

Projeto: DENUN

Aplicativo Flutter para registros de denúncias urbanas integrado ao Firebase

André Penchel, Juan Pablo, Luís Starling, Luis Fernando

22/10/2025

Contexto do Problema

Cidades sofrem com:



Buracos



Falta de Iluminação



Vandalismo

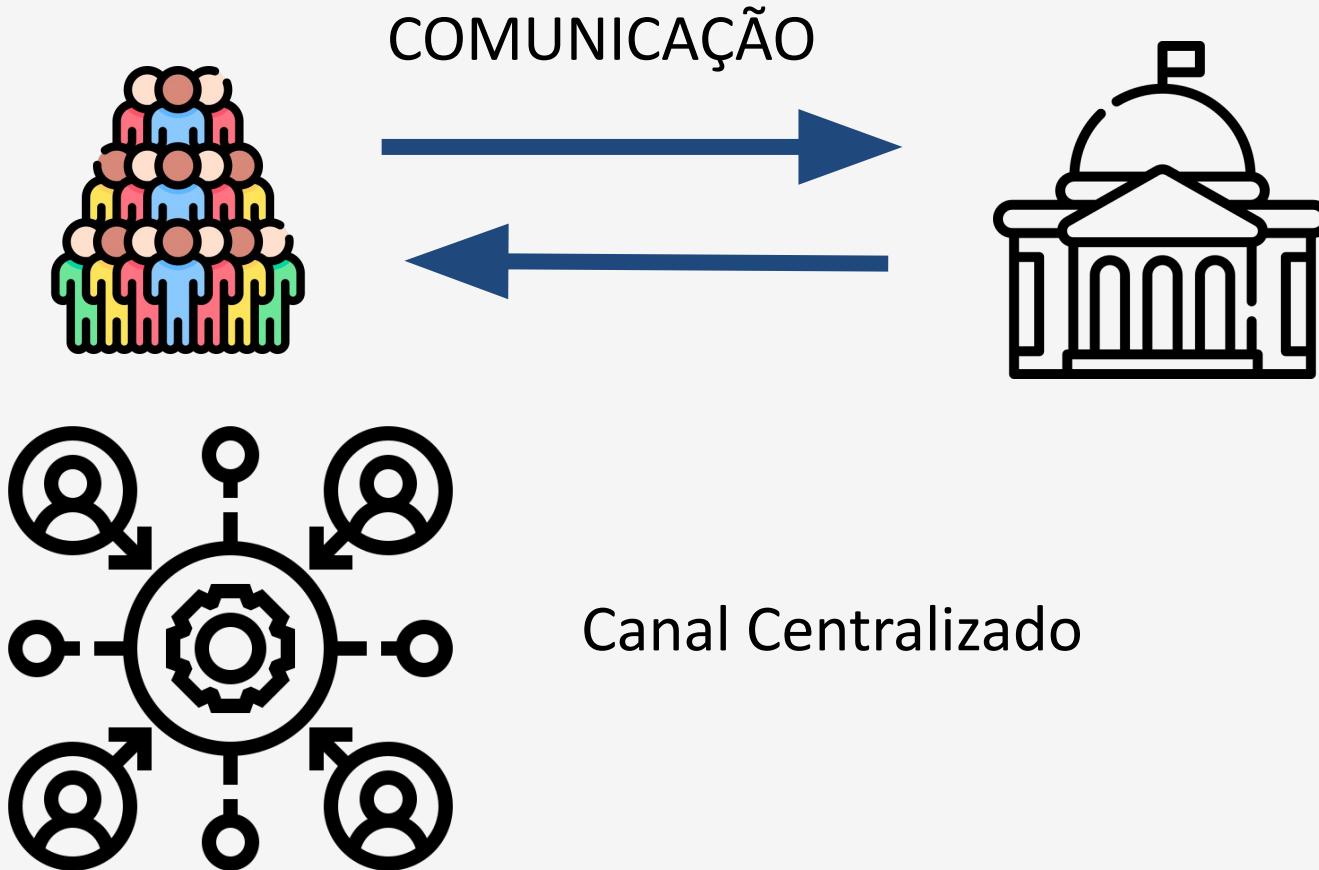


Acúmulo de lixo



Roubos

Motivação





Público Alvo

Cidadãos que querem reportar problemas urbanos.

Prefeituras e equipes de manutenção.

Comunidades locais e líderes de bairro.

Objetivo do Sistema

Permitir que usuários registrem denúncias com foto, descrição e localização.

Organizar essas denúncias em um fluxo rastreável.

Oferecer uma interface fácil de usar e integrada a serviços em nuvem.

Objetivo do Sistema

Principais User Stories

- ◆ Identificação
 - RFI 001 / UCI 001 – Login de usuário
 - RFI 002 / UCI 002 – Cadastro
 - RFI 003 / UCI 003 – Confirmação de e-mail
- ◆ Denúncias
 - RFD 001 / UCD 001 – Registrar denúncia
 - RFD 002 / UCD 002 – Protocolo automático
 - RFD 003 / UCD 003 – Acompanhar status
 - RFD 004 / UCD 004 – Atualização de status (admin)
 - RFD 005 / UCD 005 – Notificações
 - RFD 006 / UCD 006 – Avaliação
 - RFD 007 / UCD 007 – Reabrir denúncia
- ◆ Mapa
 - RFM 001 / UCM 001 – Localização no mapa
 - RFM 002 / UCM 002 – Integração com Google Maps
 - RFM 003 / UCM 003 – Mapa público anonimizado
- ◆ Administração
 - RFA 001 / UCA 001 – Triagem e encaminhamento
 - RFA 002 / UCA 002 – Relatórios e estatísticas
 -

Diagramas Utilizados

Diagrama de caso de USO RFI 001:

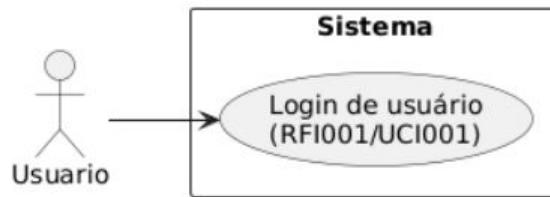
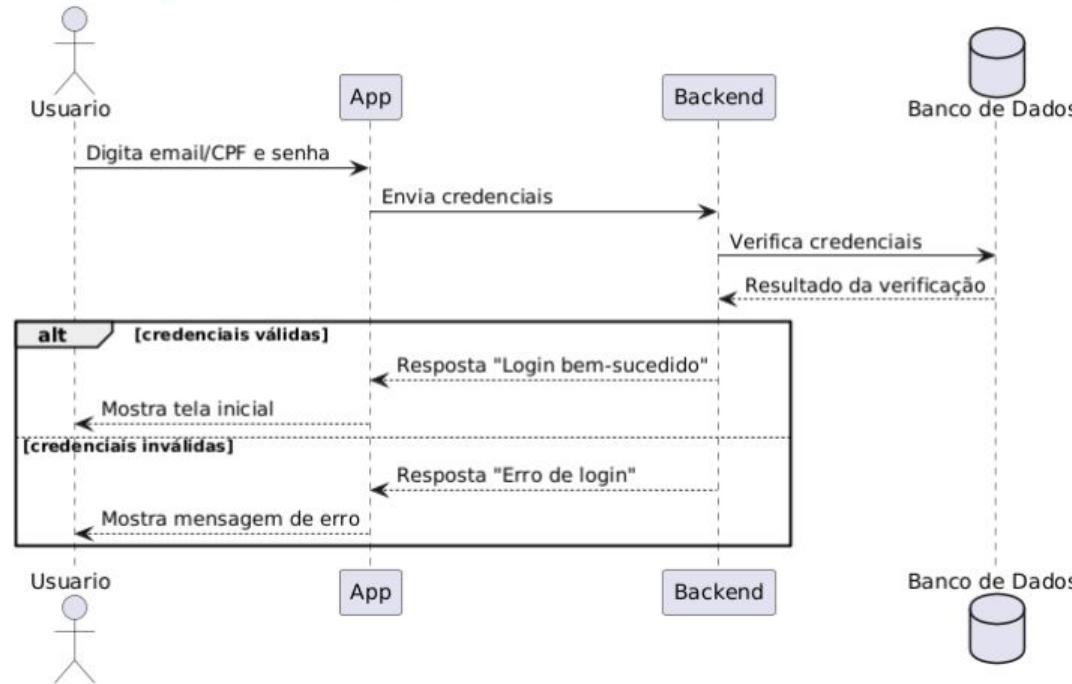


Diagrama de sequência RFI 001:



Diagramas Utilizados

Diagrama de sequência RFD 001

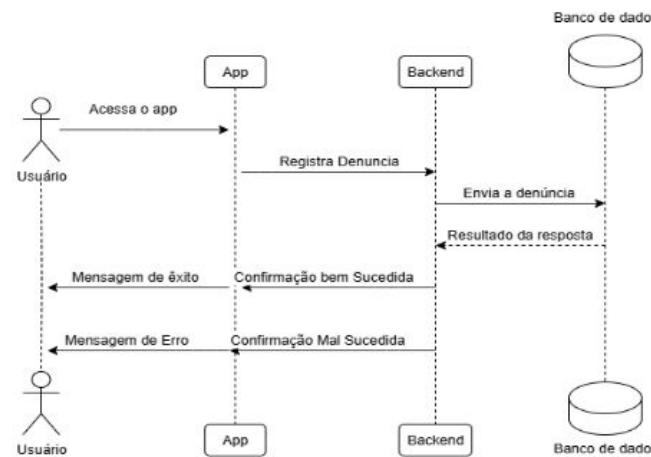
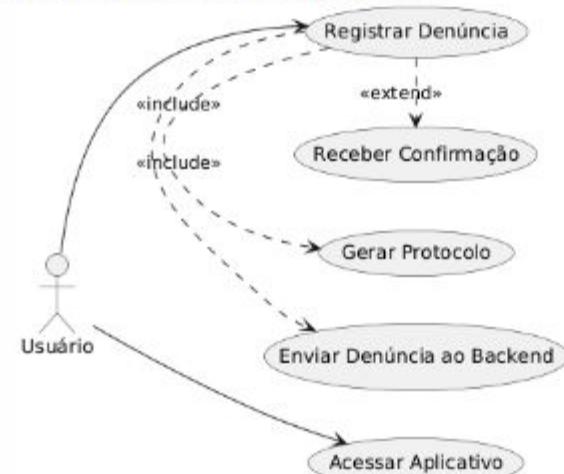


Diagrama de caso de USO RFD 001



Diagramas Utilizados

Diagrama de sequência RFD 003

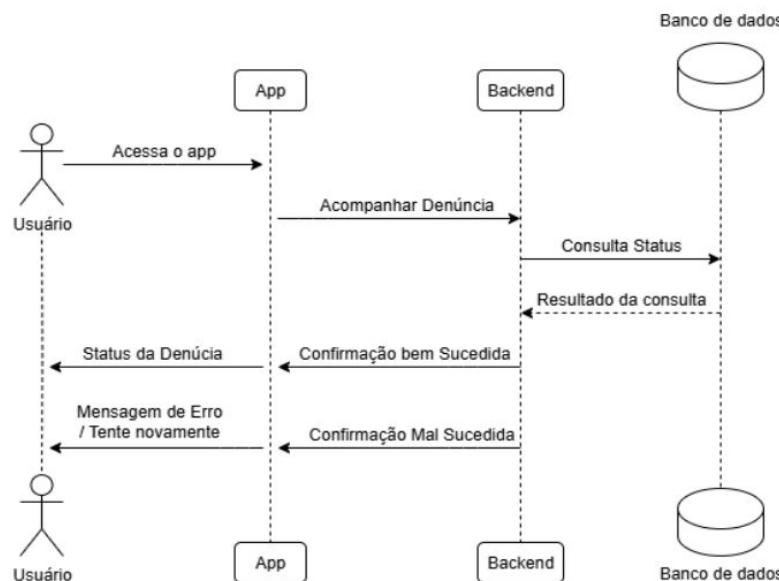
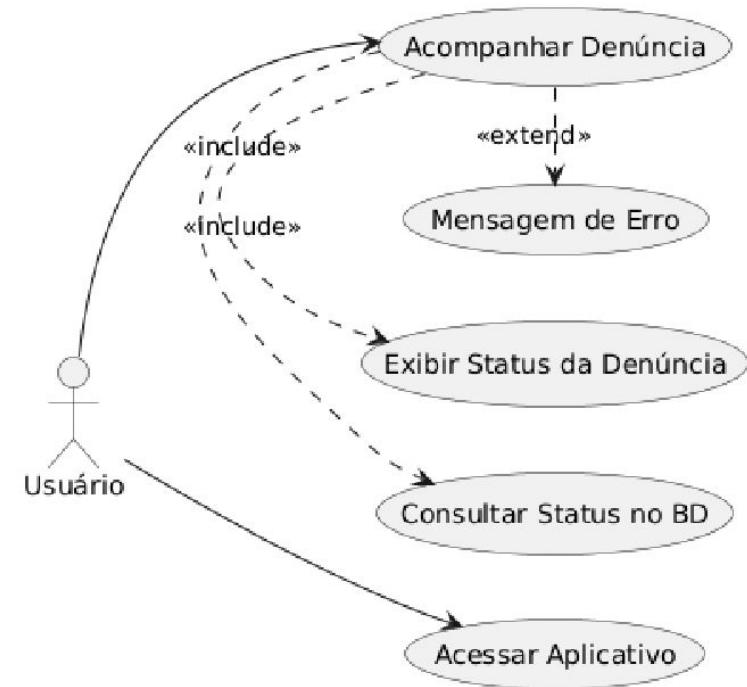
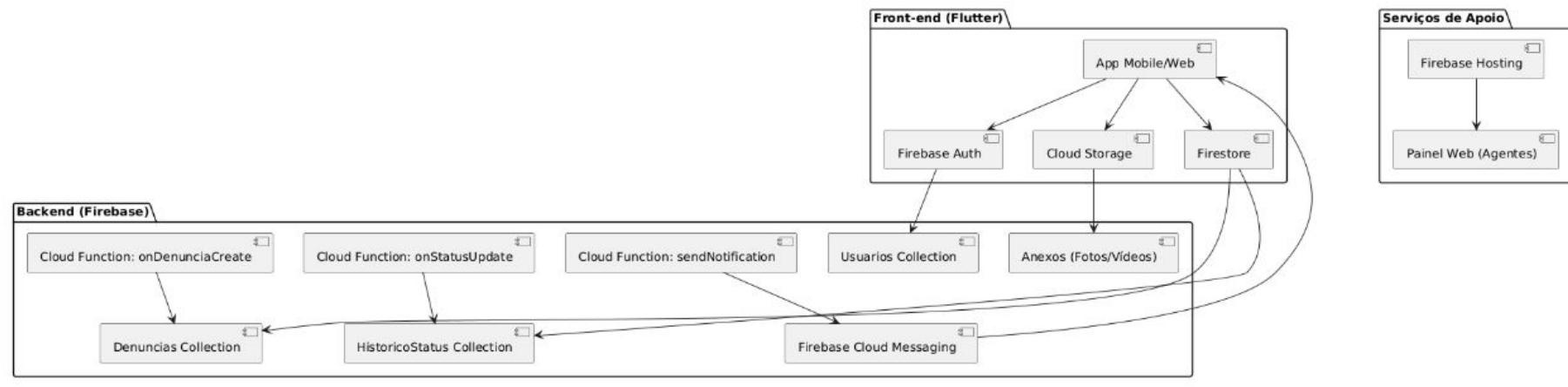


Diagrama de caso de USO RFD 003



Diagramas Utilizados

Diagrama de Componentes - Sistema de Denúncias Urbanas



Arquitetura do Sistema

Diagrama Arquitetural Final

- Camada de **Apresentação** (Flutter UI)
- Camada de **Aplicação/Serviços**
- Camada de **Domínio** (entidades Denúncia, Usuário e Arquivo)
- Camada de **Persistência** (Firestore + Firebase Storage)

Padrões Utilizados

- **Arquitetura em Camadas**
- **MVVM / MVC simplificado** dentro do Flutter
- **Repositório** para abstração do acesso aos dados
- **DTOs** para comunicação entre camadas

Justificativa das escolhas

- Flutter oferece rapidez, multiplataforma e UI moderna.
- Firebase simplifica autenticação, persistência e escalabilidade.
- Arquitetura em camadas aumenta manutenibilidade e clareza.
- Abstração via repositórios → permite trocar Firestore futuramente por outro backend.

Implementação

Principais componentes

- **Interface Flutter:** telas de login, cadastro e envio de denúncias.
- **Camada de domínio:** regras para criar, validar e tratar denúncias.
- **Persistência Firebase:** Firestore para dados e Storage para mídias.

Tecnologias e frameworks

- **Flutter/Dart**
- **Firebase Auth**
- **Cloud Firestore**
- **Firebase Storage**

Fluxo de chamadas (API → domínio → persistência)

1. Usuário envia denúncia pela UI.
2. Camada de domínio valida e prepara os dados.
3. Mídias são enviadas ao **Storage** e retornam URLs.
4. Dados + URLs são gravados no **Firestore**.
5. App recebe confirmação e atualiza a interface.

Testes e Qualidade

Estratégia de testes

- Testes de **unidade** para lógica e modelos.
- Testes de **widget/integração** para validar telas e fluxos.
- Uso de *mocks* para evitar dependência do Firebase nos testes.

Ferramentas utilizadas

- **flutter_test**
- **mockito** (ou mocks nativos)
- Testes de integração do próprio **Flutter**

Demonstração

Conclusões

Principais aprendizados: a importância dos testes e da automatização

Melhorias Futuras: A existência de um banco de dados local

O que faria de diferente: Aproveitaria de mais bibliotecas de front end

OBRIGADO!