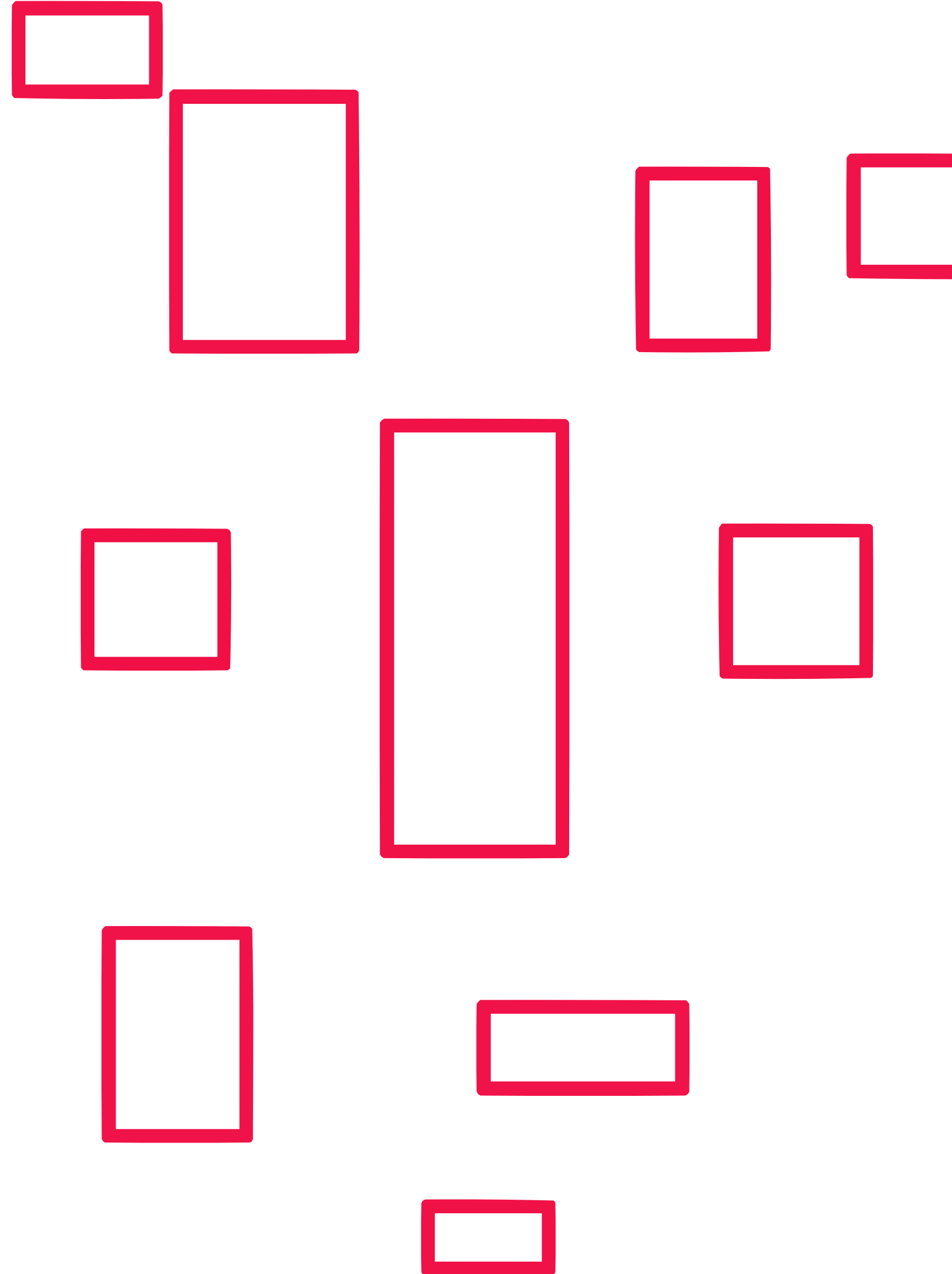


locaweb

CONECTE-SE, INSPIRE-SE E REALIZE



Objetivo

Desenvolvimento de um aplicativo de e-mail com mecanismos básicos de controle de SPAM, integração com serviços externos e interfaces modernas para uma melhor experiência do usuário.

Relação de Tecnologias Utilizadas

Linguagens de Programação

- **Java:** Utilizada para o desenvolvimento do back-end, aproveitando o framework Spring Boot para criar APIs RESTful.
- **Kotlin:** Utilizada para o desenvolvimento do front-end em Android, aproveitando o Jetpack Compose para construir a interface do usuário de forma declarativa e eficiente.

Relação de Tecnologias Utilizadas

Bibliotecas e Frameworks Utilizados

- **Back-end:**

- **Spring Boot:** Framework principal para a construção do back-end. Facilita a criação de APIs RESTful e gerenciamento de dependências.
- **Spring Data JPA:** Para integração com o banco de dados e simplificação das operações CRUD.
- **Spring Boot Starter Web:** Para suportar a criação de APIs REST.
- **Jackson:** Para serialização e desserialização de objetos Java para JSON e vice-versa.
- **Lombok:** Para reduzir o boilerplate de código, como getters, setters, e construtores.
- **Retrofit:** Para consumo de APIs REST no front-end Android.
- **Mailgun:** É uma API para poder limitar a quantidade de e-mails enviados por dia. Solução popular para enviar e-mails de forma programada, com características como: Limite de envio por dia e por hora configuráveis e Ferramentas avançadas de controle de SPAM. Vantagens: Fácil configuração e API robusta para gerenciamento de e-mails.

Relação de Tecnologias Utilizadas

Bibliotecas e Frameworks Utilizados

• Front-end:

- Jetpack Compose: Biblioteca para construção da interface do usuário de forma declarativa no Android.
- Retrofit: Para comunicação com a API do back-end.
- OkHttp: Cliente HTTP utilizado pelo Retrofit para realizar requisições HTTP.

• Bibliotecas de front end das tecnologias utilizadas dependencies:

- implementation "androidx.compose.ui:ui:1.5.0"
- implementation "androidx.compose.material:material:1.5.0"
- implementation "androidx.compose.ui:ui-tooling-preview:1.5.0"
- implementation "com.squareup.retrofit2:retrofit:2.9.0"
- implementation "com.squareup.retrofit2:converter-gson:2.9.0"
- implementation "com.squareup.okhttp3:okhttp:4.10.0"
- implementation "com.squareup.okhttp3:logging-interceptor:4.10.0"

• Banco de Dados:

- Oracle JDBC Driver: Para a conexão com o banco de dados Oracle

Detalhamento do Mecanismo de SPAM

Regras Implementadas

- **Palavra-Chave:** O mecanismo de detecção de SPAM é baseado na identificação de palavras-chave específicas no corpo do e-mail. No código, a palavra-chave utilizada é "Promoção".
- **Serviço de SPAM:** A classe SpamService no back-end realiza a verificação de SPAM. O método verificarSpam verifica se o corpo do e-mail contém a palavra-chave definida.

Problemas Solucionados

- **Identificação Simples de Spam:** O mecanismo permite identificar e-mails com características de SPAM baseadas em palavras-chave simples. Esse método é eficiente para casos básicos onde palavras-chave podem indicar SPAM.
- **Extensibilidade:** A solução pode ser estendida para incluir regras mais complexas ou integrar com APIs externas de detecção de SPAM para uma verificação mais robusta.

Diferenciais da Solução de Back-end e Destaques

Estrutura Modular

O projeto está organizado de forma modular, com separação clara entre controladores, serviços e entidades. Isso facilita a manutenção e escalabilidade do sistema.

- **Controladores:** Implementam endpoints RESTful para interagir com o front-end.
- **Serviços:** Contêm a lógica de negócio, como a verificação de SPAM.
- **Entidades:** Representam as tabelas do banco de dados.
- **Uso de Spring Boot:** Aproveita o Spring Boot para simplificar o desenvolvimento e configuração do back-end. A integração com o Spring Data JPA facilita as operações de banco de dados.
- **Configuração CORS:** Implementação de configuração CORS para permitir que o front-end, executado em um domínio diferente, interaja com o back-end sem problemas de segurança relacionados a origem cruzada.
- **Eficiência no Mecanismo de Controle de SPAM:** O mecanismo básico de controle de SPAM utiliza uma palavra-chave para identificar e-mails indesejados. Este é um ponto inicial simples que pode ser expandido para incluir técnicas mais avançadas ou integração com APIs externas de SPAM.
- **Integração com o Front-end:** Configuração adequada de endpoints REST que permitem a comunicação fluida entre o front-end em Kotlin (Jetpack Compose) e o back-end em Java (Spring Boot).
- **Uso de Retrofit no Front-end:** A biblioteca Retrofit é utilizada no front-end para facilitar a comunicação com o back-end, promovendo uma integração eficiente e simplificada.

API do Calendário e Modo Escuro

Implementação do Modo Escuro no App

- O tema escuro foi implementado utilizando o Jetpack Compose.
- Utiliza gerenciamento de estado para alternar entre o modo claro e escuro.

Estado do Tema

- O estado do tema é controlado por uma variável que indica se o tema atual é claro ou escuro.
- Uma função de alternância permite mudar entre esses dois modos.

Definição das Cores do Tema

- As cores para o modo escuro e claro são definidas em arquivos de recursos (colors.xml).
- As cores incluem elementos como:
 - Cor principal
 - Cor sobreposta para elementos interativos
 - Cor secundária e sua sobreposição



API do Calendário e Modo Escuro

Aplicação Condicional do Tema

- A interface alterna entre o tema claro e escuro com base no valor da variável de estado.
- O sistema se adapta automaticamente ao tema ativo, garantindo que a interface responda corretamente às mudanças.

Integração da API de Calendário com Jetpack Compose

- O calendário é exibido por meio de um componente Dialog no Jetpack Compose.
- A exibição é controlada por uma variável que determina se o calendário deve ser mostrado ou ocultado.

Controle de Exibição do Calendário

- A variável de estado controla quando o calendário é exibido ou ocultado.
- O calendário aparece dentro de um Dialog e pode ser fechado ao clicar fora dele.

API do Calendário e Modo Escuro

Componente do Calendário

- O `CalendarViewComposable` é o componente responsável pela exibição do calendário.
- O calendário ajusta suas cores de acordo com o tema (claro ou escuro).

Configuração de Cores e Comportamento do Calendário

- As cores do calendário mudam dinamicamente para refletir o tema escuro ou claro.
- O componente de calendário nativo do Android foi integrado ao Jetpack Compose, garantindo uma transição suave.

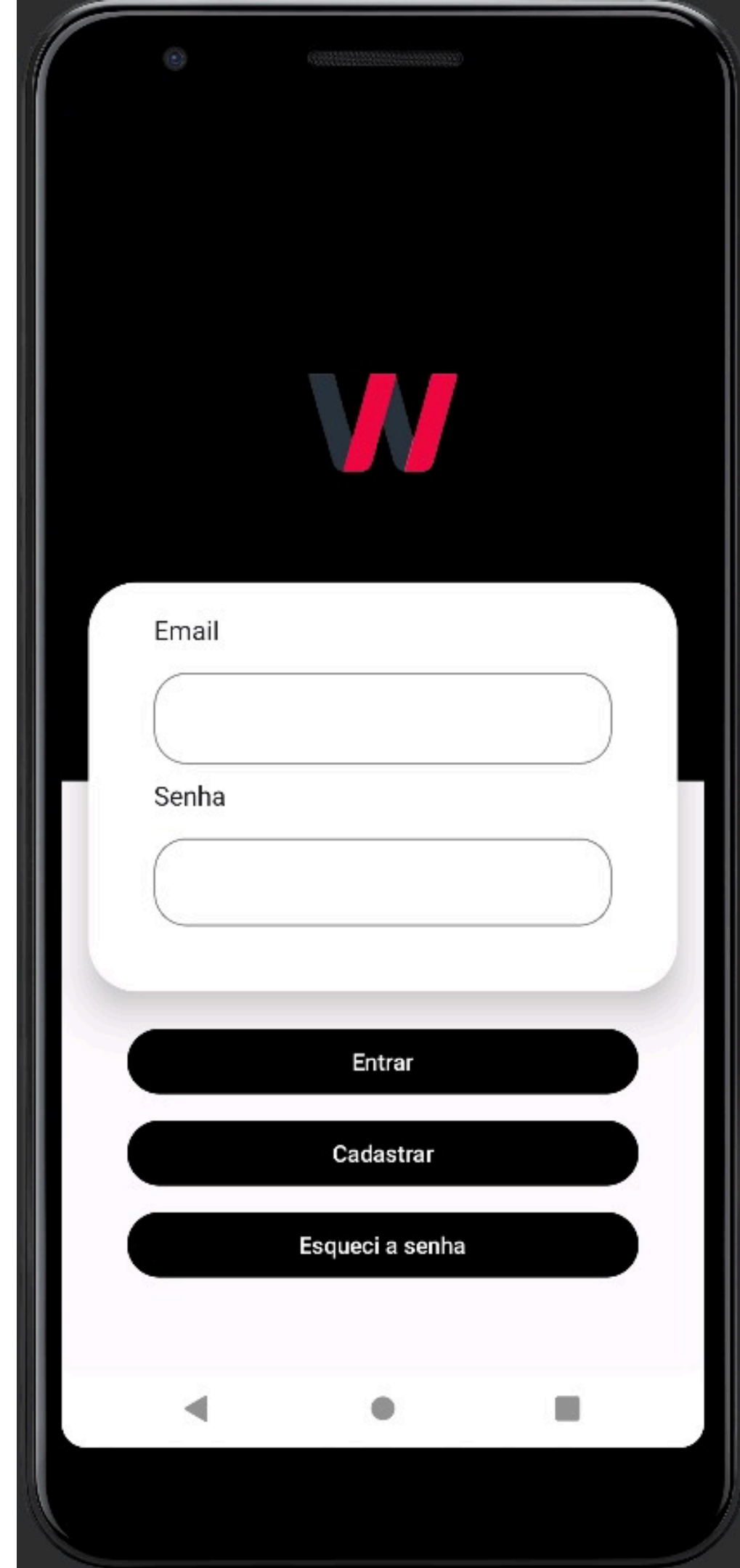
Listener de Data

- Um listener de data está presente para capturar e responder à seleção de datas no calendário.
- A interface responde ao usuário quando uma data é selecionada, podendo disparar ações no app.

Capturas de Tela do Aplicativo

(Tela de login)

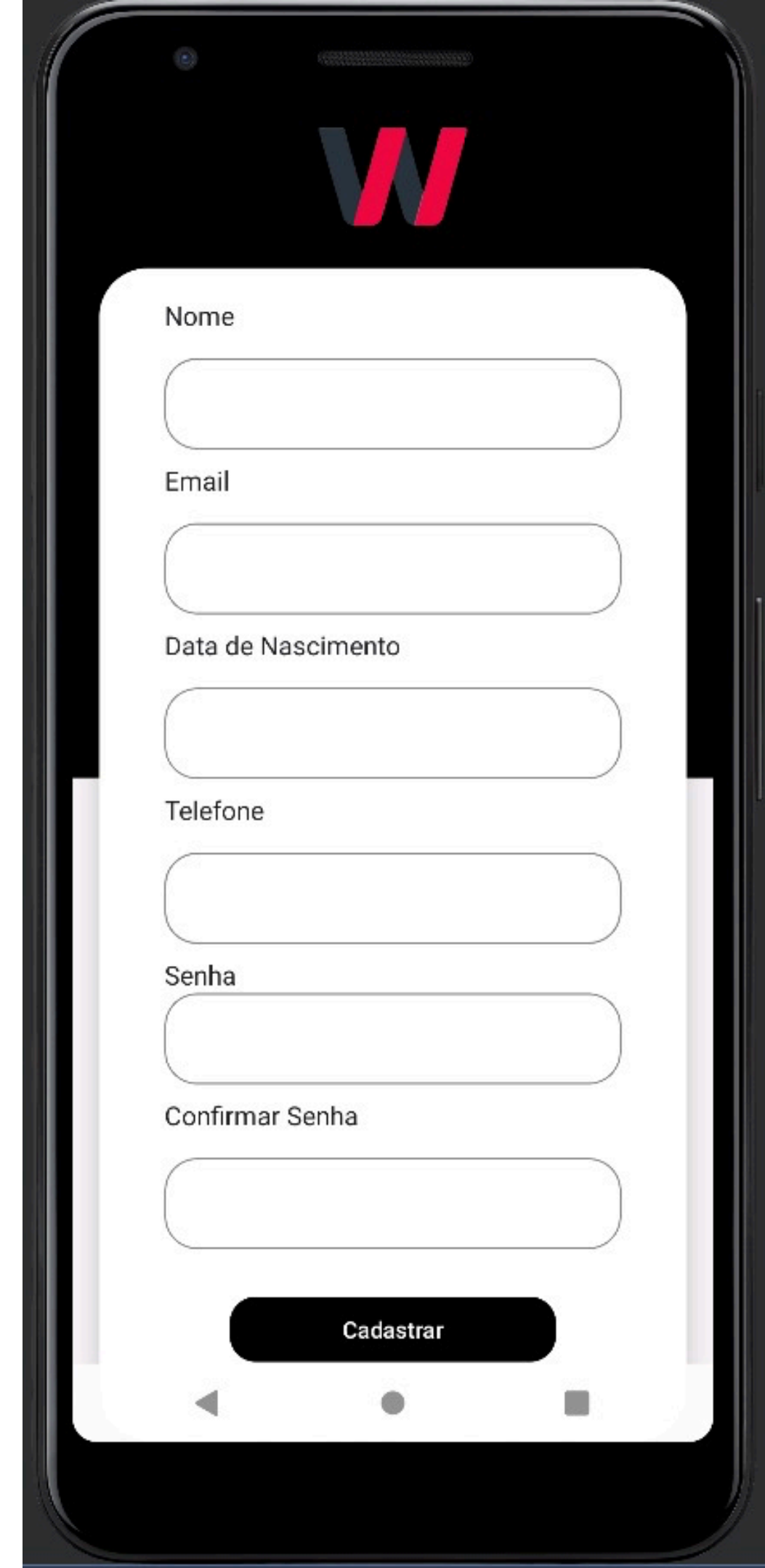
A tela de login do aplicativo de e-mail é simples, moderna e funcional, com campos para inserir o endereço de “E-mail” e a “Senha”, um botão de “Entrar”, botão opcional de “Esqueci a senha” e botão de “Cadastrar”.



Capturas de Tela do Aplicativo

(Tela de cadastro)

A tela de cadastro do aplicativo de e-mail é organizada e minimalista, com campos para inserir nome, e-mail, data de nascimento, telefone, senha e confirmação de senha. Cada campo está claramente identificado e alinhado verticalmente, com um botão de “Cadastrar” destacado ao final. O design garante uma navegação intuitiva e acessível.



A captura de tela mostra a interface de usuário do aplicativo de e-mail na tela de cadastro. No topo, há um logotipo 'W' em vermelho e cinza. Abaixo, os campos de formulário são organizados verticalmente, cada um com um rótulo à esquerda e um campo de entrada arredondado à direita:

- Nome
- Email
- Data de Nascimento
- Telefone
- Senha
- Confirmar Senha

No final da lista de campos, há um botão preto com o texto "Cadastrar" em branco. Na base da tela, são visíveis os ícones de navegação padrão do Android (seta para trás, círculo e quadrado).

Capturas de Tela do Aplicativo

(Tela de envio do código)

A tela apresenta um campo para que o usuário insira o seu Email.



Capturas de Tela do Aplicativo

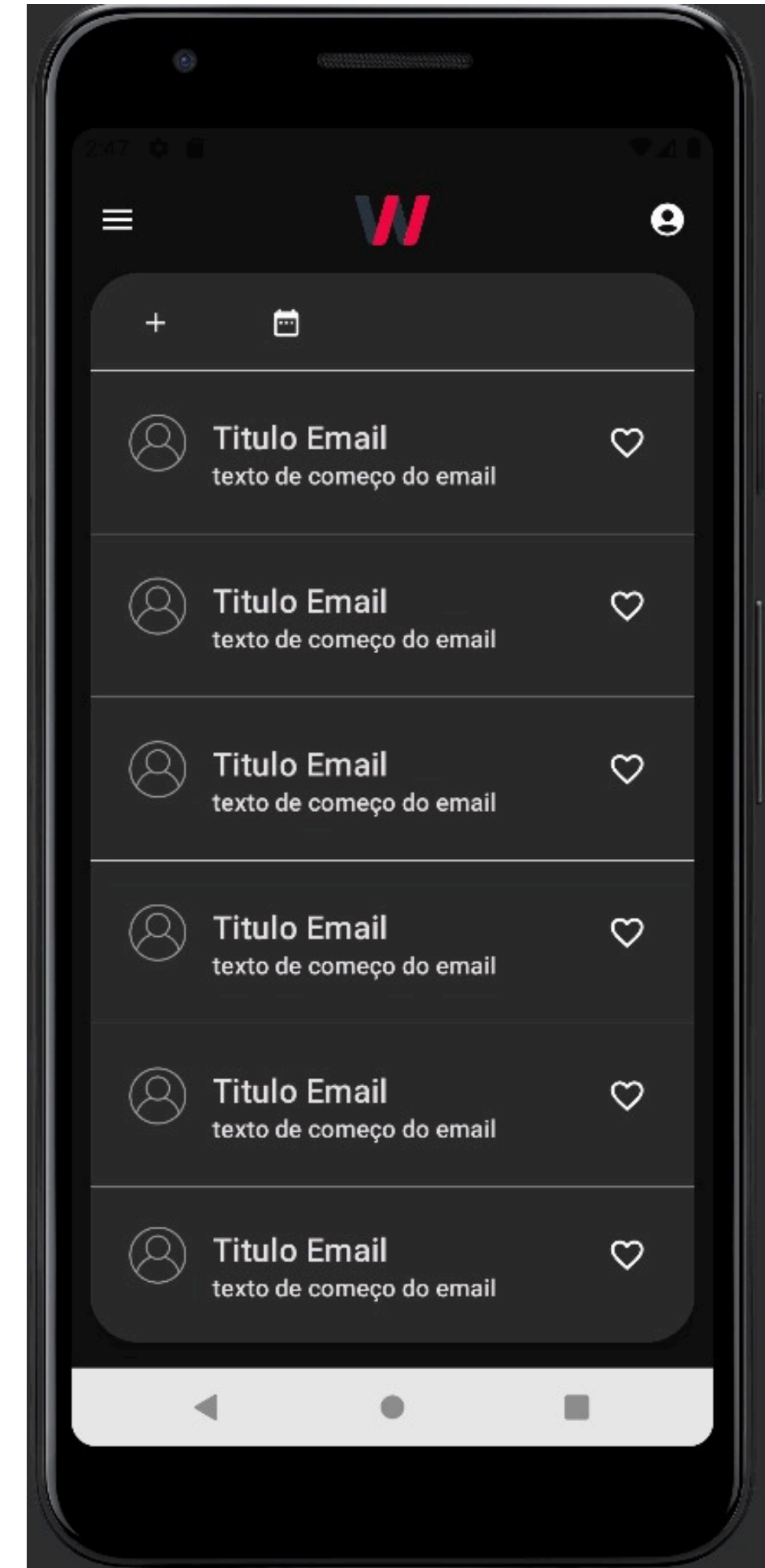
(Telas Home – modo claro e escuro)

A tela home no modo claro é conta com a lista de e-mails disposta de forma clara e fácil de navegar. No topo direito há uma barra de filtro e opções de acessar calendário, envio de e-mail e acessar campo do usuário. Além de ser possível favoritar o e-mail.

(Modo Claro)



(Modo Escuro)

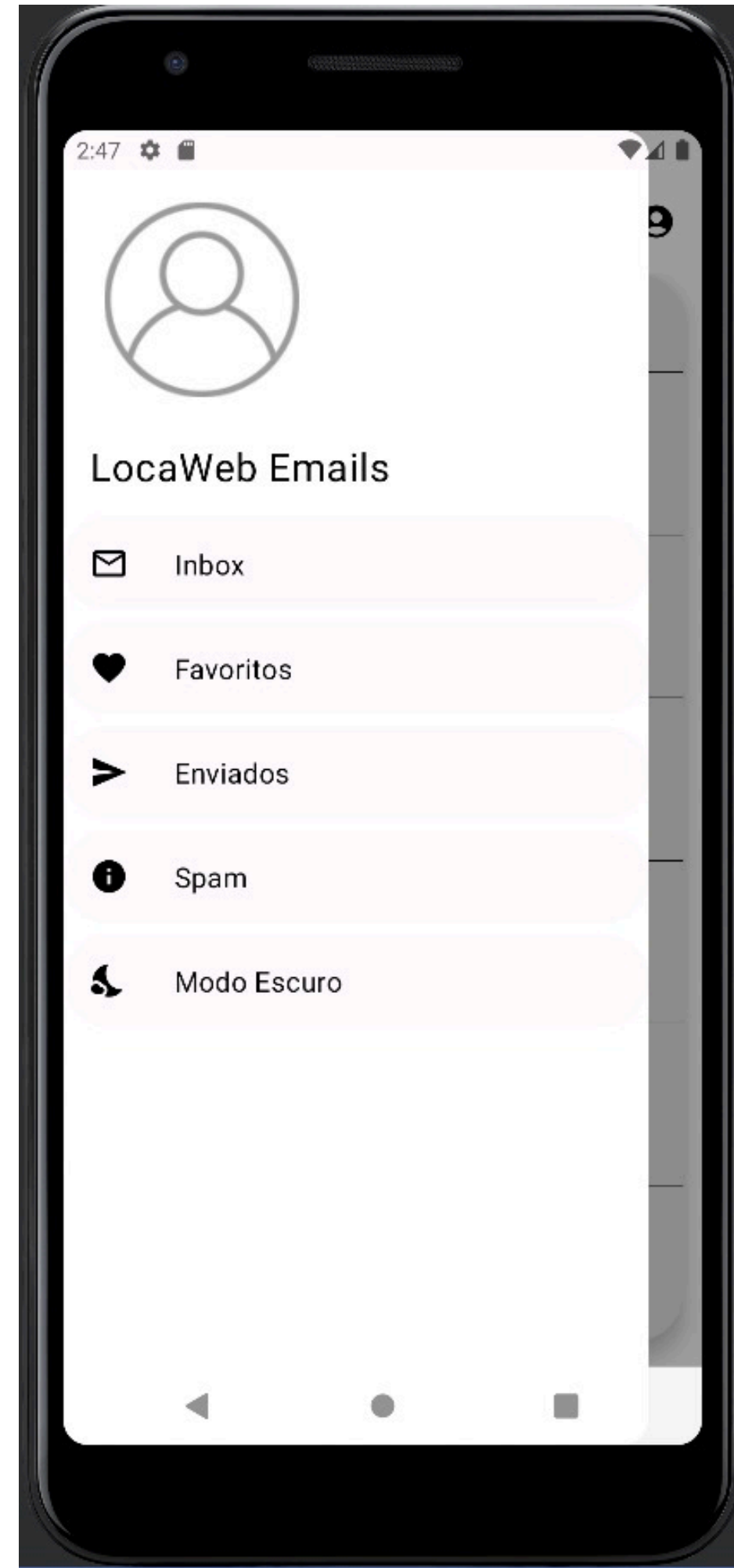


Capturas de Tela do Aplicativo

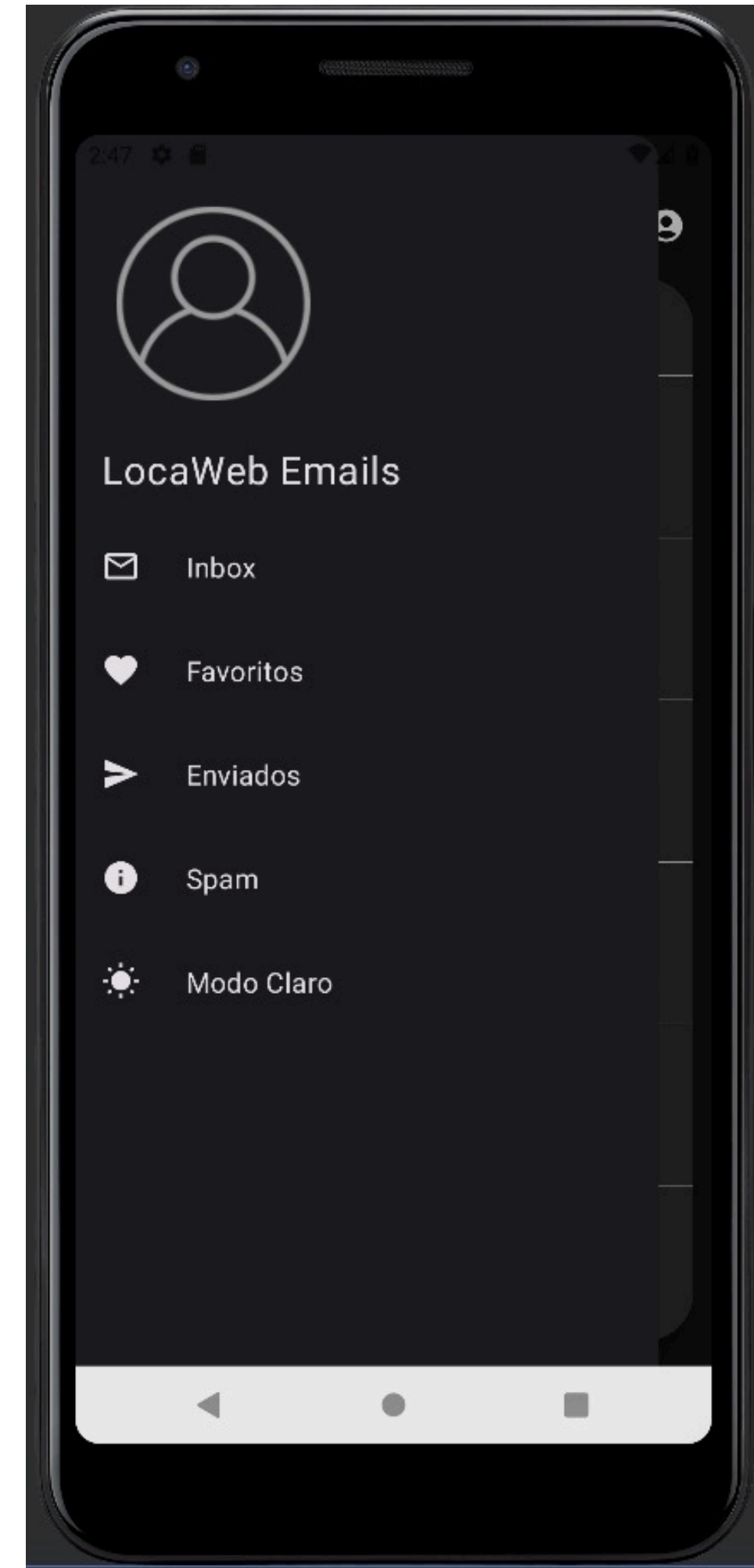
(Telas Drawer – modo claro e escuro)

Esta tela inclui opções como: “Inbox”, “Favoritos”, “Enviados”, e “Spam”, organizadas verticalmente com ícones representativos ao lado. Na parte inferior do drawer, há uma chave para alternar entre o modo claro e o modo escuro, conforme o conforto visual do usuário.

(Modo Claro)



(Modo Escuro)

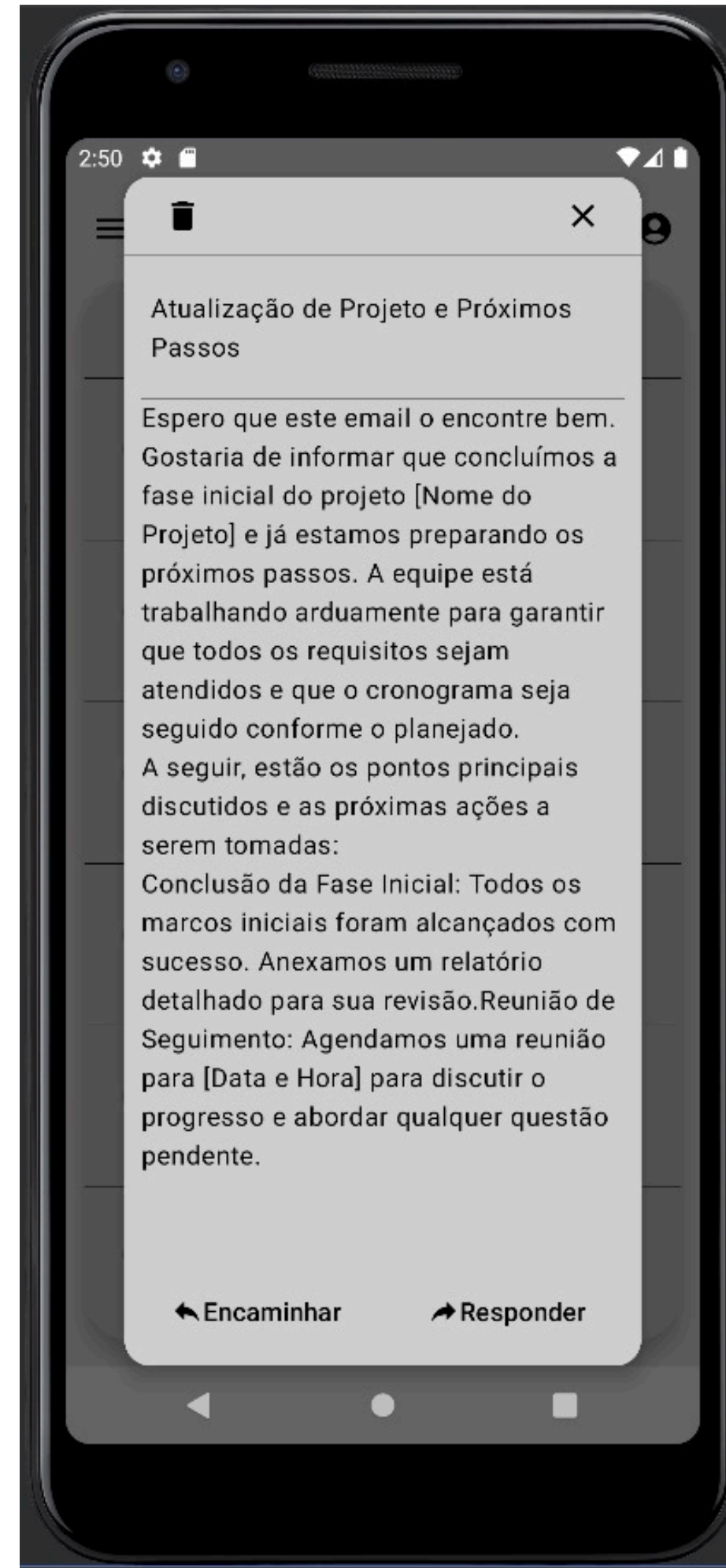


Capturas de Tela do Aplicativo

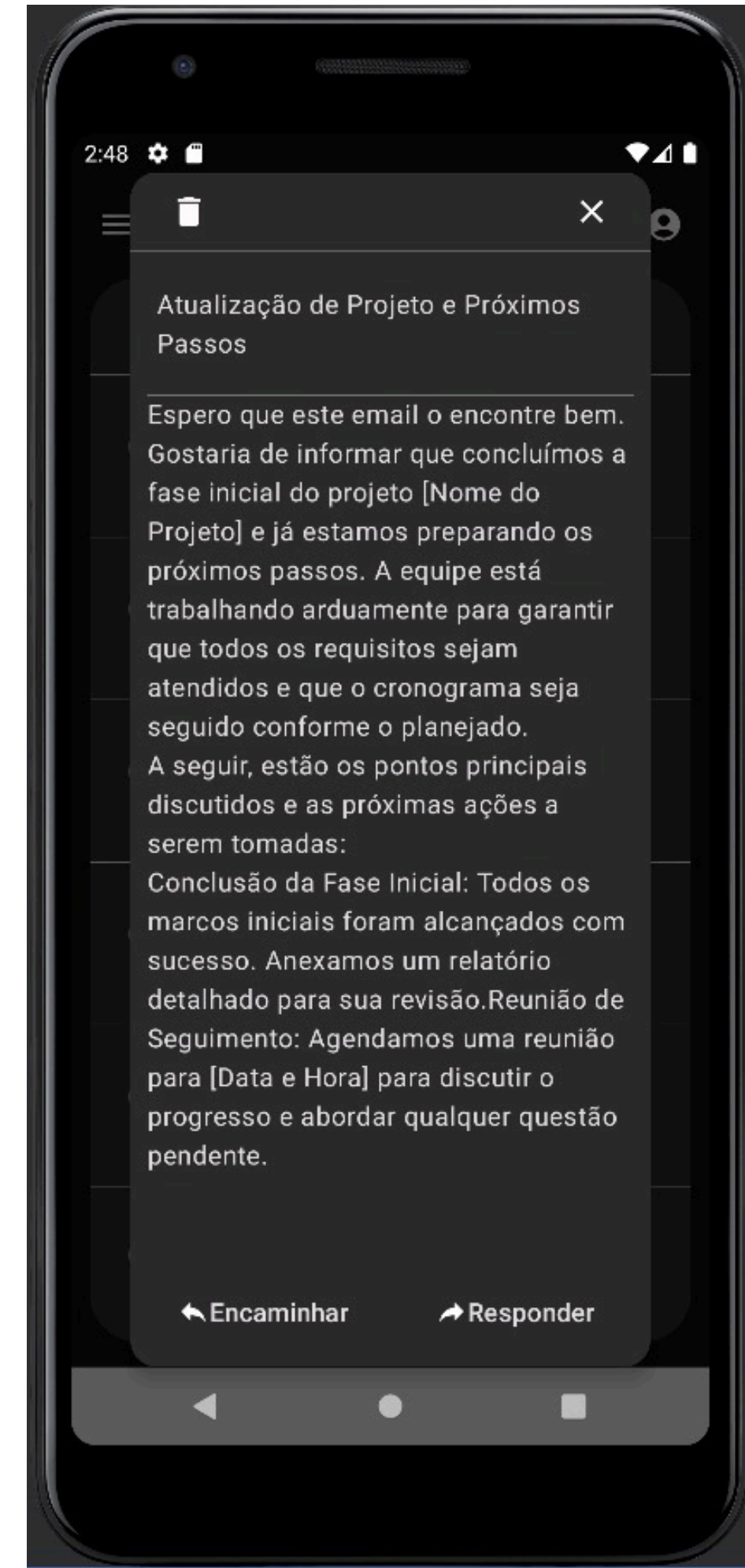
(Telas de leitura – modo claro e escuro)

A tela de leitura exibe o e-mail em destaque com o assunto no topo, seguido pelo corpo do e-mail e opção de fechar o e-mail e jogar na lixeira. Na parte inferior, conta com opção de encaminhar o e-mail e responder.

(Modo Claro)



(Modo Escuro)

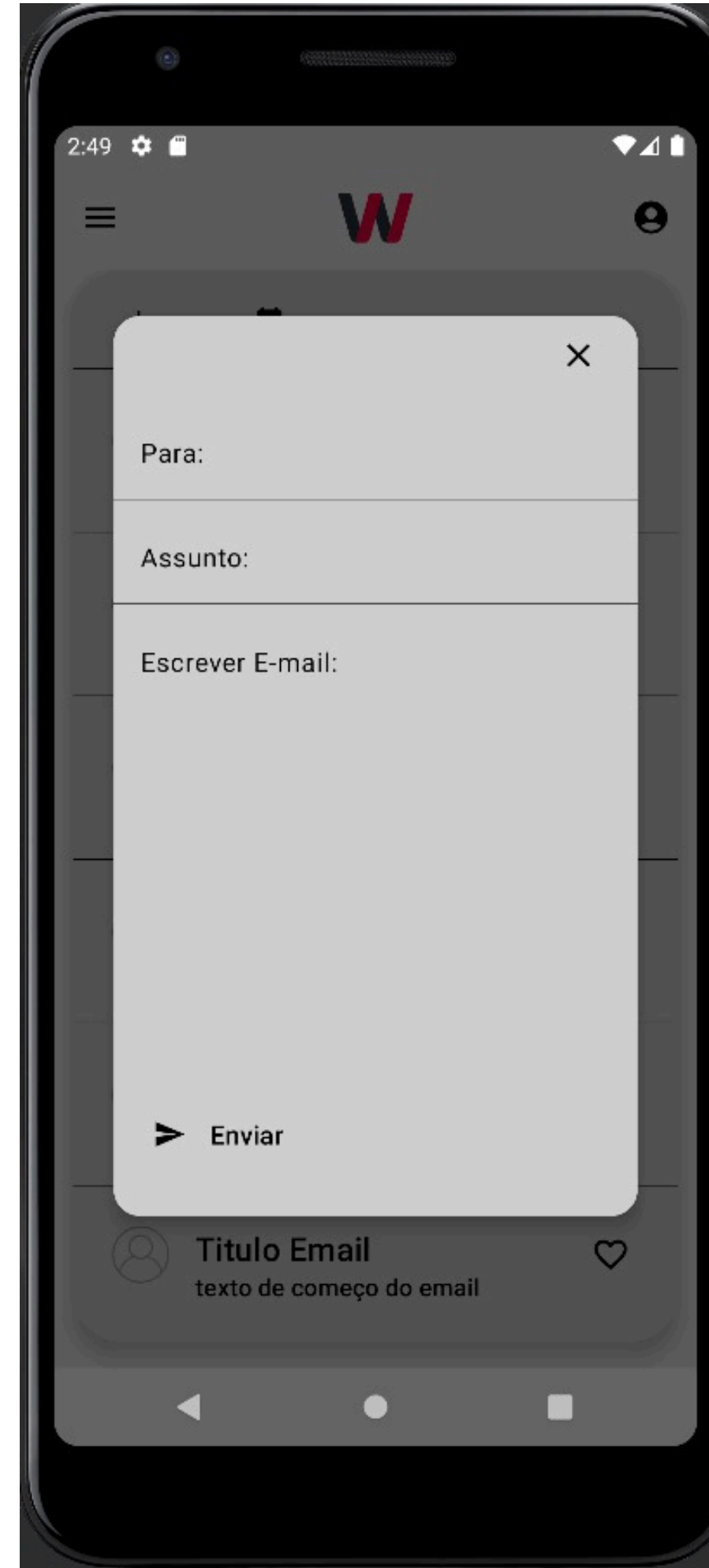


Capturas de Tela do Aplicativo

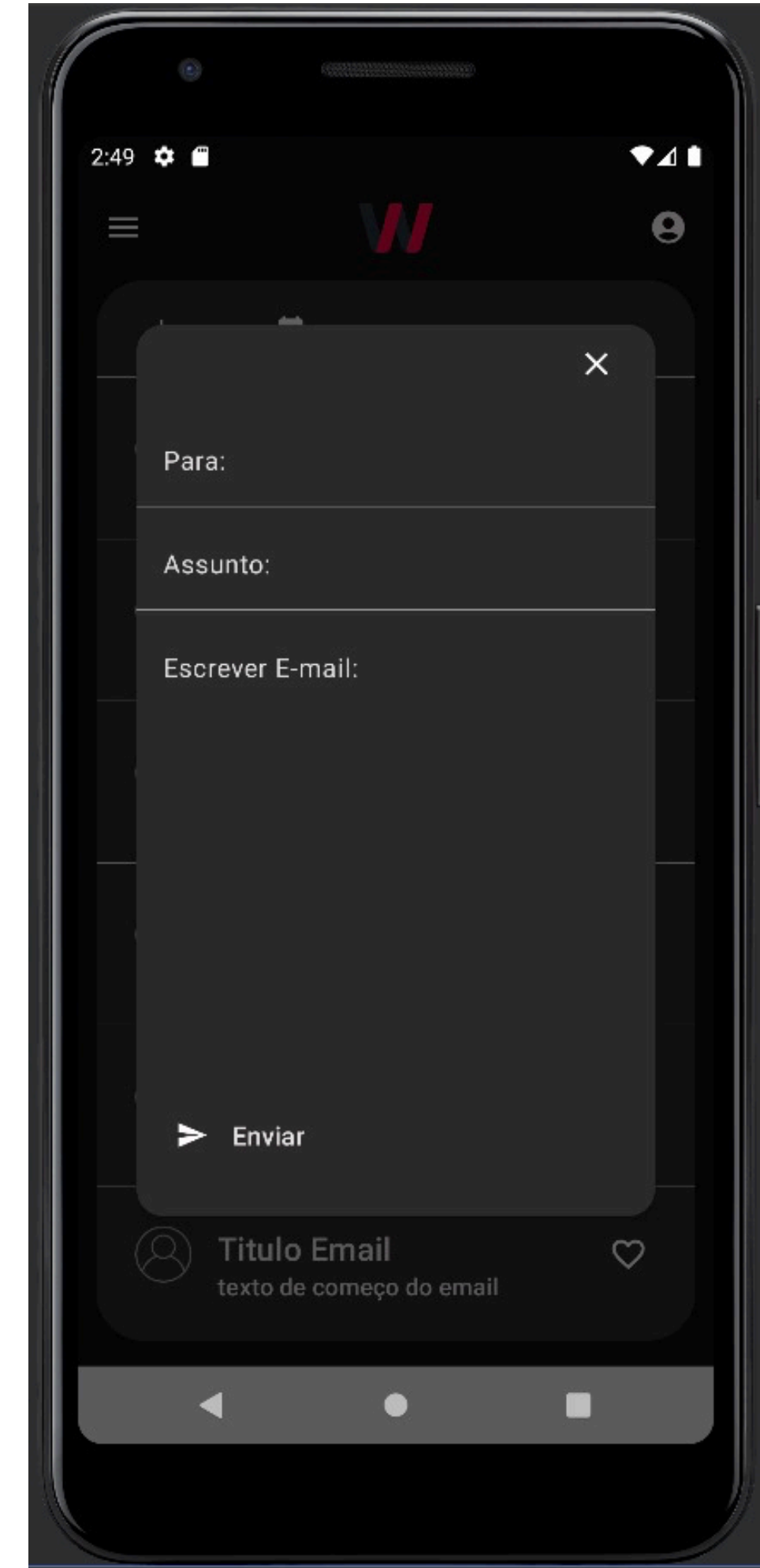
(Tela de envio de e-mail – modo claro e escuro)

A tela de envio de e-mail consta de um campo para adicionar o(s) destinatário(s), campo para o assunto da mensagem e o corpo do e-mail para o usuário escrever o teor da mensagem para seu destino final.

(Modo Claro)



(Modo Escuro)

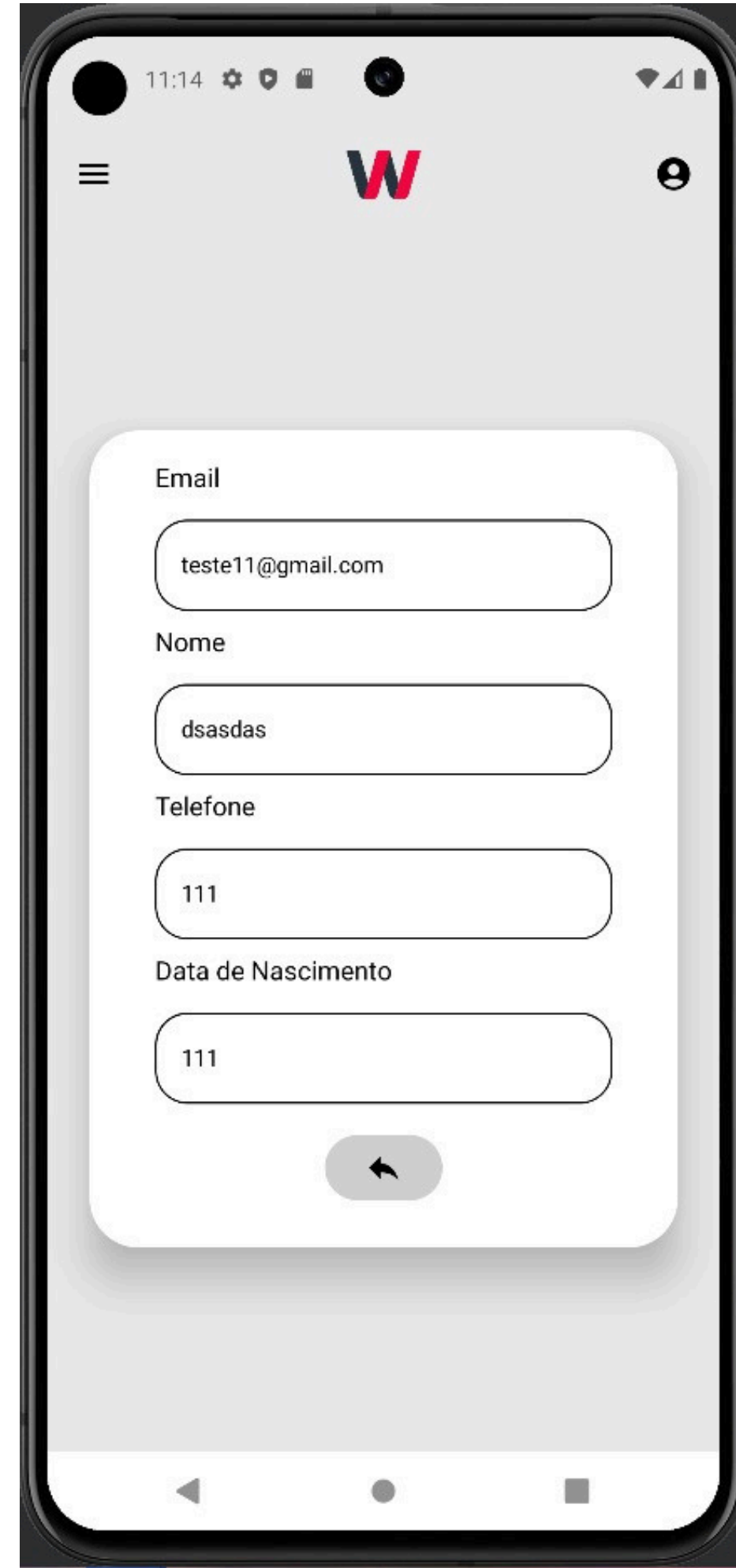


Capturas de Tela do Aplicativo

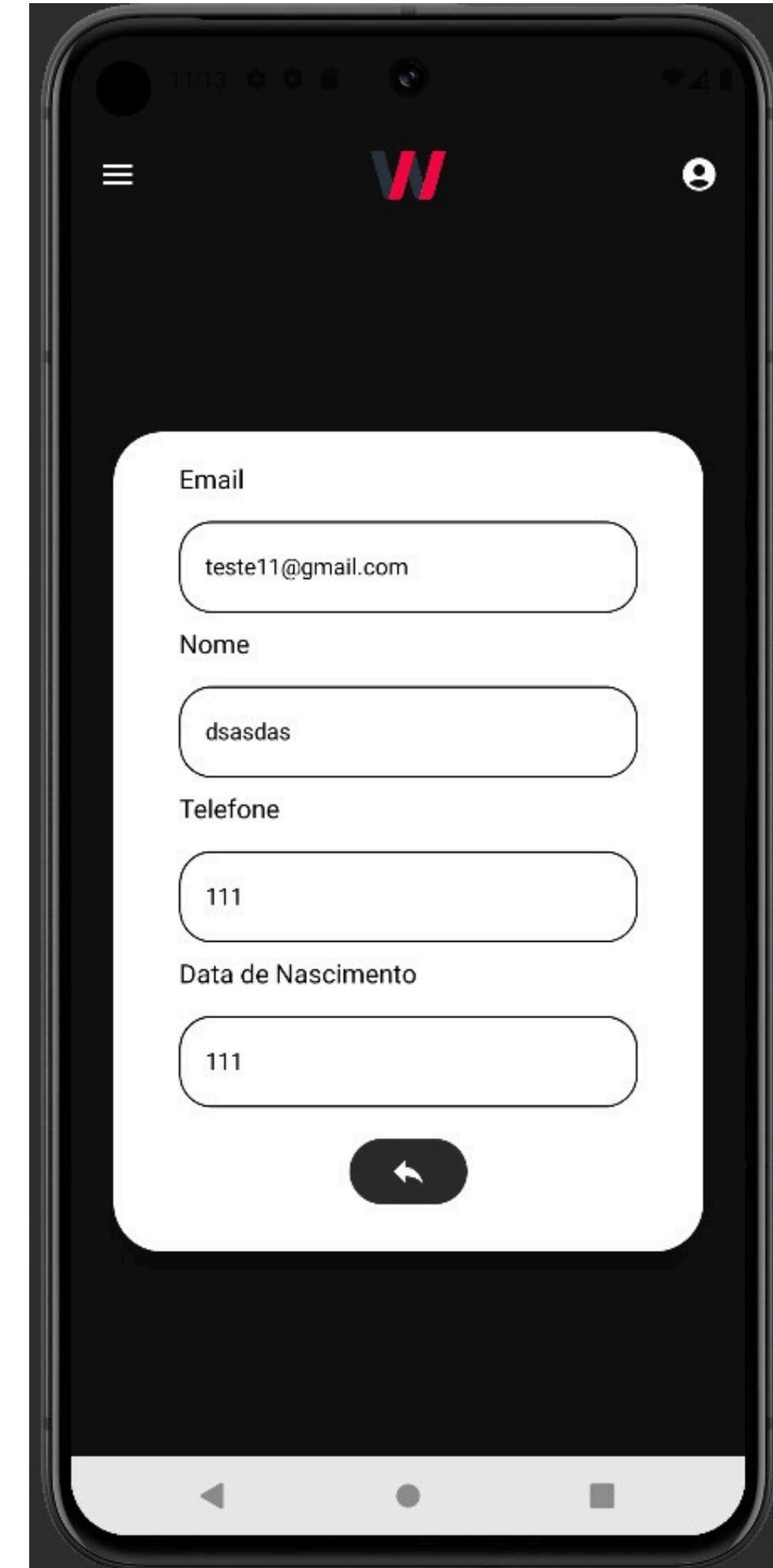
(Tela de perfil - modo claro e escuro)

A tela de perfil consta com a demonstração e opção de alteração dos dados do usuário.

(Modo Claro)



(Modo Escuro)



INTEGRANTES DA EQUIPE

- Bruno Pergentino de Sousa / RM 551618
- Ricardo Bouvier do Nascimento Silva / RM 551676
- Ronaldo César Del Papa Bofe / RM 551469
- Thainá Lopes – RM 552572
- Thiago Gomes da Silva – RM 552464