



Equipe

Tayane Silva

Scrum Master / Dev Full Stack

João Gabriel

Dev Full Stack / Analista

Daniel Bonfim

Dev Full Stack

Danilo Vinicius

Dev Full Stack

Gustavo Curado

Dev Full Stack



É uma plataforma para o **gerenciamento inteligente** de fazendas voltadas à criação de animais, reunindo em um **único ambiente digital** o controle das atividades operacionais, estoque, localização de animais e monitoramento climático.

METODOLOGIA SCRUM

Por que Scrum?

Organização
Flexibilidade
Transparência

Colaboradores

Scrum Master
Desenvolvedores

Sprint Review

Product Backlog
Uma vez por semana

SPRINTS

Sprint 0 Concepção

- Definir a Visão do Produto
- Criar o Product Backlog

Sprint 1 Análise

- Diagrama de Caso de Uso
- Diagrama BPD

Sprint 2 Projeto

- Escolha das tecnologias
- Diagrama de Classes
- Criar a arquitetura de software
- Prototipar interfaces gráficas

Sprint 3 Implementação

- Codificar os módulos principais do sistema
- Integrar frontend e backend

Sprint 4 Testes

- Realizar testes de integração e sistema
- Corrigir bugs e falhas encontradas

Sprint 5 Implantação

- Fazer o deploy do sistema
- Elaborar manual do usuário
- Sprint Review Final

DIAGRAMA BPD

Processo de Negócios - Movimentação de Estoque

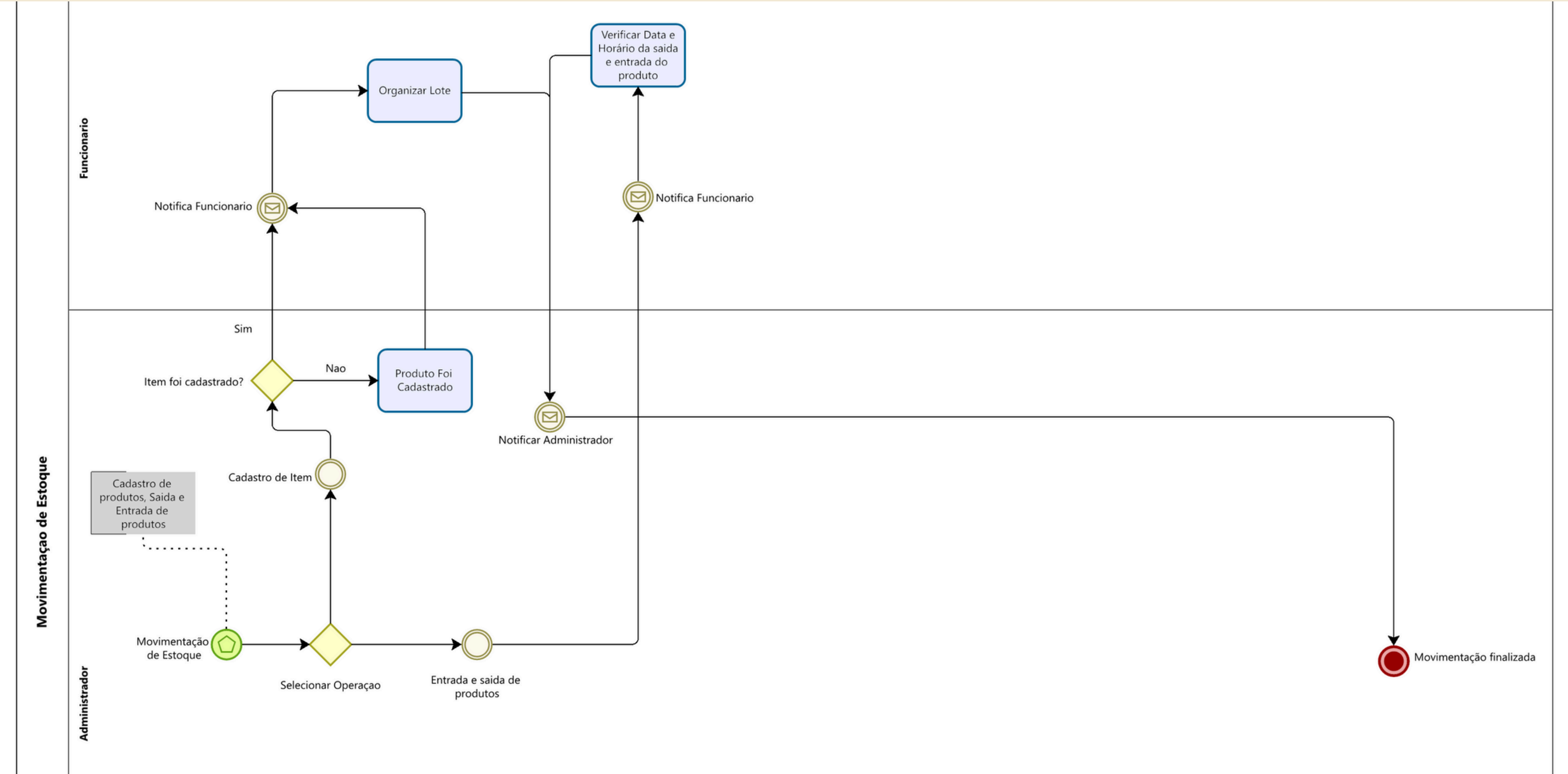
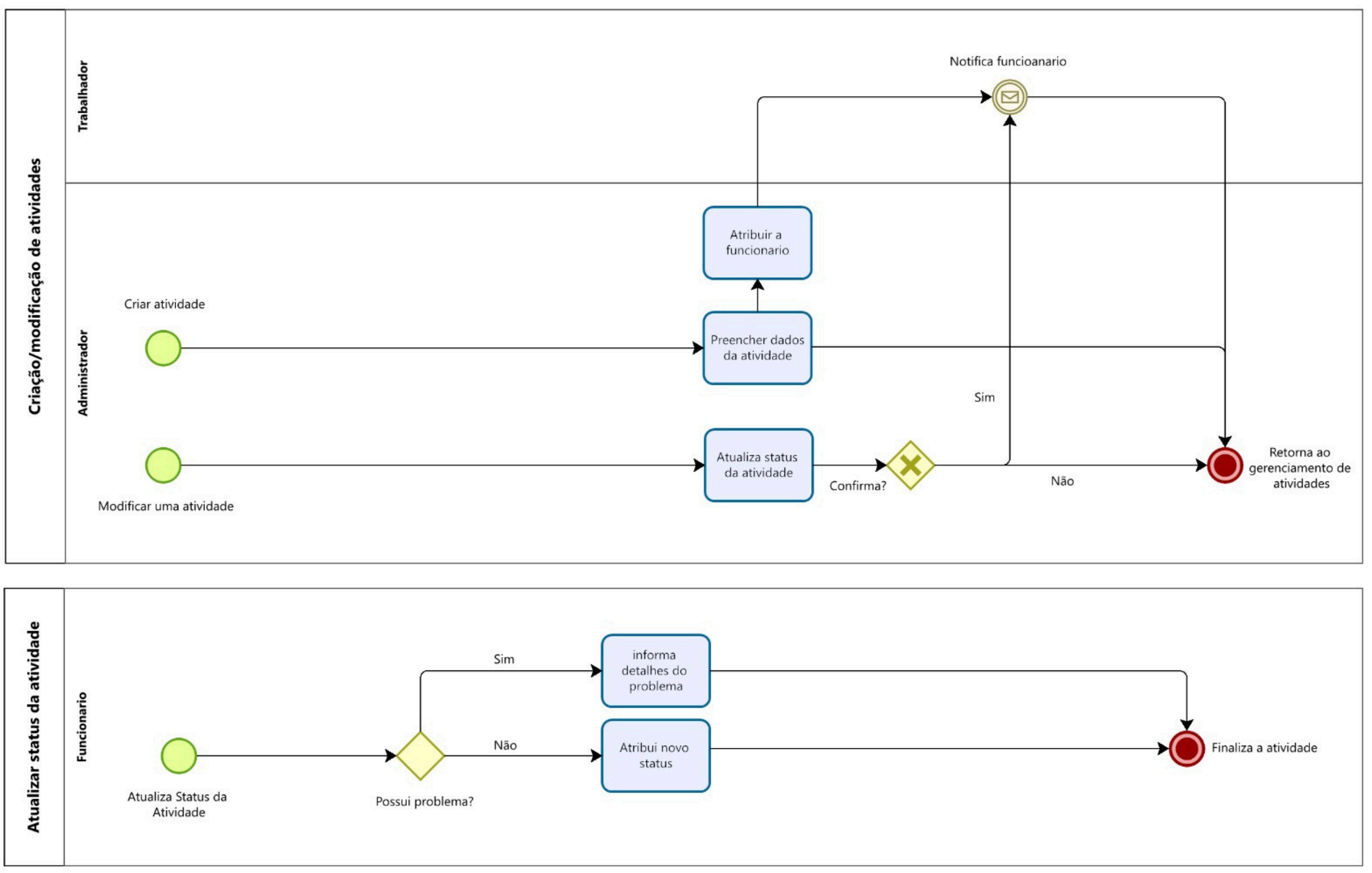


DIAGRAMA BPD

Processo de Negócios - Gerenciamento de Atividades



SPRINT 2 – PROJETO

TECNOLOGIAS UTILIZADAS

Linguagem de Programação

Python

Containerização

Docker

API's

Open-meteo
Geolinker

Banco de Dados

Sqlite

Framework

Django
Bootstrap 5

Frontend

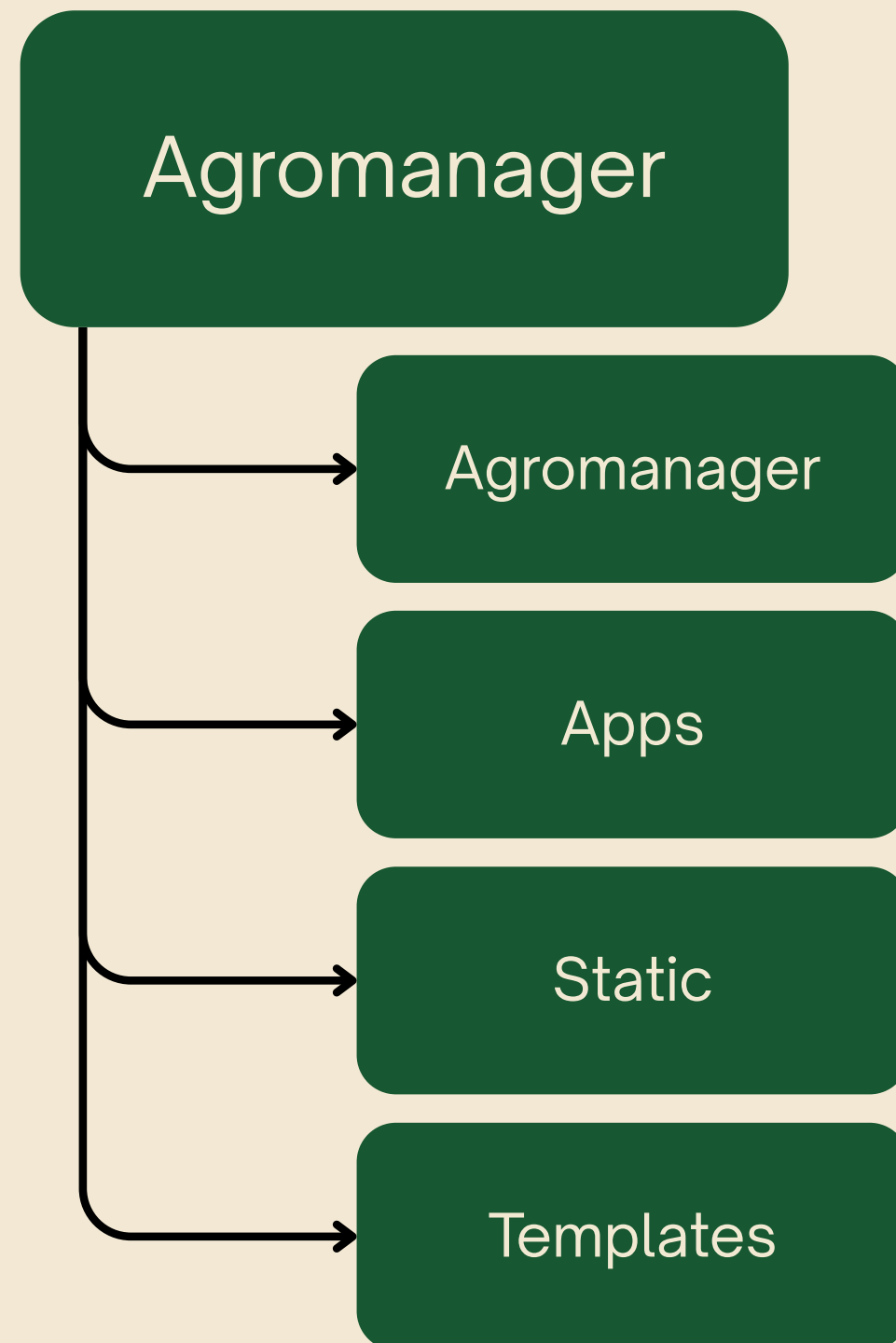
HTML,CSS
JavaScript

Versionamento

Github

ARQUITETURA DE SOFTWARE

Organização, qualidade e eficiência



- Monolito modular
- Banco de dados centralizado
- Aplicação principal
- Configurações e roteamento principal
- Padrão MVT para módulos
- Model, View e Template
- Atributos estáticos comuns a todo projeto
- Logo, CSS, Scripts JS
- Páginas HTML integradas ao Django

Exemplo de aplicação do projeto

```
estoque
├── admin.py
├── apps.py
├── forms.py
├── models.py
├── tests.py
├── urls.py
└── views.py
```

O exemplo mostra a estrutura de arquivos de uma aplicação Django chamada "estoque". O diretório contém os seguintes arquivos: admin.py, apps.py, forms.py, models.py, tests.py, urls.py e views.py.

SPRINT 3 – IMPLEMENTAÇÃO

TESTES

Garantia de Qualidade e Integração Contínua

Django

- Estrutura de testes nativa
- 68 testes
- Testes de views
- Testes de models
- Testes de templates

Github Actions

- ferramenta de CI/CD integrada ao GitHub
- Testes automatizados para cada push
- Feedback de erros
- Proteção das branches
- Toda configuração em um arquivo .yml

TESTES

Garantia de Qualidade e Integração Contínua

Exemplo de teste

```
class AtividadeModelTest(TestCase):  
    def test_criacao_basica(self):  
        a = Atividade.objects.create(titulo="Teste")  
        self.assertEqual(a.titulo, "Teste")  
        self.assertIsNone(a.data_exclusao)  
  
    def test_valores_padrao(self):  
        a = Atividade.objects.create(titulo="Teste")  
        self.assertEqual(a.prioridade, "MEDIA")  
        self.assertEqual(a.tipo_atividade, "GERAL")  
        self.assertEqual(a.status, "registrada")
```

TESTES

Garantia de Qualidade e Integração Contínua

Feedback dos testes

[ifmt-cba/agromanager] Teste Django workflow run



Teste Django: All jobs have failed

[View workflow run](#)



Teste Django / test
Failed in 16 seconds



test

failed yesterday in 16s

- > ✓ Set up job
- > ✓ Run actions/checkout@v4
- > ✓ Configurar o Python
- > ✓ Instalar dependencias
- > ✓ Rodar migracoes
- > ✗ Rodar testes
 - ⌚ Post Configurar o Python
- > ✓ Post Run actions/checkout@v4
- > ✓ Complete job

IMPLANTAÇÃO



Plataforma de hospedagem que permite deploy automático de aplicações web.

É uma solução simples, prática e gratuita

Deploy direto do GitHub



AgroManager

**THANK
YOU**