Formulario e Confirmação com Persistencia de dados

import 'package:flutter/material.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

void main() {

runApp(const MyApp());

}

// Widget principal que inicializa o app e define a página inicial (HomePage)

class MyApp extends StatelessWidget {

const MyApp({super.key});

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Cadastro e Login',

theme: ThemeData(

colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),

useMaterial3: true,

),

home: const HomePage(), // Define HomePage como a tela inicial

);

}

}

// Página inicial que oferece as opções de cadastro e login

class HomePage extends StatelessWidget {

const HomePage({super.key});

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: const Text('Bem-vindo'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

// Botão para navegar para a página de cadastro

ElevatedButton(

onPressed: () {

Navigator.of(context).push(

MaterialPageRoute(

builder: (context) => const RegistrationPage(),

),

);

},

child: const Text('Cadastrar E-mail'),

),

const SizedBox(height: 20),

// Botão para navegar para a página de login

ElevatedButton(

onPressed: () {

Navigator.of(context).push(

MaterialPageRoute(

builder: (context) => const LoginPage(),

),

);

},

child: const Text('Login'),

),

],

),

),

);

}

}

// Página de cadastro de e-mail e senha

class RegistrationPage extends StatefulWidget {

const RegistrationPage({super.key});

@override

State<RegistrationPage> createState() => \_RegistrationPageState();

}

class \_RegistrationPageState extends State<RegistrationPage> {

final \_formKey = GlobalKey<FormState>();

String? \_email;

String? \_password;

// Função para salvar as credenciais usando shared\_preferences

Future<void> \_saveCredentials(String email, String password) async {

final prefs = await SharedPreferences.getInstance();

await prefs.setString('email', email); // Salva o e-mail localmente

await prefs.setString('password', password); // Salva a senha localmente

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: const Text('Cadastro'),

),

body: Padding(

padding: const EdgeInsets.all(16.0),

child: Form(

key: \_formKey, // Chave global para manipular o formulário

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

// Campo para inserir o e-mail

TextFormField(

decoration: const InputDecoration(labelText: 'Digite seu e-mail'),

validator: (value) {

// Validação para garantir que o e-mail seja válido

if (value == null || value.isEmpty) {

return 'Por favor, insira seu e-mail';

}

if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {

return 'Insira um e-mail válido';

}

return null;

},

onSaved: (value) {

\_email = value; // Salva o e-mail inserido

},

),

const SizedBox(height: 20),

// Campo para inserir a senha

TextFormField(

decoration: const InputDecoration(labelText: 'Digite sua senha'),

obscureText: true, // Oculta o texto da senha

validator: (value) {

// Validação para garantir que a senha não esteja vazia

if (value == null || value.isEmpty) {

return 'Por favor, insira sua senha';

}

return null;

},

onSaved: (value) {

\_password = value; // Salva a senha inserida

},

),

const SizedBox(height: 20),

// Botão de ação para cadastrar o e-mail e senha

ElevatedButton(

onPressed: () async {

if (\_formKey.currentState!.validate()) {

\_formKey.currentState!.save();

// Salvar o e-mail e a senha localmente

await \_saveCredentials(\_email!, \_password!);

Navigator.of(context).pop(); // Retorna à tela inicial

}

},

child: const Text('Cadastrar E-mail'),

),

],

),

),

),

);

}

}

// Página de login, onde o usuário insere e-mail e senha

class LoginPage extends StatefulWidget {

const LoginPage({super.key});

@override

State<LoginPage> createState() => \_LoginPageState();

}

class \_LoginPageState extends State<LoginPage> {

final \_loginFormKey = GlobalKey<FormState>();

String? \_loginEmail;

String? \_loginPassword;

// Função para carregar as credenciais salvas

Future<void> \_loadCredentials() async {

final prefs = await SharedPreferences.getInstance();

setState(() {

// Carrega o e-mail e senha salvos

UserSession.email = prefs.getString('email');

UserSession.password = prefs.getString('password');

});

}

@override

void initState() {

super.initState();

\_loadCredentials(); // Carrega as credenciais ao iniciar a página de login

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: const Text('Login'),

),

body: Padding(

padding: const EdgeInsets.all(16.0),

child: Form(

key: \_loginFormKey, // Chave global para manipular o formulário

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

// Campo para inserir o e-mail no login

TextFormField(

decoration: const InputDecoration(labelText: 'Digite seu e-mail'),

validator: (value) {

// Validação para garantir que o e-mail seja válido

if (value == null || value.isEmpty) {

return 'Por favor, insira seu e-mail';

}

if (!RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(value)) {

return 'Insira um e-mail válido';

}

return null;

},

onSaved: (value) {

\_loginEmail = value; // Salva o e-mail inserido no login

},

),

const SizedBox(height: 20),

// Campo para inserir a senha no login

TextFormField(

decoration: const InputDecoration(labelText: 'Digite sua senha'),

obscureText: true, // Oculta o texto da senha

validator: (value) {

// Validação para garantir que a senha não esteja vazia

if (value == null || value.isEmpty) {

return 'Por favor, insira sua senha';

}

return null;

},

onSaved: (value) {

\_loginPassword = value; // Salva a senha inserida no login

},

),

const SizedBox(height: 20),

// Botão para fazer o login

ElevatedButton(

onPressed: () {

if (\_loginFormKey.currentState!.validate()) {

\_loginFormKey.currentState!.save();

// Verifica se o e-mail e senha correspondem às credenciais salvas

if (\_loginEmail == UserSession.email &&

\_loginPassword == UserSession.password) {

Navigator.of(context).pushReplacement(

MaterialPageRoute(

builder: (context) => UserArea(email: UserSession.email!),

),

);

} else {

// Exibe uma mensagem de erro se o login falhar

ScaffoldMessenger.of(context).showSnackBar(

const SnackBar(content: Text('E-mail ou senha incorretos.')),

);

}

}

},

child: const Text('Entrar'),

),

],

),

),

),

);

}

}

// Página de área do usuário após o login bem-sucedido

class UserArea extends StatelessWidget {

final String email;

const UserArea({super.key, required this.email});

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: const Text('Área do Usuário'),

actions: [

// Botão para fazer logout e voltar para a página inicial

IconButton(

icon: const Icon(Icons.logout),

onPressed: () {

Navigator.of(context).pushReplacement(

MaterialPageRoute(builder: (context) => const HomePage()),

);

},

),

],

),

body: Center(

child: Text('Bem-vindo, $email!', style: const TextStyle(fontSize: 24)),

),

);

}

}

// Classe para manter o estado do usuário (sessão temporária)

class UserSession {

static String? email; // E-mail do usuário

static String? password; // Senha do usuário

}