CRONOGRAMA DE DESENVOLVIMENTO: SISTEMA CHICKEN HOUSE

Prazo Estimado: 4 Semanas (20 dias úteis)

Objetivo: MVP (Mínimo Produto Viável) com Simulação

SEMANA 1: Lógica de Negócio e Testes (Backend)

Meta: Ter o "cérebro" do sistema funcionando e validado pelos testes unitários.

- [ ] **Dia 01: Configuração Inicial**
  - Criar projeto "ChickenHouse" no NetBeans.
  - Criar estrutura de pacotes (model, view, controller, test).
  - Configurar repositório Git e realizar o primeiro Push.
- [ ] **Dia 02: Modelagem de Dados**
  - Criar classes principais: `Galinha.java`, `Lote.java`, `Configuracao.java`.
  - Implementar atributos e métodos Getters/Setters.
- [ ] **Dia 03: Lógica de Nutrição**
  - Implementar classe `CalculadoraRacao.java`.
  - Codificar regra de negócio (peso x fator de consumo).
  - **Executar Testes:** CT-03 e CT-07.
- [ ] **Dia 04: Lógica de Estoque**
  - Implementar métodos de controle de saldo de animais (nascimento/mortalidade).
  - **Executar Teste:** CT-06.
- [ ] **Dia 05: Bateria de Testes**
  - Finalizar a escrita dos 10 casos de teste unitários.
  - Garantir que todos os testes retornem "Passed" (Verde).

SEMANA 2: Interface Gráfica (Frontend Swing)

Meta: Criar telas funcionais para interação do usuário.

- [ ] **Dia 06: Acess**

  Criar tela `LoginView.java` (JFrame).

  Implementar validação simples de usuário e senha.

- [ ] **Dia 07: Navegação**

  Criar tela `DashboardView.java` (Menu Principal).

  Adicionar botões/ícones para os módulos (Luz, Ração, Relatórios)

- [ ] **Dia 08: Módulo Iluminação**

  Criar tela de configuração de horários (Campos de Texto para Hora Início/Fim).

  Programar botão "Salvar Configuração".

- [ ] **Dia 09: Módulo Alimentação**

  Criar formulário para input de Quantidade de Galinhas e Peso Médio.

  Criar botão "Calcular" que exibe o resultado na tela.

- [ ] **Dia 10: Painel de Status**

  Criar área visual para mostrar estado dos equipamentos (Simulação).

  Exemplo: Labels que mudam de cor (Verde=Ligado / Cinza=Desligado).

SEMANA 3: Integração e Simulação

Meta: Fazer o sistema funcionar automaticamente com base no tempo.

[ ] **Dia 11: Relógio do Sistema**

Implementar Thread/Timer que roda em segundo plano a cada segundo.

Exibir relógio digital na tela principal.

[ ] **Dia 12: Automação de Luz**

Integrar Timer com a lógica de Iluminação.

Condicional: Se hora atual estiver no intervalo, mudar status da tela para "LIGADO".

[ ] **Dia 13: Automação de Ração**

Integrar Timer com a lista de horários de alimentação.

Exibir aviso na tela ("Liberando Ração...") quando o horário bater.

[ ] **Dia 14: Tratamento de Falhas**

Criar botão de teste "Simular Erro no Sensor".

Programar o disparo de pop-up ou alerta vermelho na tela (RF09).

[ ] **Dia 15: Teste Integrado**

Simular um dia de operação completo (acelerado).

Verificar se luz e ração ativam na ordem correta.

SEMANA 4: Persistência e Documentação

Meta: Finalizar o sistema para entrega com relatórios.

- [ ] **Dia 16: Logs de Auditoria**

  Criar classe `LogManager.java`.

  Gravar ações automáticas em arquivo de texto (`log.txt`).

- [ ] **Dia 17: Relatórios PDF**

  Integrar biblioteca iText.

  Programar botão "Gerar Relatório Técnico" que exporta os dados atuais.

- [ ] **Dia 18: Persistência de Configurações**

  Implementar salvamento de dados (Properties ou JSON) para não perder horários ao fechar o programa.

- [ ] **Dia 19: Polimento e Bugs**

  Teste exploratório (tentar quebrar o sistema).

  Correção de falhas e ajustes visuais.

- [ ] **Dia 20: Entrega Final**

  Limpar código (remover comentários inúteis).

  Atualizar README do GitHub.

  Gerar o executável (`.jar`) e realizar o Push final.