//FLUTTER-VIEW\_MODEL/BLOC\_NOTE\_BLOC (1)

import 'package:flutter\_appl/models/note\_model.dart';

import 'package:flutter\_appl/note\_database/note\_database.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

import 'note\_event.dart';

import 'note\_state.dart';

class NoteBloc extends Bloc<NoteEvent, NoteState> {

final NoteDatabase \_noteDatabase;

List<Note> \_notes = [];

NoteBloc(this.\_noteDatabase) : super(NoteInitial());

@override

Stream<NoteState> mapEventToState(NoteEvent event) async\* {

if (event is NoteInitialEvent) {

yield\* \_mapInitialEventToState();

}

if (event is NoteAddEvent) {

yield\* \_mapNoteAddEventToState(

title: event.title, content: event.content);

}

if (event is NoteEditEvent) {

yield\* \_mapNoteEditEventToState(

title: event.title, content: event.content, index: event.index);

}

if (event is NoteDeleteEvent) {

yield\* \_mapNoteDeleteEventToState(index: event.index);

}

}

//FLUTTER-VIEW\_MODEL/BLOC\_NOTE\_BLOC (2)

// Stream Functions

Stream<NoteState> \_mapInitialEventToState() async\* {

yield NotesLoading();

await \_getNotes();

yield YourNotesState(notes: \_notes);

}

Stream<NoteState> \_mapNoteAddEventToState(

{String title, String content}) async\* {

yield NotesLoading();

await \_addToNotes(title: title, content: content);

yield YourNotesState(notes: \_notes);

}

Stream<NoteState> \_mapNoteEditEventToState(

{String title, String content, int index}) async\* {

yield NotesLoading();

await \_updateNote(newTitle: title, newContent: content, index: index);

yield YourNotesState(notes: \_notes);

}

Stream<NoteState> \_mapNoteDeleteEventToState({int index}) async\* {

yield NotesLoading();

await \_removeFromNotes(index: index);

\_notes.sort((a, b) {

var aDate = a.title;

var bDate = b.title;

return aDate.compareTo(bDate);

});

yield YourNotesState(notes: \_notes);

}

//FLUTTER-VIEW\_MODEL/BLOC\_NOTE\_BLOC (3)

// Helper Functions

Future<void> \_getNotes() async {

await \_noteDatabase.getFullNote().then((value) {

\_notes = value;

});

}

Future<void> \_addToNotes({String title, String content}) async {

await \_noteDatabase.addToBox(Note(title: title, content: content));

await \_getNotes();

}

Future<void> \_updateNote(

{int index, String newTitle, String newContent}) async {

await \_noteDatabase.updateNote(

index, Note(title: newTitle, content: newContent));

await \_getNotes();

}

Future<void> \_removeFromNotes({int index}) async {

await \_noteDatabase.deleteFromBox(index);

await \_getNotes();

}

}

//FLUTTER-VIEW\_MODEL/BLOC\_NOTE\_EVENT

import 'package:equatable/equatable.dart';

import 'package:flutter/cupertino.dart';

abstract class NoteEvent extends Equatable {

@override

List<Object> get props => [];

}

// initial event

class NoteInitialEvent extends NoteEvent {

}

// add event

class NoteAddEvent extends NoteEvent {

final String title, content;

NoteAddEvent({@required this.title, @required this.content});

}

// edit event

class NoteEditEvent extends NoteEvent {

final String title, content;

final int index;

NoteEditEvent(

{@required this.title, @required this.content, @required this.index});

}

// delete event

class NoteDeleteEvent extends NoteEvent {

final int index;

NoteDeleteEvent({@required this.index});

}

//FLUTTER-VIEW\_MODEL/BLOC\_NOTE\_STATE

import 'package:equatable/equatable.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter\_appl/models/note\_model.dart';

abstract class NoteState extends Equatable {

@override

List<Object> get props => [];

}

// initial

class NoteInitial extends NoteState {}

// loading

class NotesLoading extends NoteState {}

// edit notes

class EditNotesState extends NoteState {

final Note note;

EditNotesState({@required this.note});

}

// your notes

class YourNotesState extends NoteState {

final List<Note> notes; // get all notes

YourNotesState({@required this.notes});

}

// new note

class NewNoteState extends NoteState {}

//FLUTTER-MODEL/NOTE MODEL

import 'package:hive/hive.dart';

part 'note\_model.g.dart';

@HiveType(typeId: 1)

class Note {

@HiveField(0)

final String title;

@HiveField(1)

final String content;

Note({this.title, this.content});

}

//FLUTTER-SERVICES/NOTE\_DATABASE SERVICES

import 'package:hive/hive.dart';

import 'dart:async';

import '../models/note\_model.dart';

class NoteDatabase {

String \_boxName = "Note";

// open a box

Future<Box> noteBox() async {

var box = await Hive.openBox<Note>(\_boxName);

return box; }

// get full note

Future<List<Note>> getFullNote() async {

final box = await noteBox();

List<Note> notes = box.values.toList();

return notes; }

// to add data in box

Future<void> addToBox(Note note) async {

final box = await noteBox();

await box.add(note); }

// delete data from box

Future<void> deleteFromBox(int index) async {

final box = await noteBox();

await box.deleteAt(index); }

// delete all data from box

Future<void> deleteAll() async {

final box = await noteBox();

await box.clear(); }

// update data

Future<void> updateNote(int index, Note note) async {

final box = await noteBox();

await box.putAt(index, note); }

}

//FLUTTER-SCREENS/EDIT NOTE SCREEN (1)

import 'package:flutter/material.dart';

import 'package:flutter\_appl/bloc/note\_bloc.dart';

import 'package:flutter\_appl/bloc/note\_event.dart';

import 'package:flutter\_appl/models/note\_model.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

class EditNoteScreen extends StatefulWidget {

final bool newNote;

final Note note;

final int index;

EditNoteScreen({this.note, this.index, @required this.newNote});

@override

\_EditNoteScreenState createState() => \_EditNoteScreenState();

}

class \_EditNoteScreenState extends State<EditNoteScreen> {

@override

Widget build(BuildContext context) {

String title = widget.note != null ? widget.note.title : "";

String content = widget.note != null ? widget.note.content : "";

final \_formKey = GlobalKey<FormState>();

final \_titleController = TextEditingController(text: title);

final \_contentController = TextEditingController(text: content);

Future<bool> \_onWillPop() async {

if (\_titleController.text == title ||

\_contentController.text == content) {

return showDialog(

context: context,

builder:(BuildContext context) => AlertDialog(

title: new Text('Are you sure?'),

content: new Text('Unsaved data will be lost.'),

//FLUTTER-SCREENS/EDIT NOTE SCREEN (2)

actions: <Widget>[

new ElevatedButton(

style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xFFFFFFFF))),

onPressed: () => Navigator.of(context).pop(false),

child: new Text('No',style: TextStyle(color: Colors.black),),

),

new ElevatedButton(

style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xFFFFFFFF))),

onPressed: () => Navigator.of(context).pop(true),

child: new Text('Yes',style: TextStyle(color: Colors.black),),

), ],),) ?? false;

}

else {

return true;

}

}

return WillPopScope(

onWillPop: () => \_onWillPop(),

child: Container(

decoration: BoxDecoration(

gradient: LinearGradient(

begin: Alignment.topLeft,

end: Alignment.bottomRight,

colors: [Colors.purple, Colors.orange])),

child: Scaffold(

backgroundColor: Colors.transparent,

appBar: AppBar(

title: Text(

widget.newNote ? 'New Note' : 'Edit Note',

style: TextStyle(color: Colors.white),

),

leading: IconButton(icon: Icon(Icons.arrow\_back, color: Colors.white,),

onPressed: (){

Navigator.pop(context);

//FLUTTER-SCREENS/EDIT NOTE SCREEN (3)

},),

),

floatingActionButton: Padding(

padding: EdgeInsets.only(left: 10, bottom: 10),

child: FloatingActionButton.extended(

backgroundColor: Color(0xFF560bad),

icon: Icon(Icons.add),

label: Text(widget.newNote ? 'ADD' : 'UPDATE'),

onPressed: () {

if (\_formKey.currentState.validate()) {

widget.newNote ? BlocProvider.of<NoteBloc>(context).add(NoteAddEvent(

title: \_titleController.text,

content: \_contentController.text))

: BlocProvider.of<NoteBloc>(context).add(NoteEditEvent(

title: \_titleController.text,

content: \_contentController.text,

index: widget.index));

Navigator.pop(context);

}

},

)),

body: SingleChildScrollView(

child: Form(

key: \_formKey,

child: Padding(

padding: EdgeInsets.only(left: 10, right: 10),

child: Column(

children: [

SizedBox(

height: 20,

),

NoteTitle(titleController: \_titleController),

SizedBox(

height: 20,

//FLUTTER-SCREENS/EDIT NOTE SCREEN (4)

),

NoteBody(contentController: \_contentController,)

], ), ), ), ), ), ), );

}

}

class NoteTitle extends StatelessWidget {

final TextEditingController \_titleController;

const NoteTitle({

Key key,

@required TextEditingController titleController,

}) : \_titleController = titleController,

super(key: key);

@override

Widget build(BuildContext context) {

return Container(

child: TextFormField(

validator: (value) {

if (value.isEmpty) {

return 'Please enter the note title';

}

return null;

},

controller: \_titleController,

style: TextStyle(color: Color(0xFFfdffb6)),

decoration: InputDecoration(

labelStyle: TextStyle(color: Colors.black38),

border: OutlineInputBorder(),

labelText: 'Note Title',

), ), );

}

}

//FLUTTER-SCREENS/EDIT NOTE SCREEN (5)

class NoteBody extends StatelessWidget {

const NoteBody({

Key key,

@required TextEditingController contentController,

}) : \_contentController = contentController,

super(key: key);

final TextEditingController \_contentController;

@override

Widget build(BuildContext context) {

return Padding(

padding: EdgeInsets.only(left: 10, right: 10, bottom: 20),

child: Container(

child: TextFormField(

validator: (value) {

if (value.isEmpty) {

return 'Please enter the note body!';

}

return null;

},

maxLines: 50,

controller: \_contentController,

style: TextStyle(color: Color(0xFFfdffb6)),

decoration: InputDecoration(

border: OutlineInputBorder(),

),

),

),

);

}

}

//FLUTTER-SCREENS/NOTES SCREEN (1)

import 'package:flutter/material.dart';

import 'package:flutter\_appl/bloc/note\_bloc.dart';

import 'package:flutter\_appl/bloc/note\_state.dart';

import 'package:flutter\_appl/widgets/alert\_dialog\_refactor.dart';

import 'package:flutter\_appl/widgets/note\_card.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

import 'edit\_note.dart';

class NotesScreen extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Container(

decoration: BoxDecoration(

gradient: LinearGradient(

begin: Alignment.topLeft,

end: Alignment.bottomRight,

colors: [Colors.red, Colors.white])),

child: Scaffold(

floatingActionButton: FloatingActionButton(

backgroundColor: Color(0xFF000000),

onPressed: () {

Navigator.push(context,

MaterialPageRoute(builder: (BuildContext context) {

return EditNoteScreen(

newNote: true,

);

}));

},

child: Icon(Icons.add),

),

backgroundColor: Colors.transparent,

//FLUTTER-SCREENS/NOTES SCREEN (2)

appBar: AppBar(

elevation: 0,

backgroundColor: Colors.transparent,

title: Text(

'Your Notes',

style: TextStyle(color: Colors.white),

),

actions: [

IconButton(

icon: Icon(

Icons.settings,

color: Colors.white,

),

onPressed: () {},

) ], ),

body: SingleChildScrollView(

child: Column(

children: [

// define all state and rebuild ui

BlocBuilder<NoteBloc, NoteState>(

builder: (context, state) {

if (state is NoteInitial) {

return Container();

}

if (state is YourNotesState) {

return NoteGrid(state: state);

}

if (state is NotesLoading) {

return Container();

} else {

return Container();

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

}

//FLUTTER-SCREENS/NOTES SCREEN (3)

class NoteGrid extends StatelessWidget {

final YourNotesState state;

NoteGrid({this.state});

@override

Widget build(BuildContext context) {

return GridView.builder(

shrinkWrap: true,

physics: BouncingScrollPhysics(),

itemCount: state.notes.length,

gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(

crossAxisCount: 2,

childAspectRatio: 0.72,

mainAxisSpacing: 10,

crossAxisSpacing: 10,

),

itemBuilder: (context, index) {

final note = state.notes[index];

return InkWell(

onTap: () {

Navigator.push(context,

MaterialPageRoute(builder: (BuildContext context) {

return EditNoteScreen(

note: note,

index: index,

newNote: false,

); })); },

onLongPress: () {

showDialog(

context: context,

builder: (BuildContext context) =>

AlertDialogRefactor(index: index));

},

child: NoteCard(

title: note.title, content: note.content, ), ); }); }}

//FLUTTER-WIDGETS/ALERT DIOLOG REFACTOR

import 'package:flutter/material.dart';

import 'package:flutter\_appl/bloc/note\_bloc.dart';

import 'package:flutter\_appl/bloc/note\_event.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

class AlertDialogRefactor extends StatelessWidget {

final int index;

const AlertDialogRefactor({this.index});

@override

Widget build(BuildContext context) {

return AlertDialog(

title: Text(

'Delete?',

style: TextStyle(color: Color(0xFF49565e)), ),

actions: [

ElevatedButton(

style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xFFFFFFFF))),

child: Text(

'YES',

style: TextStyle(color: Colors.redAccent), ),

onPressed: () {

BlocProvider.of<NoteBloc>(context)

.add(NoteDeleteEvent(index: index));

Navigator.pop(context); }),

ElevatedButton(

style: ButtonStyle(backgroundColor: MaterialStateProperty.all(Color(0xFFFFFFFF))),

onPressed: () {

Navigator.pop(context); },

child: Text(

'CANCEL',

style: TextStyle(color: Color(0xFF49565e)), )) ], );}}

//FLUTTER-WIDGETS/NOTE CARD (1)

import 'package:flutter/material.dart';

class NoteCard extends StatelessWidget {

final String title;

final String content;

const NoteCard({this.title, this.content});

@override

Widget build(BuildContext context) {

final Color black = Color(0xFFfdffb6);

return Padding(

padding: EdgeInsets.all(5),

child: Card(

child: Padding(

padding: EdgeInsets.only(top: 20, bottom: 10, left: 5, right: 5),

child: Column(

children: [

Text(

'$title',

style: TextStyle(

color: black,

letterSpacing: 1,

fontSize: 20,

fontWeight: FontWeight.w400),

),

SizedBox(

height: 10,

),

Container(

height: .5,

width: double.infinity,

color: black,

),

//FLUTTER-WIDGETS/NOTE CARD (2)

SizedBox(

height: 10,

),

Expanded(

child: Text(

'$content',

style: TextStyle(

color: black,

fontSize: 14,

wordSpacing: 1.25,

letterSpacing: 1),

),

)

],

),

),

),

);

}

}

//FLUTTER-SCREENS/MAIN PAGE

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

import 'package:hive/hive.dart';

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

import 'bloc/note\_bloc.dart';

import 'bloc/note\_event.dart';

import 'models/note\_model.dart';

import 'note\_database/note\_database.dart';

import 'screens/notes\_screen.dart';

void main() async {

await Hive.initFlutter();

Hive.registerAdapter<Note>(NoteAdapter());

await Hive.openBox<Note>("Note");

runApp(BlocProvider(

create: (context) => NoteBloc(NoteDatabase()),

child: MyApp(), ));

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

BlocProvider.of<NoteBloc>(context).add(NoteInitialEvent());

return MaterialApp(

debugShowCheckedModeBanner: false,

title: 'Flutter Demo',

theme: ThemeData(

//backgroundColor: Color(0xFFf0f5f9),

cardColor: Color(0xFFc9d6df),

primaryColor: Color(0xFF83c5be),

//accentColor: Color(0xFF52616b),

),

home: NotesScreen(),

); }}

//FLUTTER/PUBSPEC.YAML

|  |
| --- |
| dependencies: |
|  | flutter: |
|  | sdk: flutter |
|  | cupertino\_icons: ^1.0.2 |
|  | # LOCAL DATABASE |
|  | hive: ^2.0.4 |
|  | hive\_flutter: ^1.0.0 |
|  | # STATE MANAGMENT |
|  | flutter\_bloc: ^6.1.3 |
|  | equatable: ^2.0.0 |
|  |  |
|  | dev\_dependencies: |
|  | flutter\_test: |
|  | sdk: flutter |
|  | # GENERATOR |
|  | hive\_generator: ^1.1.0 |
|  | build\_runner: ^2.0.1 |