

Desarrollo de una aplicación móvil de controles básicos

Autores

Alexis Chimba, Javier Ramos, Ariel Reyes, Anthony Villarreal

Ingeniera Doris Chicaiza

Universidad de las Fuerzas Armadas - Espe

aschimba@espe.edu.ec, sjramos@espe.edu.ec, alreyes2@espe.edu.ec,
anvillarreal@espe.edu.ec

Resumen

Este proyecto es una app hecha en Flutter que se enfoca en el manejo de usuarios, como iniciar sesión, crear cuentas nuevas, recuperar contraseñas y mostrar una página de bienvenida. Lo interesante es que guarda automáticamente el usuario y la contraseña usando SharedPreferences, por lo que, si ya habías iniciado sesión antes, te lleva directo a tu perfil sin tener que volver a escribir los datos.

El archivo principal (main.dart) decide a qué pantalla entrar primero, dependiendo de si hay una sesión guardada. La lógica del inicio de sesión está en archivos como login_controller.dart, mientras que la parte visual está en los archivos de la carpeta views, donde están las pantallas que ve el usuario. También hay una carpeta de themes que sirve para que todos los botones, colores y textos se vean igual y tengan un diseño bonito y consistente. En general, el código está bien organizado, separando la lógica de la interfaz, lo cual hace que sea más fácil de entender, mantener y mejorar con el tiempo

Palabras clave. – Flutter, inicio de sesión, interfaz gráfica, controlador.

I. Introducción

En el desarrollo de aplicaciones móviles modernas, una de las funciones más comunes y necesarias es la autenticación de usuarios. Este proyecto fue creado en Flutter y tiene como objetivo ofrecer una experiencia simple pero completa para el manejo de usuarios, incluyendo el inicio de sesión, la creación de cuentas, la recuperación de contraseñas y el acceso a la información personal del usuario.

Desde el inicio, la aplicación está pensada para ser práctica: si el usuario ya se ha logueado antes, no necesita volver a ingresar sus datos, ya que se guardan de forma segura usando `SharedPreferences`. Esta funcionalidad no solo mejora la experiencia del usuario, sino que también optimiza el tiempo de acceso a la aplicación.

El código está organizado por módulos que separan claramente la lógica de negocio (controladores y modelos) de la interfaz visual (vistas), lo que permite mantener un orden y facilita futuras modificaciones. Además, se utiliza un sistema de rutas para navegar entre pantallas y una estructura de temas para un diseño visual uniforme.

Esta introducción ofrece una visión general del funcionamiento de la app y del enfoque técnico usado para su desarrollo.

II. Trabajos Relacionados

En el desarrollo de aplicaciones móviles con autenticación, existen numerosos proyectos que abordan funcionalidades similares. Por ejemplo, muchas apps creadas con Flutter implementan Firebase Auth como método principal de autenticación. Sin embargo, este proyecto se diferencia al utilizar `SharedPreferences` para gestionar las credenciales de manera local, permitiendo una experiencia más rápida al recordar la sesión del usuario sin necesidad de conectarse a un servidor.

Otros trabajos similares también incluyen pantallas de inicio de sesión, recuperación de contraseña y creación de cuenta, pero no siempre cuentan con una estructura modular como la que se utiliza aquí, donde el código se organiza en controladores, modelos, vistas y temas. Esta separación mejora la mantenibilidad y escalabilidad del proyecto, lo que lo convierte en una base sólida para futuros desarrollos más avanzados.

III. Materiales y Métodos

- Android Studio
- Extensión Flutter
- Extensión Dart
- Sistema Operativo Windows 11

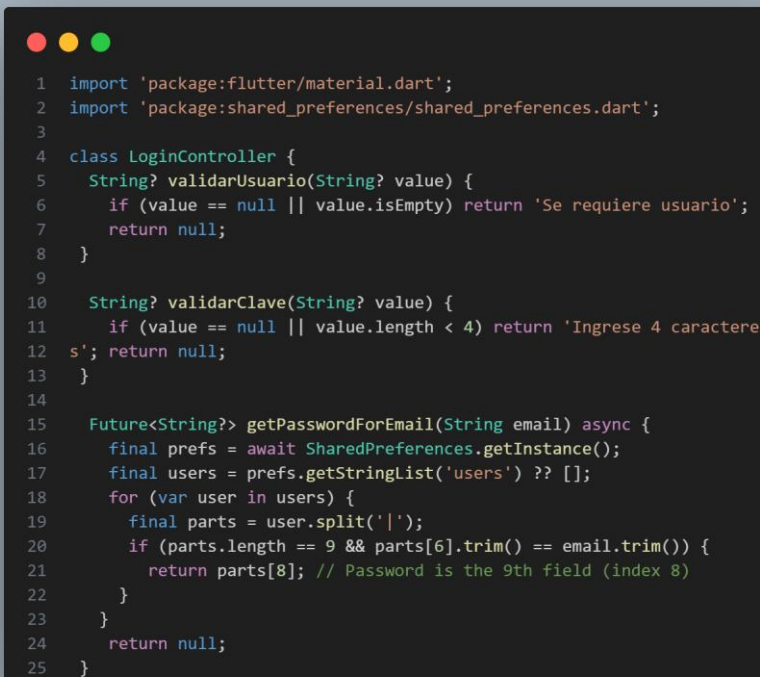
IV. Desarrollo

El código desarrolla una app en Flutter que gestiona el inicio de sesión, registro y recuperación de usuarios, recordando credenciales con SharedPreferences.

- controllers

Figura 1

logic_controller.dart



```
1 import 'package:flutter/material.dart';
2 import 'package:shared_preferences/shared_preferences.dart';
3
4 class LoginController {
5   String? validarUsuario(String? value) {
6     if (value == null || value.isEmpty) return 'Se requiere usuario';
7     return null;
8   }
9
10  String? validarClave(String? value) {
11    if (value == null || value.length < 4) return 'Ingresa 4 caracteres';
12    return null;
13  }
14
15  Future<String?> getPasswordForEmail(String email) async {
16    final prefs = await SharedPreferences.getInstance();
17    final users = prefs.getStringList('users') ?? [];
18    for (var user in users) {
19      final parts = user.split('|');
20      if (parts.length == 9 && parts[6].trim() == email.trim()) {
21        return parts[8]; // Password is the 9th field (index 8)
22      }
23    }
24    return null;
25  }
```

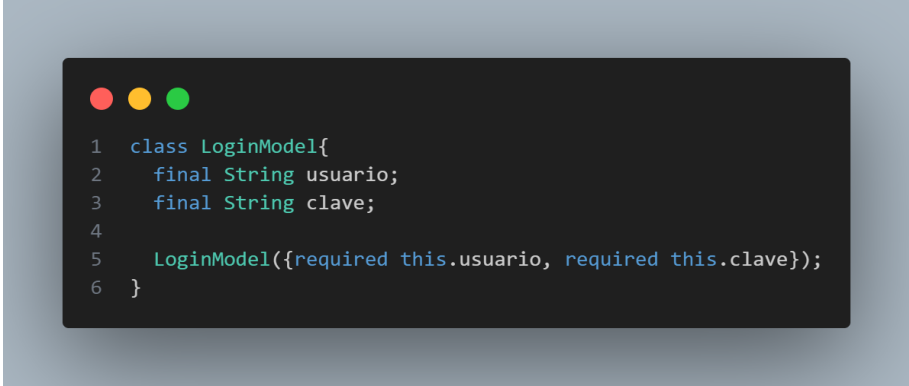
Nota. La Figura 1 maneja la lógica principal del inicio de sesión en la aplicación. Valida que el usuario y la contraseña cumplan con los requisitos mínimos, y luego verifica si coinciden con los datos almacenados en SharedPreferences. Si todo está correcto, redirige al usuario a su perfil. Además, guarda las credenciales si el usuario activa la opción de

"recordarme" y también incluye un acceso directo con una cuenta predefinida para pruebas (Anthony/1234).

- **models**

Figura 2

logic_model.dart



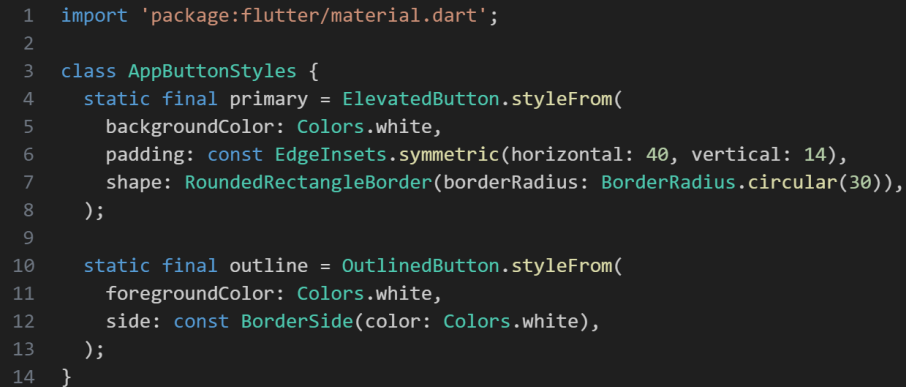
```
1 class LoginModel{
2   final String usuario;
3   final String clave;
4
5   LoginModel({required this.usuario, required this.clave});
6 }
```

Nota. La Figura 2 define una clase sencilla llamada LoginModel, que estructura los datos del inicio de sesión del usuario. Contiene dos campos obligatorios: usuario y clave, y sirve para manejar estas credenciales de forma ordenada y clara dentro de la app.

- **themes**

Figura 3

button_styles.dart

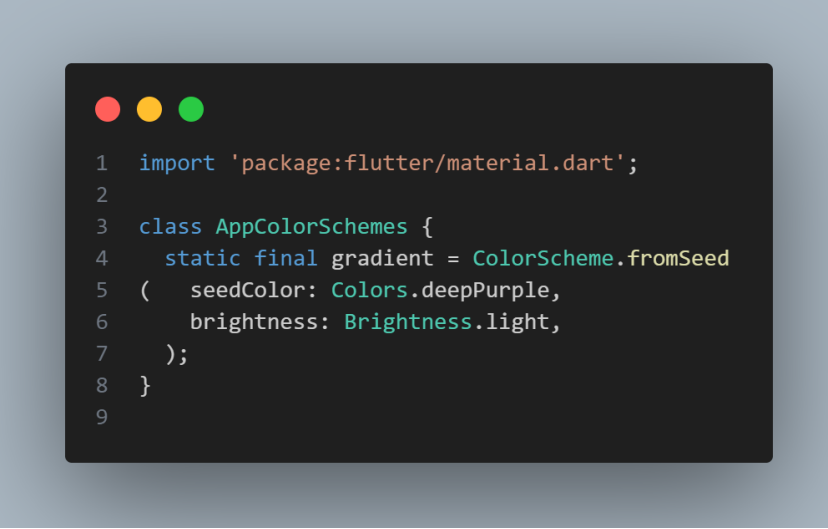


```
1 import 'package:flutter/material.dart';
2
3 class AppButtonStyles {
4   static final primary = ElevatedButton.styleFrom(
5     backgroundColor: Colors.white,
6     padding: const EdgeInsets.symmetric(horizontal: 40, vertical: 14),
7     shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(30)),
8   );
9
10  static final outline = OutlinedButton.styleFrom(
11    foregroundColor: Colors.white,
12    side: const BorderSide(color: Colors.white),
13  );
14 }
```

Nota. La Figura 3 define estilos personalizados para botones en la aplicación. Se configuran dos estilos: uno principal (primary) con fondo blanco y bordes redondeados, y otro estilo de contorno (outline) que solo muestra el borde blanco. Estos estilos aseguran una apariencia consistente y moderna en toda la interfaz.

Figura 4

color_schemes.dart



```
1 import 'package:flutter/material.dart';
2
3 class AppColorSchemes {
4   static final gradient = ColorScheme.fromSeed
5   (   seedColor: Colors.deepPurple,
6       brightness: Brightness.light,
7   );
8 }
9
```

Nota. La Figura 4 define un esquema de colores personalizado para la app. Usa un color base (deepPurple) para generar una paleta completa mediante ColorScheme.fromSeed, lo que permite mantener una apariencia visual coherente en toda la aplicación.

Figura 5

text_styles.dart



```
1 import 'package:flutter/material.dart';
2
3 class AppTextStyles {
4   static const input = TextStyle(color: Colors.black, fontSize: 18);
5   static const button = TextStyle(fontSize: 18, fontWeight: FontWeight.bold);
6   static const heading = TextStyle(fontSize: 28, fontWeight: FontWeight.bold, color: Colors.black);
7 }
8
```

Nota. La Figura 5 define estilos de texto reutilizables para la app. Incluye estilos para entradas (input), botones (button) y títulos (heading), ayudando a mantener una tipografía coherente, clara y moderna en toda la interfaz.

- views

Figura 6

create_account_page.dart

```
1 import 'package:flutter/material.dart';
2 import 'package:shared_preferences/shared_preferences.dart';
3 import '../controllers/login_controller.dart';
4 import '../themes/text_styles.dart';
5 import '../themes/button_styles.dart';
6
7 // Ecuadorian city, province, and postal code mapping (expanded)
8 const Map<String, Map<String, String>> ecuadorRegions = {
9   'Quito': {'province': 'Pichincha', 'postalCode': '1701'},
10  'Guayaquil': {'province': 'Guayas', 'postalCode': '0901'},
11  'Cuenca': {'province': 'Azuay', 'postalCode': '0101'},
12  'Ambato': {'province': 'Tungurahua', 'postalCode': '1801'},
13  'Loja': {'province': 'Loja', 'postalCode': '1101'},
14  'Machala': {'province': 'El Oro', 'postalCode': '0701'},
15  'Portoviejo': {'province': 'Manabí', 'postalCode': '1301'},
16  'Manta': {'province': 'Manabí', 'postalCode': '1308'},
17  'Esmeraldas': {'province': 'Esmeraldas', 'postalCode': '0801'},
18  'Ibarra': {'province': 'Imbabura', 'postalCode': '1001'},
19  'Riobamba': {'province': 'Chimborazo', 'postalCode': '0601'},
20  'Latacunga': {'province': 'Cotopaxi', 'postalCode': '0501'},
21  'Tulcan': {'province': 'Carchi', 'postalCode': '0401'},
22  'Santo Domingo': {'province': 'Santo Domingo de los Tsáchilas', 'postalCode': '2301'},
23  'Bahabuyo': {'province': 'Los Ríos', 'postalCode': '1201'},
24  'Quevedo': {'province': 'Los Ríos', 'postalCode': '1205'},
25  'Milagro': {'province': 'Guayas', 'postalCode': '0910'},
26  'Durán': {'province': 'Guayas', 'postalCode': '0902'},
27  'Puyo': {'province': 'Pastaza', 'postalCode': '1601'},
28  'Tena': {'province': 'Napo', 'postalCode': '1501'},
29  'Macas': {'province': 'Morona Santiago', 'postalCode': '1401'},
30  'Zamora': {'province': 'Zamora Chinchipe', 'postalCode': '1901'},
31  'Azogues': {'province': 'Cañar', 'postalCode': '0301'},
32  'Guaranda': {'province': 'Bolívar', 'postalCode': '0201'},
33  'Nueva Loja': {'province': 'Sucumbíos', 'postalCode': '2101'},
34  'Chone': {'province': 'Manabí', 'postalCode': '1303'},
35  'Santa Elena': {'province': 'Santa Elena', 'postalCode': '2401'},
36  'La Libertad': {'province': 'Santa Elena', 'postalCode': '2402'},
37  'Salinas': {'province': 'Santa Elena', 'postalCode': '2403'},
38  'Francisco de Orellana': {'province': 'Orellana', 'postalCode': '2201'},
39  };
```

Nota. La Figura 6 muestra la pantalla de creación de cuenta. Permite al usuario ingresar sus datos personales como nombre, dirección, ciudad, provincia, código postal, correo, teléfono y contraseña. Los datos se validan con reglas básicas y se almacenan localmente usando SharedPreferences. Además, los campos de ciudad, provincia y código postal se

llenen dinámicamente según una lista de ciudades del Ecuador, mejorando la experiencia del usuario y asegurando consistencia en los datos.

Figura 7

forgot_password_page.dart

```
1  import 'package:flutter/material.dart';
2  import '../controllers/login_controller.dart';
3  import '../themes/text_styles.dart';
4  import '../themes/button_styles.dart';
5
6  class ForgotPasswordPage extends StatefulWidget {
7    const ForgotPasswordPage({super.key});
8
9    @override
10   State<ForgotPasswordPage> createState() => _ForgotPasswordPageState();
11 }
12
13 class _ForgotPasswordPageState extends State<ForgotPasswordPage> {
14   final _emailCtrl = TextEditingController();
15   final _controller = LoginController();
16   String? _password;
17   bool _hasSearched = false;
18
19   @override
20   void dispose() {
21     _emailCtrl.dispose();
22     super.dispose();
23   }
24
25   @override
26   Widget build(BuildContext context) {
27     return Scaffold(
28       appBar: AppBar(
29         backgroundColor: Colors.transparent,
30         title: const Text('Forgot Password', style: TextStyle(color: Colors.white),
31         elevation: 0,
32       ),
33       body: Container(
34         decoration: const BoxDecoration(
35           gradient: LinearGradient(
36             colors: [Colors.cyan, Colors.purpleAccent],
37             begin: Alignment.topCenter,
38             end: Alignment.bottomCenter,
39           ),
40       ),
```

Nota. La Figura 7 muestra la pantalla de recuperación de contraseña. Permite al usuario ingresar su correo electrónico y, si se encuentra en los datos guardados, se muestra su contraseña en pantalla. Utiliza el controlador de inicio de sesión para buscar la clave asociada al correo y proporciona retroalimentación visual según el resultado. Todo se presenta con un diseño amigable y validaciones básicas.

Figura 8

login_page.dart

```
1 import 'package:flutter/material.dart';
2 import '../controllers/login_controller.dart';
3 import '../themes/text_styles.dart';
4 import '../themes/button_styles.dart';
5
6 class LoginPage extends StatefulWidget {
7   const LoginPage({super.key});
8
9   @override
10  State<LoginPage> createState() => _LoginPageState();
11 }
12
13 class _LoginPageState extends State<LoginPage> {
14   final _formKey = GlobalKey<FormState>();
15   final _usuarioCtrl = TextEditingController();
16   final _claveCtrl = TextEditingController();
17   final _controller = LoginController();
18   bool recordar = false;
19
20   @override
21   void dispose() {
22     _usuarioCtrl.dispose();
23     _claveCtrl.dispose();
24     super.dispose();
25   }
26
27   @override
28   Widget build(BuildContext context) {
29     return Scaffold(
30       appBar: AppBar(
31         backgroundColor: Colors.transparent,
32         title: const Text('Login', style: TextStyle(color: Colors.white)),
33       ),
34       body: Container(
35         decoration: const BoxDecoration(
36           gradient: LinearGradient(
37             colors: [Colors.cyan, Colors.purpleAccent],
38             begin: Alignment.topCenter,
39             end: Alignment.bottomCenter,
40           ),
41         ),
42       ),
43     );
44   }
45 }
```

Nota. La Figura 8 representa la pantalla de inicio de sesión de la aplicación. Permite al usuario ingresar su correo y contraseña, con validaciones básicas. También incluye opciones para recordar la sesión y recuperar la contraseña. Si los datos son correctos, se inicia sesión y se redirige al perfil del usuario. Además, ofrece un acceso directo para crear una cuenta nueva si aún no está registrado. Todo se muestra con un diseño amigable y estilizado.

Figura 9

user_info_pase.dart

```
1 import 'package:flutter/material.dart';
2 import '../themes/text_styles.dart';
3
4 class UserInfoPage extends StatelessWidget {
5   const UserInfoPage({super.key});
6
7   @override
8   Widget build(BuildContext context) {
9     final String username = Uri.decodeComponent(ModalRoute.of(context)!.settings.name!.replaceFirst('/user_info/',
10 ''))final Map<String, String>? args = ModalRoute.of(context)!.settings.arguments as Map<String, String>;
11     final String email = args?['email'] ?? 'No email available';
12     final String password = args?['password'] ?? 'No password available';
13
14     return Scaffold(
15       appBar: AppBar(
16         backgroundColor: Colors.transparent,
17         title: Text('User Info - $username', style: const TextStyle(color: Colors.white)),
18         elevation: 0,
19       ),
20       body: Container(
21         decoration: const BoxDecoration(
22           gradient: LinearGradient(
23             colors: [Colors.cyan, Colors.purpleAccent],
24             begin: Alignment.topCenter,
25             end: Alignment.bottomCenter,
26           ),
27         ),
28         padding: const EdgeInsets.symmetric(horizontal: 32),
29         child: Center(
30           child: Column(
31             mainAxisAlignment: MainAxisAlignment.center,
32             children: [
33               const Icon(Icons.person, size: 100, color: Colors.white),
34               const SizedBox(height: 20),
35               Text(
36                 'Bienvenido!',
37                 style: AppTextStyles.heading,
38               ),
39             ],
40           ),
41         ),
42       ),
43     );
44   }
45 }
```

Nota. La Figura 9 muestra la pantalla de información del usuario una vez que ha iniciado sesión. Presenta un mensaje de bienvenida junto con el correo y la contraseña del usuario (recibidos como argumentos). También incluye un botón para cerrar sesión y volver a la pantalla de login. La vista mantiene el diseño visual coherente con el resto de la aplicación.

Figura 10

welcome_page.dart



```
1 import 'package:flutter/material.dart';
2 import '../themes/text_styles.dart';
3
4 class WelcomePage extends StatelessWidget {
5   const WelcomePage({super.key});
6
7   @override
8   Widget build(BuildContext context) {
9     final String username = ModalRoute.of(context)!.settings.arguments as String;
10
11     return Scaffold(
12       appBar: AppBar(
13         title: const Text('Bienvenido'),
14         backgroundColor: Colors.deepPurple,
15       ),
16       body: Container(
17         width: double.infinity,
18         decoration: const BoxDecoration(
19           gradient: LinearGradient(
20             colors: [Colors.cyan, Colors.purpleAccent],
21             begin: Alignment.topCenter,
22             end: Alignment.bottomCenter,
23           ),
24         ),
25         child: Center(
26           child: Text(
27             '¡Bienvenido, $username!',
28             style: AppTextStyles.heading,
29           ),
30         ),
31       ),
32     );
33   }
34 }
```

Nota. La Figura 10 muestra la pantalla de bienvenida que se despliega después de iniciar sesión. Recibe el nombre del usuario como argumento y lo muestra en un mensaje personalizado. Utiliza un diseño colorido y centrado para dar una experiencia visual agradable al usuario al ingresar a la app.

Figura 11

main.dart



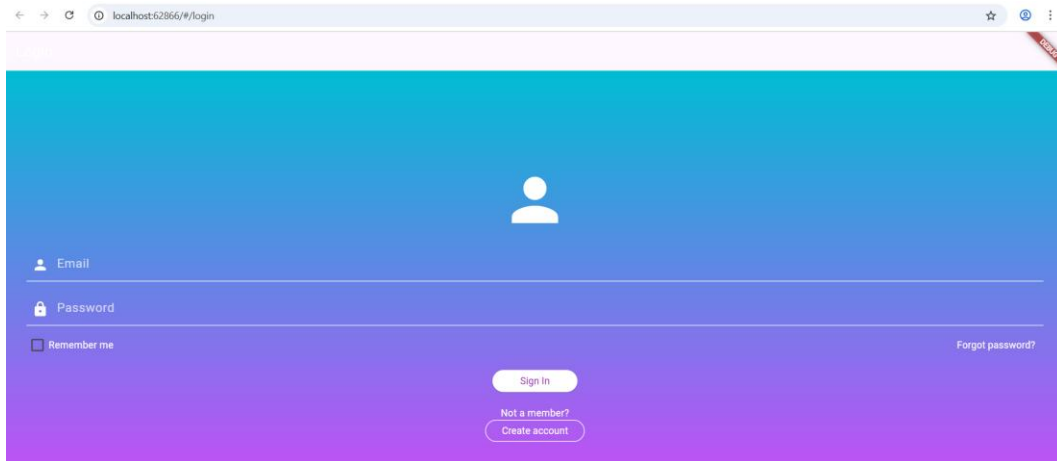
```
1 import 'package:flutter/material.dart';
2 import 'package:shared_preferences/shared_preferences.dart';
3 import 'views/login_page.dart';
4 import 'views/welcome_page.dart';
5 import 'views/create_account_page.dart';
6 import 'views/forgot_password_page.dart';
7 import 'views/user_info_page.dart';
8
9 void main() async {
10   WidgetsFlutterBinding.ensureInitialized();
11   final prefs = await SharedPreferences.getInstance();
12   final String? lastUsername = prefs.getString('last_username');
13   final String? lastPassword = prefs.getString('last_password');
14
15   String initialRoute = '/login';
16   Object? initialArguments;
17
18   // Si hay sesión guardada, ir directamente a user_info con argumento
19   if (lastUsername != null && lastPassword != null) {
20     initialRoute = '/user_info/${Uri.encodeComponent(lastUsername)}';
21     initialArguments = {
22       'email': lastUsername,
23       'password': lastPassword,
24     };
25   }
26
27   runApp(MyApp(
28     initialRoute: initialRoute,
29     initialArguments: initialArguments,
30   ));
31 }
32
```

Nota. La Figura 11 muestra el punto de entrada de la aplicación Flutter. Aquí se verifica si hay una sesión guardada en SharedPreferences para redirigir automáticamente al perfil del usuario; de lo contrario, se abre la pantalla de inicio de sesión. La clase MyApp configura las rutas principales de navegación, permitiendo moverse entre login, creación de cuenta, recuperación de contraseña, bienvenida y perfil de usuario, manteniendo una estructura organizada y dinámica.

V. Resultados

Figura 12

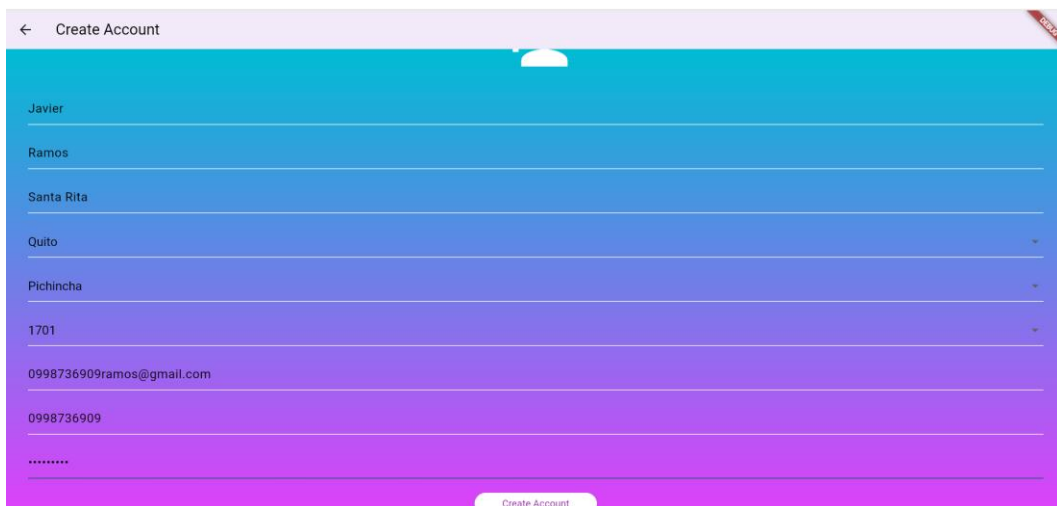
Interfaz de inicio de sesión



Nota. La Figura 12 muestra la pantalla principal de inicio de sesión de la app. Tiene un diseño moderno con un fondo degradado que va de azul a morado, y presenta de forma clara los campos para ingresar el correo y la contraseña. También permite al usuario marcar la opción de “Recordarme” para no volver a iniciar sesión cada vez. Además, incluye accesos rápidos para recuperar la contraseña o crear una cuenta nueva si aún no está registrado. Todo está pensado para que la experiencia sea simple, visualmente agradable y fácil de usar.

Figura 13

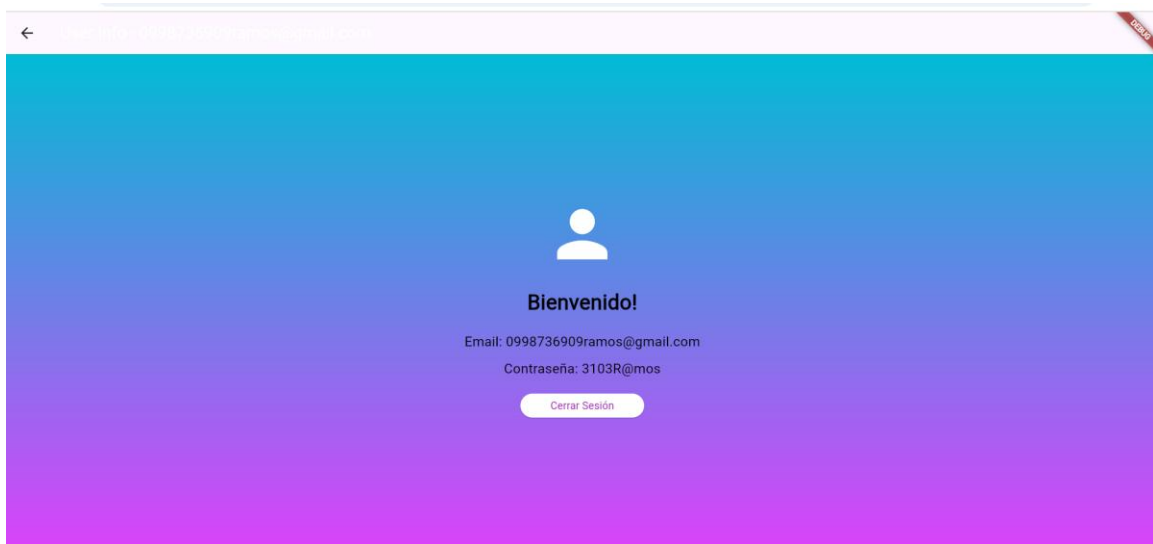
Pantalla de creación de cuenta



Nota. La Figura 13 muestra la interfaz donde un nuevo usuario puede registrarse. Se pueden ver los campos completados con nombres, dirección, ciudad, provincia, código postal, correo, teléfono y contraseña. El diseño mantiene el mismo estilo visual degradado, con una estructura clara y ordenada que guía al usuario paso a paso. Esta pantalla facilita el proceso de registro de manera intuitiva y rápida, ofreciendo una experiencia amigable desde el primer contacto con la app.

Figura 14

Pantalla de información del usuario



Nota. La Figura 14 muestra una pantalla de bienvenida personalizada después de que el usuario ha iniciado sesión exitosamente. En esta vista se presenta el correo electrónico y la contraseña del usuario, junto a un mensaje de saludo y un botón para cerrar sesión. El fondo con degradado mantiene la coherencia visual de la app, y el diseño sencillo permite al usuario identificar fácilmente sus datos y gestionar su sesión de forma rápida y clara.

VI. Conclusiones

- La aplicación está estructurada de forma ordenada, separando claramente la lógica (controladores), la interfaz (vistas) y los estilos visuales (temas), lo que facilita su mantenimiento y escalabilidad a futuro.
- Gracias al uso de SharedPreferences, la app permite guardar las credenciales del usuario, mejorando la experiencia al evitar repetir el inicio de sesión cada vez que se abre la app.

- El uso de rutas personalizadas y la lógica para redirigir automáticamente según la sesión guardada permiten una navegación más eficiente y adaptada al contexto del usuario, optimizando el flujo dentro de la aplicación.

Referencias

[1] Google, "Flutter," *flutter.dev*, 2024. [En línea]. Disponible en: <https://flutter.dev/>

[2] [1] D. K. Chicaiza, Lección 1.2: Modelos de desarrollo, Módulo Configuración, Universidad de las Fuerzas Armadas ESPE. [Presentación] .

Link del git hub

- <https://github.com/ANTHONYNESTORVILLARREALMACIAS/Desarrollo-Aplicaciones-Moviles-Chimba-Ramos-Reyes-Villarreal.git>