REPORTE DE CÓDIGO DART

Fecha: mar 16 dic 2025 06:43:11 CET
Directorio: /home/jose/StudioProjects/compartimos_gastos_app

Archivo: lib/main.dart

```dart
import 'package:compartimos_gastos/controllers/main_navigator_controller.dart';
import 'package:compartimos_gastos/screens/login_screen.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';

import 'controllers/themes_controller.dart';
import 'firebase_options.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Compartimos Gastos',
      debugShowCheckedModeBanner: false,
      theme: ThemeController.tema,
      home: StreamBuilder<User?>(
        // escucha cambios
        stream: FirebaseAuth.instance.authStateChanges(),
        builder: (context, snapshot) {
          // Si está cargando -> pantalla de carga
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Scaffold(
              body: Center(child: CircularProgressIndicator()),
            );
          }

          // Si existen credenciales guardadas, logea directomente
          if (snapshot.hasData) {
            return const MainNavigationController();
          }

          return const LoginScreen();
        },
      ),
    );
  }
```

```
}
```

Archivo: lib/screens/main/group_screen.dart

```dart
import 'package:flutter/material.dart';

import '../../widgets/appbar_custom.dart';

class GroupScreen extends StatelessWidget {
  const GroupScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: CustomAppBar(title: 'Grupos'),
      // El cuerpo de la página
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center, // Centra verticalmente
          children: [
            // Texto
            const Text(
              'Aquí se mostrarán los grupos',
              style: TextStyle(fontSize: 30),
            ),

            const SizedBox(height: 20),
            // Botón
            ElevatedButton(
              onPressed: () {
                print("Nada inegrado aún");
              },
              child: const Text('Botón de prueba'),
            ),
          ],
        ),
      ),
    );
  }
}
```

Archivo: lib/screens/main/profile_screen.dart

```dart
import 'package:compartimos_gastos/widgets/appbar_custom.dart';
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import '../../controllers/login_controller.dart';
import '../../controllers/user_controller.dart';
import '../../widgets/profile_screen/alert_dialog.dart';
import '../../widgets/profile_screen/custom_text_field.dart';
import '../../widgets/profile_screen/profile_action_button.dart';

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

  @override
  State<ProfileScreen> createState() => _ProfileScreenState();
}

class _ProfileScreenState extends State<ProfileScreen> {
  final LoginController _loginController = LoginController();
  final UserController _userController = UserController();

  final TextEditingController _nameController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

  User? user = FirebaseAuth.instance.currentUser;

  @override
  void initState() {
    super.initState();
    _nameController.text = user?.displayName ?? '';
  }

  // recargar el usuario y refrescar la pantalla
  Future<void> _refreshUser() async {
    await user?.reload();
    setState(() {
      user = FirebaseAuth.instance.currentUser;
    });
  }

  @override
  Widget build(BuildContext context) {
    // INVITADO ------------------------------------------------
    if (user!.isAnonymous) {
      return Scaffold(
        appBar: const CustomAppBar(title: 'Perfil de Invitado'),
        body: Center(
          child: Padding(
            padding: const EdgeInsets.all(24.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
```

```
              const Icon(Icons.no_accounts, size: 80, color: Colors.grey),
              const SizedBox(height: 20),

              const Text(
                "Has entrado como invitado",
                style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
              ),
              const SizedBox(height: 20),

              const Text(
                "Si cierras sesión o pierdes tu dispositivo, perderás tus datos. Regístrate para guard
                textAlign: TextAlign.center,
                style: TextStyle(fontSize:16, color: Colors.grey),
              ),
              const SizedBox(height: 20),

              ProfileActionButton(
                icon: Icons.save_as,
                text: 'Registrarse',
                onPressed: () => _convertirInvitadoARegistrado(context),
              ),
            ],
          ),
        ),
      ),
    );
  }

  // USUARIO REGISTRADO-------------------------------------------------
  return Scaffold(
    appBar: const CustomAppBar(title: 'Mi Perfil'),
    body: Center(
      child: SingleChildScrollView(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            // CARD DE INFORMACIÓN
            _buildUserInfoCard(),
            const SizedBox(height: 20),

            // BOTÓN VERIFICAR EMAIL (Solo si falta verificar)
            if (!user!.emailVerified) ...[
              ProfileActionButton(
                icon: Icons.mark_email_unread,
                text: 'Verificar tu correo electrónico',
                color: Colors.orange,
                onPressed: () async {
                  await _loginController.sendEmailVerification();
                  if (context.mounted) {
                    ScaffoldMessenger.of(context).showSnackBar(
                      const SnackBar(content: Text("Correo enviado")),
                    );
                  }
```

```
                },
              ),
              const SizedBox(height: 10),
            ],

            // BOTÓN ACTUALIZAR PERFIL
            ProfileActionButton(
              icon: Icons.person_4,
              text: 'Actualizar Perfil',
              onPressed: () => _editarPerfil(context),
            ),
            const SizedBox(height: 10),

            // BOTÓN CAMBIAR CONTRASEÑA
            ProfileActionButton(
              icon: Icons.lock_reset,
              text: 'Cambiar Contraseña',
              onPressed: () => _cambiarContrasena(context),
            ),
            const SizedBox(height: 10),

            // BOTÓN CAMBIAR EMAIL
            ProfileActionButton(
              icon: Icons.alternate_email,
              text: 'Cambiar Email',
              onPressed: () => _cambiarEmail(context),
            ),
          ],
        ),
      ),
    ),
  );
}

// WIDGET DE LA PÁGINA NO REUTILIZABLE----------------------

Widget _buildUserInfoCard() {
  return Card(
    elevation: 4,// sombreado
    shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(15)),
    child: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        children: [
          const CircleAvatar(
            radius: 40,
            backgroundColor: Colors.grey,
            child: Icon(Icons.person, size: 40, color: Colors.white),
          ),
          const SizedBox(height: 10),

          Text(
            user?.displayName ?? 'Actualiza perfil e introduce nombre.',
```

```
            style: const TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
            textAlign: TextAlign.center,
          ),

          Text(
            user?.email ?? 'Sin Email',
            style: TextStyle(fontSize: 18, color: Colors.grey[600]),
          ),
          const SizedBox(height: 5),

          Container(
            padding: const EdgeInsets.symmetric(horizontal: 8, vertical: 4),
            decoration: BoxDecoration(
              color: user!.emailVerified
                  ? Colors.green[100]
                  : Colors.orange[100],
              borderRadius: BorderRadius.circular(10),
            ),
            child: Text(
              user!.emailVerified ? "Verificado" : "No Verificado",
              style: TextStyle(
                color: user!.emailVerified ? Colors.green : Colors.orange,
                fontWeight: FontWeight.bold,
                fontSize: 12,
              ),
            ),
          ),
        ],
      ),
    ),
  );
}

// LÓGICA DE DIÁLOGS ----------------------------

void _convertirInvitadoARegistrado(BuildContext context) {
  _emailController.clear();
  _passwordController.clear();

  showDialog(
    context: context,
    builder: (context) => CustomInputDialog(
      title: 'Registrar Cuenta',
      confirmText: 'Vincular Cuenta',
      children: [
        const Text("Introduce un email y contraseña para registrarte."),
        const SizedBox(height: 20),
        CustomTextField(
          controller: _emailController,
          label: 'Email',
          icon: Icons.email,
          keyboardType: TextInputType.emailAddress,
        ),
```

```
          const SizedBox(height: 15),
          CustomTextField(
            controller: _passwordController,
            label: 'Contraseña',
            icon: Icons.lock,
            obscureText: true, //*******
          ),
        ],
        onConfirm: () async {
          if (_emailController.text.isEmpty ||
              _passwordController.text.length < 6) {
            if (context.mounted) {
              ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(
                  content: Text(
                    'Email inválido o contraseña minimo 6 caracteres',
                  ),
                ),
              );
            }
            return;
          }

          final usuario = await _loginController.linkAnonymousAccount(
            _emailController.text.trim(),
            _passwordController.text.trim(),
          );

          if (usuario != null) {
            await _refreshUser();
            if (context.mounted) {
              Navigator.pop(context);
              ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text('Cuenta guardada con éxito!')),
              );
            }
          } else {
            if (context.mounted) {
              ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(
                  content: Text('Error al vincular. Email en uso.'),
                ),
              );
            }
          }
        },
      ),
    );
}

void _cambiarContrasena(BuildContext context) {
  _passwordController.clear();
  showDialog(
```

```
    context: context,
    builder: (context) => CustomInputDialog(
      title: 'Cambiar Contraseña',
      confirmText: 'Actualizar',
      children: [
        const Text("Asegúrate de haber iniciado sesión recientemente."),
        const SizedBox(height: 10),
        CustomTextField(
          controller: _passwordController,
          label: 'Nueva contraseña',
          icon: Icons.lock,
          obscureText: true, // *****
        ),
      ],
      onConfirm: () async {
        await _loginController.updatePassword(
          _passwordController.text.trim(),
        );
        if (context.mounted) {
          Navigator.pop(context);
          ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text('Contraseña actualizada con éxito')),
          );
        }
      },
    ),
  );
}

void _cambiarEmail(BuildContext context) {
  _emailController.clear();
  showDialog(
    context: context,
    builder: (context) => CustomInputDialog(
      title: 'Cambiar Email',
      confirmText: 'Enviar verificación',
      children: [
        const Text("Se enviará un enlace al nuevo correo para confirmar."),
        const SizedBox(height: 10),
        CustomTextField(
          controller: _emailController,
          label: 'Nuevo correo electrónico',
          icon: Icons.email,
          keyboardType: TextInputType.emailAddress,
        ),
      ],
      onConfirm: () async {
        await _loginController.changeEmail(_emailController.text.trim());
        if (context.mounted) {
          Navigator.pop(context);
          ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(
              content: Text('Verificación enviada. Revisa tu nuevo correo.'),
```

```
            ),
          );
        }
      },
    ),
  );
}

void _editarPerfil(BuildContext context) {
  _nameController.text = '';
  showDialog(
    context: context,
    builder: (context) => CustomInputDialog(
      title: 'Actualizar Perfil',
      confirmText: 'Guardar',
      children: [
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const CircleAvatar(
              radius: 30,
              backgroundColor: Colors.grey,
              child: Icon(Icons.camera_alt, color: Colors.white),
            ),
          ],
        ),
        const SizedBox(height: 20),
        CustomTextField(
          controller: _nameController,
          label: 'Nombre',
          icon: Icons.person,
        ),
        const SizedBox(height: 20),
      ],
      onConfirm: () async {
        String nuevoNombre = _nameController.text.trim();
        if (nuevoNombre.isNotEmpty) {
          Navigator.pop(context);
          ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text('Perfil actualizado correctamente')),
          );
          try {
            await user?.updateDisplayName(nuevoNombre);

            await _userController.actualizarUsuario(user!.uid, {
              'nombre': nuevoNombre,
            });

            await _refreshUser();
          } catch (e) {
            // SI FALLA ALGO, AVISAMOS
            if (context.mounted) {
              ScaffoldMessenger.of(context).showSnackBar(
```

```
                  SnackBar(
                    content: Text('Hubo un problema de conexión: $e'),
                    backgroundColor: Colors.red,
                  ),
                );
              }
            }
          }
        },
      ),
    );
  }
}
```

Archivo: lib/screens/login_screen.dart

```dart
import 'package:compartimos_gastos/widgets/logo_widget.dart';
import 'package:compartimos_gastos/widgets/login_screen/auth_dialogs.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import '../controllers/login_controller.dart';
import '../controllers/main_navigator_controller.dart';

class LoginScreen extends StatefulWidget {
  const LoginScreen({super.key});
  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final _formKey = GlobalKey<FormState>();

  //final _emailCtrl = TextEditingController();
  final _emailCtrl = TextEditingController(
    text: 'joseroyo3@hotmail.com',
  ); // de momento se deja para acceder mas rápido
  final _passCtrl = TextEditingController();
  final _controller = LoginController();

  bool _isLoading = false;
  bool _rememberMe = false;

  @override
  void initState() {
    super.initState();
    _initPrefs();
  }

  @override //
  void dispose() {
    _emailCtrl.dispose();
    _passCtrl.dispose();
    super.dispose();
  }

  Future<void> _initPrefs() async {
    final prefs = await SharedPreferences.getInstance();
    setState(() {
      _rememberMe = prefs.getBool('rememberMe') ?? false;
      if (_rememberMe) _emailCtrl.text = prefs.getString('savedEmail') ?? '';
    });
  }

  Future<void> _toggleRememberMe(bool value) async {
    setState(() => _rememberMe = value);
    final prefs = await SharedPreferences.getInstance();
    value
        ? await prefs.setString('savedEmail', _emailCtrl.text.trim())
```

```
      : await prefs.remove('savedEmail');
  await prefs.setBool('rememberMe', value);
}

Future<void> _executeAuth(
  Future<dynamic> Function() action, {
  bool validate = true,
}) async {
  if (validate && !_formKey.currentState!.validate()) return;

  setState(() => _isLoading = true);

  if (validate && _rememberMe) {
    await _toggleRememberMe(true);
  }

  final result = await action();

  if (mounted) {
    setState(() => _isLoading = false);
    // Feedback visual simple (error generico, no comprueba regex)
    if (result == null) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Error en la autenticación')),
      );
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: SingleChildScrollView(
        padding: const EdgeInsets.all(24),
        child: Form(
          key: _formKey,
          child: Column(
            children: [
              const LogoWidget(),

              TextFormField(
                controller: _emailCtrl,
                decoration: const InputDecoration(
                  labelText: "Email",
                  prefixIcon: Icon(Icons.email),
                ),

                validator: (v) => v!.isEmpty ? 'Requerido' : null,
              ),

              const SizedBox(height: 16),
```

```
            TextFormField(
              controller: _passCtrl,
              obscureText: true,
              decoration: const InputDecoration(
                labelText: "Contraseña",
                prefixIcon: Icon(Icons.lock),
              ),
              validator: (v) => v!.isEmpty ? 'Requerido' : null,
            ),

            Row(
              children: [
                Checkbox(
                  value: _rememberMe,
                  onChanged: (v) => _toggleRememberMe(v ?? false),
                ),
                const Text('Recordar usuario'),
              ],
            ),

            const SizedBox(height: 24),

            if (_isLoading)
              const CircularProgressIndicator()
            else ...[
              //lista de widgets
              SizedBox(
                width: double.infinity,
                child: ElevatedButton(
                  onPressed: () => _executeAuth(
                    () => _controller.login(
                      _emailCtrl.text.trim(),
                      _passCtrl.text.trim(),
                    ),
                  ),
                  child: const Text("Entrar"),
                ),
              ),

              const SizedBox(height: 10),

              // DIALOGS REGISTRO
              TextButton(
                onPressed: () =>
                    AuthDialogs.showRegister(context, _controller),
                child: const Text("¿No tienes cuenta? Regístrate"),
              ),

              // DIALOGS RECUPERAR CONTRASEÑA
              TextButton(
                onPressed: () => AuthDialogs.showForgotPass(
                  context,
                  _emailCtrl,
```

```
                    _controller,
                  ),
                  child: const Text('¿Olvidaste tu contraseña?'),
                ),

                const SizedBox(height: 30),

                // ANONIMO
                SizedBox(
                  child: TextButton(
                    onPressed: () => _executeAuth(
                      () => _controller.signInAnonymously(),
                      validate: false,
                    ),
                    child: const Text("Entrar como Invitado"),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

Archivo: lib/widgets/logo_widget.dart

```dart
import 'package:flutter/material.dart';

class LogoWidget extends StatelessWidget {
  const LogoWidget({
    super.key,
  });

  @override
  Widget build(BuildContext context) {
    return Container(
      margin: EdgeInsets.only(bottom: 40),
      height: 200,
      child: Image.asset(
        'lib/assets/images/logo/logo.png',
        fit: BoxFit.contain,
        errorBuilder: (context, error, stackTrace) {
          // Si falla la imagen, muestra un icono
          return Icon(
            Icons.account_balance_wallet,
            size: 80,
            color: Theme.of(context).primaryColor,
          );
        },
      ),
    );
  }
}
```

Archivo: lib/widgets/appbar_custom.dart

```dart
import 'package:flutter/material.dart';
import '../../controllers/login_controller.dart';

class CustomAppBar extends StatelessWidget implements PreferredSizeWidget {
  final String title;
  final List<Widget>? actions;
  final bool showLogout;

  const CustomAppBar({
    super.key,
    required this.title,
    this.actions,
    this.showLogout = true,
  });

  @override
  Size get preferredSize => const Size.fromHeight(kToolbarHeight);

  @override
  Widget build(BuildContext context) {
    return AppBar(
      title: Text(title),
      centerTitle: true,
      actions: [
        ...?actions, // ACCIONES independientes pasables
        // Para no mostrar el botón de logout hay que pasar false
        if (showLogout)
          IconButton(
            icon: const Icon(Icons.logout),
            tooltip: 'Cerrar sesión',
            onPressed: () => _confirmLogout(context),
          ),
      ],
    );
  }

  void _confirmLogout(BuildContext context) async {
    final confirm = await showDialog<bool>(
      context: context,
      builder: (context) => AlertDialog(
        title: const Text('Cerrar sesión'),
        content: const Text('¿Seguro que quieres salir?'),
        actions: [
          TextButton(
            onPressed: () => Navigator.pop(context, false),
            child: const Text('Cancelar'),
          ),
          ElevatedButton(
            onPressed: () => Navigator.pop(context, true),
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.red,
              foregroundColor: Colors.white,
```

```
            ),
            child: const Text('Salir'),
          ),
        ],
      ),
    );

    if (confirm == true) {
      await LoginController().logout(); // sin await no cierra
    }
  }
}
```

Archivo: lib/widgets/login_screen/auth_dialogs.dart

```dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import '../../controllers/login_controller.dart';


// Clase que usaremos solo para los dialogs de login screen
abstract class AuthDialogs {
  static void showForgotPass(
    BuildContext context,
    TextEditingController
    emailCtrl, // Reutilizamos el controller para que el usuario no reescriba su email
    LoginController controller,
  ) {
    showDialog(
      context: context,
      builder: (ctx) => AlertDialog(
        title: const Text('Recuperar contraseña'),
        content: TextField(
          controller: emailCtrl,
          keyboardType: TextInputType.emailAddress, // teclado estandar pero comprobando @ y .
          decoration: const InputDecoration(labelText: 'Tu email'),
        ),
        actions: [
          TextButton(
            onPressed: () => Navigator.pop(ctx),
            child: const Text('Cancelar'),
          ),
          ElevatedButton(
            onPressed: () async {
              if (emailCtrl.text.isEmpty) return;

              // Navigator.pop antes de operaciones asíncronas largas para cerrar rapido
              Navigator.pop(ctx);

              await controller.sendPasswordResetEmail(emailCtrl.text.trim());

              // Verificamos 'mounted' del contexto original (context) no del diálogo (ctx)
              if (context.mounted) {
                ScaffoldMessenger.of(
                  context,
                ).showSnackBar(const SnackBar(content: Text('Email enviado')));
              }
            },
            child: const Text('Enviar'),
          ),
        ],
      ),
    );
  }

  static void showRegister(BuildContext context, LoginController controller) {
    // Definimos controllers locales
```

```
final rEmail = TextEditingController();
final rPass = TextEditingController();

showDialog(
  context: context,
  builder: (ctx) => AlertDialog(
    title: const Text('Registro'),
    content: Column(
      mainAxisSize: MainAxisSize.min,
      children: [
        TextField(
          controller: rEmail,
          keyboardType: TextInputType.emailAddress,
          decoration: const InputDecoration(labelText: 'Email'),
        ),
        TextField(
          controller: rPass,
          obscureText: true, // salen *******
          decoration: const InputDecoration(labelText: 'Contraseña'),
        ),
      ],
    ),
    actions: [
      TextButton(
        onPressed: () => Navigator.pop(ctx),
        child: const Text('Cancelar'),
      ),
      ElevatedButton(
        onPressed: () async {
          if (rEmail.text.isEmpty || rPass.text.isEmpty) return;

          Navigator.pop(ctx);

          // Creamos usuario en auth
          User? usuario = await controller.register(
            rEmail.text.trim(),
            rPass.text.trim(),
          );

          // Comprobamamos que no sea null (que hay usuario y conexion)
          if (usuario != null) {
            if (context.mounted) {
              ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text('Registrado con éxito. Verifica tu email.')),
              );
            }
          } else {
            // Si usuario es null, mostramos error generico
            if (context.mounted) {
              ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text('Email en uso o sin red.')),
              );
            }
```

```
              }
            },
            child: const Text('Registrarse'),
          ),
        ],
      ),
    );
  }
}
```

Archivo: lib/widgets/profile_screen/alert_dialog.dart

```dart
import 'package:flutter/material.dart';

class CustomInputDialog extends StatelessWidget {
  final String title;
  final List<Widget> children; // Aquí vna los TextFields
  final String confirmText;
  final VoidCallback onConfirm;
  final bool isLoading;

  const CustomInputDialog({
    super.key,
    required this.title,
    required this.children,
    required this.onConfirm, // LÓGICA
    this.confirmText = 'Guardar',
    this.isLoading = false,
  });

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: Text(title),
      content: SizedBox(
        width: 500,
        child: SingleChildScrollView(
          child: Column(mainAxisSize: MainAxisSize.min, children: children),
        ),
      ),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context),
          child: const Text('Cancelar'),
        ),
        ElevatedButton(
          onPressed: isLoading ? null : onConfirm,
          child: isLoading
              ? const SizedBox(
            width: 20,
            height: 20,
            child: CircularProgressIndicator(strokeWidth: 2),
          )
              : Text(confirmText),
        ),
      ],
    );
  }
}
```

Archivo: lib/widgets/profile_screen/custom_text_field.dart

```dart
import 'package:flutter/material.dart';

class CustomTextField extends StatelessWidget {
  final TextEditingController controller;
  final String label;
  final IconData icon;
  final bool obscureText;
  final TextInputType keyboardType;

  const CustomTextField({
    super.key,
    required this.controller,
    required this.label,
    required this.icon,
    this.obscureText = false, // si es true, salen **** (para contraseñas)
    this.keyboardType = TextInputType.text, // teclado estandar
  });

  @override
  Widget build(BuildContext context) {
    return TextField(
      controller: controller,
      obscureText: obscureText,
      keyboardType: keyboardType,
      decoration: InputDecoration(
        labelText: label,
        prefixIcon: Icon(icon),
        border: const OutlineInputBorder(),
        filled: true,
        fillColor: Colors.grey[200],
        contentPadding: const EdgeInsets.symmetric(vertical: 15, horizontal: 20),
      ),
    );
  }
}
```

\newpage

Archivo: lib/widgets/profile_screen/profile_action_button.dart

```dart
import 'package:flutter/material.dart';

class ProfileActionButton extends StatelessWidget {
  final IconData icon;
  final String text;
  final VoidCallback onPressed;
  final Color? color;

  const ProfileActionButton({
    super.key,
    required this.icon,
    required this.text,
    required this.onPressed,
```

```
    this.color,
  });

  @override
  Widget build(BuildContext context) {
    return ElevatedButton.icon(
      onPressed: onPressed,
      icon: Icon(icon, color: Colors.white),
      label: Text(text, style: TextStyle(fontSize: 16)),
      style: ElevatedButton.styleFrom(
        backgroundColor: color ?? Theme.of(context).primaryColor,
        padding: const EdgeInsets.symmetric(vertical: 12, horizontal: 12),
      ),
    );
  }
}
```

Archivo: lib/controllers/login_controller.dart

```dart
import 'package:compartimos_gastos/controllers/user_controller.dart';
import 'package:firebase_auth/firebase_auth.dart';

import '../models/user_model.dart';

class LoginController {
  final FirebaseAuth _auth = FirebaseAuth.instance;

  // REGISTRO
  Future<User?> register(String email, String password) async {
    try {
      final userCredential = await _auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );

      User? user = userCredential.user;
      if (user != null) {
        UserModel nuevoUsuarioRegistrandose = UserModel.fromFirebase(user);

        // Llamamos al UserController para crearlo en la base de datos
        await UserController().crearUsuario(nuevoUsuarioRegistrandose);
        await user.sendEmailVerification();
      }

      return user;
    } on FirebaseAuthException catch (e) {
      print("Error de registro: ${e.code} - ${e.message}");
      return null;
    }
  }

  // LOGIN
  Future<User?> login(String email, String password) async {
    try {
      final userCredential = await _auth.signInWithEmailAndPassword(
        email: email,
        password: password,
      );
      return userCredential.user;
    } on FirebaseAuthException catch (e) {
      print("Error de login: ${e.code} - ${e.message}");
      return null;
    }
  }

  // LOGOUT
  Future<void> logout() async {
    await _auth.signOut();
  }

  // OBTENER USUARIO ACTUAL
```

```dart
  User? get currentUser => _auth.currentUser;

  // RECUPERAR CONTRASEÑA
  Future<void> sendPasswordResetEmail(String email) async {
    await _auth.sendPasswordResetEmail(email: email);
  }

  // ACTUALIZAR CONTRASEÑA
  Future<void> updatePassword(String newPassword) async {
    await _auth.currentUser?.updatePassword(newPassword);
  }


  // VERIFICACION POR EMAIL EN CASO DE OLVIDO
  Future<void> sendEmailVerification() async {
    await _auth.currentUser?.sendEmailVerification();
  }

  // CAMBIAR EMAIL
  Future<void> changeEmail(String newEmail) async {
    await _auth.currentUser?.verifyBeforeUpdateEmail(newEmail);
  }

  /* ---------- USUARIOS ANONIMOS ---------- */

// INICIAR SESIÓN ANÓNIMA
  Future<User?> signInAnonymously() async {
    try {
      final userCredential = await _auth.signInAnonymously();
      final user = userCredential.user;

      if (user != null) {
        UserModel usuarioAnonimo = UserModel(
          id: user.uid,
          email: '',
          nombre: 'Anónimo',
          fotoPerfil: '',
          grupos: [],
        );
        // Debemos registrar al anonimo para que pueda crear/entrar en grupos
        await UserController().crearUsuario(usuarioAnonimo);
      }

      return user;
    } on FirebaseAuthException catch (e) {
      print("Error anónimo: $e");
      return null;
    }
  }

  // USUARIO ANONIMO A REGISTRARSE
  Future<User?> linkAnonymousAccount(String email, String password) async {
    try {
```

```
      final credential = EmailAuthProvider.credential(
        email: email,
        password: password,
      );
      final userCredential = await _auth.currentUser?.linkWithCredential(credential);
      final user = userCredential?.user;

      if (user != null) {
        await UserController().actualizarUsuario(user.uid, {
          'email': email,
        });
        await user.sendEmailVerification();
      }

      return user;
    } catch (e) {
      print("Error vinculando cuenta: $e");
      return null;
    }
  }
}
```

Archivo: lib/controllers/main_navigator_controller.dart

```dart
import 'package:compartimos_gastos/screens/main/group_screen.dart';
import 'package:compartimos_gastos/screens/main/profile_screen.dart';
import 'package:flutter/material.dart';

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

  @override
  State<MainNavigationController> createState() =>
      _MainNavigationControllerState();
}

class _MainNavigationControllerState extends State<MainNavigationController> {
  int _selectedIndex = 0; //iniciamos en 0, que es group

  // Usamos 'late' para inizializar las paginas en el initState
  // en el futuro necesitaremos pasar 'context' o argumentos a las pantallas para el cambio de color
  late final List<Widget> _pages;

  @override
  void initState() {
    super.initState();
    // El orden de esta lista debe coincidir estrictamente con el de los BottomNavigationBarItem
    _pages = [const GroupScreen(), const ProfileScreen()];
  }

  void _onItemTapped(int index) {
    // setState es necesario para reconstruir el Scaffold y mostrar la nueva página seleccionada
    setState(() {
      _selectedIndex = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // Renderizamos el widget correspondiente al índice actual
      body: _pages[_selectedIndex],
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: _selectedIndex,
        onTap: _onItemTapped,
        // Usamos fixed para que los iconos no se muevan al seleccionarlos (estilo material estándar)
        type: BottomNavigationBarType.fixed,
        items: const [
          BottomNavigationBarItem(
            icon: Icon(Icons.group),
            label: 'Grupos',
          ), // 0
          BottomNavigationBarItem(
            icon: Icon(Icons.person),
            label: 'Perfil',
          ), // 1
```

```
        ],
      ),
    );
  }
}
```

Archivo: lib/controllers/user_controller.dart

```dart
import '../models/user_model.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

class UserController {
  final FirebaseFirestore firestore = FirebaseFirestore.instance;

  // Función para GUARDAR
  Future<void> crearUsuario(UserModel usuario) async {
    await firestore.collection('usuarios').doc(usuario.id).set(usuario.toMap());
  }

  // Función para LEER
  Future<UserModel?> obtenerUsuario(String id) async {
    var doc = await firestore.collection('usuarios').doc(id).get();
    if (!doc.exists) return null;
    return UserModel.fromFirestore(doc);
  }

  // Función para ACTUALIZAR
  Future<void> actualizarUsuario(
    String uid,
    Map<String, dynamic> datosAActualizar,
  ) async {
    // .update solo modifica los campos que le pasas en el mapa, sin modificar el resto
    await firestore.collection('usuarios').doc(uid).update(datosAActualizar);
  }
}
```

Archivo: lib/controllers/themes_controller.dart

```dart
import 'package:flutter/material.dart';

// Encargado de controlar el tema principal de la app
class ThemeController {
  // Especifico el verde del logo como color principal
  static const Color primaryColor = Color(0xFF30CBA1);

  // Creamos el Tema Global
  static final ThemeData tema = ThemeData(
    useMaterial3: true,

    // definimos la paleta principal en verde, a futuro pasaremos otros colores (segun grupo)
    colorScheme: ColorScheme.fromSeed(
      seedColor: primaryColor,
      primary: primaryColor,
    ),

    // --- BARRA SUPERIOR (APPBAR) ---
    appBarTheme: const AppBarTheme(
      backgroundColor: primaryColor, // Fondo Verde
      foregroundColor: Colors.white, // Texto Blanco
      centerTitle: true,
      elevation: 0,
    ),

    // --- BOTONES (ELEVATED BUTTON) ---
    elevatedButtonTheme: ElevatedButtonThemeData(
      style: ElevatedButton.styleFrom(
        backgroundColor: primaryColor, // Fondo Verde
        foregroundColor: Colors.white, // Texto Blanco
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(10)),
      ),
    ),

    // --- BOTÓN FLOTANTE ---
    // Lo he puesto también en Verde (primaryColor) para que no dé error
    floatingActionButtonTheme: const FloatingActionButtonThemeData(
      backgroundColor: primaryColor,
      foregroundColor: Colors.white,
    ),
  );
}
```

Archivo: lib/firebase_options.dart

```dart
// File generated by FlutterFire CLI.
// ignore_for_file: type=lint
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      return web;
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        return ios;
      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.',
        );
    }
  }

  // Automaticamente configura Firebase para web, android y iOS
  static const FirebaseOptions web = FirebaseOptions(
    apiKey: 'AIzaSyCFSSEKT-ZYg5kYPlwBwiWh8q3UZsEIcec',
    appId: '1:1029109934862:web:753b80ae38618719c5c55f',
```

```
    messagingSenderId: '1029109934862',
    projectId: 'compartimosgastosapp',
    authDomain: 'compartimosgastosapp.firebaseapp.com',
    storageBucket: 'compartimosgastosapp.firebasestorage.app',
    measurementId: 'G-GBDJ0RHTLF',
  );

  static const FirebaseOptions android = FirebaseOptions(
    apiKey: 'AIzaSyCkjvICetJbIaSLirQqB7eIPCo6A0Vxw-k',
    appId: '1:1029109934862:android:1224c6f381cefbc3c5c55f',
    messagingSenderId: '1029109934862',
    projectId: 'compartimosgastosapp',
    storageBucket: 'compartimosgastosapp.firebasestorage.app',
  );

  static const FirebaseOptions ios = FirebaseOptions(
    apiKey: 'AIzaSyCMIKoX-LS9EWZvbuGPrk_g56B1ZL27Jvc',
    appId: '1:1029109934862:ios:ea10a7c0253d8634c5c55f',
    messagingSenderId: '1029109934862',
    projectId: 'compartimosgastosapp',
    storageBucket: 'compartimosgastosapp.firebasestorage.app',
    iosBundleId: 'josebyteando.compartimosgastos.compartimosGastos',
  );
}
```

Archivo: lib/models/user_model.dart

```dart
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';

class UserModel {
  final String id;
  final String nombre;
  final String email;
  final String fotoPerfil;
  final List<String> grupos;

  UserModel({
    required this.id,
    required this.nombre,
    required this.email,
    required this.fotoPerfil,
    required this.grupos,
  });

  //DESDE AUTH (Cuando el usuario se registra/loguea)
  factory UserModel.fromFirebase(User user) {
    return UserModel(
      id: user.uid,
      nombre: user.displayName ?? '',
      email: user.email ?? 'no-email',
      fotoPerfil: user.photoURL ?? '',
      grupos: [], // Auth no tiene grupos, iniciamos vacío
    );
  }

  // DESDE FIRESTORE DATABASE (Cuando LEES la base de datos)
  factory UserModel.fromFirestore(DocumentSnapshot doc) {
    // Convertimos la data del documento a un Mapa seguro
    final data = doc.data() as Map<String, dynamic>? ?? {};

    return UserModel(
      id: doc.id, // El ID viene del documento, no de la data interna
      nombre: data['nombre'] ?? '',
      email: data['email'] ?? '',
      fotoPerfil: data['fotoPerfil'] ?? '',
      grupos: List<String>.from(data['grupos'] ?? []),
    );
  }

  // HACIA FIRESTORE DATABASE (Para guardar/actualizar)
  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'nombre': nombre,
      'email': email,
      'fotoPerfil': fotoPerfil,
      'grupos': grupos,
    };
```

```
  }

  // COPYWITH (Para actualizar la UI fácilmente)
  UserModel copyWith({
    String? id,
    String? nombre,
    String? email,
    String? fotoPerfil,
    List<String>? grupos,
  }) {
    return UserModel(
      id: id ?? this.id,
      nombre: nombre ?? this.nombre,
      email: email ?? this.email,
      fotoPerfil: fotoPerfil ?? this.fotoPerfil,
      grupos: grupos ?? this.grupos,
    );
  }
}
```

Archivo: .dart_tool/dartpad/web_plugin_registrant.dart

```dart
// Flutter web plugin registrant file.
//
// Generated file. Do not edit.
//

// @dart = 2.13
// ignore_for_file: type=lint

import 'package:cloud_firestore_web/cloud_firestore_web.dart';
import 'package:firebase_auth_web/firebase_auth_web.dart';
import 'package:firebase_core_web/firebase_core_web.dart';
import 'package:shared_preferences_web/shared_preferences_web.dart';
import 'package:flutter_web_plugins/flutter_web_plugins.dart';

void registerPlugins([final Registrar? pluginRegistrar]) {
  final Registrar registrar = pluginRegistrar ?? webPluginRegistrar;
  FirebaseFirestoreWeb.registerWith(registrar);
  FirebaseAuthWeb.registerWith(registrar);
  FirebaseCoreWeb.registerWith(registrar);
  SharedPreferencesPlugin.registerWith(registrar);
  registrar.registerMessageHandler();
}
```

Archivo: test/widget_test.dart

```dart
// This is a basic Flutter widget test.
//
// To perform an interaction with a widget in your test, use the WidgetTester
// utility in the flutter_test package. For example, you can send tap and scroll
// gestures. You can also use WidgetTester to find child widgets in the widget
// tree, read text, and verify that the values of widget properties are correct.

import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

import 'package:compartimos_gastos/main.dart';

void main() {
  testWidgets('Counter increments smoke test', (WidgetTester tester) async {
    // Build our app and trigger a frame.
    await tester.pumpWidget(const MyApp());

    // Verify that our counter starts at 0.
    expect(find.text('0'), findsOneWidget);
    expect(find.text('1'), findsNothing);

    // Tap the '+' icon and trigger a frame.
    await tester.tap(find.byIcon(Icons.add));
    await tester.pump();

    // Verify that our counter has incremented.
    expect(find.text('0'), findsNothing);
    expect(find.text('1'), findsOneWidget);
  });
}
```