Repaso de Examen 2ª Evaluación

Rodrigo Iglesias Gorrón

Desarrollo de Interfaces - 2º DAM 2024 / 2025

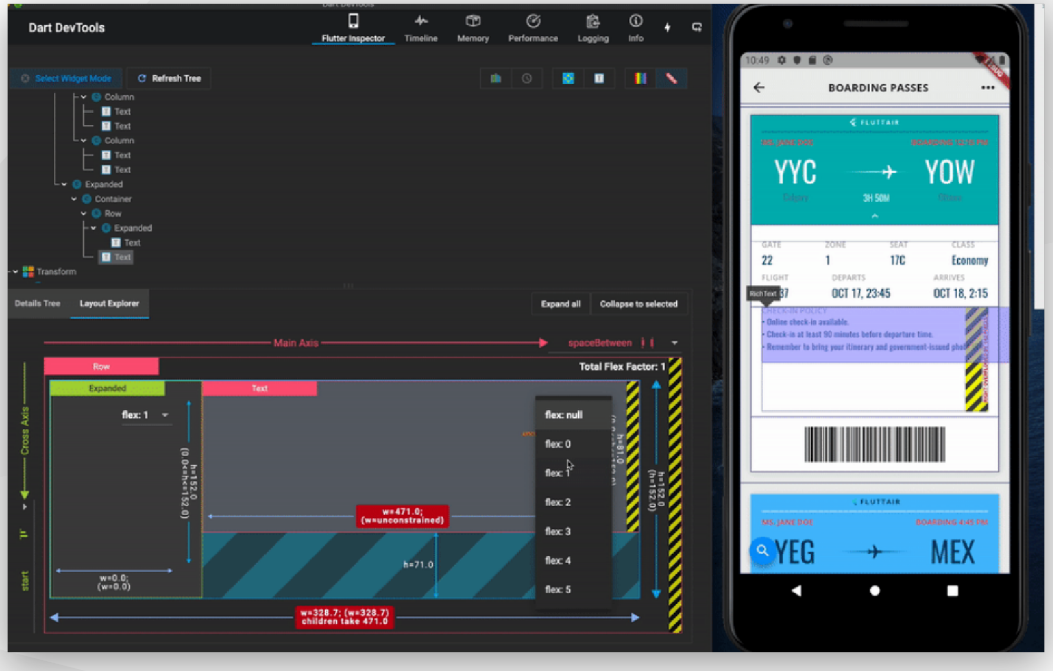
Índice

1. Teoría necesaria

1. **Patrones de diseño** (Tema 1 y Tema 2.5)
   1.1. Patrones gráficos
   1.2. Model-View-ViewModel
2. **Dart** (Tema 2.1)
   2.1. Tipos de datos y colecciones (List, Set, Map...)
   2.2. Operadores, estructuras, control de flujo...
   2.3. Enumeraciones y excepciones

1. Teoría necesaria

3. **Flutter** (Tema 2.3)
   - 3.1. Widgets básicos
   - 3.2. Padding y Margin
   - 3.3. Stack
   - 3.4. ListView y GridView
   - 3.5. Formularios
   - 3.6. Navegación
   - 3.7. Menú (lateral e inferior)



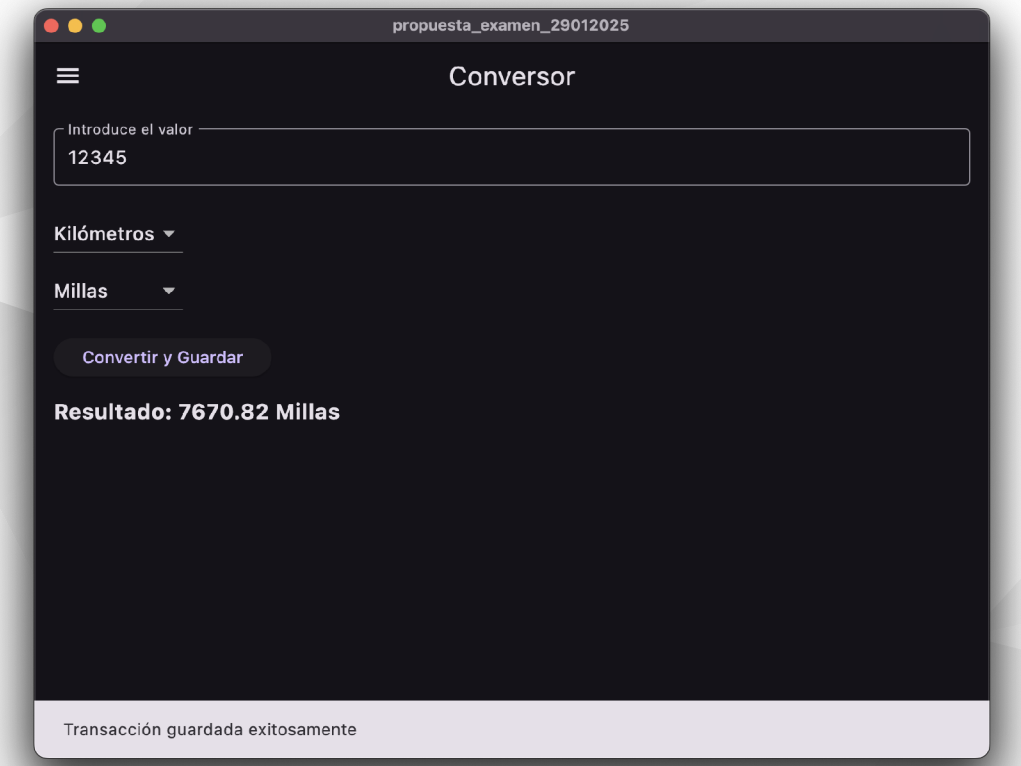
1. Teoría necesaria

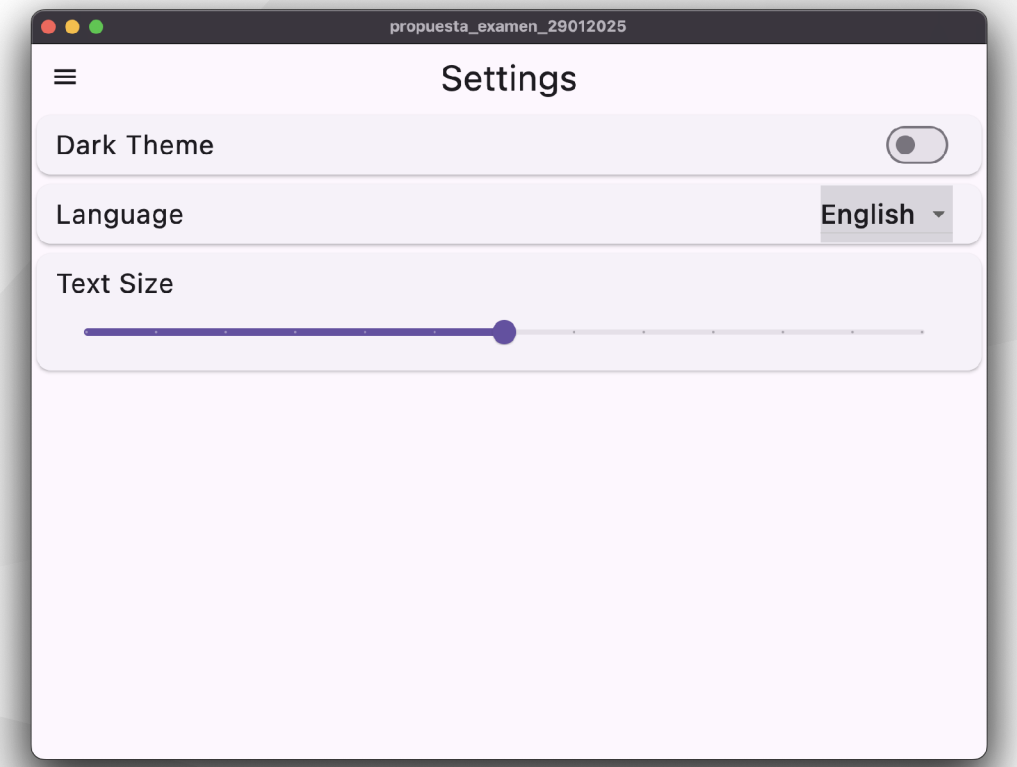
4. **Permanencia de datos** (Tema 2.4)
   - 4.1. Provider con ChangeNotifier
   - 4.2. SharedPreferences
   - 4.3. SQLite
5. **Accesibilidad** (Tema 1 y Tema 3.1)
   - 5.1. Pautas, características...
   - 5.2. Semantics
6. **Diseño Adaptativo** (Tema 3.2)
   - 6.1. MediaQuery

1. Teoría necesaria

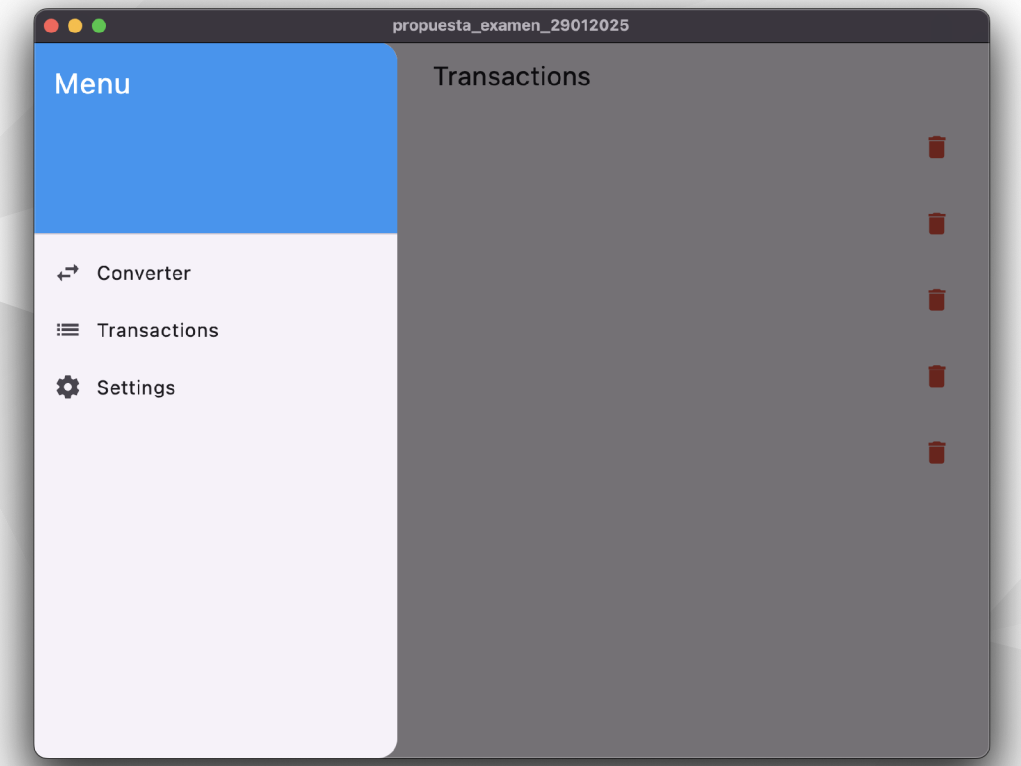
7. **Documentación e Informes** (Tema 3.3)
   7.1. DartDoc
   7.2. Internacionalización
8. **Testing** (Tema 3.5)
   8.1. Test unitarios
   8.2. Test de widgets
   8.3. Test de integración

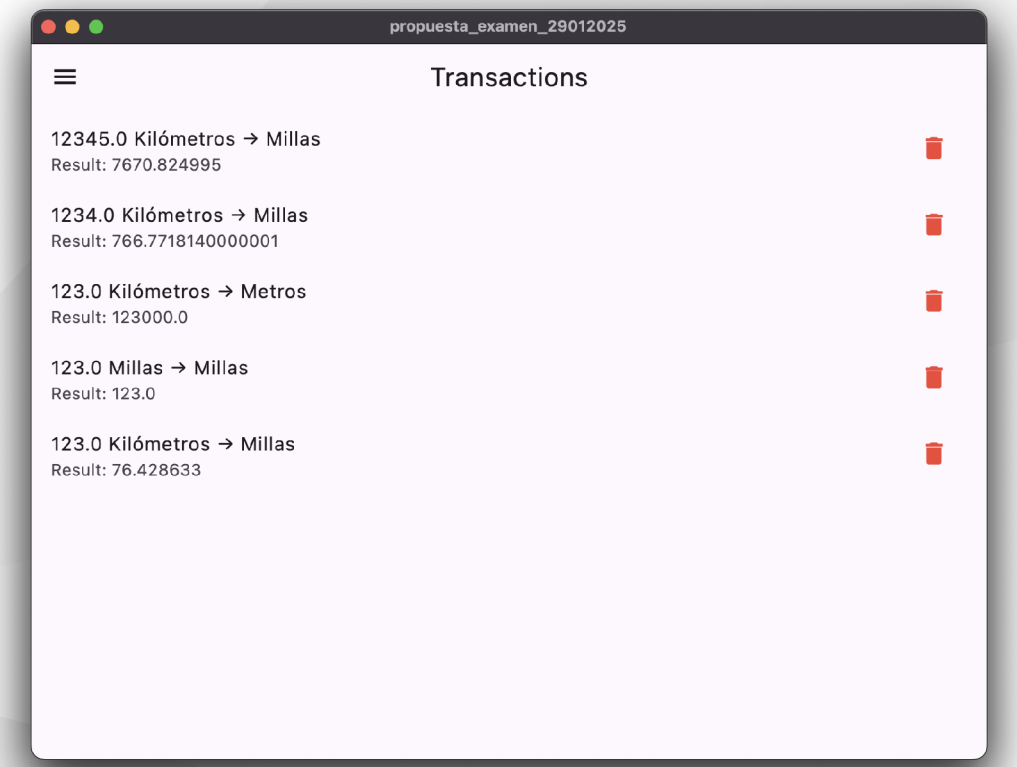
2. Propuesta *similar* al examen





2. Propuesta *similar* al examen





2. Propuesta *similar* al examen

Implementa un conversor con tres pantallas unidas por un `Drawer`.

- Las transacciones han de ser validadas e insertadas en la BBDD.
- Se usará un `Map` para pasar de unas unidades a otras.

```
const conversionRates = {
      'Kilómetros': {'Kilómetros': 1.0, 'Metros': 1000.0, 'Millas': 0.621371},
      'Metros': {'Kilómetros': 0.001, 'Metros': 1.0, 'Millas': 0.000621371},
      'Millas': {'Kilómetros': 1.60934, 'Metros': 1609.34, 'Millas': 1.0},
    };
```

2. Propuesta *similar* al examen

- En la pestaña de **transacciones** aparecerán las transacciones en un `ListView` leídas de la base de datos.
  - Se podrán eliminar de la base de datos.
- En la pestaña de **ajustes** tendremos las `SharedPreferences`:
  - Cambio de modo claro a oscuro.
  - Cambio de idioma (implementado con `intl`).
  - Cambio del tamaño de texto.
- Usar `Provider` para los cambios de tema y separación MVVM.

2. Propuesta *similar* al examen

- Usa `MultiProvider` cuando tengas que implementar varios `Provider`.
- Centraliza los cambios de escala en `MaterialApp` con `MediaQuery`.
- Crea un servicio de base de datos para reutilizar código.
- Crea test automatizados para probar el código.
- Genera la documentación de `DartDoc` adecuada.

3. Ejercicios de repaso

- **Actividad 1**: implementa el patrón MVVM para una pantalla que muestre un contador. Usa `ChangeNotifier` para gestionar el estado del contador, y añade un botón que incremente el contador al ser pulsado, y otro que lo decremente.
- **Actividad 2**: crea un widget que represente una tarjeta de perfil (estilo red social) en un `Stack` con las siguientes características:
  - Nombre y foto se pasan por parámetro.
  - Usa el patrón MVVM para separar los datos del diseño del widget.

Actividad 1

```
// Modelo
class CounterModel {
  int value;
  CounterModel({this.value = 0});
}

// ViewModel
class CounterViewModel extends ChangeNotifier {
  final CounterModel _counterModel;

  CounterViewModel(this._counterModel);

  int get counter => _counterModel.value;

  void increment() {
    _counterModel.value++;
    notifyListeners();
  }

  void decrement() {
    _counterModel.value--;
    notifyListeners();
  }
}
```

Actividad 1

```
void main() {
  runApp(
    ChangeNotifierProvider(
      create: (_) => CounterViewModel(CounterModel()),
      child: MyApp(),
    ),
  );
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: CounterScreen(),
    );
  }
}
```

Actividad 1

```dart
// Vista: Pantalla principal
class CounterScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final counterVM = Provider.of<CounterViewModel>(context);

    return Scaffold(
      appBar: AppBar(title: Text('Contador MVVM')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              'Contador: ${counterVM.counter}',
              style: TextStyle(fontSize: 24),
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                ElevatedButton(
                  onPressed: counterVM.increment,
                  child: Text('+'),
                ),
                SizedBox(width: 20),
                ElevatedButton(
                  onPressed: counterVM.decrement,
                  child: Text('-'),
                ),],),],),),);}}
```

Actividad 2

```dart
/// Modelo
class ProfileModel {
  final String name;
  final String photoUrl;

  ProfileModel({required this.name, required this.photoUrl});
}

/// ViewModel
class ProfileViewModel extends ChangeNotifier {
  final ProfileModel profile;

  ProfileViewModel(this.profile);
}
```

Actividad 2

```dart
/// Vista principal
void main() {
  runApp(
    ChangeNotifierProvider(
      create: (_) => ProfileViewModel(
        ProfileModel(
          name: "José Pérez",
          photoUrl:
              "https://placehold.co/150.jpg",
        ),
      ),
      child: MyApp(),
    ),
  );
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text('Perfil con Stack y MVVM')),
        body: Center(
          child: ProfileCard(),
        ),
      ),
    );
  }
}
```

Actividad 2

```
/// Vista
class ProfileCard extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final profileVM = Provider.of<ProfileViewModel>(context);

    return Stack(
      alignment: Alignment.center,
      children: [
        // Fondo de la tarjeta
        Container(
          width: 300,
          height: 150,
          decoration: BoxDecoration(
            color: Colors.blueAccent,
            borderRadius: BorderRadius.circular(16),
          ),
        ),
        Positioned(
          top: 10,
          child: CircleAvatar(
            radius: 40,
            backgroundImage: NetworkImage(profileVM.profile.photoUrl),
          ),
        ),
        Positioned(
          bottom: 10,
          child: Text(
            profileVM.profile.name,
            style: TextStyle(
              fontSize: 20,
              fontWeight: FontWeight.bold,
              color: Colors.white,),),),],);}}
```

3. Ejercicios de repaso

- **Actividad 3**: implementa un programa que maneje una excepción al intentar convertir un `String` no válido en un entero. Muestra un mensaje de error en un `Text`.
- **Actividad 4**: escribe una función que reciba una lista de enteros y devuelva una lista que contenga solo los números pares. Crea un widget en Flutter que muestre el resultado en un `ListView`.
- **Actividad 5**: implementa la funcionalidad de guardar y recuperar la configuración del tema (oscuro o claro) usando `SharedPreferences`.
  - Incluye un botón para alternar entre los temas.

Actividad 3

```
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: ExceptionHandlingScreen(),
    );
  }
}

class ExceptionHandlingScreen extends StatefulWidget {
  @override
  _ExceptionHandlingScreenState createState() =>
      _ExceptionHandlingScreenState();
}

class _ExceptionHandlingScreenState extends State<ExceptionHandlingScreen> {
  String? errorMessage;
  String input = "";

  void convertStringToInt(String value) {
    try {
      int number = int.parse(value);
      setState(() {
        errorMessage = "El número es: $number";
      });
    } catch (e) {
      setState(() {
        errorMessage = "Error: '$value' no es un número válido.";
      });
    }
  }
```

Actividad 3

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Manejo de Excepciones"),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            TextField(
              decoration: InputDecoration(
                labelText: "Introduce un número",
                border: OutlineInputBorder(),
              ),
              onChanged: (value) {
                input = value;
              },
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () => convertStringToInt(input),
              child: Text("Convertir"),
            ),
            SizedBox(height: 20),
            Text(
              errorMessage ?? "",
              style: TextStyle(
                color: errorMessage != null && errorMessage!.contains("Error")
                    ? Colors.red
                    : Colors.green,
                fontSize: 16,
              ),),],),),);}
}
```

Actividad 4

```dart
void main() {
  runApp(MyApp());
}

// Función para filtrar números pares
List<int> filterEvenNumbers(List<int> numbers) {
  return numbers.where((number) => number % 2 == 0).toList();
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: EvenNumbersScreen(),
    );
  }
}

class EvenNumbersScreen extends StatefulWidget {
  @override
  _EvenNumbersScreenState createState() => _EvenNumbersScreenState();
}

class _EvenNumbersScreenState extends State<EvenNumbersScreen> {
  final List<int> numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
  List<int> evenNumbers = [];

  @override
  void initState() {
    super.initState();
    evenNumbers = filterEvenNumbers(numbers);
  }
```

Actividad 4

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Números Pares"),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            Text(
              "Lista original: ${numbers.join(", ")}",
              style: TextStyle(fontSize: 16),
            ),
            SizedBox(height: 20),
            Expanded(
              child: ListView.builder(
                itemCount: evenNumbers.length,
                itemBuilder: (context, index) {
                  return ListTile(
                    title: Text(
                      evenNumbers[index].toString(),
                      style: TextStyle(fontSize: 18),
                    ),);},),),],),),);}
}
```

Actividad 5

```dart
void main() {
  runApp(MyApp());
}

class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  bool isDarkTheme = false;

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

  // Cargar el tema almacenado en SharedPreferences
  Future<void> _loadTheme() async {
    final prefs = await SharedPreferences.getInstance();
    setState(() {
      isDarkTheme = prefs.getBool('isDarkTheme') ?? false;
    });
  }
```

Actividad 5

```
  // Guardar el tema en SharedPreferences
  Future<void> _saveTheme(bool value) async {
    final prefs = await SharedPreferences.getInstance();
    await prefs.setBool('isDarkTheme', value);
  }

  // Alternar entre los temas
  void _toggleTheme() {
    setState(() {
      isDarkTheme = !isDarkTheme;
    });
    _saveTheme(isDarkTheme);
  }

    @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: isDarkTheme ? ThemeData.dark() : ThemeData.light(),
      home: ThemeSwitcherScreen(
        isDarkTheme: isDarkTheme,
        toggleTheme: _toggleTheme,
      ),
    );
  }
}
```

Actividad 5

```
class ThemeSwitcherScreen extends StatelessWidget {
  final bool isDarkTheme;
  final VoidCallback toggleTheme;

  ThemeSwitcherScreen({required this.isDarkTheme, required this.toggleTheme});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Configuración de Tema')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              isDarkTheme ? 'Tema Oscuro Activado' : 'Tema Claro Activado',
              style: TextStyle(fontSize: 20),
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: toggleTheme,
              child: Text('Cambiar Tema'),
            ),],),),);}
}
```

3. Ejercicios de repaso

- **Actividad 6**: crea una galería simple con un `GridView` que muestre al menos 6 imágenes diferentes cargas desde URLs. Puedes usar placehold.co para ello.
  - Asegúrate de que cambie el número de columnas mostradas de 1 a 2 si la pantalla es ancha (>600px).
- **Actividad 7**: implementa un formulario con nombre y teléfono con soporte de accesibilidad usando `Semantics`.
  - Añade accesibilidad a cada uno de los `TextField` y al `Button`.

Actividad 6

```
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: GalleryScreen(),
    );
  }
}

class GalleryScreen extends StatelessWidget {
  final List<String> imageUrls = [
    "https://placehold.co/300x300/FF0000/FFFFFF.jpg?text=1",
    "https://placehold.co/300x300/00FF00/FFFFFF.jpg?text=2",
    "https://placehold.co/300x300/0000FF/FFFFFF.jpg?text=3",
    "https://placehold.co/300x300/FFFF00/000000.jpg?text=4",
    "https://placehold.co/300x300/FF00FF/FFFFFF.jpg?text=5",
    "https://placehold.co/300x300/00FFFF/000000.jpg?text=6",
  ];
```

Actividad 6

```
  @override
  Widget build(BuildContext context) {
    // Determinar el número de columnas según el ancho de la pantalla
    final isWideScreen = MediaQuery.of(context).size.width > 600;
    final crossAxisCount = isWideScreen ? 2 : 1;

    return Scaffold(
      appBar: AppBar(title: Text('Galería de Imágenes')),
      body: Padding(
        padding: const EdgeInsets.all(8.0),
        child: GridView.builder(
          gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
            crossAxisCount: crossAxisCount,
            crossAxisSpacing: 8,
            mainAxisSpacing: 8,
          ),
          itemCount: imageUrls.length,
          itemBuilder: (context, index) {
            return Card(
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(8),
              ),
              elevation: 4,
              child: ClipRRect(
                borderRadius: BorderRadius.circular(8),
                child: Image.network(
                  imageUrls[index],
                  fit: BoxFit.cover,
                ),),);},),),);}
}
```

Actividad 7

```
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: AccessibleForm(),
    );
  }
}

class AccessibleForm extends StatefulWidget {
  @override
  _AccessibleFormState createState() => _AccessibleFormState();
}

class _AccessibleFormState extends State<AccessibleForm> {
  final _formKey = GlobalKey<FormState>();
  final _nameController = TextEditingController();
  final _phoneController = TextEditingController();

  void _submitForm() {
    if (_formKey.currentState!.validate()) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Formulario enviado con éxito')),
      );
    }
  }
```

Actividad 7

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Formulario Accesible'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Form(
        key: _formKey,
        child: Column(
          children: [
            Semantics(
              label: 'Campo de texto para nombre',
              hint: 'Introduce tu nombre completo',
              child: TextFormField(
                controller: _nameController,
                decoration: InputDecoration(
                  labelText: 'Nombre',
                  border: OutlineInputBorder(),
                ),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'El nombre no puede estar vacío';
                  }
                  return null;
                },
              ),
            ),
```

Actividad 7

```
            SizedBox(height: 20),
            Semantics(
              label: 'Campo de texto para teléfono',
              hint: 'Introduce tu número de teléfono',
              child: TextFormField(
                controller: _phoneController,
                decoration: InputDecoration(
                  labelText: 'Teléfono',
                  border: OutlineInputBorder(),
                ),
                keyboardType: TextInputType.phone,
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'El teléfono no puede estar vacío';
                  }
                  if (!RegExp(r'^\d+$').hasMatch(value)) {
                    return 'Introduce solo números';
                  }
                  return null;
                },
              ),
            ),
            SizedBox(height: 20),
            Semantics(
              label: 'Botón de enviar formulario',
              hint: 'Pulsa para enviar los datos',
              child: ElevatedButton(
                onPressed: _submitForm,
                child: Text('Enviar'),
              ),),],),),),);}
}
```

3. Ejercicios de repaso

- **Actividad 8**: implementa los test unitarios, de widget y de integración en la actividad 7. Añade la funcionalidad de limpiar los `TextField` después de dar al botón y prueba esa funcionalidad.
- **Actividad 9**: traduce todos los textos de la actividad 7 utilizando `intl` y `flutter_localizations`.

Actividad 8

Cambiamos:

```
void _submitForm() {
  if (_formKey.currentState!.validate()) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Formulario enviado con éxito')),
    );
    _clearFields();
  }
}

void _clearFields() {
  _nameController.clear();
  _phoneController.clear();
}
```

Actividad 8 (`test/unit_test.dart`)

```dart
void main() {
  group('Validación de formulario', () {
    test('Nombre vacío muestra error', () {
      final result = validateName('');
      expect(result, 'El nombre no puede estar vacío');
    });

    test('Nombre válido no muestra error', () {
      final result = validateName('John Doe');
      expect(result, null);
    });

    test('Teléfono vacío muestra error', () {
      final result = validatePhone('');
      expect(result, 'El teléfono no puede estar vacío');
    });

    test('Teléfono no numérico muestra error', () {
      final result = validatePhone('abc123');
      expect(result, 'Introduce solo números');
    });

    test('Teléfono válido no muestra error', () {
      final result = validatePhone('1234567890');
      expect(result, null);
    });
  });
}
```

Actividad 8 (`test/unit_test.dart`)

```
String? validateName(String? value) {
  if (value == null || value.isEmpty) return 'El nombre no puede estar vacío';
  return null;
}

String? validatePhone(String? value) {
  if (value == null || value.isEmpty) return 'El teléfono no puede estar vacío';
  if (!RegExp(r'^\d+$').hasMatch(value)) return 'Introduce solo números';
  return null;
}
```

Actividad 8 (`test/widget_test.dart`)

```dart
void main() {
  testWidgets('Enviar formulario limpia los campos', (WidgetTester tester) async {
    // Construir el widget
    await tester.pumpWidget(MyApp());

    // Buscar los TextFormFields y el botón
    final nameField = find.byType(TextFormField).first;
    final phoneField = find.byType(TextFormField).last;
    final submitButton = find.text('Enviar');

    // Introducir texto en los campos
    await tester.enterText(nameField, 'John Doe');
    await tester.enterText(phoneField, '123456');

    // Comprobar que los campos contienen texto
    expect(find.text('John Doe'), findsOneWidget);
    expect(find.text('123456'), findsOneWidget);

    // Pulsar el botón de enviar
    await tester.tap(submitButton);
    await tester.pump();

    // Comprobar que los campos están vacíos
    expect(find.text('John Doe'), findsNothing);
    expect(find.text('123456'), findsNothing);

    // Comprobar que el SnackBar aparece
    expect(find.text('Formulario enviado con éxito'), findsOneWidget);
  });
}
```

Actividad 8 (`integration_test/app_test.dart`)

```dart
void main() {
  IntegrationTestWidgetsFlutterBinding.ensureInitialized();

  testWidgets('Completar formulario y enviarlo', (WidgetTester tester) async {
    await tester.pumpWidget(MyApp());

    // Rellenar campos
    await tester.enterText(find.byType(TextFormField).at(0), 'John Doe');
    await tester.enterText(find.byType(TextFormField).at(1), '123456');

    // Pulsar enviar
    await tester.tap(find.text('Enviar'));
    await tester.pumpAndSettle();

    // Verificar que los campos están vacíos
    expect(find.text('John Doe'), findsNothing);
    expect(find.text('123456'), findsNothing);

    // Verificar que aparece el SnackBar
    expect(find.text('Formulario enviado con éxito'), findsOneWidget);
  });
}
```

Actividad 9

```dart
import 'package:flutter/material.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'package:flutter_localizations/flutter_localizations.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      localizationsDelegates: [
        AppLocalizations.delegate,
        GlobalMaterialLocalizations.delegate,
        GlobalWidgetsLocalizations.delegate,
        GlobalCupertinoLocalizations.delegate,
      ],
      supportedLocales: [
        Locale('en', ''), // Inglés
        Locale('es', ''), // Español
      ],
      home: AccessibleForm(),
      locale: Locale('en'),
    );
  }
}
```

Actividad 9

```dart
class AccessibleForm extends StatefulWidget {
  @override
  _AccessibleFormState createState() => _AccessibleFormState();
}

class _AccessibleFormState extends State<AccessibleForm> {
  final _formKey = GlobalKey<FormState>();
  final _nameController = TextEditingController();
  final _phoneController = TextEditingController();

  void _submitForm() {
    if (_formKey.currentState!.validate()) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text(AppLocalizations.of(context)!.formSubmitSuccess),
        ),
      );
      _clearFields();
    }
  }

  void _clearFields() {
    _nameController.clear();
    _phoneController.clear();
  }
```

Actividad 9

```
@override
Widget build(BuildContext context) {
  final localizations = AppLocalizations.of(context)!;

  return Scaffold(
    appBar: AppBar(
      title: Text(localizations.appTitle),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Form(
        key: _formKey,
        child: Column(
          children: [
            Semantics(
              label: localizations.formNameLabel,
              hint: localizations.formNameHint,
              child: TextFormField(
                controller: _nameController,
                decoration: InputDecoration(
                  labelText: localizations.formNameLabel,
                  border: OutlineInputBorder(),
                ),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return localizations.formNameValidation;
                  }
                  return null;
                },
              ),
            ),
```

Actividad 9

```
            SizedBox(height: 20),
            Semantics(
              label: localizations.formPhoneLabel,
              hint: localizations.formPhoneHint,
              child: TextFormField(
                controller: _phoneController,
                decoration: InputDecoration(
                  labelText: localizations.formPhoneLabel,
                  border: OutlineInputBorder(),
                ),
                keyboardType: TextInputType.phone,
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return localizations.formPhoneValidationEmpty;
                  }
                  if (!RegExp(r'^\d+$').hasMatch(value)) {
                    return localizations.formPhoneValidationInvalid;
                  }
                  return null;
                },
              ),
            ),
            SizedBox(height: 20),
            Semantics(
              label: localizations.formSubmitButton,
              hint: localizations.formSubmitButton,
              child: ElevatedButton(
                onPressed: _submitForm,
                child: Text(localizations.formSubmitButton),
              ),),],),),),);}
}
```

Actividad 9

```
{
  "appTitle": "Accessible Form",
  "formNameLabel": "Name",
  "formNameHint": "Enter your full name",
  "formNameValidation": "Name cannot be empty",
  "formPhoneLabel": "Phone",
  "formPhoneHint": "Enter your phone number",
  "formPhoneValidationEmpty": "Phone cannot be empty",
  "formPhoneValidationInvalid": "Only numbers are allowed",
  "formSubmitButton": "Submit",
  "formSubmitSuccess": "Form submitted successfully"
}
```

Actividad 9

```
{
  "appTitle": "Formulario Accesible",
  "formNameLabel": "Nombre",
  "formNameHint": "Introduce tu nombre completo",
  "formNameValidation": "El nombre no puede estar vacío",
  "formPhoneLabel": "Teléfono",
  "formPhoneHint": "Introduce tu número de teléfono",
  "formPhoneValidationEmpty": "El teléfono no puede estar vacío",
  "formPhoneValidationInvalid": "Solo se permiten números",
  "formSubmitButton": "Enviar",
  "formSubmitSuccess": "Formulario enviado con éxito"
}
```

3. Ejercicios de repaso

- **Actividad 10**: crea una aplicación que permita al usuario hacer escritura y lectura de las transacciones almacenadas en una base de datos SQLite.
  - Separa `database_service.dart` del `main.dart`.
  - Haz que el valor se muestre en un `Snackbar`.

Actividad 10

```
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Ejemplo BBDD',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: SimpleDatabaseScreen(),
    );
  }
}

class SimpleDatabaseScreen extends StatefulWidget {
  @override
  _SimpleDatabaseScreenState createState() => _SimpleDatabaseScreenState();
}

class _SimpleDatabaseScreenState extends State<SimpleDatabaseScreen> {
  final DatabaseService _databaseService = DatabaseService();
  final _formKey = GlobalKey<FormState>();
  final _valueController = TextEditingController();
  double? _lastValue;
```

Actividad 10

```
Future<void> _saveValue() async {
  if (_formKey.currentState!.validate()) {
    final value = double.parse(_valueController.text);
    await _databaseService.insertValue(value);

    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Valor $value guardado en la base de datos')),
    );

    _valueController.clear();
  }
}

Future<void> _getValue() async {
  final value = await _databaseService.getLastValue();
  setState(() {
    _lastValue = value;
  });

  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(value != null
          ? 'Valor cargado: $value'
          : 'No hay valores en la base de datos'),
    ),
  );
}
```

Actividad 10

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Ejemplo BBDD'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          // Formulario para insertar el valor
          Form(
            key: _formKey,
            child: TextFormField(
              controller: _valueController,
              keyboardType: TextInputType.number,
              decoration: InputDecoration(
                labelText: 'Introduce un valor',
                border: OutlineInputBorder(),
              ),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Por favor, inserta un valor';
                }
                if (double.tryParse(value) == null) {
                  return 'Inserta un número válido';
                }
                return null;
              },
            ),
          ),
          SizedBox(height: 16),
```

Actividad 10

```
            // Botón para guardar el valor
            ElevatedButton(
              onPressed: _saveValue,
              child: Text('Guardar Valor'),
            ),
            SizedBox(height: 16),

            // Botón para cargar el último valor
            ElevatedButton(
              onPressed: _getValue,
              child: Text('Cargar Valor'),
            ),
            SizedBox(height: 16),

            // Mostrar el último valor guardado
            if (_lastValue != null)
              Text(
                'Último valor guardado: $_lastValue',
                style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
              ),
          ],
        ),
      ),
    );
  }
}
```