

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №3**

**з дисципліни «Розробка ПЗ для мобільних пристроїв»**

**на тему:**

**«Створення додатків в Material Design.»**

Виконав:

Студент гр. ПЗ2011

Кулик С.В.

Прийняв:

Іванов О.П.

Дніпро, 2023

**Тема.** Створення додатків в Material Design.

**Мета.** Реалізація мобільного інтерфейсу Material Design у Flutter, використання вікон зі станом, додаткові пакети, робота із файлами.

**Завдання**

Мобільний додаток повинен реалізувати функції роботи зі списком елементів необхідного виду, має дозволити перегляд списку, додавання, редагування та видалення елементів.

Види елементів:

* Пам'ятка (Memo) – має заголовок, розділ, пояснення.
* Справа (Todo) – має заголовок, розділ, пояснення, відмітка «Виконано».
* Подія (Event) – має заголовок, розділ, пояснення та дату події.

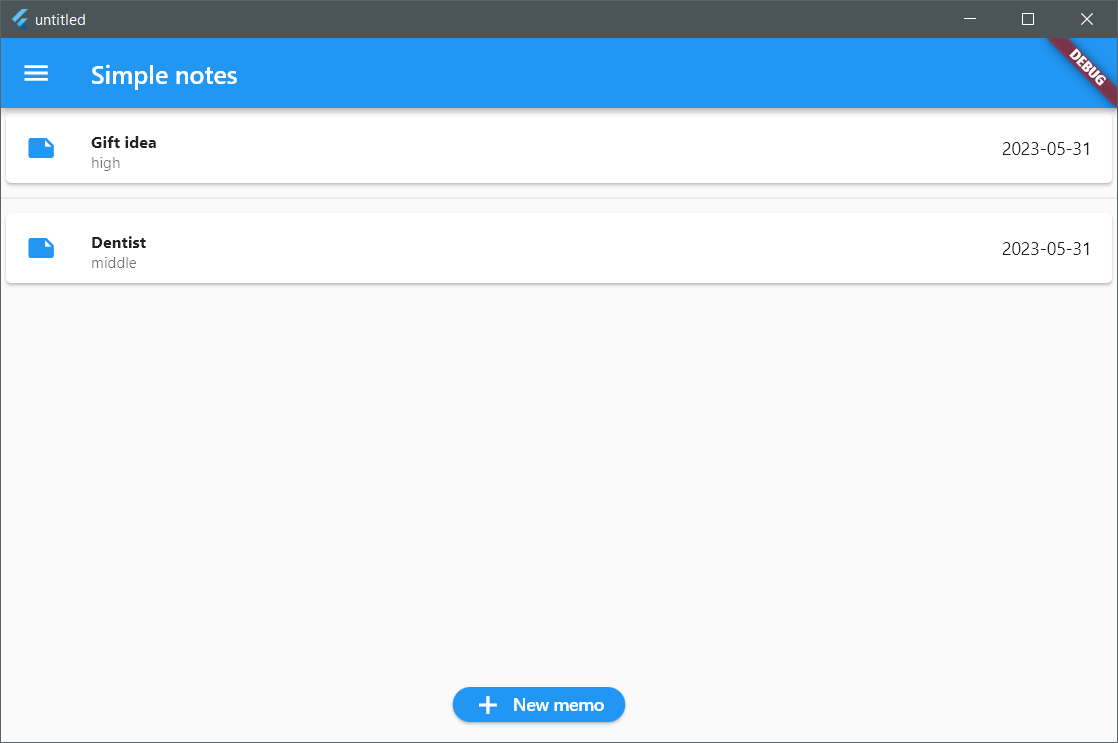
Додаткові можливості – реалізувати обробку списку елементів, типу фільтрація по полю «Розділ»/ «Категорія», сортування по вказаному полю або пошук елементів по тексту в будь якому полі.

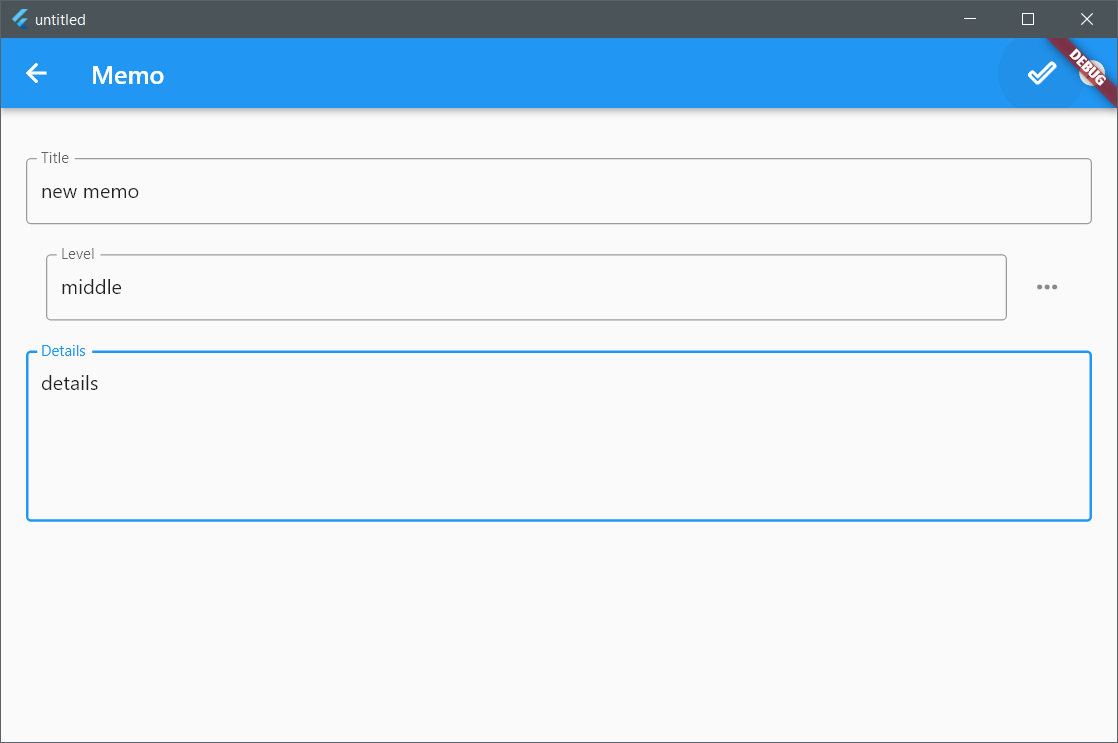
Збереження даних – всі данні мають зберігатися та відновлюватися при наступному запуску, зберігати в текстовий файл заданого формату. \* - Реалізація без збереження в файл на зменшену оцінку, для демонстрації в DartPad.

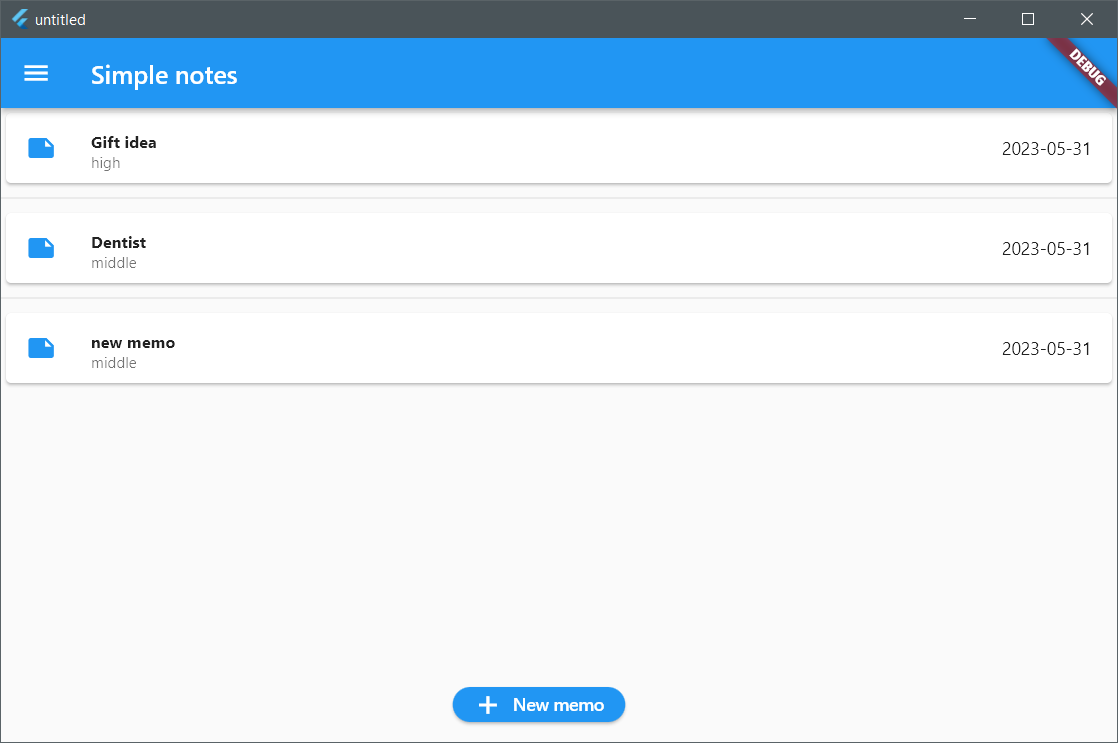
**Текст програми**

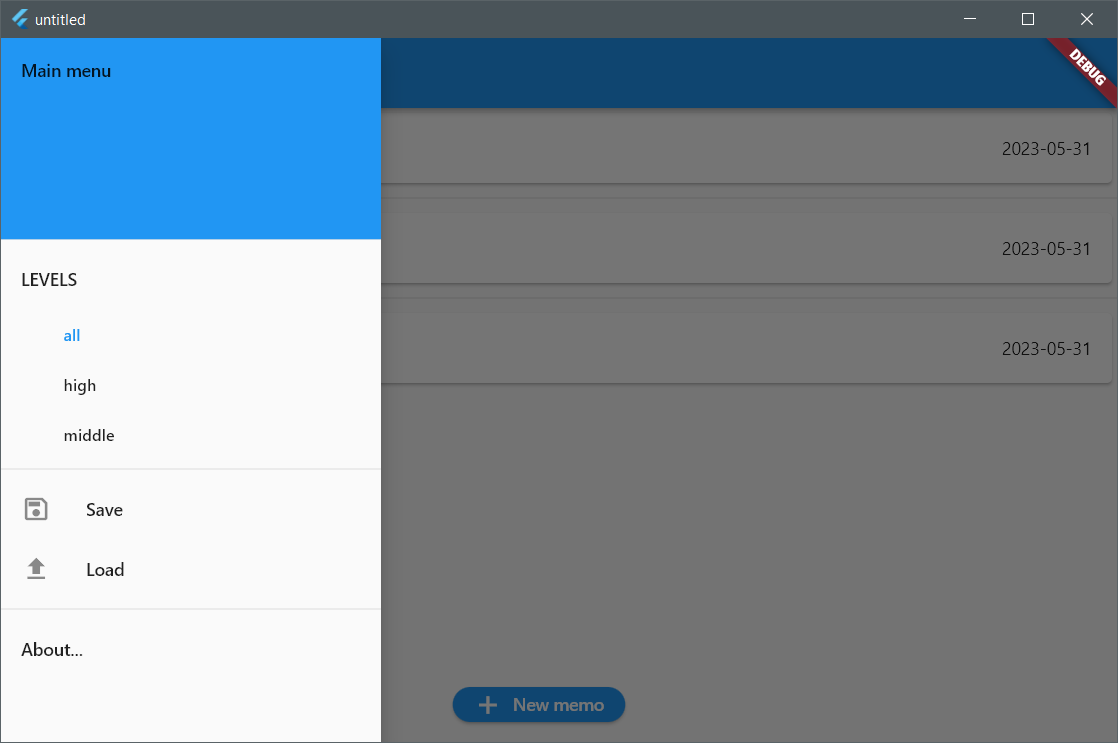
import 'dart:io';  
import 'package:flutter/material.dart';  
import 'package:path\_provider/path\_provider.dart';  
import 'package:xml/xml.dart';  
  
///////// Data classes  
abstract class Note {  
 String title;  
 String level;  
 DateTime dateCreate = DateTime.now();  
 DateTime dateModify = DateTime.now();  
  
 Note({this.title = '', this.level = 'low'});  
  
 update(other) {  
 this  
 ..title = other.title  
 ..level = other.level  
 ..dateModify = DateTime.now();  
 }  
  
 toXml(builder);  
  
 fromXml() {}  
}  
  
class Memo extends Note {  
 String details = "";  
  
 Memo({this.details = '', title = '', level = 'low'}) : super(title: title, level: level);  
  
 @override  
 update(other) {  
 super.update(other);  
 this.details = other.details;  
 }  
  
 @override  
 toString() => "memo||$title||$level||$details";  
  
 @override  
 toXml(builder) {  
 builder.element('Memo', nest: () {  
 builder.element('title', nest: title);  
 builder.element('level', nest: level);  
 builder.element('details', nest: details);  
 });  
 }  
}  
  
mixin Check {  
 bool done = false;  
}  
  
mixin Date {  
 String date = DateTime.now().toString();  
}  
  
//Container  
class Notes {  
 List<Note> items = [];  
 var \_curLevel = "all";  
  
 Notes() {  
 items.add(Memo(title: 'Gift idea', level: 'high', details: 'car, flowers, house'));  
 items.add(Memo(title: 'Dentist', level: 'middle', details: 'Jack Martin, good dentist, phone number 3858394'));  
 }  
  
 get count => items.where(\_filter).length;  
  
 get item => (int i) => items.where(\_filter).toList()[i];  
  
 get levels => items.map((x) => x.level).toSet().toList();  
  
 get filter => \_curLevel == "" ? "all" : \_curLevel;  
  
 setFilter(lev) {  
 \_curLevel = lev;  
 }  
  
 bool \_filter(x) {  
 if (\_curLevel == 'all') {  
 return true;  
 } else {  
 return \_curLevel == x.level;  
 }  
 }  
  
 load() async {  
 final directory = await getApplicationDocumentsDirectory();  
 print("Load from $directory");  
 final file = File(directory.path + "/notes.xml");  
 final document = XmlDocument.parse(file.readAsStringSync());  
 items.clear();  
 for (var note in document.findElements('Notes').first.childElements) {  
 var param = note.children.map((event) => event.text).toList();  
 if (note.localName == 'Memo') {  
 items.add(Memo(title: param[0], level: param[1], details: param[2]));  
 }  
 }  
 }  
  
 save() async {  
 try {  
 final directory = await getApplicationDocumentsDirectory();  
 print(directory.path);  
 final file = File(directory.path + "/notes.xml");  
 file.openWrite();  
 final builder = XmlBuilder();  
 builder.processing('xml', 'version="1.0"');  
 builder.element('Notes', nest: () {  
 for (var x in items) {  
 x.toXml(builder);  
 }  
 });  
  
 final document = builder.buildDocument();  
 file.writeAsStringSync(document.toString());  
 } catch (e) {  
 print("Error while saving the file: " + e.toString());  
 }  
 }  
  
 remove(item) {  
 items.remove(item);  
 }  
  
 add<T extends Note>() {  
 final factories = <Type, Function>{Memo: () => Memo()};  
 final one = (factories[T]!)();  
 items.add(one);  
 return one;  
 }  
}  
  
////////////////////// Widgets  
class ItemPage extends StatefulWidget {  
 final \_item, \_levels;  
  
 const ItemPage(this.\_item, this.\_levels);  
  
 @override  
 \_ItemPageState createState() => \_ItemPageState(\_item, \_levels);  
}  
  
class \_ItemPageState extends State<ItemPage> {  
 final Memo \_item;  
 var \_levels;  
 bool \_check = false;  
 var date = DateTime.now().toString();  
 final cntTitle = TextEditingController();  
 final cntLevel = TextEditingController();  
 final cntDetails = TextEditingController();  
 final cntDate = TextEditingController();  
  
 \_ItemPageState(this.\_item, this.\_levels) {  
 cntTitle.text = \_item.title;  
 cntLevel.text = \_item.level;  
 cntDetails.text = \_item.details;  
 }  
  
 @override  
 void dispose() {  
 // Clean up the controller when the Widget is disposed  
 cntTitle.dispose();  
 super.dispose();  
 }  
  
 @override  
 void initState() {  
 super.initState();  
 }  
  
 showDelDialog(context) {  
 // set up the AlertDialog  
 AlertDialog alert = AlertDialog(  
 title: Text("Confirm"),  
 content: Text("Would you like to remove item?"),  
 actions: [  
 TextButton(  
 child: Text("Cancel"),  
 onPressed: () {  
 Navigator.*pop*(  
 context,  
 );  
 },  
 ),  
 TextButton(  
 child: Text("Yes"),  
 onPressed: () {  
 Navigator.*pop*(  
 context,  
 );  
 Navigator.*pop*(context, "del");  
 },  
 )  
 ],  
 );  
 // show the dialog  
 showDialog(  
 context: context,  
 builder: (BuildContext context) {  
 return alert;  
 },  
 );  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text('${\_item.runtimeType}'),  
 actions: [  
 new IconButton(  
 icon: new Icon(Icons.*done\_outline\_rounded*),  
 onPressed: () {  
 Navigator.*pop*(context, \_upd());  
 }),  
 new IconButton(icon: new Icon(Icons.*cancel*), onPressed: () => showDelDialog(context))  
 ],  
 ),  
 body: SingleChildScrollView(  
 padding: EdgeInsets.all(20),  
 child: Column(  
 children: [  
 SizedBox(height: 20),  
 TextFormField(  
 controller: cntTitle,  
 decoration: InputDecoration(  
 labelText: 'Title',  
 border: OutlineInputBorder(),  
 ),  
 ),  
 SizedBox(height: 20),  
 ListTile(  
 trailing: PopupMenuButton(  
 icon: const Icon(Icons.*more\_horiz\_rounded*),  
 onSelected: (String newValue) {  
 setState(() {  
 cntLevel.text = newValue;  
 });  
 },  
 itemBuilder: (context) => [  
 for (String x in \_levels)  
 PopupMenuItem(  
 value: x,  
 child: Text(x),  
 )  
 ],  
 ),  
 title: TextFormField(  
 controller: cntLevel,  
 decoration: InputDecoration(  
 labelText: 'Level',  
 border: OutlineInputBorder(),  
 ),  
 ),  
 ),  
 SizedBox(height: 20),  
 TextFormField(  
 controller: cntDetails,  
 decoration: InputDecoration(  
 labelText: 'Details',  
 border: OutlineInputBorder(),  
 ),  
 minLines: 5,  
 maxLines: 10,  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
  
 \_upd() {  
 if (\_item is Memo) return Memo(details: cntDetails.text, title: cntTitle.text, level: cntLevel.text);  
 }  
}  
  
class MyApp extends StatelessWidget {  
 // This widget is the root of your application.  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 title: 'Notes',  
 theme: ThemeData(  
 primarySwatch: Colors.*blue*,  
 ),  
 home: HomePage(title: 'Simple notes'),  
 );  
 }  
}  
  
class HomePage extends StatefulWidget {  
 HomePage({Key? key, this.title = ""}) : super(key: key);  
 final String title;  
  
 @override  
 \_HomePageState createState() => \_HomePageState();  
}  
  
class \_HomePageState extends State<HomePage> {  
 var notes = Notes();  
  
 showItem(item) async {  
 final result = await Navigator.*push*(  
 context,  
 MaterialPageRoute(  
 builder: (BuildContext context) => ItemPage(item, notes.levels),  
 ));  
 if (result != null) {  
 setState(() {  
 if (result == "del") {  
 notes.remove(item);  
 } else {  
 item.update(result);  
 }  
 });  
 }  
 }  
  
 listItem(index, context) {  
 final item = notes.item(index);  
  
 return Card(  
 elevation: 2,  
 child: ListTile(  
 leading: \_buildLeadingIcon(item),  
 title: Text(  
 item.title,  
 style: TextStyle(  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 subtitle: Text(item.level),  
 trailing: Text(  
 item.dateCreate.toString().substring(0, 10),  
 ),  
 dense: true,  
 onTap: () => showItem(item),  
 ),  
 );  
 }  
  
 Widget \_buildLeadingIcon(Note item) {  
 if (item is Memo) {  
 return Icon(  
 Icons.*note*,  
 color: Colors.*blue*,  
 );  
 }  
 return Container();  
 }  
  
 \_HomePageState();  
  
 @override  
 void initState() {  
 super.initState();  
 //(notes.load()).whenComplete(() => setState(() {}));  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text(widget.title),  
 ),  
 body: ListView.separated(  
 separatorBuilder: (context, index) => const Divider(),  
 itemCount: notes.count,  
 itemBuilder: (context, i) => listItem(i, context),  
 ),  
 drawer: Drawer(  
 child: ListView(  
 children: [  
 DrawerHeader(  
 child: Text('Main menu'),  
 decoration: BoxDecoration(  
 color: Colors.*blue*,  
 ),  
 ),  
 ListTile(title: Text('LEVELS')),  
 for (var x in ["all", ...notes.levels])  
 ListTile(  
 title: Text(x),  
 contentPadding: EdgeInsets.only(left: 50),  
 dense: true,  
 selected: notes.filter == x,  
 onTap: () => setState(() {  
 Navigator.*pop*(context);  
 notes.setFilter(x);  
 }),  
 ),  
 Divider(),  
 ListTile(  
 title: Text('Save'),  
 leading: Icon(Icons.*save\_outlined*),  
 onTap: () {  
 notes.save();  
 ScaffoldMessenger.*of*(context).showSnackBar(  
 SnackBar(  
 content: const Text('Saved.'),  
 ),  
 );  
 ;  
 }),  
 ListTile(  
 title: Text('Load'),  
 leading: Icon(Icons.*file\_upload*),  
 onTap: () async {  
 await notes.load();  
 setState(() {});  
 }),  
 Divider(),  
 ListTile(title: Text('About...')),  
 ],  
 )),  
 floatingActionButton: Column(mainAxisAlignment: MainAxisAlignment.end, children: [  
 Row(mainAxisAlignment: MainAxisAlignment.spaceAround, children: [  
 ElevatedButton(  
 onPressed: () => showItem(notes.add<Memo>()),  
 style: ElevatedButton.*styleFrom*(  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.all(Radius.circular(15.0)),  
 ),  
 ),  
 child: Row(  
 mainAxisSize: MainAxisSize.min,  
 children: [  
 Icon(Icons.*add*),  
 SizedBox(width: 8.0),  
 Text('New memo'),  
 ],  
 ),  
 ),  
 ])  
 ]));  
 }  
}  
  
///////////////////////  
void main() {  
 runApp(MyApp());  
}

**Результати роботи програми**









Можна побачити що програма працює коректно, та виконує поставлені завдання.

**Висновок**

У результаті роботи було створено мобільний додаток з інтуїтивним і красивим інтерфейсом, що відповідає принципам Material Design.