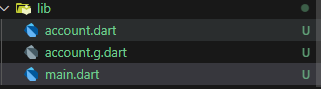
**Actividad de la Lección de Hive**  
Esta actividad te permitirá aplicar conocimientos de manipulación de listas y mejorar la  
funcionalidad de una aplicación de almacenamiento de datos en Flutter usando Hive.



**pubspec.yaml:**

name: h\_i\_v\_e

description: "A new Flutter project."

# Prevent accidental publishing to pub.dev

publish\_to: 'none' # Remove this line if you wish to publish to pub.dev

version: 1.0.0+1

environment:

  sdk: ^3.5.3

# Dependencies for your project

dependencies:

  flutter:

    sdk: flutter

  hive: ^2.2.3

  hive\_flutter: ^1.1.0

  path\_provider: ^2.1.5

  cupertino\_icons: ^1.0.8

# Development dependencies

dev\_dependencies:

  hive\_generator: ^2.0.1

  build\_runner: ^2.4.13

  flutter\_test:

    sdk: flutter

  flutter\_lints: ^4.0.0

# Flutter-specific settings

flutter:

  uses-material-design: true

Main.dart:

import 'package:flutter/material.dart';

import 'package:hive/hive.dart';

import 'package:hive\_flutter/hive\_flutter.dart';

import 'account.dart';

void main() async {

  WidgetsFlutterBinding.ensureInitialized();

  await Hive.initFlutter();

  Hive.registerAdapter(AccountAdapter());

  await Hive.openBox<Account>('accounts');

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      theme: ThemeData.light(),

      home: AccountManager(),

    );

  }

}

class AccountManager extends StatefulWidget {

  @override

  \_AccountManagerState createState() => \_AccountManagerState();

}

class \_AccountManagerState extends State<AccountManager> {

  final \_serviceNameController = TextEditingController();

  final \_userNameController = TextEditingController();

  final \_passwordController = TextEditingController();

  String \_selectedServiceType = 'WebPage';

  Box<Account> accountBox = Hive.box<Account>('accounts');

  bool \_obscurePassword = true;

  int? \_editingIndex;

  void \_addOrUpdateAccount() {

    String currentDate =

        '${DateTime.now().year}-${DateTime.now().month}-${DateTime.now().day}';

    if (\_editingIndex == null) {

      // Agregar nueva cuenta

      final account = Account(

        serviceName: \_serviceNameController.text,

        serviceType: \_selectedServiceType,

        userName: \_userNameController.text,

        password: \_passwordController.text,

        createdAt: DateTime.now(),

        updateHistory: currentDate, // Fecha de creación inicial

      );

      accountBox.add(account);

    } else {

      // Actualizar cuenta existente

      final account = accountBox.getAt(\_editingIndex!)!;

      account

        ..serviceName = \_serviceNameController.text

        ..serviceType = \_selectedServiceType

        ..userName = \_userNameController.text

        ..password = \_passwordController.text

        ..updateHistory = currentDate; // Fecha de última actualización

      accountBox.putAt(\_editingIndex!, account);

    }

    \_clearFields();

  }

  void \_deleteAccount(int index) {

    accountBox.deleteAt(index);

    \_clearFields();

  }

  void \_clearFields() {

    \_serviceNameController.clear();

    \_userNameController.clear();

    \_passwordController.clear();

    setState(() {

      \_selectedServiceType = 'WebPage';

      \_editingIndex = null;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text("Gestor de Cuentas"),

      ),

      body: Column(

        children: [

          Padding(

            padding: const EdgeInsets.all(18.0),

            child: Column(

              children: [

                Row(

                  children: [

                    const Text(

                      'Servicio',

                      style: TextStyle(fontSize: 16),

                    ),

                    const SizedBox(

                      width: 10,

                    ),

                    Expanded(

                      child: DropdownButton<String>(

                        value: \_selectedServiceType,

                        items: [

                          'WebPage',

                          'Mobile App',

                          'Bank Account',

                          'eMail',

                          'PC Login',

                          'Others'

                        ].map((String value) {

                          return DropdownMenuItem<String>(

                            value: value,

                            child: Text(value),

                          );

                        }).toList(),

                        onChanged: (newValue) {

                          setState(() {

                            \_selectedServiceType = newValue!;

                          });

                        },

                      ),

                    ),

                  ],

                ),

                TextField(

                  controller: \_serviceNameController,

                  decoration: InputDecoration(labelText: "Nombre del Servicio"),

                ),

                TextField(

                  controller: \_userNameController,

                  decoration: InputDecoration(labelText: "Nombre de Usuario"),

                ),

                Row(

                  children: [

                    Expanded(

                      child: TextField(

                        controller: \_passwordController,

                        decoration: InputDecoration(labelText: "Contraseña"),

                        obscureText: \_obscurePassword,

                      ),

                    ),

                    IconButton(

                      icon: Icon(\_obscurePassword

                          ? Icons.visibility

                          : Icons.visibility\_off),

                      onPressed: () {

                        setState(() {

                          \_obscurePassword = !\_obscurePassword;

                        });

                      },

                    ),

                  ],

                ),

                ElevatedButton(

                  onPressed: \_addOrUpdateAccount,

                  child: Text(\_editingIndex == null

                      ? "Agregar Cuenta"

                      : "Actualizar Cuenta"),

                ),

              ],

            ),

          ),

          Expanded(

            child: ValueListenableBuilder(

              valueListenable: accountBox.listenable(),

              builder: (context, Box<Account> box, \_) {

                if (box.values.isEmpty) {

                  return Center(child: Text("No hay cuentas almacenadas."));

                }

                return ListView.builder(

                  itemCount: box.length,

                  itemBuilder: (context, index) {

                    final account = box.getAt(index)!;

                    return ListTile(

                      title: Text(account.serviceName),

                      subtitle: Column(

                        crossAxisAlignment: CrossAxisAlignment.start,

                        children: [

                          Text("Tipo: ${account.serviceType}"),

                          Text(

                              "Última Actualización: ${account.updateHistory}"),

                        ],

                      ),

                      trailing: Row(

                        mainAxisSize: MainAxisSize.min,

                        children: [

                          IconButton(

                            icon: Icon(Icons.edit),

                            onPressed: () {

                              setState(() {

                                \_serviceNameController.text =

                                    account.serviceName;

                                \_selectedServiceType = account.serviceType;

                                \_userNameController.text = account.userName;

                                \_passwordController.text = account.password;

                                \_editingIndex = index;

                              });

                            },

                          ),

                          IconButton(

                            icon: Icon(Icons.delete),

                            onPressed: () => \_deleteAccount(index),

                          ),

                        ],

                      ),

                    );

                  },

                );

              },

            ),

          ),

        ],

      ),

    );

  }

}

account.g.dart:

// GENERATED CODE - DO NOT MODIFY BY HAND

part of 'account.dart';

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// TypeAdapterGenerator

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class AccountAdapter extends TypeAdapter<Account> {

  @override

  final int typeId = 0;

  @override

  Account read(BinaryReader reader) {

    final numOfFields = reader.readByte();

    final fields = <int, dynamic>{

      for (int i = 0; i < numOfFields; i++) reader.readByte(): reader.read(),

    };

    return Account(

      serviceName: fields[0] as String,

      serviceType: fields[1] as String,

      userName: fields[2] as String,

      password: fields[3] as String,

      createdAt: fields[4] as DateTime,

      updateHistory: fields[5] as String, // Nuevo campo

    );

  }

  @override

  void write(BinaryWriter writer, Account obj) {

    writer

      ..writeByte(6) // Incrementar el número total de campos

      ..writeByte(0)

      ..write(obj.serviceName)

      ..writeByte(1)

      ..write(obj.serviceType)

      ..writeByte(2)

      ..write(obj.userName)

      ..writeByte(3)

      ..write(obj.password)

      ..writeByte(4)

      ..write(obj.createdAt)

      ..writeByte(5)

      ..write(obj.updateHistory); // Escribir el nuevo campo

  }

  @override

  int get hashCode => typeId.hashCode;

  @override

  bool operator ==(Object other) =>

      identical(this, other) ||

      other is AccountAdapter &&

          runtimeType == other.runtimeType &&

          typeId == other.typeId;

}

Account.dart:

import 'package:hive/hive.dart';

part 'account.g.dart';

@HiveType(typeId: 0)

class Account {

  @HiveField(0)

  String serviceName;

  @HiveField(1)

  String serviceType;

  @HiveField(2)

  String userName;

  @HiveField(3)

  String password;

  @HiveField(4)

  DateTime createdAt;

  @HiveField(5)

  String updateHistory; // Nuevo campo

  Account({

    required this.serviceName,

    required this.serviceType,

    required this.userName,

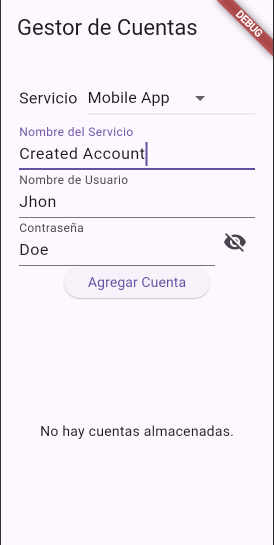
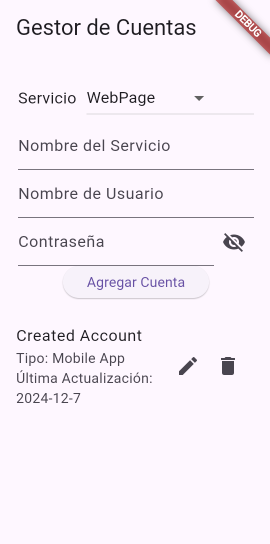
    required this.password,

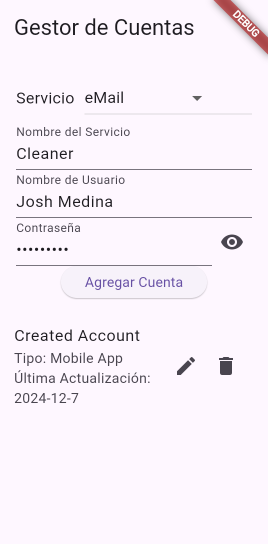
    required this.createdAt,

    required this.updateHistory, // Nuevo campo en el constructor

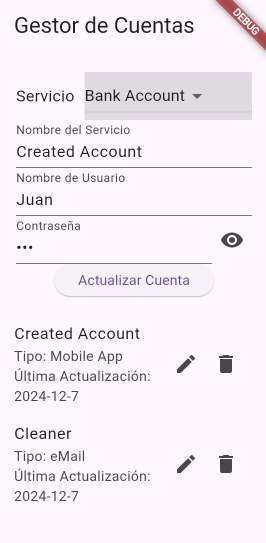
  });

}

 **Agregando cuenta** 

**Segunda cuenta** A screenshot of a phone

Description automatically generated

**Actualizando y eliminando la cuenta** **(cleaner**)A screenshot of a phone

Description automatically generated