Módulo 5: Almacenamiento de Datos Locales en Aplicaciones Móviles

Lección 2: Uso de shared_preferences para Almacenamiento de Datos

Objetivos de la Lección

- Comprender qué es `shared_preferences` y cómo se utiliza para almacenar datos simples en una aplicación móvil.
- Implementar el almacenamiento y la recuperación de preferencias del usuario, como el nombre de usuario y el modo de tema (oscuro o claro).
- Aplicar `shared_preferences` en un caso práctico que permite recordar y restaurar configuraciones al reiniciar la aplicación.

Introducción de la Lección

`shared_preferences` es un paquete de Flutter que permite almacenar datos simples en el dispositivo de manera persistente. Utiliza un almacenamiento basado en pares clave-valor, lo que es ideal para guardar configuraciones de usuario, preferencias y otros pequeños datos que deben mantenerse incluso después de que la aplicación se cierra y vuelve a abrir.

¿Cuándo utilizar shared_preferences?

- Almacenar el nombre de usuario para mostrarlo al inicio de la aplicación.
- Guardar el estado del tema de la aplicación (oscuro o claro) para restaurarlo al iniciar.

Instalación de `shared_preferences`

Para usar `shared_preferences` en un proyecto Flutter, primero debes agregar la dependencia en tu archivo `pubspec.yaml`:

```
dependencies:
  flutter:
    sdk: flutter
  shared_preferences: ^2.0.15
```

Luego, almacena el archivo `pubspec.yaml` o en el terminal de Visual Studio Code ejecuta el comando `flutter pub get` en la terminal para descargar la dependencia.

Ejemplo Práctico: Recordar el Nombre de Usuario y el Modo de Tema

Supongamos que estamos desarrollando una aplicación móvil que permite al usuario ingresar su nombre y elegir entre el modo claro u oscuro. Queremos que la aplicación recuerde estas configuraciones incluso después de cerrarla.

Desarrollo del Caso Práctico

1. **Configuración Inicial del Proyecto**
    - Crea un nuevo proyecto Flutter en Visual Studio Code o cualquier editor de tu elección.
    - Asegúrate de haber instalado la dependencia `shared_preferences` como se mencionó anteriormente.
2. **Diseño de la Interfaz de Usuario**
    - Usaremos un widget `Scaffold` con un campo de texto para ingresar el nombre y dos botones para seleccionar el modo de tema.

Código de Ejemplo Completo

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

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

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

class _MyAppState extends State<MyApp> {
  bool isDarkMode = false;
  String userName = "";
  late TextEditingController _controller;

  @override
  void initState() {
    super.initState();
    _controller = TextEditingController();
    _loadPreferences();
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }
```

```
  Future<void> _loadPreferences() async {
    SharedPreferences prefs = await
SharedPreferences.getInstance();
    setState(() {
      isDarkMode = prefs.getBool('isDarkMode') ?? false;
      userName = prefs.getString('userName') ?? "";
      _controller.text = userName;
    });
  }

  Future<void> _saveThemePreference(bool value) async {
    SharedPreferences prefs = await
SharedPreferences.getInstance();
    prefs.setBool('isDarkMode', value);
  }

  Future<void> _saveUserName(String name) async {
    SharedPreferences prefs = await
SharedPreferences.getInstance();
    prefs.setString('userName', name);
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: isDarkMode ? ThemeData.dark() : ThemeData.light(),
      home: Scaffold(
        appBar: AppBar(
          title: const Text("Preferencias de Usuario"),
```

```
        ),
        body: Builder(
          builder: (BuildContext context) => Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                TextField(
                  decoration: const InputDecoration(
                    labelText: "Nombre de Usuario",
                  ),
                  onChanged: (value) {
                    userName = value;
                  },
                  controller: _controller,
                ),
                const SizedBox(height: 20),
                Row(
                  mainAxisAlignment:
MainAxisAlignment.spaceAround,
                  children: [
                    ElevatedButton(
                      onPressed: () {
                        setState(() {
                          isDarkMode = false;
                          _saveThemePreference(isDarkMode);
                        });
                      },
                      child: const Text("Modo Claro"),
                    ),
```

```
                    ElevatedButton(
                      onPressed: () {
                        setState(() {
                          isDarkMode = true;
                          _saveThemePreference(isDarkMode);
                        });
                      },
                      child: const Text("Modo Oscuro"),
                    ),
                  ],
                ),
                const SizedBox(height: 20),
                ElevatedButton(
                  onPressed: () {
                    _saveUserName(userName);
                    ScaffoldMessenger.of(context).showSnackBar(
                      const SnackBar(
                        content: Text('Preferencias
guardadas.'),
                      ),
                    );
                  },
                  child: Text("Guardar Preferencias"),
                ),
              ],
            ),
          ),
        ),
      ),
    );
```

```
  }
}
```

Explicación del Código

1. **Variables y Métodos Principales**:
    - `isDarkMode`: Una variable booleana que controla si el tema es oscuro o claro.
    - `userName`: Una variable que almacena el nombre del usuario.
    - `_loadPreferences()`: Un método que carga las preferencias almacenadas usando `shared_preferences`.
    - `_saveThemePreference(bool value)`: Un método que guarda el estado del tema.
    - `_saveUserName(String name)`: Un método que guarda el nombre del usuario.
2. **Uso de `SharedPreferences`**:
    - `SharedPreferences.getInstance()`: Obtiene la instancia de `shared_preferences` para leer y escribir datos.
    - `prefs.setBool()` y `prefs.setString()`: Métodos para guardar datos en formato clave-valor.
    - `prefs.getBool()` y `prefs.getString()`: Métodos para recuperar datos almacenados.
3. **Interfaz de Usuario**:
    - Un `TextField` para ingresar el nombre del usuario.
    - Dos botones (`ElevatedButton`) para seleccionar el modo de tema (claro u oscuro).
    - Un botón para guardar las preferencias, que muestra un mensaje de confirmación.

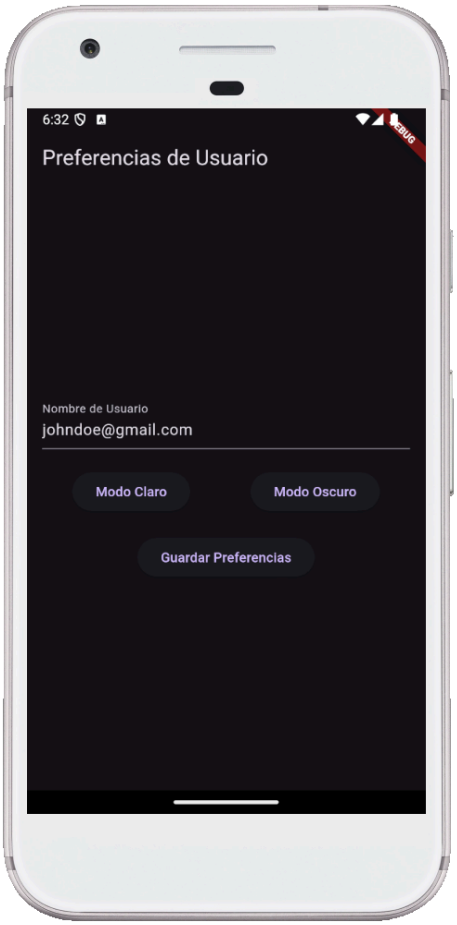

Imagen de aplicación móvil que almacena datos de preferencias con shared_prefereces con la opción de modo oscuros (darkmode) en Flutter. Creada por Javier A. Dastas (2024)

Resumen de la Lección

En esta lección, aprendimos a usar `shared_preferences` en Flutter para almacenar y recuperar datos simples de forma persistente. Vimos un ejemplo práctico de cómo recordar el nombre del usuario y el estado del tema (oscuro o claro) en una aplicación móvil. Este enfoque es útil para mejorar la experiencia del usuario, asegurando que las configuraciones personales se mantengan entre sesiones.

Actividad de la Lección

Esta actividad te ayudará a practicar el almacenamiento y recuperación de datos usando `shared_preferences`, preparándote para implementar características personalizadas en aplicaciones móviles Flutter.

Instrucciones:

1. Modifica el ejemplo proporcionado para agregar una preferencia adicional, como recordar el idioma preferido del usuario (por ejemplo, "español" o "inglés").

2. Implementa una función que permita restablecer todas las preferencias a sus valores predeterminados.
3. Luego de realizar todos los cambios de la actividad desarrolla un documento con evidencia de la realización del código en esta lección y los requerimientos de esta actividad. Debes incluir imágenes de las pantallas de la aplicación, imagen de la estructura de archivos y carpetas de tu proyecto, y copia de todo el código.
4. Entrega el documento desarrollado en formato PDF en el enlace provisto para esta actividad.