# Documento de Requisitos – Projeto Locus

# 1. Introdução

Este documento descreve os requisitos do sistema Locus - Plataforma web mobile comunitária de mapeamento e cuidado de animais de rua. O objetivo é consolidar, de forma clara e organizada, os requisitos funcionais, não funcionais, restrições e critérios de aceitação que nortearão o desenvolvimento do sistema.

Propósito: servir como base de entendimento entre equipe técnica, stakeholders e cliente.

Escopo resumido: criar uma PWA responsiva que permita cadastrar, monitorar e apoiar animais de rua, com engajamento da comunidade, ONGs, clínicas e órgãos públicos.

# 2. Visão Geral do Sistema

O abandono de animais de rua é um problema crescente nos centros urbanos. Muitas vezes as informações sobre esses animais são dispersas ou inexistentes. O sistema Locus visa centralizar dados, facilitar a ação comunitária e oferecer apoio para ONGs e prefeituras, aumentando as chances de cuidado e adoção.

Objetivos principais do sistema:  
- Mapear e monitorar animais de rua.  
- Permitir interação comunitária para ajudar animais.  
- Apoiar ONGs e clínicas veterinárias em ações de resgate e adoção.  
- Gerar dados úteis para políticas públicas.

# 3. Requisitos Funcionais

* RF1 - Cadastro de animais com foto, tipo, situação e localização aproximada.
* RF2 - Visualização de feed comunitário com animais cadastrados.
* RF3 - Interação no mural (curtir, comentar, compartilhar postagens).
* RF4 - Filtro do feed por tipo, situação e localização (bairro ou raio).
* RF5 - Criação de novas postagens (upload de imagem, descrição, status inicial, localização GPS).
* RF6 - Validação de campos obrigatórios e feedback visual.
* RF7 - Atualizações colaborativas em postagens existentes (histórico de status e alterações).
* RF8 - Exibição de badges/ícones indicando se o animal foi ajudado, adotado ou ainda precisa de ajuda.
* RF9 - Gerenciamento de postagens: criar, editar, excluir e listar.
* RF10 - Registro de histórico de mudanças e responsável pela atualização.
* RF11 - Endpoint para listar postagens filtradas (API).
* RF12 - Paginação no feed (scroll infinito).

# 4. Requisitos Não Funcionais

* RNF1 - Desempenho: carregamento rápido com lazy loading de imagens.
* RNF2 - Escalabilidade: suportar grande volume de postagens, comentários e usuários simultâneos.
* RNF3 - Usabilidade: interface leve, responsiva e intuitiva (padrão de apps sociais).
* RNF4 - Segurança: autenticação robusta (OAuth2), controle de acesso e moderação de conteúdo.
* RNF5 - Acessibilidade: contraste adequado, texto alternativo para imagens, navegação por teclado.
* RNF6 - Disponibilidade: sistema acessível 24/7, com downtime mínimo.
* RNF7 - Compatibilidade: acesso fluido em navegadores modernos e dispositivos móveis (PWA).

# 5. Restrições

- O sistema deve ser desenvolvido em React para frontend.

- O backend deve utilizar Node.js (Express ou NestJS).

- Banco de dados MySQL.

- Uso de API de mapas (Google Maps ou Leaflet).

- Autenticação via OAuth2 (Google/Facebook) e login por e-mail.

- Hospedagem a ser definida.

- Gerenciamento do projeto com Jira, GitHub e Notion.

# 6. Priorização dos Requisitos

Os requisitos foram classificados conforme prioridade para o desenvolvimento inicial (MVP) e evoluções futuras.

MVP (essenciais): mural/feed, cadastro de animais, mapa interativo, login/autenticação.

Evoluções futuras: gamificação, relatórios avançados para órgãos públicos, integrações adicionais.

# 7. Critérios de Aceitação

- O usuário deve conseguir cadastrar um animal com foto, localização e status inicial.

- O feed deve listar postagens em ordem cronológica com possibilidade de filtro.

- Comentários devem aparecer instantaneamente após inserção.

- Usuários devem conseguir compartilhar uma postagem.

- O mapa deve exibir a localização aproximada dos animais cadastrados.

- O sistema deve suportar pelo menos 1000 usuários simultâneos sem perda significativa de desempenho.

- ONGs e administradores devem ter acesso restrito a relatórios e dados consolidados.

# 8. Riscos Relacionados a Requisitos

- Risco de baixa adesão da comunidade: mitigação via campanhas de engajamento e parcerias.  
- Vazamento de dados sensíveis: mitigação via autenticação robusta e criptografia.  
- Atraso no desenvolvimento de requisitos críticos: mitigação via sprints curtos e controle rigoroso.  
- Escopo crescer demais (scope creep): mitigação via validação contínua com stakeholders.