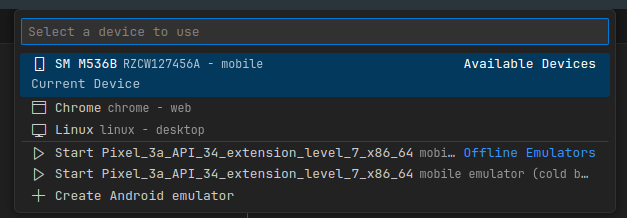
**Лабораторна робота № 5**

*Тема:* Створення мобільного додатку засобами Flutter

**Хід роботи**

Для компіляції додатку під мобільний пристрій потрібно вибрати відповідний пристрій в VS Code:



Також можна вибрати один з емуляторів з списку, якщо вони налаштовані в Android Studio.

В цьому проекті будемо розробляти гру “Хрестики-нулики”. Заповнимо файл **main.dart** наступним кодом:

import 'package:flutter/material.dart';

import 'package:mobile\_app/presentation/pages/game\_page.dart';

void main() {

runApp(const MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({super.key});

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

title: 'Tic tac toe',

theme: ThemeData(

colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),

useMaterial3: true,

),

home: const GamePage(),

);

}

}

Тут в якості головної сторінки використовується сторінка **GamePage.** Напишемо відповідний файл **game\_page.dart** для цієї сторінки. Заповнимо його наступним чином:

// ignore\_for\_file: library\_private\_types\_in\_public\_api

import 'package:flutter/material.dart';

import 'package:mobile\_app/presentation/widgets/circle\_image.dart';

import '../widgets/cross\_image.dart';

class GamePage extends StatefulWidget {

const GamePage({super.key});

@override

\_GamePageState createState() => \_GamePageState();

}

class \_GamePageState extends State<GamePage> {

List<List<Widget>> board =

List.generate(3, (\_) => List.filled(3, Container()));

bool isCircleMove = false;

@override

Widget build(BuildContext context) {

return Scaffold(

body: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

isCircleMove ? "Circle's turn" : "Cross's turn",

style: const TextStyle(fontSize: 20.0),

),

const SizedBox(height: 20.0),

Column(

children: List.generate(3, (row) {

return Row(

mainAxisAlignment: MainAxisAlignment.center,

children: List.generate(3, (col) {

return InkWell(

onTap: () {

\_onTileTap(row, col);

},

child: Container(

width: 50.0,

height: 50.0,

decoration: BoxDecoration(

border: Border.all(),

),

child: Center(

child: board[row][col],

),

),

);

}),

);

}),

),

const SizedBox(height: 20.0),

ElevatedButton(

onPressed: \_resetGame,

child: const Text('Reset Game'),

),

],

),

);

}

void \_onTileTap(int row, int col) {

if (board[row][col] is Container) {

setState(() {

board[row][col] =

isCircleMove ? const CircleImage() : const CrossImage();

isCircleMove = !isCircleMove;

\_checkForWinner();

});

}

}

void \_checkForWinner() {

for (int i = 0; i < 3; i++) {

if (board[i][0] == board[i][1] &&

board[i][1] == board[i][2] &&

board[i][0] is! Container) {

\_showWinnerDialog(board[i][0]);

return;

}

}

for (int i = 0; i < 3; i++) {

if (board[0][i] == board[1][i] &&

board[1][i] == board[2][i] &&

board[0][i] is! Container) {

\_showWinnerDialog(board[0][i]);

return;

}

}

if (board[0][0] == board[1][1] &&

board[1][1] == board[2][2] &&

board[0][0] is! Container) {

\_showWinnerDialog(board[0][0]);

return;

}

if (board[0][2] == board[1][1] &&

board[1][1] == board[2][0] &&

board[0][2] is! Container) {

\_showWinnerDialog(board[0][2]);

return;

}

if (!board.any((row) => row.any((cell) => cell is Container))) {

\_showDrawDialog();

}

}

void \_resetGame() {

setState(() {

board = List.generate(3, (\_) => List.filled(3, Container()));

isCircleMove = false;

});

}

void \_showWinnerDialog(Widget winner) {

showDialog(

context: context,

builder: (BuildContext context) {

return AlertDialog(

title: const Text('Game Over'),

content: Text('${winner is CircleImage ? "Circles" : "Cross"} wins!'),

actions: [

TextButton(

onPressed: () {

Navigator.of(context).pop();

\_resetGame();

},

child: const Text('Play Again'),

),

],

);

},

);

}

void \_showDrawDialog() {

showDialog(

context: context,

builder: (BuildContext context) {

return AlertDialog(

title: const Text('Game Over'),

content: const Text('It\'s a draw!'),

actions: [

TextButton(

onPressed: () {

Navigator.of(context).pop();

\_resetGame();

},

child: const Text('Play Again'),

),

],

);

},

);

}

}

В класі стану цієї сторінки створено два поля, **board** i **isCircleMove**. **Board** являє собою двовимірний список віджетів, а саме віджетів зображень нулика, хрестика або віджета **Container,** який являє пусте місце на ігровому полі. При ініціалізації, поле заповнюється трьома списками, які в свою чергу заповнюються пустими контейнерами.

**IsCircleMove** є змінною стану, тобто визначає, чи наступний хід зробить нулик, якщо true або хрестик, якщо false.

Як можна помітити, більшість коду сторінки займає логіка, тобто методи-обробники, які реагують на клацання кнопочки, показують діалоги і перевіряють наявність переможця.  
Коротко опишу кожен з них і його призначення:

1. void \_onTileTap(int row, int col) – метод, який обробляє клік на клітинку ігрового поля. Приймає координати клітинки як параметри. Перевіряє клітинку на пустоту, заповняє якщо пуста і перевіряє чи хтось переміг.
2. void \_checkForWinner() - перевіряє список board на наявність переможця. Циклами і умовами проходиться по всіх клітинках і шукає чи є “пряма лінія”.
3. void \_resetGame() - перезапускає гру. Очищає поле від хрестиків і нуликів і задає початкового гравця.
4. void \_showWinnerDialog(Widget winner) – показує діалогове вікно, з повідомленням про переможця. Приймає віджет хрестика чи нулика як параметр і в залежності від цього виводить відповідне повідомлення про перемогу.
5. void \_showDrawDialog() - показує діалогове вікно, з повідомленням про нічию.

Також, на цій сторінці можна помітити користувацькі віджети, такі як **CircleImage** i **CrossImage**. Ці віджети являють собою зображення нулика чи хрестика відповідно.

Їх код такий:

import 'package:flutter/material.dart';

class CrossImage extends StatelessWidget {

const CrossImage({super.key});

@override

Widget build(BuildContext context) {

return SizedBox(

width: 200,

height: 200,

child: Image.asset('assets/cross.png'),

);

}

}

import 'package:flutter/material.dart';

class CircleImage extends StatelessWidget {

const CircleImage({super.key});

@override

Widget build(BuildContext context) {

return SizedBox(

width: 200,

height: 200,

child: Image.asset('assets/circle.png'),

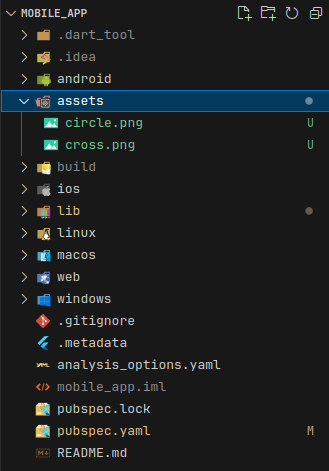
);

}

}

Їхня реалізація просто використовує зображення з списку **assets** на проекті і обмежує його в розмірах 200x200.

Для того, аби ці зображженя були доступними для їх виведення, спочатку потрібно додати файли зображень в проект. Рекомендується розміщувати їх на такому шляху:



Після того, як додали файли в папку, потрібно визначити їх в списку assets, про який було сказано раніше. Для цього, перейдіть до файлу **pubspec.yaml**, та в ключі **flutter** додайте наступне:

assets:

- assets/cross.png

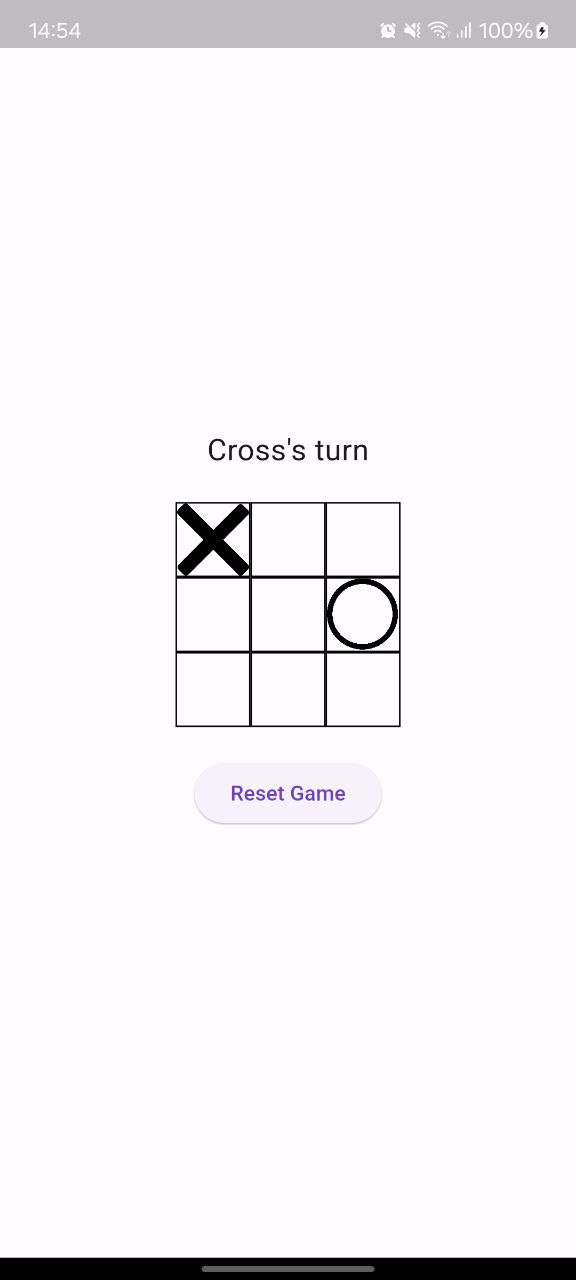
- assets/circle.png

Весь **pubspec.yaml** повинен виглядати приблизно так:



Після того, як все налаштували та підключили, можна компілювати проект.

Якщо все добре зкомпілюється і не буде ніяких помилок, додаток буде виглядати так:



**Завдання**

Створити додаток на реакт згідно варіанту, в якому необхідно обов’язково використати props, state, створити мінімум дві компоненти, що мають бути вкладеними.

* 1. Список завдань. Створення завдань, перегляд створених, видалення.
  2. Форма реєстрації. Поля на формі різних типів даних, створити варефікацію даних, перевірку на правильність введених даних.
  3. Програма тягни білет. Генерується N кількість білетів, при натиснені на кнопку, випадковим чином вибирається номер білету, але можна один білет вибрати лише один раз.
  4. Бики-корови. Програма генерує випадкове 4х значне число, користувач має відгадати те число. Користувач вводить число, і якщо є цифри, які входять в загадане число, але стоять не на відповідних місцях позначаються як корови, а ті що на місцях – бики. Власне за допомогою таких підсказок користувач при наступних спробах може відгадати число.(Наприклад, загадано число 7481, а користувач ввів – 0784, то повідомлення буде 1-бик і 2 корови, так як 8 на відповідному місці, а 7 і 4 є і числі, але на інших позиціях)
  5. Таймер. Задається час, і при запуску таймера, по секунді зменшується час до 0.
  6. Хрестики нулики. Створюється поле 3 на 3, і користувач може ставити хрестик або нулик відповідно до ходу.
  7. Електронне табло. Можливість вивести інформацію у вигляді електронного таблоа.
  8. Калькулятор. Реалізувати арифметичні операції.
  9. Телефона книга. Можливість додавати контакти до книги, переглядати усі і тд.
  10. Табло для спортивних змагань. Зазначаються назви команд, та є можливість змінювати рахунок.

**Література**