APP NUTRICONECTA EM FRAMEWORK FLUTTER

APP NUTRICONECTA IN FRAMEWORK FLUTTER

Ryan Pablo Ramos dos Santos [1]
Yuri Henrique Rezende [2]
Daniel Filipe Vieira[3]

Data da versão final: 10 de junho de 2024.

RESUMO

Aplicativos de nutrição desempenham um papel fundamental para a sociedade, não apenas para um público-alvo específico, mas também para uma sociedade que pretende aprimorar seu conhecimento no âmbito nutricional. Esses aplicativos variam amplamente em termos de qualidade e responsividade, alguns com muitas propagandas e pouco conteúdo, entre outros com muito conteúdo e falta de intuitividade que atrapalham a navegabilidade do usuário. Nesse contexto, esse trabalho desenvolveu um app gratuito, amigável e estável para smartphones do tipo Android usando o framework Flutter.

Palavras-chave: linguagem dart; flutter; mobile; app.

ABSTRACT

Nutrition apps play a fundamental role for society, not only for a specific target audience, but also for a society that wants to improve its nutritional knowledge. These apps vary widely in terms of quality and responsiveness, some with a lot of advertising and little content, and others with a lot of content and a lack of intuitiveness that hinders user navigability. In this context, this work has developed a free, user-friendly and stable app for android smartphones using the Flutter framework.

Keywords: language dart; flutter; mobile; app.

1 INTRODUÇÃO

Hoje em dia existe uma gama de aplicativos de alimentação com ênfase na saúde alimentar, com diversas integridades como dietas, receitas e ligação com nutricionistas particulares, mas nem todos recebem um feedback bom do app, por conta de travamento, propaganda, e sem estar otimizado.

Pelo fato desses apps serem chatos e com muita informação, o projeto teve a ideia de ser interativo, possui um chat de conversa, ter receitas e ser algo divertido e saudável.

[1] Graduando em Análise e Desenvolvimento de Sistemas, Faculdade de Tecnologia SENAI “Roberto Mange”. ryanpablo1687@gmail.com
[2] Graduando em Análise e Desenvolvimento de Sistemas, Faculdade de Tecnologia SENAI “Roberto Mange”. yuri.henrique.rezende@outlook.com.br
[3] Professor mestre em Engenharia Elétrica, Faculdade de Tecnologia SENAI “Roberto Mange”. Daniel.vieira@sp.senai.br

2 REVISÃO DE LITERATURA

Flutter é um framework de desenvolvimento de interfaces criado pelo Google, escrito na linguagem Dart. Ele permite a criação de aplicativos nativos para Android, iOS, Web e desktop com uma única base de código, oferecendo uma abordagem multiplataforma. O Flutter destaca-se pelo uso de widgets altamente customizáveis e pela capacidade de renderizar interfaces de usuário diretamente na tela, sem depender dos componentes nativos do sistema operacional.

Dart é uma linguagem de programação criada pelo Google, projetada inicialmente para o desenvolvimento de aplicativos web, mas que ganhou destaque com o Flutter. Ela é uma linguagem orientada a objetos, com sintaxe semelhante a outras linguagens populares, como JavaScript e Java, o que facilita a curva de aprendizado para desenvolvedores.

Firebase é uma plataforma de desenvolvimento de aplicativos móveis e web oferecida pelo Google, que fornece uma ampla gama de serviços para facilitar a criação e o gerenciamento de aplicativos. Ele é muito usado para simplificar o backend de aplicativos, especialmente em projetos de startups e de equipes menores que precisam de uma solução escalável e integrada. A plataforma oferece ferramentas para desenvolvimento, análise e monetização, permitindo que desenvolvedores se concentrem no desenvolvimento de recursos sem precisar se preocupar com a infraestrutura.

Android Studio é o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicativos Android, desenvolvido pelo Google. Ele é baseado no IntelliJ IDEA, uma popular IDE para Java, mas vem com ferramentas e recursos específicos para criar, testar e depurar aplicativos Android de forma eficiente. Desde seu lançamento em 2013, o Android Studio se tornou o principal ambiente de desenvolvimento para Android devido à sua integração completa com o SDK Android e ferramentas adicionais de produtividade.

Android é um sistema operacional desenvolvido pelo Google e é amplamente utilizado em dispositivos de várias marcas, como Samsung, Motorola e Xiaomi. Baseado em Linux, ele permite uma grande flexibilidade para personalização e compatibilidade com uma vasta gama de aparelhos. A programação para Android é realizada principalmente em Java e Kotlin, utilizando a ferramenta Android Studio como IDE oficial.

iOS é um sistema operacional desenvolvido pela Apple, exclusivo para dispositivos da marca, como iPhones e iPads. Ele oferece uma experiência unificada, uma vez que os desenvolvedores precisam otimizar seus aplicativos para uma gama limitada de dispositivos. As linguagens de programação mais comuns para iOS são Swift e Objective-C, e os desenvolvedores utilizam o Xcode como ferramenta oficial de desenvolvimento.

3 METODOLOGIA

O desenvolvimento do aplicativo foi feito em linguagem dart no framework flutter, com o uso de diversas bibliotecas para o desenvolvimento do projeto com suas versões devidamente otimizadas (Figura 1).

Figura 1 – Arquivo pubspec.yaml.

```yaml
name: app_base
description: "A new Flutter project."
publish_to: 'none'
version: 1.0.0+1

environment:
  sdk: '>=3.3.4 <4.0.0'
dependencies:
  flutter:
    sdk: flutter

  cupertino_icons: ^1.0.6
  firebase_core: ^3.4.0
  firebase_auth: ^5.2.0
  cloud_firestore: ^5.4.0
  google_fonts: ^6.2.1
  firebase_storage: ^12.2.0
  image_picker: ^1.1.2
  file_picker: ^8.0.5
  flutter_launcher_icons: ^0.10.0
  font_awesome_flutter: ^10.0.0
  win32: ^5.7.1
  carousel_slider: ^5.0.0
  flutter_lucide: ^1.4.0
  lucide_icons: ^0.257.0
  lottie: ^3.1.3

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^3.0.0
```

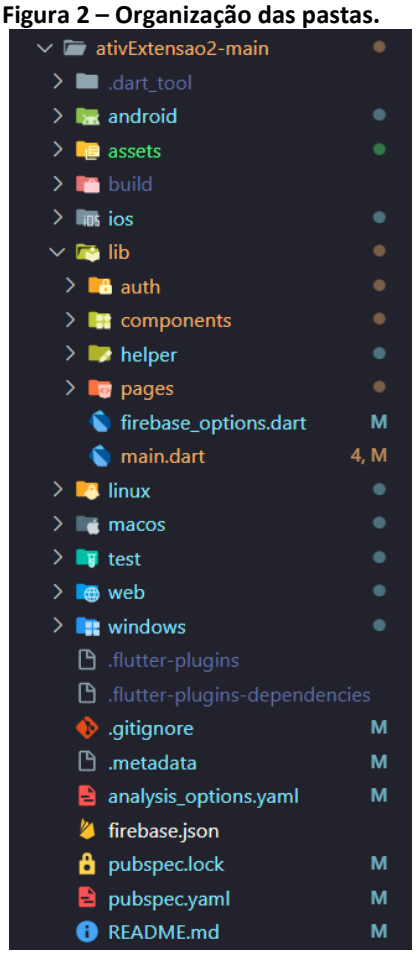
Fonte: (Própria, 2024)

O projeto foi complexo e foi criado múltiplas telas com isso pastas e subpastas foi criada para organizar o projeto e ficar mais fácil para o programador ou tester fazer as alterações e estudar o projeto.

A criação da pasta Lib (Figura 2) tem como objetivo a criação da pasta do firebase, das páginas de cada tela, autenticação do login e componentes para customização dele.

Sendo assim o projeto ficou mais limpo e facilita no low-code tendo em vista o número de pastas que já tem junto com o projeto quando cria um projeto base flutter.

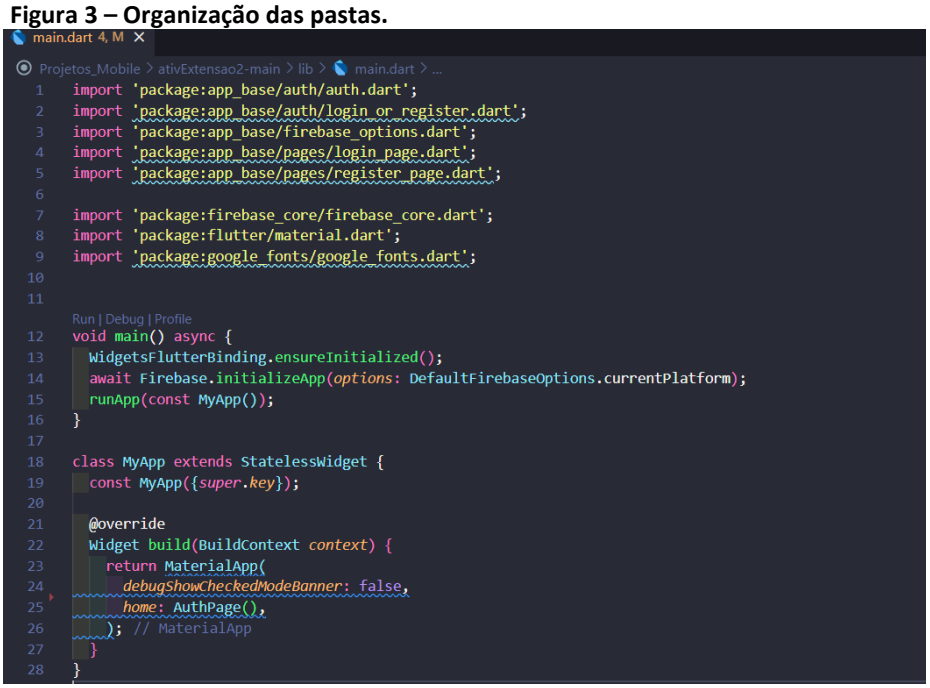
Figura 2 – Organização das pastas.



Fonte: (Própria, 2024)

O arquivo “main”, tem como objetivo iniciar o projeto e toda “lib” (figura 3).

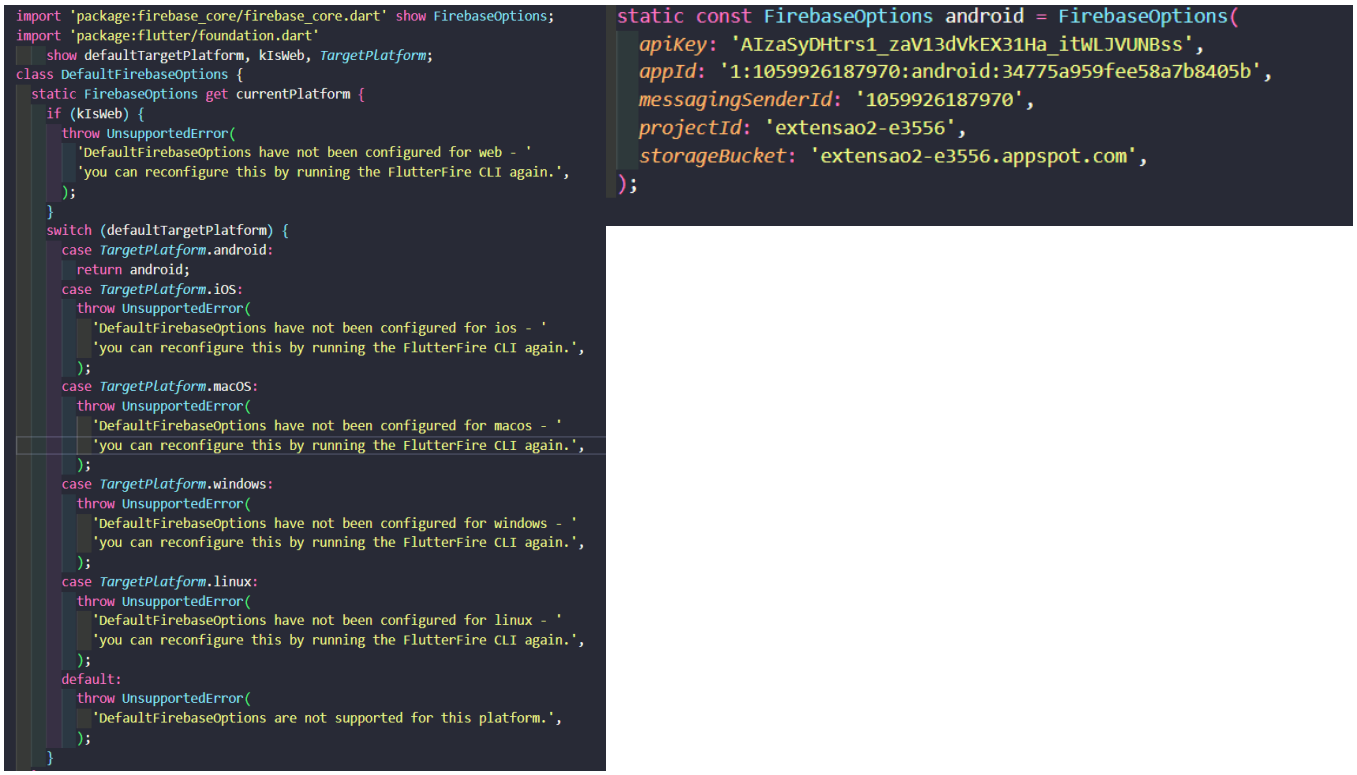
Figura 3 – Organização das pastas.



Fonte: (Própria, 2024)

Foi importado as autenticações e as configuração do firebase (Figura 4).

Figura 4 – Configuração FireBase.



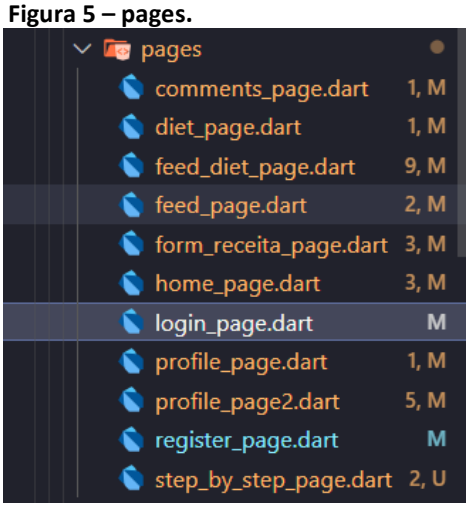
```dart
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for web - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for ios - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      case TargetPlatform.macOS:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for macos - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      case TargetPlatform.windows:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for windows - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      case TargetPlatform.linux:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for linux - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      default:
        throw UnsupportedError(
          'DefaultFirebaseOptions are not supported for this platform.',
        );
    }
```

```dart
static const FirebaseOptions android = FirebaseOptions(
  apiKey: 'AIzaSyDHtrs1_zaV13dVkEX31Ha_itWLJVUNBss',
  appId: '1:1059926187970:android:34775a959fee58a7b8405b',
  messagingSenderId: '1059926187970',
  projectId: 'extensao2-e3556',
  storageBucket: 'extensao2-e3556.appspot.com',
);
```

Fonte: (Própria, 2024)

Depois de tudo pronto e importado, foi feitos todas as telas na pasta pages (Figura 5).

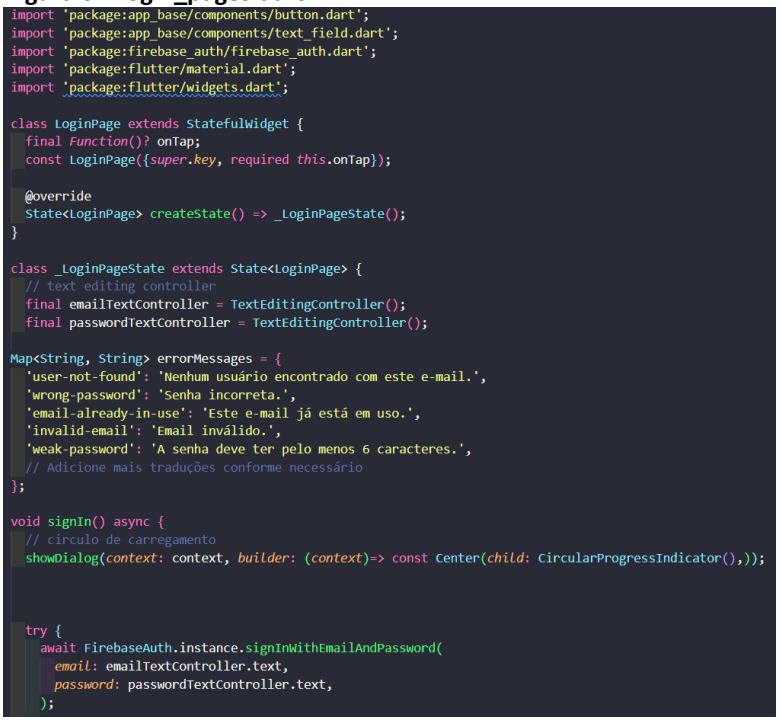
Figura 5 – pages.



Fonte: (Própria, 2024)

Esse código Flutter (Figura 6) cria uma tela de login usando Firebase Authentication, ele define controladores para capturar email e senha, além de um mapa de mensagens de erro para exibir feedback amigável. A função signIn tenta autenticar o usuário, exibindo um indicador de carregamento e, em caso de erro, mostra uma mensagem de erro em um AlertDialog. A interface inclui campos de entrada, um botão de login, uma mensagem de boas-vindas e uma opção para redirecionar o usuário para a página de registro.

Figura 6 – login_pages.dart.



```dart
import 'package:app_base/components/button.dart';
import 'package:app_base/components/text_field.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter/widgets.dart';

class LoginPage extends StatefulWidget {
  final Function()? onTap;
  const LoginPage({super.key, required this.onTap});

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  // text editing controller
  final emailTextController = TextEditingController();
  final passwordTextController = TextEditingController();

Map<String, String> errorMessages = {
  'user-not-found': 'Nenhum usuário encontrado com este e-mail.',
  'wrong-password': 'Senha incorreta.',
  'email-already-in-use': 'Este e-mail já está em uso.',
  'invalid-email': 'Email inválido.',
  'weak-password': 'A senha deve ter pelo menos 6 caracteres.',
  // Adicione mais traduções conforme necessário
};

void signIn() async {
  // circulo de carregamento
  showDialog(context: context, builder: (context)=> const Center(child: CircularProgressIndicator(),));



  try {
    await FirebaseAuth.instance.signInWithEmailAndPassword(
      email: emailTextController.text,
      password: passwordTextController.text,
    );
```

Fonte: (Própria, 2024)

Sobre a página principal foi a HomePage (Figura 7) tem uma barra de navegação (drawer) personalizada com botões que redirecionam para a página de perfil, feed e dietas. Funções como goToProfilePage, goToFeedPage, e goToDietPage gerenciam essa navegação. A interface inclui um banner, ícones interativos para acessar o feed, dietas e cardápio, além de um botão de logout (signOut) para sair do Firebase.

Figura 7 – home_page.dart.

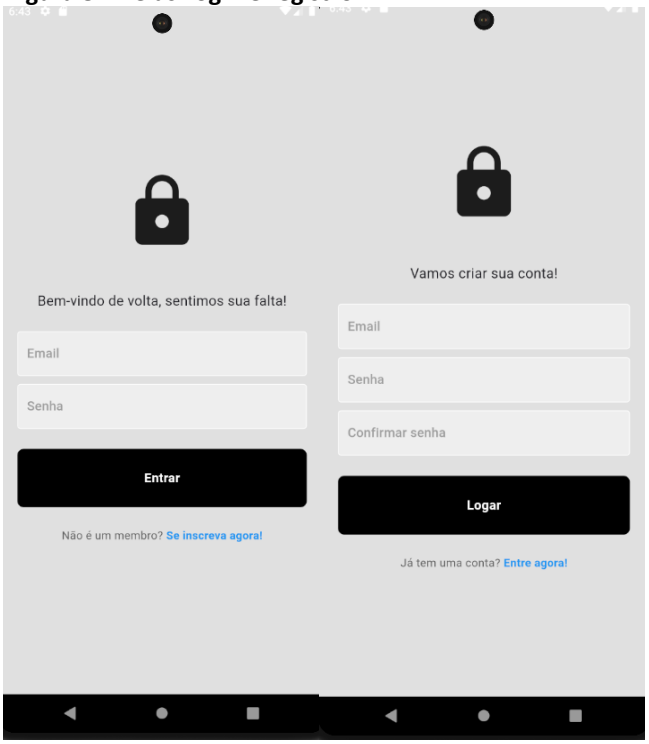
```dart
import 'package:app_base/components/alt_button.dart';
import 'package:app_base/components/drawer.dart';
import 'package:app_base/pages/diet_page.dart';
import 'package:app_base/pages/feed_page.dart';
import 'package:app_base/pages/profile_page.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:lottie/lottie.dart';
import 'package:lucide_icons/lucide_icons.dart';

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
late final AnimationController _controller;

@override


  void goToProfilePage() {
    Navigator.pop(context);

    Navigator.push(
        context, MaterialPageRoute(builder: (context) => ProfilePage()));
  }

  void goToFeedPage() {
    Navigator.pop(context);

    Navigator.push(
        context, MaterialPageRoute(builder: (context) => FeedPage()));
  }

  void goToDietPage() {
    Navigator.pop(context);
```

Fonte: (Própria, 2024)

4 RESULTADOS E DISCUSSÕES

Para verificar o funcionamento do app, foi instalado em 5 celulares, 2 IOS e 3 Android, ao abrir o app se inicia na tela de login com a possibilidade do usuario fazer login ou registrar (Figura 8).

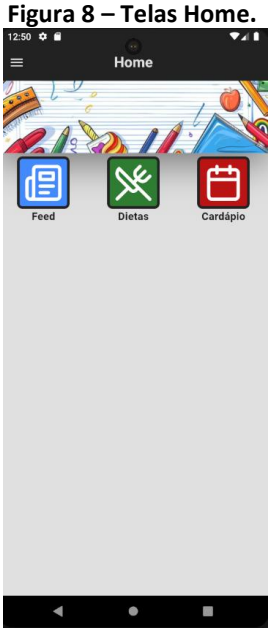
Figura 8 – Telas Login e registro.



Fonte: (Própria, 2024)

Depois que o usuário faz o registro ele e direcionado a tela home, nisso tem acesso as 3 funções do app, ao Feed Dietas e Cardápio

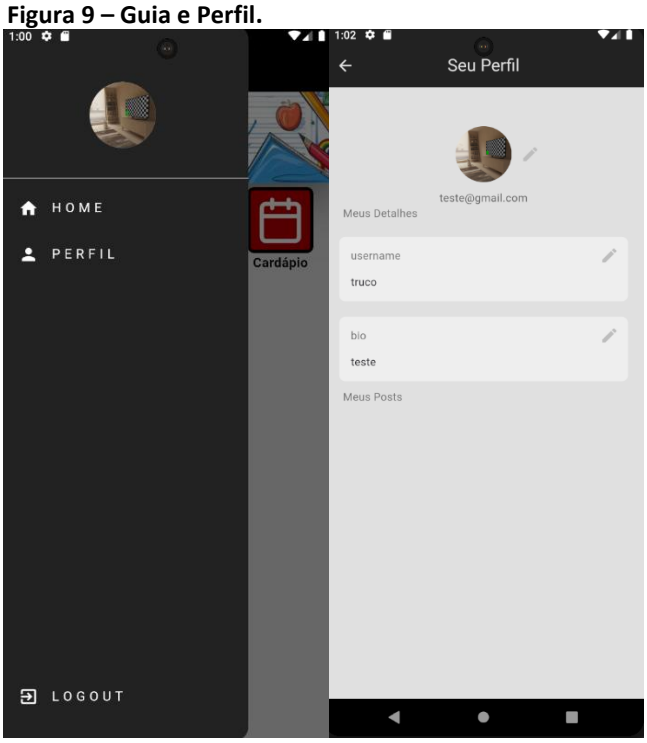
Figura 8 – Telas Home.



Fonte: (Própria, 2024)

O usuário pode acessar seu perfil ou sair e fazer as alterações (Figura 9).

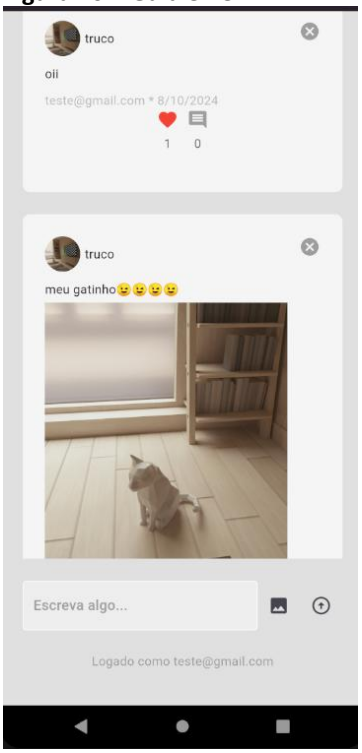
Figura 9 – Guia e Perfil.



Fonte: (Própria, 2024)

Na tela home quando o usuário acessa Feed, poderá publicar que nem em uma rede social, imagens e escrever comentários (Figura 10).

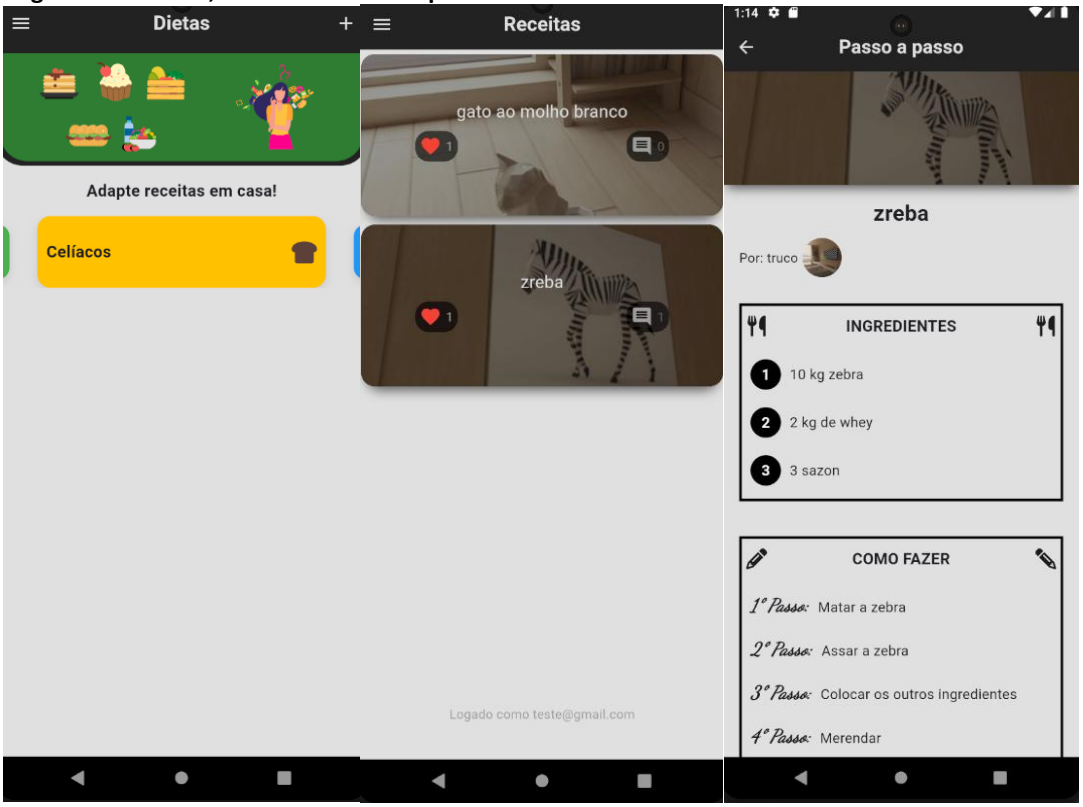
Figura 10 – Guia e Perfil.



Fonte: (Própria, 2024)

Em dietas o usuário pode compartilhar suas receitas ou ler para preparar em sua residência (Figura 11).

Figura 11 – Dietas, Receitas e Passo a passo.



Fonte: (Própria, 2024)

Está em desenvolvimento a opção Cardápio, na qual a nutricionista local irá publicar receitas saudável e fomentar a rede.

5 CONCLUSÃO

Após a validação da metodologia e dos conceitos utilizados, com base nos resultados obtidos, novas pessoas com interesse em trocar ideias e ter uma vida saudável poderão compartilhar para as outras pessoas da comunidade e ter uma sociedade com foco em bem-estar e qualidade de vida.

O app conseguiu promover o bem-estar com telas interativas e ser responsivo com as telas e funcionar em sistemas operacionais como Android e IOS. Foi validado o chat de conversa que foi implementado para ser interativo, o tom de cores equilibrados e brilho.

Com esse compartilhamento promovera uma cultura de crescimento pessoal e incentivo de práticas e fortalecimento de laços sociais.

REFERENCIAS

GOOGLE. **Flutter: beautiful native apps in record time. Documentação oficial.** Disponível em: https://docs.flutter.dev. Acesso em: 23 jul. 2024.

LADDIS, C.; CATALIN, L. **Dart Apprentice: Fundamentals, Functionality and the Foundation for Flutter Development.** New York: Razeware, 2023.

GOOGLE. **Android Studio User Guide. Documentação oficial do Android Studio.** Disponível em: https://developer.android.com/studio/intro. Acesso em: 07 set. 2024.

APPLE INC. **iOS Human Interface Guidelines. Documentação oficial para desenvolvimento iOS.** Disponível em: https://developer.apple.com/design/human-interface-guidelines. Acesso em: 03 set. 2024.

GOOGLE. **Firebase Documentation**. Documentação oficial do Firebase. Disponível em: https://firebase.google.com/docs. Acesso em: 28 ago. 2024.