dashboard completo do módulo de agricultura
- Função `agricultura_ciclo_novo(request, propriedade_id)`: Criar novo ciclo de produção
- Função `agricultura_ciclo_editar(request, propriedade_id, ciclo_id)`: Editar ciclo de produção
- Função `agricultura_ciclo_lista(request, propriedade_id)`: Lista todos os ciclos de produção
- Função `agricultura_ciclo_excluir(request, propriedade_id, ciclo_id)`: Excluir ciclo de produção
- Função `agricultura_analise(request, propriedade_id)`: Análise detalhada de agricultura

## views_analise.py
- Função `analise_dashboard(request, propriedade_id)`: Dashboard do módulo de análise
- Função `indicadores_lista(request, propriedade_id)`: Lista todos os indicadores da propriedade
- Função `indicador_novo(request, propriedade_id)`: Adiciona novo indicador
- Função `indicador_editar(request, propriedade_id, indicador_id)`: Edita indicador existente
- Função `calcular_indicadores_automaticos(request, propriedade_id)`: Calcula indicadores automaticamente baseado no
- Função `relatorio_analise(request, propriedade_id)`: Gera relatório de análise financeira
- Função `calcular_indicadores_basicos(propriedade)`: Calcula indicadores financeiros básicos

## views_assinaturas.py
- Função `assinaturas_dashboard(request)`: Sem descrição registrada.
- Função `iniciar_checkout(request, plano_slug)`: Sem descrição registrada.
- Função `checkout_sucesso(request)`: Sem descrição registrada.

- Função `checkout_cancelado(request)`: Sem descrição registrada.
- Função `stripe_webhook(request)`: Sem descrição registrada.
- Função `_handle_checkout_completed(dados)`: Sem descrição registrada.
- Função `_handle_subscription_event(dados)`: Sem descrição registrada.
- Função `_handle_subscription_deleted(dados)`: Sem descrição registrada.
- Função `_handle_invoice_failed(dados)`: Sem descrição registrada.

## views_capacidade_pagamento.py
- Função `capacidade_pagamento_dashboard(request, propriedade_id)`: Dashboard do módulo de capacidade de pagam
- Função `calcular_capacidade_pagamento(propriedade)`: Calcula indicadores de capacidade de pagamento
- Função `analisar_fluxo_caixa(propriedade)`: Analisa o fluxo de caixa da propriedade
- Função `gerar_cenarios_stress(propriedade)`: Gera cenários de stress para análise de capacidade
- Função `gerar_recomendacoes(propriedade, indicadores)`: Gera recomendações baseadas nos indicadores

## views_cenarios.py
- Função `analise_cenarios(request, propriedade_id)`: Análise de múltiplos cenários de projeção
- Função `gerar_cenario(propriedade, parametros, anos, nome_cenario, fator_venda, fator_custo)`: Gera um cenário espe
- Função `buscar_cenario(propriedade, nome_cenario)`: Busca movimentações de um cenário específico
- Função `calcular_total_receitas(movimentacoes)`: Calcula total de receitas de um cenário
- Função `calcular_total_custos(movimentacoes)`: Calcula total de custos de um cenário
- Função `calcular_total_animais(movimentacoes)`: Calcula total de animais em um cenário
- Função `preparar_comparacao_cenarios(cenarios)`: Prepara dados para comparação entre cenários

## views_compras.py
- Função `gerar_conta_pagar_para_ordem(ordem)`: Cria ou atualiza uma conta a pagar vinculada à ordem de compra.
- Função `obter_historico_preco_item(ordem, item)`: Retorna informações da última compra (mesmo fornecedor e geral)
- Função `_buscar_orcamento(propriedade, setor, data_referencia)`: Sem descrição registrada.
- Função `montar_contexto_orcamento(propriedade, setor, data_referencia, ignorar_ordem)`: Sem descrição registrada.
- Função `validar_orcamento_para_valor(propriedade, setor, data_emissao, valor, ignorar_ordem)`: Sem descrição regist
- Função `compras_dashboard(request, propriedade_id)`: Dashboard consolidado de Compras
- Função `requisicoes_compra_lista(request, propriedade_id)`: Lista de requisições de compra por fazenda
- Função `requisicao_compra_nova(request, propriedade_id)`: Cadastro de nova requisição (funcionário da fazenda)
- Função `requisicao_compra_detalhes(request, propriedade_id, requisicao_id)`: Detalhes e linha do tempo da requisição
- Função `setores_compra_lista(request, propriedade_id)`: Listagem de setores responsáveis por autorizações de compra
- Função `setor_compra_novo(request, propriedade_id)`: Cadastro de novo setor de compras
- Função `setor_compra_editar(request, propriedade_id, setor_id)`: Edição de setor de compras
- Função `setor_compra_alterar_status(request, propriedade_id, setor_id)`: Ativa ou inativa um setor
- Função `convites_cotacao_lista(request, propriedade_id)`: Lista convites de cotação enviados aos fornecedores.
- Função `convite_cotacao_novo(request, propriedade_id)`: Sem descrição registrada.
- Função `convite_cotacao_cancelar(request, propriedade_id, convite_id)`: Sem descrição registrada.
- Função `cotacao_fornecedor_responder_token(request, token)`: Portal público para fornecedores responderem uma cot
- Função `cotacao_fornecedor_nova(request, propriedade_id, requisicao_id)`: Registro de cotação para uma requisição a
- Função `recebimento_compra_novo(request, propriedade_id, ordem_id)`: Registro do recebimento físico vinculado à OC

- Função `fornecedores_lista(request, propriedade_id)`: Lista de fornecedores
- Função `fornecedor_novo(request, propriedade_id)`: Cadastrar novo fornecedor
- Função `ordens_compra_lista(request, propriedade_id)`: Lista de ordens de compra
- Função `orcamentos_compra_lista(request, propriedade_id)`: Configuração de orçamento mensal por propriedade e set
- Função `ordem_compra_nova(request, propriedade_id)`: Criar nova ordem de compra
- Função `ordem_compra_detalhes(request, propriedade_id, ordem_id)`: Detalhes completos da ordem de compra
- Função `notas_fiscais_lista(request, propriedade_id)`: Lista de notas fiscais
- Função `nota_fiscal_upload(request, propriedade_id)`: Upload de Nota Fiscal (XML)
- Função `nota_fiscal_detalhes(request, propriedade_id, nota_id)`: Detalhes da nota fiscal

## views_curral.py
- Função `_obter_resumo_animais(sessao)`: Gera um resumo inteligente dos animais manejados na sessão
- Função `_obter_resumo_lotes(propriedade, limite)`: Sem descrição registrada.
- Função `_garantir_manejos_padrao(propriedade)`: Garante que manejos fundamentais estejam cadastrados.
- Função `_montar_catalogo_manejos(propriedade)`: Retorna o catálogo de manejos organizado por categoria.
- Função `_obter_sessao_super_tela(propriedade, usuario)`: Garante que exista uma sessão ativa para vincular eventos r
- Função `curral_painel(request, propriedade_id)`: Sem descrição registrada.
- Função `curral_dashboard(request, propriedade_id)`: Sem descrição registrada.
- Função `_normalizar_codigo(codigo)`: Remove caracteres não numéricos e devolve o código limpo.
- Função `_extrair_numero_manejo(codigo_sisbov)`: Obtém o número de manejo (7 últimos dígitos sem o verificador).
- Função `_parse_decimal(valor)`: Sem descrição registrada.
- Função `_parse_data(data_str)`: Sem descrição registrada.
- Função `_categoria_padrao_para(sexo)`: Sem descrição registrada.
- Função `_mapear_tipo_movimentacao(origem)`: Sem descrição registrada.
- Função `_avaliar_situacao_bnd(consta_bnd, presente_no_sistema)`: Retorna a situação de conformidade com o BND S
- Função `curral_identificar_codigo(request, propriedade_id)`: Verifica se o código pertence a um animal existente ou ao e
- Função `curral_registrar_manejo(request, propriedade_id)`: Registra ações rápidas do Curral Inteligente (cadastros e tro
- Função `_processar_cadastro_estoque(propriedade, codigo, payload, usuario)`: Sem descrição registrada.
- Função `_atualizar_ficha_animal(propriedade, payload, usuario)`: Sem descrição registrada.
- Função `_status_para_brinco_antigo(motivo)`: Sem descrição registrada.
- Função `_processar_troca_brinco(propriedade, codigo, payload, usuario)`: Sem descrição registrada.
- Função `_registrar_manejo_programar_iatf(propriedade, codigo, payload, usuario)`: Registra um manejo de protocolo IA
- Função `_registrar_pesagem_rapida(propriedade, codigo, payload, usuario)`: Registra um evento de pesagem vinculado
- Função `_registrar_movimentacao_animal(propriedade, codigo, payload, usuario)`: Registra uma movimentação individu
- Função `curral_sessao(request, propriedade_id, sessao_id)`: Sem descrição registrada.
- Função `curral_criar_lote(request, propriedade_id, sessao_id)`: Sem descrição registrada.
- Função `curral_registrar_evento(request, propriedade_id, sessao_id)`: Sem descrição registrada.
- Função `curral_encerrar_sessao(request, propriedade_id, sessao_id)`: Sem descrição registrada.
- Função `curral_relatorio(request, propriedade_id, sessao_id)`: Sem descrição registrada.

## views_custos.py
- Função `custos_dashboard(request, propriedade_id)`: Dashboard de custos da propriedade
- Função `custos_fixos_lista(request, propriedade_id)`: Lista de custos fixos

- Função `custos_fixos_novo(request, propriedade_id)`: Criar novo custo fixo
- Função `custos_variaveis_lista(request, propriedade_id)`: Lista de custos variáveis
- Função `custos_fixos_editar(request, propriedade_id, custo_id)`: Editar custo fixo
- Função `custos_fixos_excluir(request, propriedade_id, custo_id)`: Excluir custo fixo
- Função `custos_variaveis_editar(request, propriedade_id, custo_id)`: Editar custo variável
- Função `custos_variaveis_excluir(request, propriedade_id, custo_id)`: Excluir custo variável
- Função `custos_variaveis_novo(request, propriedade_id)`: Criar novo custo variável
- Função `calcular_fluxo_caixa(request, propriedade_id)`: Calcular fluxo de caixa da propriedade

## views_endividamento.py
- Função `dividas_financeiras_dashboard(request, propriedade_id)`: Dashboard do módulo de dívidas financeiras
- Função `financiamentos_lista(request, propriedade_id)`: Lista todos os financiamentos da propriedade
- Função `financiamento_novo(request, propriedade_id)`: Adiciona novo financiamento
- Função `financiamento_editar(request, propriedade_id, financiamento_id)`: Edita financiamento existente
- Função `financiamento_excluir(request, propriedade_id, financiamento_id)`: Exclui financiamento
- Função `tipos_financiamento_lista(request)`: Lista tipos de financiamento
- Função `tipo_financiamento_novo(request)`: Adiciona novo tipo de financiamento
- Função `calcular_amortizacao(request, financiamento_id)`: Calcula tabela de amortização do financiamento

## views_exportacao.py
- Função `calcular_altura_tabela(table_data, font_size, padding)`: Calcula a altura aproximada de uma tabela baseada no
- Função `ajustar_tabela_para_pagina(table_data, col_widths, max_height_cm)`: Ajusta uma tabela para caber na página,
- Função `ajustar_fonte_por_conteudo(table_data, fonte_base)`: Ajusta o tamanho da fonte baseado na quantidade de co
- Função `calcular_larguras_dinamicas(table_data, num_colunas, largura_total_cm)`: Calcula larguras de colunas dinamic
- Função `exportar_inventario_excel(request, propriedade_id)`: Exporta inventário para Excel
- Função `exportar_projecao_excel(request, propriedade_id)`: Exporta projeção para Excel
- Função `exportar_projecao_pdf(request, propriedade_id)`: Exporta projeção para PDF em modo paisagem com todas as
- Função `exportar_iatf_excel(request, propriedade_id)`: Exporta o relatório completo de IATF em formato Excel.
- Função `exportar_iatf_pdf(request, propriedade_id)`: Exporta o relatório completo de IATF em PDF com tabelas analítica
- Função `exportar_inventario_pdf(request, propriedade_id)`: Exporta inventário para PDF

## views_financeiro.py
- Função `_obter_propriedade(usuario, propriedade_id)`: Garante que a propriedade pertence ao contexto do usuário.
- Função `financeiro_dashboard(request, propriedade_id)`: Dashboard unificado de visão financeira.
- Função `lancamentos_lista(request, propriedade_id)`: Lista filtrada de lançamentos financeiros.
- Função `lancamento_novo(request, propriedade_id)`: Cria um novo lançamento financeiro.
- Função `lancamento_editar(request, propriedade_id, lancamento_id)`: Edição de lançamento existente.
- Função `lancamento_quitar(request, propriedade_id, lancamento_id)`: Marca um lançamento como quitado.
- Função `lancamento_cancelar(request, propriedade_id, lancamento_id)`: Cancela um lançamento.
- Função `contas_financeiras_lista(request, propriedade_id)`: Sem descrição registrada.
- Função `conta_financeira_nova(request, propriedade_id)`: Sem descrição registrada.
- Função `conta_financeira_editar(request, propriedade_id, conta_id)`: Sem descrição registrada.
- Função `categorias_lista(request, propriedade_id)`: Sem descrição registrada.

- Função `categoria_nova(request, propriedade_id)`: Sem descrição registrada.
- Função `categoria_editar(request, propriedade_id, categoria_id)`: Sem descrição registrada.
- Função `centros_custo_lista(request, propriedade_id)`: Sem descrição registrada.
- Função `centro_custo_novo(request, propriedade_id)`: Sem descrição registrada.
- Função `centro_custo_editar(request, propriedade_id, centro_id)`: Sem descrição registrada.

## views_funcionarios.py
- Função `funcionarios_dashboard(request, propriedade_id)`: Dashboard de funcionários
- Função `funcionarios_lista(request, propriedade_id)`: Lista de funcionários
- Função `funcionario_novo(request, propriedade_id)`: Cadastrar novo funcionário
- Função `folha_pagamento_processar(request, propriedade_id)`: Processar folha de pagamento
- Função `processar_holerite(funcionario, folha, competencia)`: Processa holerite de um funcionário
- Função `folha_pagamento_detalhes(request, propriedade_id, folha_id)`: Detalhes da folha de pagamento
- Função `holerite_pdf(request, propriedade_id, holerite_id)`: Gerar PDF do holerite

## views_iatf_completo.py
- Função `iatf_dashboard(request, propriedade_id)`: Dashboard completo de IATF
- Função `lote_iatf_novo(request, propriedade_id)`: Criar novo lote de IATF
- Função `lote_iatf_detalhes(request, propriedade_id, lote_id)`: Detalhes do lote de IATF
- Função `iatf_individual_novo(request, propriedade_id)`: Registrar nova IATF individual
- Função `iatf_individual_detalhes(request, propriedade_id, iatf_id)`: Detalhes da IATF individual
- Função `iatf_registrar_aplicacao(request, propriedade_id, iatf_id)`: Registrar aplicação de medicamento
- Função `iatf_registrar_inseminacao(request, propriedade_id, iatf_id)`: Registrar inseminação realizada
- Função `iatf_registrar_diagnostico(request, propriedade_id, iatf_id)`: Registrar diagnóstico de prenhez
- Função `lotes_iatf_lista(request, propriedade_id)`: Lista de lotes de IATF
- Função `iatfs_lista(request, propriedade_id)`: Lista de IATFs individuais
- Função `protocolos_iatf_lista(request, propriedade_id)`: Lista de protocolos IATF
- Função `touros_semen_lista(request, propriedade_id)`: Lista de touros para sêmen
- Função `lotes_semen_lista(request, propriedade_id)`: Lista de lotes de sêmen

## views_imobilizado.py
- Função `imobilizado_dashboard(request, propriedade_id)`: Dashboard do módulo de imobilizado
- Função `bens_lista(request, propriedade_id)`: Lista todos os bens da propriedade
- Função `bem_novo(request, propriedade_id)`: Adiciona novo bem
- Função `bem_editar(request, propriedade_id, bem_id)`: Edita bem existente
- Função `bem_excluir(request, propriedade_id, bem_id)`: Exclui bem
- Função `categorias_lista(request)`: Lista categorias de imobilizado
- Função `categoria_nova(request)`: Adiciona nova categoria
- Função `categoria_editar(request, categoria_id)`: Editar categoria de imobilizado
- Função `categoria_excluir(request, categoria_id)`: Excluir categoria de imobilizado
- Função `calcular_depreciacao_automatica(request, propriedade_id)`: Calcula depreciação automaticamente para todos
- Função `relatorio_imobilizado(request, propriedade_id)`: Gera relatório de imobilizado

## views_nutricao.py
- Função `nutricao_dashboard(request, propriedade_id)`: Dashboard consolidado de Nutrição
- Função `estoque_suplementacao_lista(request, propriedade_id)`: Lista de estoques de suplementação
- Função `compra_suplementacao_nova(request, propriedade_id)`: Registrar compra de suplementação
- Função `distribuicao_suplementacao_nova(request, propriedade_id)`: Registrar distribuição de suplementação
- Função `cochos_lista(request, propriedade_id)`: Lista de cochos
- Função `controle_cocho_novo(request, propriedade_id)`: Registrar controle de cocho

## views_operacoes.py
- Função `operacoes_dashboard(request, propriedade_id)`: Dashboard consolidado de Operações
- Função `combustivel_lista(request, propriedade_id)`: Lista de tanques de combustível
- Função `consumo_combustivel_novo(request, propriedade_id)`: Registrar consumo de combustível
- Função `equipamentos_lista(request, propriedade_id)`: Lista de equipamentos
- Função `manutencao_nova(request, propriedade_id)`: Registrar nova manutenção

## views_pecuaria_completa.py
- Função `pecuaria_completa_dashboard(request, propriedade_id)`: Dashboard consolidado de Pecuária (Inventário + Ra
- Função `_categoria_tem_keywords(nome_categoria, keywords)`: Sem descrição registrada.
- Função `_estimar_peso_categoria(categoria)`: Sem descrição registrada.
- Função `_calcular_valor_movimentacao(mov)`: Sem descrição registrada.
- Função `_json_response(payload, status)`: Sem descrição registrada.
- Função `_carregar_json(request)`: Sem descrição registrada.
- Função `_parse_decimal(valor, default)`: Sem descrição registrada.
- Função `_parse_int(valor, default)`: Sem descrição registrada.
- Função `_obter_planejamento(propriedade, planejamento_id)`: Sem descrição registrada.
- Função `_serializar_meta_comercial(meta)`: Sem descrição registrada.
- Função `_serializar_meta_financeira(meta)`: Sem descrição registrada.
- Função `_serializar_indicador_planejado(indicador)`: Sem descrição registrada.
- Função `_serializar_cenario_planejamento(cenario)`: Sem descrição registrada.
- Função `_montar_contexto_planejamento(propriedade, planejamento, cenario)`: Sem descrição registrada.
- Função `pecuaria_planejamento_dashboard(request, propriedade_id)`: Dashboard estratégico do planejamento pecuári
- Função `pecuaria_planejamentos_api(request, propriedade_id)`: API com a lista de planejamentos anuais da propriedad
- Função `pecuaria_planejamento_resumo_api(request, propriedade_id, planejamento_id)`: API com resumo consolidado
- Função `animais_individuais_lista(request, propriedade_id)`: Lista de animais individuais (Rastreabilidade)
- Função `animal_individual_novo(request, propriedade_id)`: Cadastrar novo animal individual
- Função `animal_individual_detalhes(request, propriedade_id, animal_id)`: Detalhes do animal individual
- Função `reproducao_dashboard(request, propriedade_id)`: Dashboard de reprodução
- Função `touros_lista(request, propriedade_id)`: Lista de touros
- Função `touro_novo(request, propriedade_id)`: Cadastrar novo touro
- Função `estacao_monta_nova(request, propriedade_id)`: Criar nova estação de monta
- Função `iatf_nova(request, propriedade_id)`: Registrar nova IATF

## views_pesagem.py

- Função `_calcular_metricas(ultima_pesagem, pesagem_anterior, peso_alvo, hoje)`: Sem descrição registrada.
- Função `pesagem_dashboard(request, propriedade_id)`: Sem descrição registrada.
- Função `pesagem_nova(request, propriedade_id)`: Sem descrição registrada.

## views_projetos_bancarios.py
- Função `projetos_bancarios_dashboard(request, propriedade_id)`: Dashboard centralizado de projetos bancários
- Função `consolidar_dados_propriedade(propriedade)`: Consolida dados de todos os módulos da propriedade
- Função `calcular_indicadores_automaticos(propriedade)`: Calcula indicadores financeiros automaticamente
- Função `gerar_projecoes_financeiras(propriedade)`: Gera projeções financeiras para 5 anos

## views_proprietario.py
- Função `proprietario_dashboard(request, produtor_id)`: Dashboard consolidado do proprietário com todas as propriedad
- Função `consolidar_dados_proprietario(produtor, propriedades)`: Consolida dados de todas as propriedades do proprieta
- Função `proprietario_dividas_consolidadas(request, produtor_id)`: Dívidas consolidadas de todas as propriedades
- Função `proprietario_capacidade_consolidada(request, produtor_id)`: Capacidade de pagamento consolidada
- Função `calcular_capacidade_consolidada(produtor, propriedades)`: Calcula capacidade de pagamento consolidada
- Função `proprietario_imobilizado_consolidado(request, produtor_id)`: Imobilizado consolidado de todas as propriedades
- Função `proprietario_analise_consolidada(request, produtor_id)`: Análise consolidada de todas as propriedades
- Função `calcular_indicadores_consolidados(produtor, propriedades)`: Calcula indicadores financeiros consolidados
- Função `proprietario_relatorios_consolidados(request, produtor_id)`: Relatórios consolidados de todas as propriedades
- Função `gerar_relatorios_consolidados(produtor, propriedades)`: Gera relatórios consolidados

## views_rastreabilidade.py
- Função `rastreabilidade_dashboard(request, propriedade_id)`: Dashboard principal de rastreabilidade bovina
- Função `importar_bnd_sisbov(request, propriedade_id)`: Tela centralizada para importação de dados BND/SISBOV
- Função `animais_individuais_lista(request, propriedade_id)`: Lista de animais individuais
- Função `animal_individual_novo(request, propriedade_id)`: Cadastro de novo animal individual
- Função `animal_individual_detalhes(request, propriedade_id, animal_id)`: Detalhes de um animal individual
- Função `animal_individual_editar(request, propriedade_id, animal_id)`: Edição de animal individual
- Função `movimentacao_individual_nova(request, propriedade_id, animal_id)`: Cadastro de nova movimentação individu
- Função `brincos_lista(request, propriedade_id)`: Lista de brincos da propriedade
- Função `brinco_cadastrar_lote(request, propriedade_id)`: Cadastro de brincos em lote
- Função `relatorio_rastreabilidade(request, propriedade_id)`: Relatório completo de rastreabilidade
- Função `_parse_date(value)`: Sem descrição registrada.
- Função `relatorio_dia_barcodes(request, propriedade_id)`: Emissão de Documentos de Identificação Animal (DIA) com c
- Função `relatorio_inventario_sisbov(request, propriedade_id)`: Inventário oficial SISBOV
- Função `relatorio_movimentacoes_sisbov(request, propriedade_id)`: Livro oficial de movimentações SISBOV
- Função `relatorio_entradas_sisbov(request, propriedade_id)`: Relatório de nascimentos e entradas (compras/transferênc
- Função `relatorio_saidas_sisbov(request, propriedade_id)`: Relatório de saídas (vendas, transferências, óbitos)
- Função `relatorio_sanitario_sisbov(request, propriedade_id)`: Relatório de vacinação e tratamentos
- Função `api_gerar_numero_brinco(request, propriedade_id)`: API para gerar sugestão de número de brinco

## views_relatorios.py

- Função `relatorios_dashboard(request, propriedade_id)`: Dashboard do módulo de relatórios
- Função `relatorio_final(request, propriedade_id)`: Relatório final consolidado simples (inventário + indicadores básicos).
- Função `relatorio_inventario(request, propriedade_id)`: Relatório de inventário do rebanho
- Função `relatorio_financeiro(request, propriedade_id)`: Relatório financeiro completo
- Função `relatorio_custos(request, propriedade_id)`: Relatório de custos de produção
- Função `relatorio_endividamento(request, propriedade_id)`: Relatório de endividamento
- Função `relatorio_consolidado(request, propriedade_id)`: Relatório consolidado geral
- Função `gerar_resumo_propriedade(propriedade)`: Gera resumo geral da propriedade
- Função `gerar_dados_financeiros(propriedade)`: Gera dados financeiros para relatórios
- Função `exportar_relatorio_inventario_pdf(request, propriedade_id)`: Exporta relatório de inventário em PDF
- Função `exportar_relatorio_inventario_excel(request, propriedade_id)`: Exporta relatório de inventário em Excel
- Função `exportar_relatorio_financeiro_pdf(request, propriedade_id)`: Exporta relatório financeiro em PDF
- Função `exportar_relatorio_financeiro_excel(request, propriedade_id)`: Exporta relatório financeiro em Excel
- Função `exportar_relatorio_custos_pdf(request, propriedade_id)`: Exporta relatório de custos em PDF
- Função `exportar_relatorio_custos_excel(request, propriedade_id)`: Exporta relatório de custos em Excel
- Função `exportar_relatorio_endividamento_pdf(request, propriedade_id)`: Exporta relatório de endividamento em PDF
- Função `exportar_relatorio_endividamento_excel(request, propriedade_id)`: Exporta relatório de endividamento em Exce
- Função `exportar_relatorio_consolidado_pdf(request, propriedade_id)`: Exporta relatório consolidado em PDF
- Função `exportar_relatorio_consolidado_excel(request, propriedade_id)`: Exporta relatório consolidado em Excel

## views_relatorios_rastreabilidade.py
- Função `relatorio_identificacao_individual(request, propriedade_id)`: Relatório de Identificação Individual dos Animais - P
- Função `relatorio_movimentacao_animais(request, propriedade_id)`: Relatório de Movimentação de Animais - PNIB OBI
- Função `relatorio_sanitario(request, propriedade_id)`: Relatório Sanitário - PNIB OBRIGATÓRIO
- Função `relatorio_gta(request, propriedade_id, movimentacao_id)`: Relatório de GTA (Guia de Trânsito Animal) - PNIB O
- Função `exportar_identificacao_individual_pdf(request, propriedade_id)`: Exporta Relatório de Identificação Individual em
- Função `exportar_movimentacao_animais_pdf(request, propriedade_id)`: Exporta Relatório de Movimentação de Animai

## views_suplementacao.py
- Função `suplementacao_dashboard(request, propriedade_id)`: Dashboard de suplementação
- Função `estoque_suplementacao_lista(request, propriedade_id)`: Lista de estoques de suplementação
- Função `estoque_suplementacao_novo(request, propriedade_id)`: Cadastrar novo estoque de suplementação
- Função `compra_suplementacao_nova(request, propriedade_id)`: Registrar compra de suplementação
- Função `distribuicao_suplementacao_nova(request, propriedade_id)`: Registrar distribuição de suplementação no pasto
- Função `estoque_suplementacao_detalhes(request, propriedade_id, estoque_id)`: Detalhes do estoque de suplementaç

## views_vendas.py
- Função `vendas_por_categoria_lista(request, propriedade_id)`: Lista os parâmetros de venda por categoria
- Função `vendas_por_categoria_novo(request, propriedade_id)`: Adiciona novo parâmetro de venda por categoria
- Função `vendas_por_categoria_editar(request, propriedade_id, parametro_id)`: Edita parâmetro de venda por categoria
- Função `vendas_por_categoria_bulk(request, propriedade_id)`: Configuração em massa de vendas por categoria
- Função `vendas_por_categoria_excluir(request, propriedade_id, parametro_id)`: Exclui parâmetro de venda por categori
- Função `vendas_por_categoria_toggle_status(request, propriedade_id, parametro_id)`: Ativa/desativa parâmetro de ven