UNIVERSIDADE DE FORTALEZA
VICE REITORIA DE GRADUAÇÃO
CENTRO DE CIÊNCIAS TECNOLÓGICAS
TECNÓLOGO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Alice Rocha Jamil
Breno Guimarães Ribeiro
Daniel de Moura Mascarenhas
Larissa Evangelista Moreira
Matheus Silva Matos
Walther Oliveira Pires

DOCUMENTAÇÃO TÉCNICA

Projeto Recycle

FORTALEZA
2025

Alice Rocha Jamil
Breno Guimarães Ribeiro
Daniel de Moura Mascarenhas
Larissa Evangelista Moreira
Matheus Silva Matos
Walther Oliveira Pires

DOCUMENTAÇÃO TÉCNICA

Projeto Recycle

Este documento contém a documentação técnica do Sistema de Gestão de Estoque denominado Recycle, voltado a empresas de reciclagem, desenvolvido na componente curricular N393 - Projeto Aplicado Multiplataforma, como requisito para obtenção de nota.

Supervisor: Prof. Bruno Lopes, Me

FORTALEZA
2025

SUMÁRIO

1. INTRODUÇÃO

1.1. CONTEXTO E JUSTIFICATIVA

O setor de reciclagem desempenha um papel essencial na sustentabilidade ambiental, mas muitas empresas ainda realizam o controle de seus estoques de forma manual, utilizando planilhas ou anotações em papel. Essa prática torna o processo suscetível a erros, perda de informações e falta de atualização em tempo real, comprometendo a eficiência das operações e a tomada de decisões.

Diante desse cenário, o sistema *Recycle* foi idealizado como uma solução tecnológica voltada à digitalização e automação da gestão de estoque em empresas de reciclagem. O sistema permitirá controlar entradas e saídas, calcular automaticamente o preço médio dos materiais e oferecer uma visão geral do estoque, por meio de uma interface intuitiva e segura para usuários autenticados. Assim, o *Recycle* busca reduzir erros humanos, otimizar processos e contribuir para a eficiência e sustentabilidade do setor.

1.2. OBJETIVOS

O objetivo deste projeto é desenvolver um sistema de gestão de estoque denominado Recycle, voltado a empresas de reciclagem, com o intuito de otimizar o controle de materiais recicláveis por meio da automatização dos processos de entrada e saída, cálculo do preço médio e acompanhamento do saldo atual. O sistema visa proporcionar uma visão ampla e precisa do estoque, oferecendo interfaces web e mobile com acesso seguro e usabilidade intuitiva.

Dito isso, os objetivos específicos do projeto são:

- Implementar funcionalidades que possibilitem o cadastro de materiais recicláveis, bem como o registro das movimentações de entrada e saída, garantindo o cálculo automático do preço médio e do saldo atualizado.
- Desenvolver uma interface web administrativa que permita o gerenciamento completo do sistema, abrangendo o cadastro de administradores, materiais e movimentações de estoque.

- Criar uma aplicação mobile voltada ao uso operacional em campo, possibilitando o registro de materiais e movimentações de forma prática e acessível.
- Estabelecer níveis de acesso diferenciados, assegurando que administradores possuam controle total sobre as funcionalidades, enquanto funcionários disponham de permissões restritas às suas atribuições.
- Proporcionar uma experiência de uso segura e eficiente, por meio de autenticação de usuários e design responsivo, favorecendo a confiabilidade e a acessibilidade do sistema em diferentes dispositivos.

1.3. ESCOPO E DELIMITAÇÃO

- **Escopo:**
    - Cadastro, edição, visualização e exclusão de materiais recicláveis.
    - Registro de entradas e saídas de materiais, com atualização automática do saldo e cálculo do preço médio.
    - Controle de acesso de usuários baseado em perfis (administrador, funcionário).
    - Autenticação de usuários, garantindo acesso seguro às funcionalidades do sistema.
    - Interface web destinada à gestão completa do sistema, voltada aos administradores.
    - Aplicativo mobile voltado à operação prática, permitindo o registro de materiais e movimentações.
    - Interface intuitiva e responsiva, proporcionando facilidade de uso em diferentes dispositivos.
- **Delimitação (Fora do Escopo):**
    - *O sistema não possui a funcionalidade de recuperação de senha (“Esqueci minha senha”).*
    - *Não é possível suspender, desativar ou bloquear o acesso de um funcionário após o cadastro.*
    - *Não há geração de relatórios sobre movimentações ou saldos de estoque.*

- *O sistema não realiza integração com sistemas externos, como plataformas financeiras ou de contabilidade.*
- *Não há suporte a notificações automáticas (por e-mail ou aplicativo) relacionadas a movimentações de estoque.*

2. ENGENHARIA DE REQUISITOS

2.1. REQUISITOS FUNCIONAIS (RFs).

ID	Nome do Requisito	Descrição
RF01	Autenticação de Usuário	O sistema, tanto web quanto o mobile, deve permitir que o usuário se autentique com e-mail e senha. O acesso deve ser permitido somente após validação das credenciais.
RF02	Controle de Perfis de Acesso	O sistema deve permitir a diferenciação de permissões entre usuários administradores e funcionários, garantindo que cada perfil visualize e execute apenas funcionalidades compatíveis com seu nível de acesso.
RF03	Cadastro de Materiais Recicláveis	O sistema deve permitir o cadastro de materiais recicláveis, contendo informações como nome, categoria e unidade de medida.
RF04	Edição e Exclusão de Materiais Recicláveis	O sistema deve permitir editar e excluir materiais previamente cadastrados, desde que o usuário possua permissão de administrador.
RF05	Visualização de Materiais Cadastrados	O sistema deve permitir que os usuários visualizem a lista de materiais cadastrados, incluindo suas informações atualizadas.
RF06	Registro de Entrada de Materiais	O sistema deve permitir registrar entradas de materiais no estoque, atualizando automaticamente o saldo e contribuindo para o cálculo do preço médio.
RF07	Registro de Saída de Materiais	O sistema deve permitir registrar saídas de materiais do estoque, atualizando automaticamente o saldo disponível.
RF08	Interface Responsiva e Intuitiva	O sistema deve fornecer uma interface responsiva e de fácil navegação, permitindo uso eficiente tanto em web quanto mobile.

RF09	Cálculo Automático do Preço Médio	Ao registrar movimentações de entrada de materiais, o sistema deve recalcular automaticamente o preço médio do material com base no valor e quantidade adicionados.
RF10	Exibição do Saldo Atual de Estoque	O sistema deve apresentar ao usuário o saldo atual de cada material no estoque, considerando todas as movimentações registradas.
RF11	Interface Web Administrativa	O sistema deve disponibilizar uma interface web destinada a administradores, permitindo o gerenciamento completo de materiais, usuários e movimentações.
RF12	Aplicativo Mobile Operacional	O sistema deve disponibilizar um aplicativo mobile que permita aos funcionários registrar materiais e movimentações de estoque de forma prática durante o trabalho em campo.

2.2. REQUISITOS NÃO FUNCIONAIS (RNFs)

- **Desempenho:**
  - RNF01: O sistema deve registrar entradas e saídas de materiais sem demora perceptível ao usuário.
  - RNF02: As telas devem carregar em tempo adequado para uso cotidiano, evitando lentidão.
- **Usabilidade:**
  - RNF03: A interface do sistema deve ser simples e fácil de entender.
  - RNF04: As ações principais (como registrar entrada/saída) devem ser acessadas rapidamente pelo usuário.
- **Compatibilidade:**
  - RNF05: A versão web deve funcionar nos navegadores mais comuns (Chrome, Edge e Firefox).
  - RNF06: O aplicativo mobile deve funcionar em dispositivos Android utilizados pelos funcionários.
- **Segurança:**
  - RNF07: O sistema deve permitir acesso apenas a usuários cadastrados.

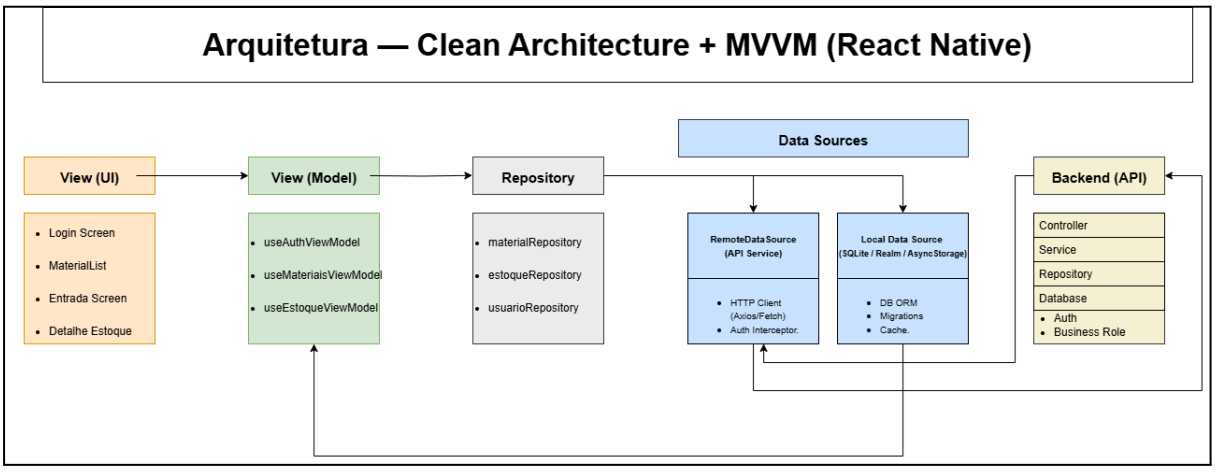
3. PROJETO E ARQUITETURA DO SOFTWARE

3.1. ARQUITETURA GERAL

O aplicativo foi desenvolvido seguindo o padrão **Clean Architecture** com adaptação para o modelo **MVVM (Model–View–ViewModel)**. Esse padrão foi escolhido por promover separação de responsabilidades, facilidade de manutenção, testabilidade e escalabilidade do projeto.

A organização interna do código é dividida em camadas:

1. **View (Apresentação):** Composta pelos componentes e telas desenvolvidas em React Native. É responsável por exibir os dados e capturar interações do usuário, sem conter lógica de negócio.
2. **ViewModel:** Responsável por controlar o estado da interface e coordenar as ações iniciadas pela View. Toda lógica de apresentação fica nessa camada, que não depende diretamente da UI.
3. **Repository (Modelo / Regra de Negócio):** Centraliza o acesso aos dados. Decide se a informação deve ser buscada da API ou do banco de dados local, aplicando o padrão Single Source of Truth.
4. **Data Sources:** São as fontes de dados efetivas. Incluem a comunicação com a API REST (dados remotos) e a persistência local em SQLite / AsyncStorage (cache local).



3.2. PROJETO DE DADOS: INTEGRAÇÃO COM API E PERSISTÊNCIA LOCAL

A arquitetura de dados do sistema foi projetada considerando desempenho, disponibilidade e integridade das informações. A API REST atua como a fonte

principal dos dados, garantindo consistência e atualização centralizada, enquanto o banco de dados local funciona como um cache de persistência, permitindo operação constante mesmo quando o dispositivo estiver sem conexão.

O fluxo de funcionamento segue o padrão Single Source of Truth, onde o aplicativo sempre exibe os dados diretamente do banco de dados local, enquanto a API é utilizada para manter essas informações sincronizadas.

- **Fluxo de Dados**
    - Solicitação de Dados: Quando o usuário acessa seções como lista de materiais ou movimentações de estoque, a camada de aplicação solicita os dados ao Repositório.
    - Consulta à API: O Repositório tenta obter a versão mais atualizada dos dados através da API REST, garantindo que alterações feitas por outros usuários ou sistemas sejam consideradas.
    - Persistência Local (Cache): Após a resposta da API, os dados são salvos no banco de dados local, substituindo registros antigos. Esse armazenamento evita acessos desnecessários à API.
    - Exibição pela Camada Local: A interface do usuário exibe sempre os dados armazenados localmente. Quando há atualização, a interface é atualizada automaticamente sem necessidade de nova requisição à API.

- **Benefícios da Estratégia**
    - Performance: A leitura local é mais rápida do que requisitar a API constantemente.
    - Operação Offline: Caso não haja conexão, o sistema continua funcional, exibindo os últimos dados salvos.
    - Sincronização Confiável: Atualizações acontecem sempre que houver conectividade disponível.
    - Menos carga na API: Diminui o número de requisições e reduz o consumo de rede.

- **Persistência Local**

  O armazenamento local mantém tabelas essenciais para o funcionamento do sistema, incluindo:

    - Usuários
    - Materiais
    - Estoque atualizado
    - Registros de entradas e saídas

Esse modelo garante que o sistema de controle de estoque continue operando de forma consistente mesmo em ambientes desconectados ou com baixa internet, cenário comum em operações logísticas, almoxarifados e canteiros de obra.

Dicionário de Dados

- Tabela `Materiais`:

Nome do campo	Tipo de dados	Chave (PK/FK)	Nulo?	Descrição
data_atualizacao	DATETIME(6)		Não	Timestamp de quando o dado foi atualizado.
data_criacao	DATETIME(6)		Não	Timestamp de quando o dado foi criado.
id	INT	PK	Não	Chave Primária da tabela Materiais.
usuario_id	INT	FK	Não	Foreign Key da tabela Usuarios.
descricao	VARCHAR(255)		Não	Descrição do material cadastrado.
nome	VARCHAR(255)		Não	Nome do Material.
unidade	VARCHAR(255)		Não	Unidade de Medida do Material.

- Tabela `Empresas`:

Nome do campo	Tipo de dados	Chave (PK/FK)	Nulo?	Descrição
id	int	PK	Não	Chave Primária da tabela Empresas.
cnpj	VARCHAR(255)		Não	CNPJ da Empresa
nome_fantasia	VARCHAR(255)	PK	Não	Nome Fantasia da Empresa

- Tabela `Entradas`:

Nome do campo	Tipo de dados	Chave (PK/FK)	Nulo?	Descrição
preco	FLOAT		Não	Valor unitário do material no momento da entrada.
quantidade	FLOAT		Não	Quantidade de unidades do material adicionadas ao estoque

				nesta entrada.
data	DATETIME(6)		Não	Timestamp de quando o material foi adicionado no sistema.
id	INT	PK	Não	Chave Primária da tabela Entradas.
material_id	INT	FK	Não	Foreign Key da tabela Material.
usuario_id	INT	FK	Não	Foreign Key da tabela Usuarios.

- Tabela `Estoque`:

Nome do campo	Tipo de dados	Chave (PK/FK)	Nulo?	Descrição
preco_medio	FLOAT		Não	Preço médio do material no estoque, calculado com base nas entradas realizadas.
quantidade	FLOAT		Não	Quantidade atual disponível do material no estoque.
valor_total	FLOAT		Não	Valor total do estoque desse material, resultante da multiplicação entre preco_medio e quantidade.
material_id	INT	PK	Não	Foreign Key da tabela Material.

- Tabela `Saidas`:

Nome do campo	Tipo de dados	Chave (PK/FK)	Nulo?	Descrição
quantidade	FLOAT		Não	Quantidade de unidades do material removidas do estoque nesta saida.
data	DATETIME(6)		Não	Timestamp de quando o material saiu do sistema.
id	INT	PK	Não	Chave Primária da tabela Saidas.
material_id	INT	FK	Não	Foreign Key da tabela Material.
usuario_id	INT	FK	Não	Foreign Key da tabela Usuarios.

- Tabela `Usuarios`:

Nome do campo	Tipo de dados	Chave (PK/FK)	Nulo?	Descrição
id	INT	PK	Não	Chave Primária da tabela Usuarios.
data_criacao	DATETIME(6)		Não	Timestamp de quando o dado foi criado.
nome	VARCHAR(255)		Não	Nome do Usuario
email	VARCHAR(255)		Não	Email do Usuario
senha	VARCHAR(255)		Não	Senha do Usuario
role	ENUM: ('GERENTE','OPERADOR')		Não	Define o nível de acesso do usuário no sistema.
empresa_id	INT	FK	Não	Foreign Key da tabela Empresa.
data_atualizacao	DATETIME(6)		Não	Timestamp de quando o dado foi atualizado.

3.3. CONSUMO DA API E FLUXO DE NAVEGAÇÃO

- **URL da Documentação da API:**

  `https://github.com/SantoGuru/Recycle/tree/main/docs`

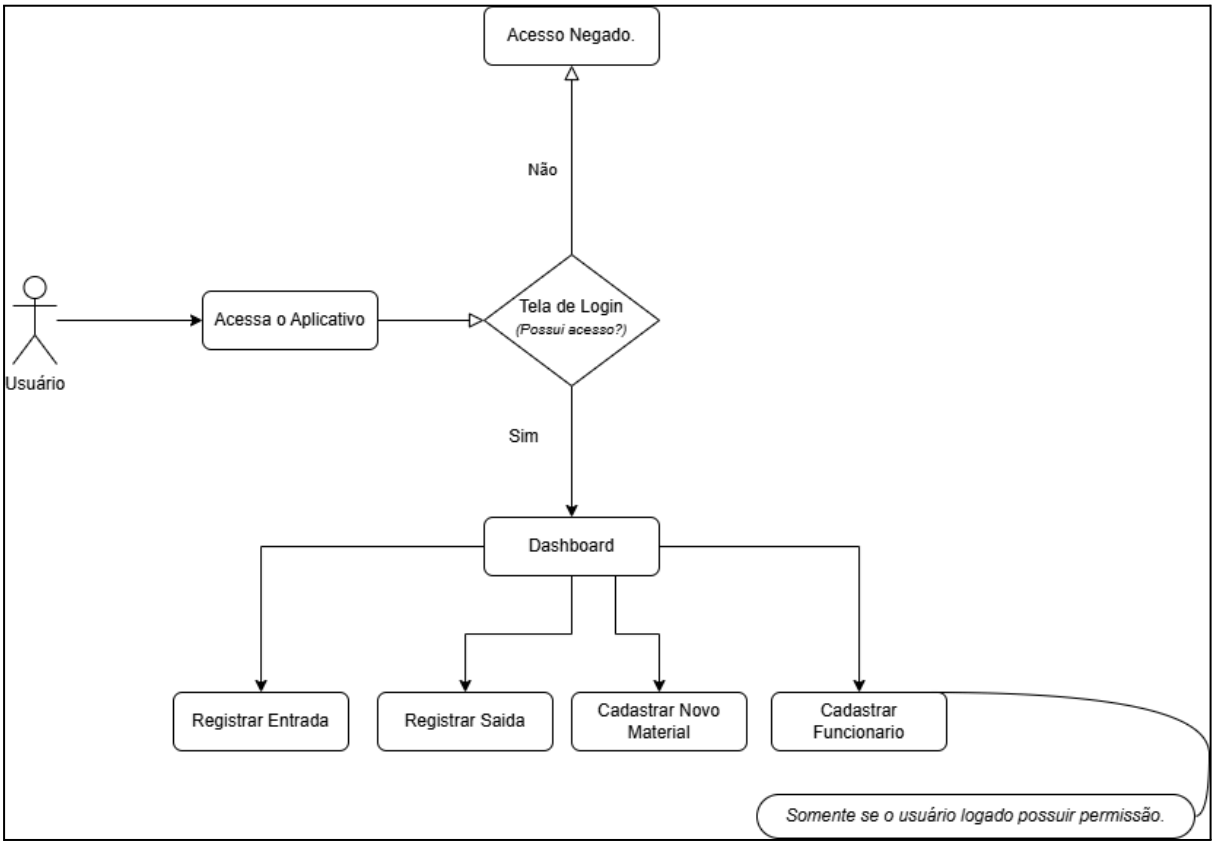
- **Diagrama de Fluxo de Navegação do Aplicativo**

O fluxo de navegação do usuário dentro do aplicativo foi projetado para ser simples, direto e coerente com os perfis de acesso definidos no sistema. A interação se inicia na tela inicial, que apresenta o botão “Começar” junto à identidade visual do projeto Recycle. Ao selecionar essa opção, o usuário é direcionado para a tela de Login, onde deve informar E-mail e Senha para acessar o sistema.

Importante destacar que o aplicativo não permite criação de novas contas diretamente pela interface mobile. O cadastro de usuários é realizado somente pela versão Web (Desktop). Dessa forma, o login assume papel central como porta de entrada para o sistema.

Após a autenticação, o comportamento da navegação passa a depender do perfil (role) do usuário:

- *Gerente:* Possui acesso completo às funcionalidades do sistema. Esse perfil pode cadastrar novos materiais, registrar entradas e saídas de materiais, cadastrar novos usuários (funcionários).
- *Operador:* Possui acesso restrito, podendo apenas registrar entradas e saídas.



4. TECNOLOGIAS E FERRAMENTAS

4.1. STACK DE TECNOLOGIAS

- ***Linguagem:*** *Typescript (react-native) para o desenvolvimento mobile e Java (Springboot) para o backend.*
- ***Arquitetura:*** *Multicamadas.*
- ***UI Toolkit:*** *React Native Paper, que segue o padrão de UI do Google.*
- ***Consumo de API:*** *Requisições HTTP.*
- ***Banco de Dados Local:*** *MySQL.*

4.2. FERRAMENTAS DE DESENVOLVIMENTO

- ***IDE:*** *Visual Studio Code foi a IDE padrão para toda a equipe.*

- ***Controle de Versão:*** *Git, com o repositório hospedado no GitHub. Adotamos o fluxo de trabalho "GitFlow", com branches separadas para* `backend`*,* `frontend`*,* `documentação`*,* `features` *e* `main`*.*
- ***Ferramenta de API:*** *Postman foi usado para testar os endpoints da API durante o desenvolvimento.*

5. IMPLEMENTAÇÃO E RESULTADOS

5.1. TELAS DO SISTEMA

Figura 1: Tela Inicial *(Legenda: Tela de boas-vindas do aplicativo, apresentando a identidade visual do projeto e o botão “Começar”, que direciona o usuário para o acesso ao sistema.)*

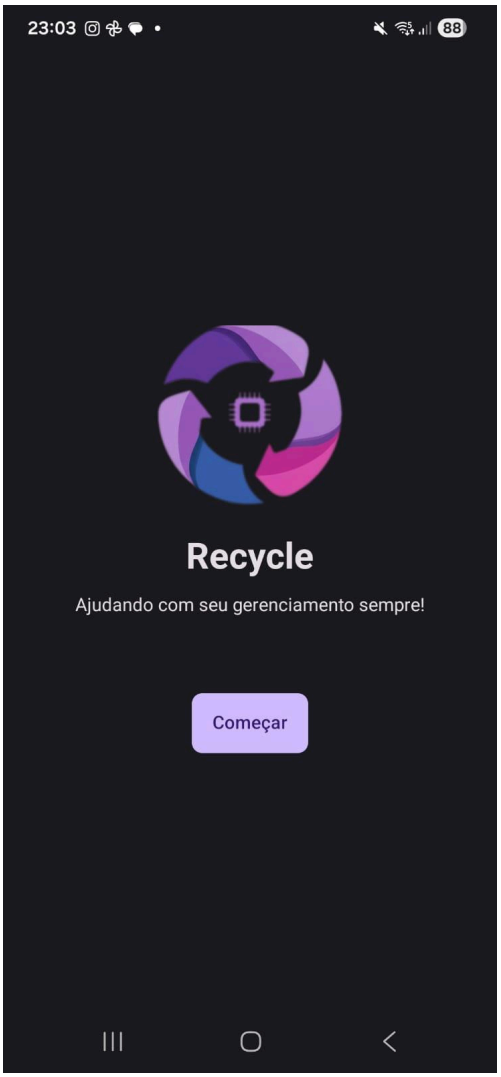


Figura 2: Tela de Login *(Legenda: A tela de login é responsável por controlar o acesso ao sistema. O usuário informa seu e-mail e senha previamente cadastrados. O aplicativo não permite criação de contas via mobile; esse processo ocorre exclusivamente no sistema Web.)*

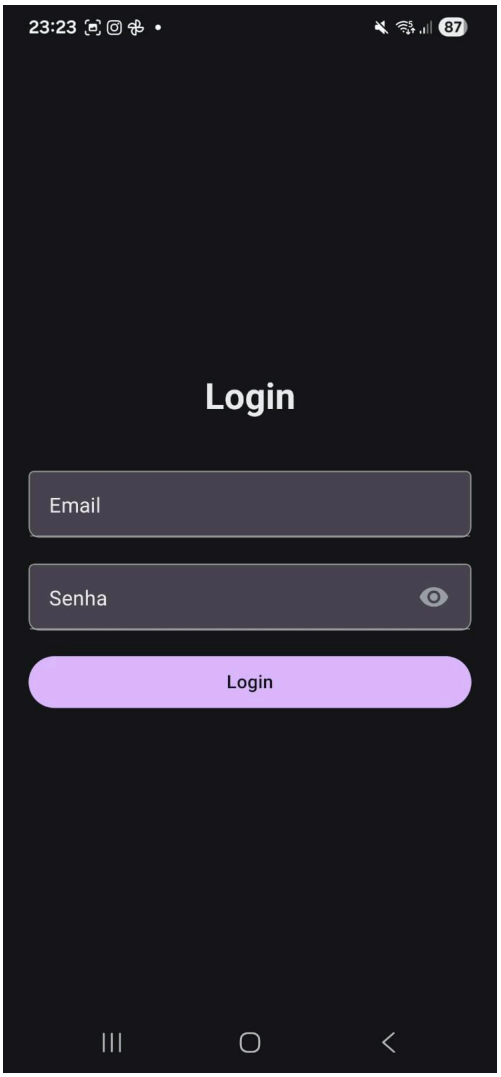


Figura 3: Tela Principal (Visão Gerente) *(Legenda: Após a autenticação, usuários com perfil Gerente têm acesso completo às funcionalidades do sistema, incluindo cadastro de materiais, registro de entradas, registro de saídas e cadastro de novos usuários.)*

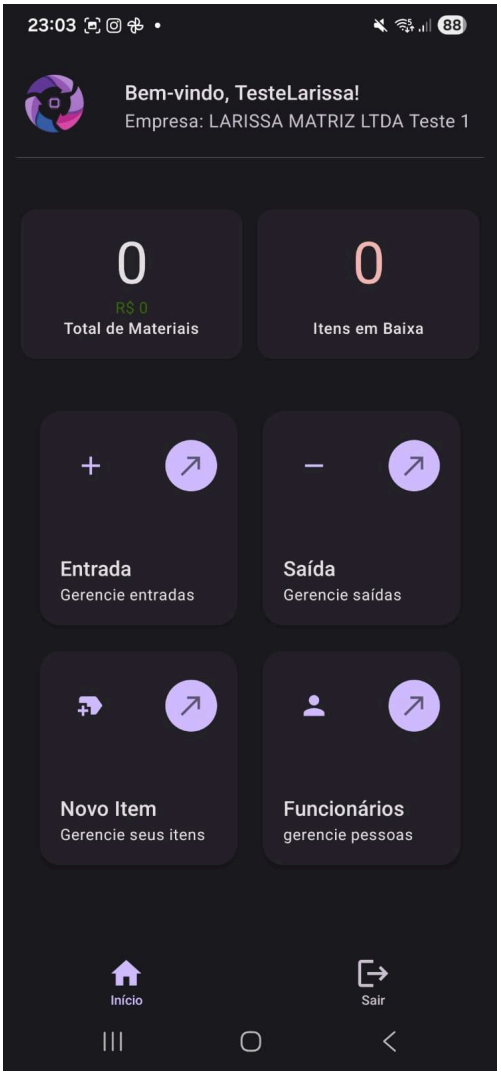
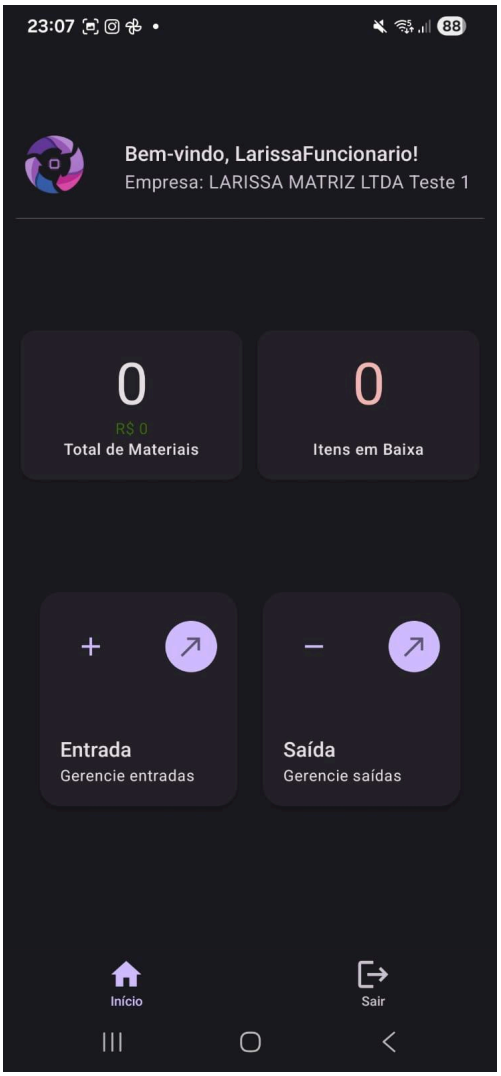


Figura 4: Tela Principal (Visão Operador) *(Legenda: Usuários com perfil Operador possuem interface simplificada, permitindo apenas registrar entradas e saídas de materiais, promovendo usabilidade e evitando ações não autorizadas.)*



6. AMBIENTE E GUIA DE GERAÇÃO (BUILD)

6.1. REQUISITOS DO AMBIENTE

- **IDE:** Visual Studio Code
- **Node:** 21.0.0
- **JDK:** JDK 17

6.2. PROCESSO DE GERAÇÃO DE APLICATIVO

- ***Pré-requisitos:***
  - Ter uma conta cadastrada na versão web;
  - Ter o backend funcionando;

- ***Instruções:***

1. *Clone o repositório do projeto: git clone https://github.com/SantoGuru/Recycle.git*
2. *Entre na pasta do projeto: cd mobile*
3. *No Windows, execute o comando para instalar os pacotes: npm i*
4. *Após isso, obtenha o ipv4 usando o comando no terminal: ipconfig*
5. *Copie o ip e adicione no arquivo config.ts, em API_URL.*
6. *Execute o projeto usando o comando no terminal: npx expo start*
7. *Baixe o aplicativo Expo Go no seu dispositivo móvel.*
8. *Escaneie o QR Code exibido no terminal do seu desktop para executar o aplicativo.*
9. *Após o término do carregamento, o aplicativo deve aparecer no seu dispositivo.*

6.3. ACESSO À APLICAÇÃO IMPLANTADA

- ***Link para Download do APK:*** *Em análise/desenvolvimento.*
- ***Credenciais de Acesso (Perfil de Vendedor):*** *Em análise/desenvolvimento.*

7. CONCLUSÃO

7.1. TRABALHOS FUTUROS

Com a base sólida do aplicativo estabelecida, identificamos diversas oportunidades para evoluir e agregar ainda mais valor ao sistema RochaForte no futuro. As principais propostas são:

- ***Funcionalidades Offline Avançadas:*** *Expandir a capacidade offline para permitir não apenas a consulta, mas também a* ***criação e edição de pedidos*** *sem conexão com a internet. Os dados seriam sincronizados automaticamente com o servidor assim que uma conexão fosse restabelecida.*
- ***Identificação de Lajes por QR Code:*** *Implementar uma funcionalidade que utilize a câmera do dispositivo para ler um QR Code fixado em cada laje de pedra. Isso permitiria ao vendedor ou gerente de estoque acessar*

instantaneamente os detalhes da peça, eliminando a necessidade de busca manual e agilizando o processo de inventário.

- ***Notificações Push em Tempo Real:*** *Desenvolver um sistema de notificações push para alertar os vendedores sobre eventos importantes, como a confirmação de um pedido, a chegada de um novo lote de material ou uma alteração no status de uma laje que ele esteja monitorando.*
- ***Otimização para Tablets e Modo Paisagem:*** *Adaptar a interface do usuário para oferecer uma experiência otimizada em telas maiores, como as de tablets, que são frequentemente utilizados em balcões de vendas. Isso incluiria layouts de duas colunas e melhor aproveitamento do espaço horizontal.*
- ***Versão para a Plataforma iOS:*** *Iniciar o desenvolvimento da versão do aplicativo para iOS, a fim de atender a todos os potenciais usuários da empresa, independentemente do sistema operacional de seus dispositivos móveis.*

7.2. LIÇÕES APRENDIDAS

O desenvolvimento do aplicativo RochaForte foi uma experiência de aprendizado imensa, que nos levou muito além da simples escrita de código. Nossas principais lições aprendidas podem ser divididas em três áreas:

1. ***Desafios Técnicos de Arquitetura:*** *A maior dificuldade técnica que enfrentamos foi, sem dúvida, o gerenciamento de estado e a orquestração de operações assíncronas. Inicialmente, tínhamos dificuldade em manter a interface consistente enquanto os dados eram buscados da API e salvos no banco de dados local. Compreender e aplicar corretamente a arquitetura MVVM, utilizando `StateFlows` para expor o estado da UI a partir da `ViewModel`, foi um divisor de águas. Isso nos ensinou que uma arquitetura bem definida não é opcional, mas sim essencial para criar um aplicativo robusto e manutenível.*
2. ***A Importância da Persistência Local:*** *No início, subestimamos a complexidade de oferecer um suporte offline funcional. Implementar o padrão de Repositório como a "Fonte Única da Verdade" que abstrai a origem dos dados (API ou cache local) foi o nosso maior "Aha! Moment". Aprendemos na prática que um aplicativo moderno não apenas consome uma API, mas gerencia dados de forma inteligente para ser rápido e resiliente, melhorando drasticamente a experiência do usuário.*
3. ***Comunicação entre Equipes (Frontend/Backend):*** *A integração com a API, desenvolvida pela equipe de Web, foi um grande exercício de comunicação. Depender da documentação OpenAPI foi fundamental e nos ensinou o valor*

de ter um "contrato" claro entre o cliente e o servidor. Tivemos momentos em que precisávamos de um campo extra ou de um formato de dado diferente, e a negociação com a outra equipe para evoluir a API foi um aprendizado valioso sobre o desenvolvimento colaborativo no mundo real.

A principal lição que levamos deste projeto é que a qualidade de um aplicativo mobile não está apenas em sua aparência, mas em sua arquitetura resiliente e na forma como ele lida com as incertezas do mundo real, como falhas de rede.