Сучасні технології мобільного програмування

Використання Flutter для кросплатформної розробки мобільних застосунків

Слайди до лекцій (6 змістовий модуль)

Що таке Flutter?

- Open-source SDK для розробки застосунків від Google (версія 1.0 вийшла у 2018 р).
- Дозволяє створювати Android, iOS, Web та Desktop (Windows, MacOS, Linux) застосунки з єдиного кодової бази.
- Мова програмування: Dart.





Google

Переваги Flutter

- Швидкий розробницький цикл (‘Hot Reload’).
- Висока продуктивність.
- Безшовна крос-платформність.
- Широкий вибір готових віджетів.





Ключові компоненти Flutter

- **Widgets:** Основа взаємодії Flutter.
- **Dart:** Основна мова Flutter.
- **Flutter Engine:** Обробляє графіку та анімацію.
- **Foundation Library:** Базові компоненти для застосунків.





Встановлення Flutter SDK

- Завантажити Flutter SDK з офіційного сайту.
- Розпакувати в будь-яке місце (рекомендується %USERPROFILE%\develop (C:\Users\{username}\develop)).
- Додати Flutter до PATH ((C:\Users\{username}\develop\bin)).
- Запустити утиліту flutter doctor.

```
C:\Windows\System32>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.50.0, on Microsoft Windows [Version 10.0.26300.9974], locale uk-UA)
[√] Windows Version (Windows 11 or higher, 26H2, 2009)
[√] Android toolchain - develop for Android devices (Android SDK version 37.0.0)
[√] Chrome - develop for the web
[√] Visual Studio - develop Windows apps (Visual Studio Build Tools 2026 18.9.1)
[√] Connected device (3 available)
[√] Network resources
C:\Windows\System32>
```

https://docs.flutter.dev/install/manual

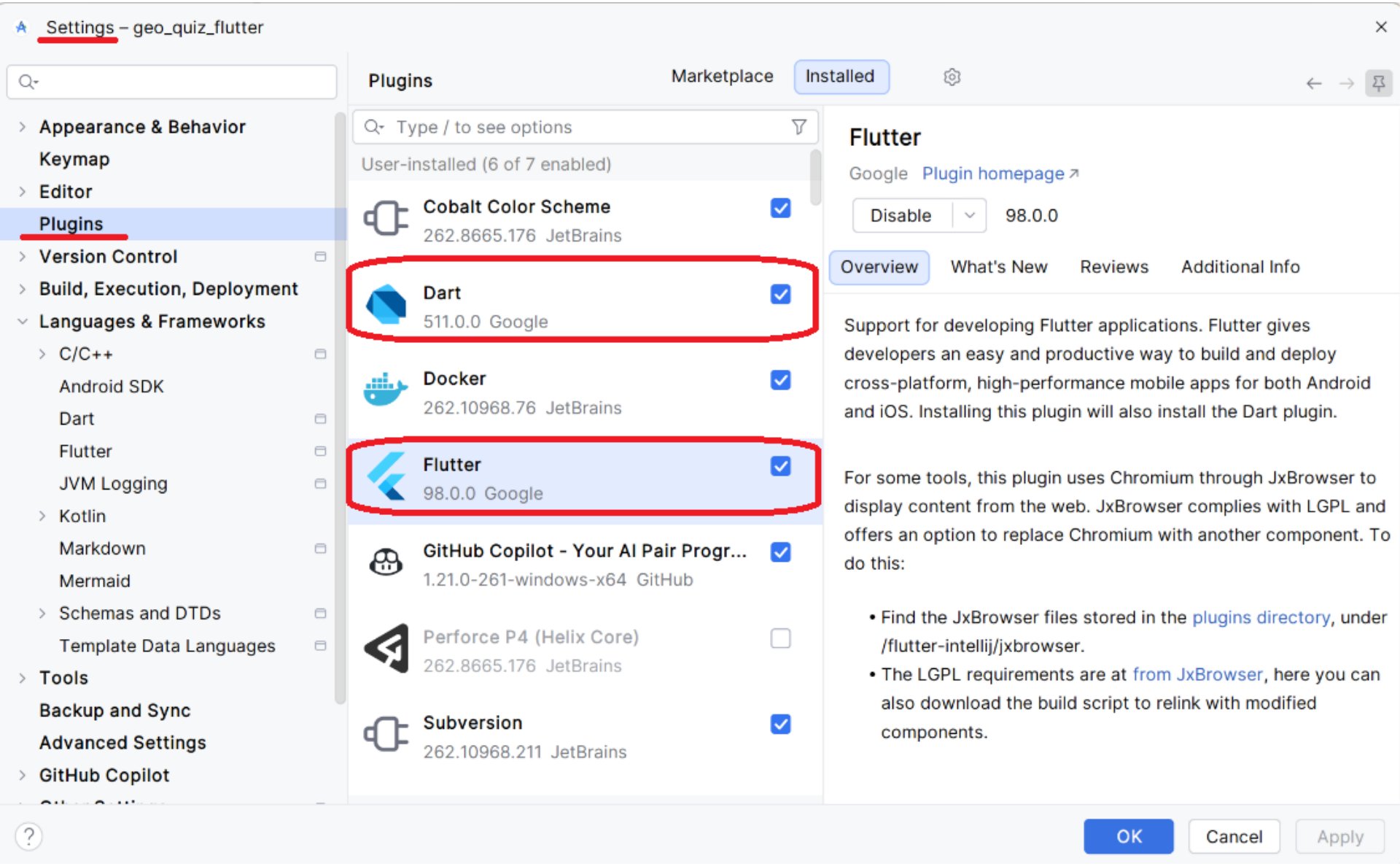


Налаштування Android Studio та XCode

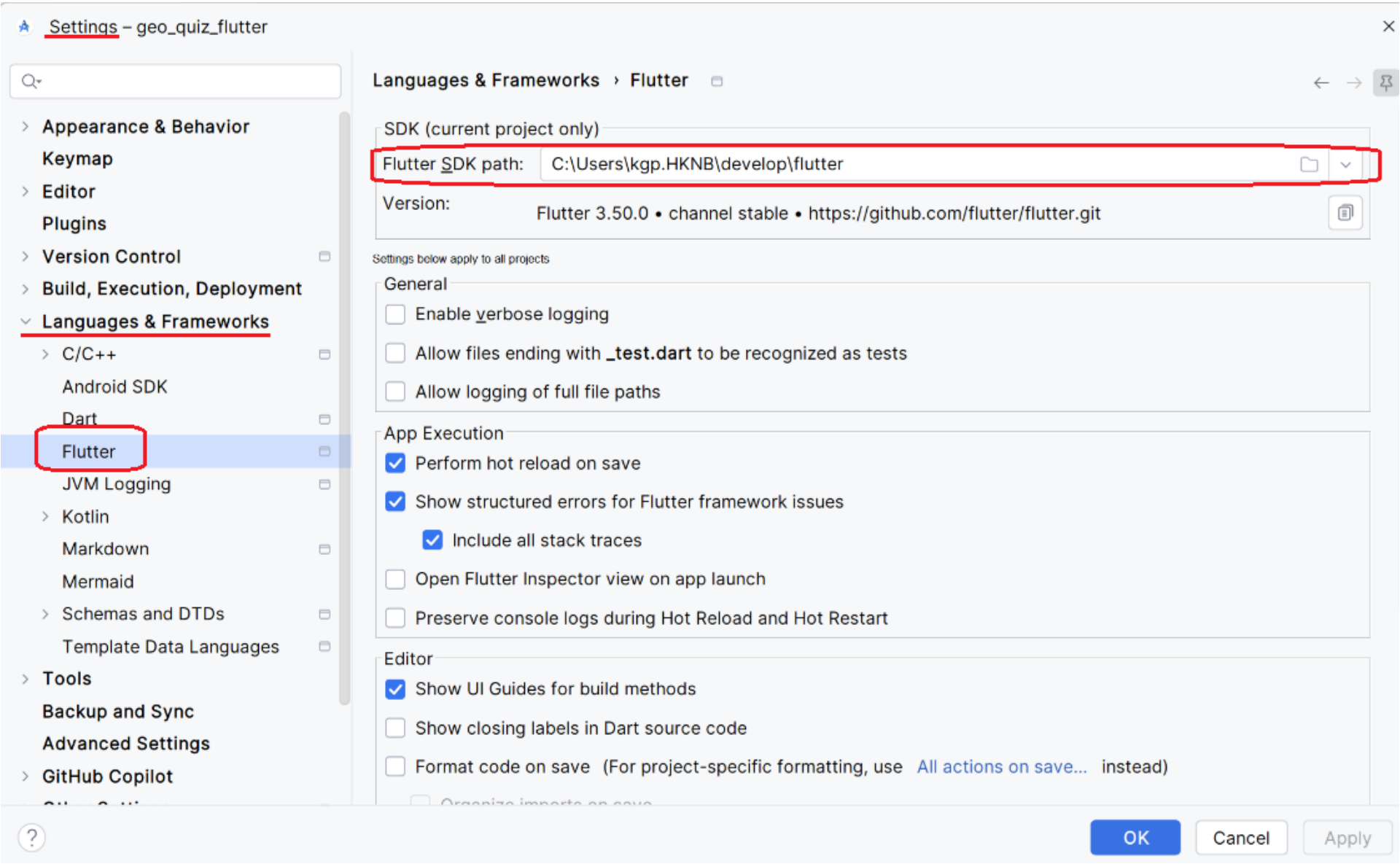
- **Android Studio:**
  - Завантажити та встановити Android Studio.
  - Додатково встановити Flutter та Dart Plugins (File > Settings > Plugins).
  - Перевірити Android SDK та емулятори.
- **XCode:**
  - Завантажити та встановити XCode з App Store.
  - Додати XCode Command Line Tools (запуск команди xcode-select --install).
  - Перевірити iOS Simulator.



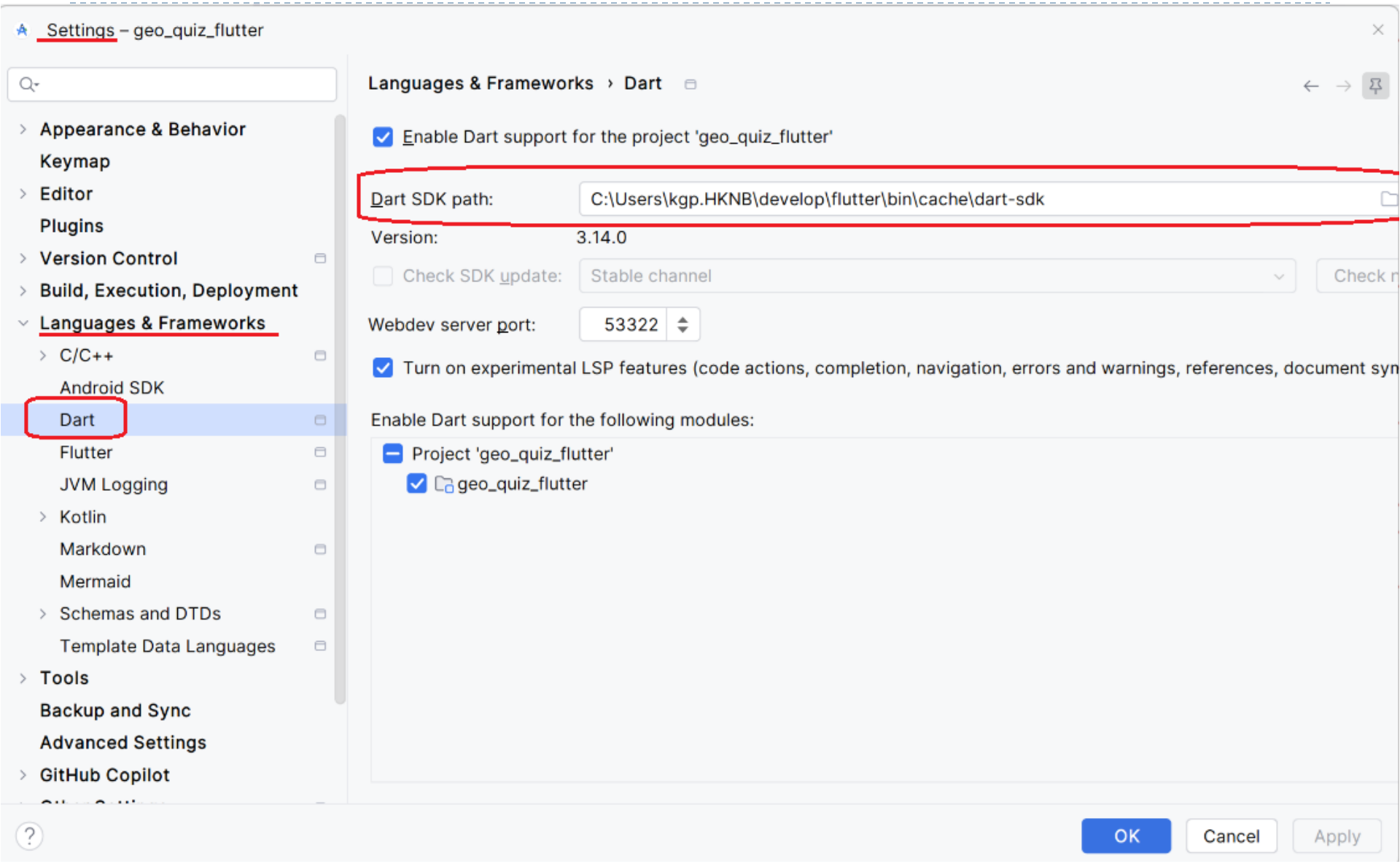
Налаштування Android Studio



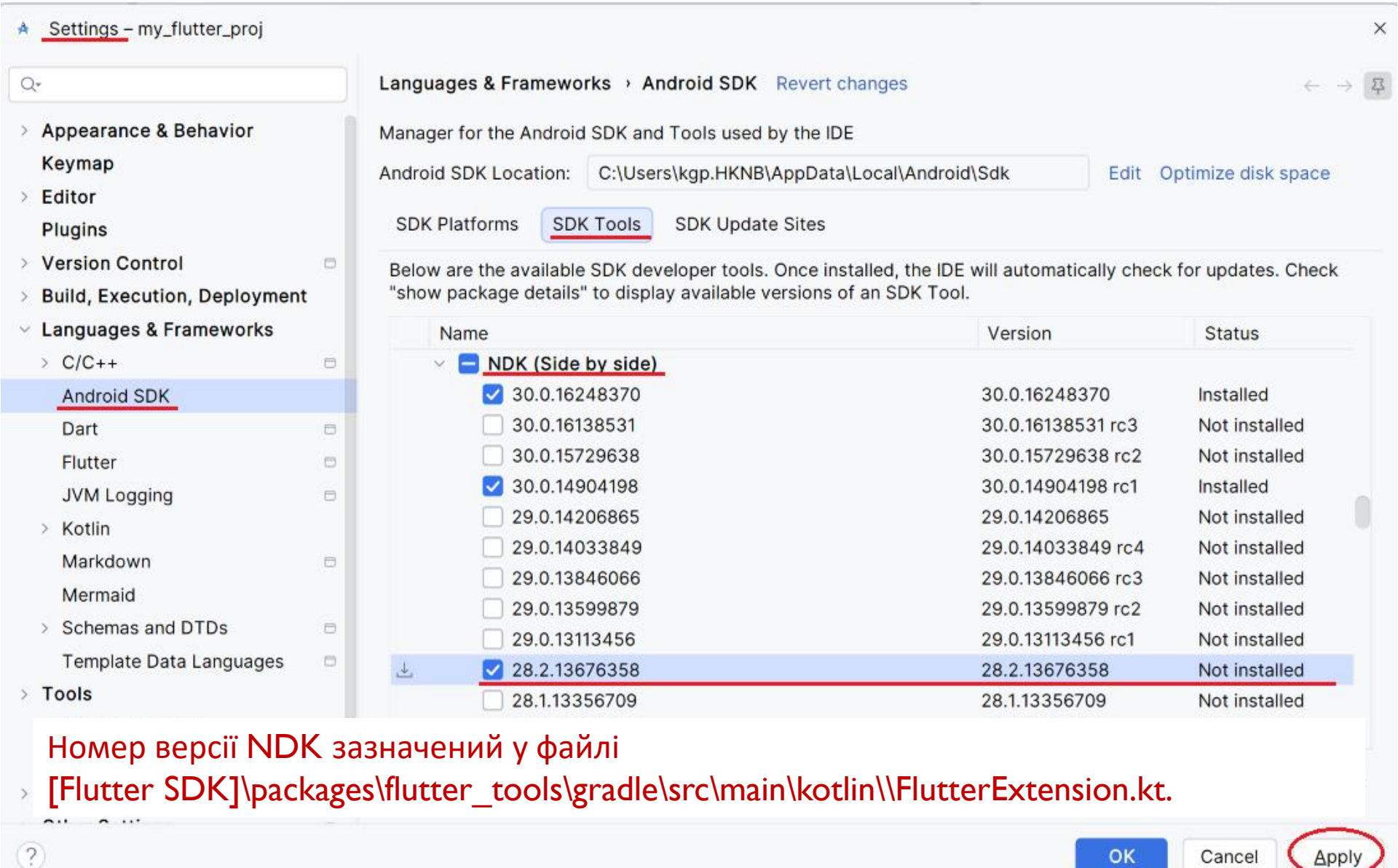
Налаштування плагіну Flutter



Налаштування плагіну Dart



Встановлення сумісної для версії Flutter версії NDK



Номер версії NDK зазначений у файлі [Flutter SDK]\packages\flutter_tools\gradle\src\main\kotlin\\FlutterExtension.kt.

Команди Flutter – розробка та запуск



- **flutter devices** - Показує список усіх доступних пристроїв, емуляторів та браузерів, на яких можна запустити код.
- **flutter run -d <назва_пристрою>** - запуск застосунка на конкретному пристрої зі списку devices (наприклад: flutter run -d chrome).
- **flutter run** - запуск застосунка на підключеному пристрої або емуляторі.
  - **Клавіша r** (в терміналі під час роботи програми) - **Hot Reload** - оновлює інтерфейс без втрати стану застосунка (займає < 1 секунди).
  - **Клавіша R** (велика) - **Hot Restart -** перезапускає застосунок повністю і скидає стан (потрібно, якщо Ви змінили логіку ініціалізації чи глобальні змінні).



Команди Flutter – керування пакетами та залежностями

- **flutter pub get** - завантажує всі пакети та бібліотеки, вказані у головному конфігураційному файлі проєкта pubspec.yaml.
- **flutter pub outdated** - аналізує Ваші пакети та показує, які з них застаріли і чи доступні для них нові мажорні версії.
- **flutter pub upgrade** - оновлює всі залежності застосунка до найсвіжіших дозволених версій.
- **flutter pub add <назва_пакета>** - швидке додавання сторонніх плагінів (наприклад, http або provider) у Ваш проєкт без ручного редагування файлу pubspec.yaml.



Файл pubspec.yaml у корені проєкту Flutter – це головний конфігураційний файл застосунка, який визначає: опис проєкту (метадані), залежності, ресурси, версії Flutter SDK та Dart SDK тощо. Після змін у цьому файлі необхідно клікнути по посиланню Pub get у вікні редактора файлу.

Команди Flutter – чищення та перевірка коду



- **flutter analyze** - статичний аналізатор коду. Перевіряє весь проєкт на наявність помилок, попереджень (warnings) та недотримання стилю написання Dart.
- **dart format .** - форматує всі файли коду в поточному каталозі відповідно до офіційного гайдлайну Dart (вирівнює відступи, розставляє коми тощо).
- **flutter clean** - видаляє кеш збірки, каталог build та тимчасові файли. **Головна команда для вирішення "магічних" помилок**, коли код правильний, але застосунок не збирається. Після неї обов'язково треба робити flutter pub get.



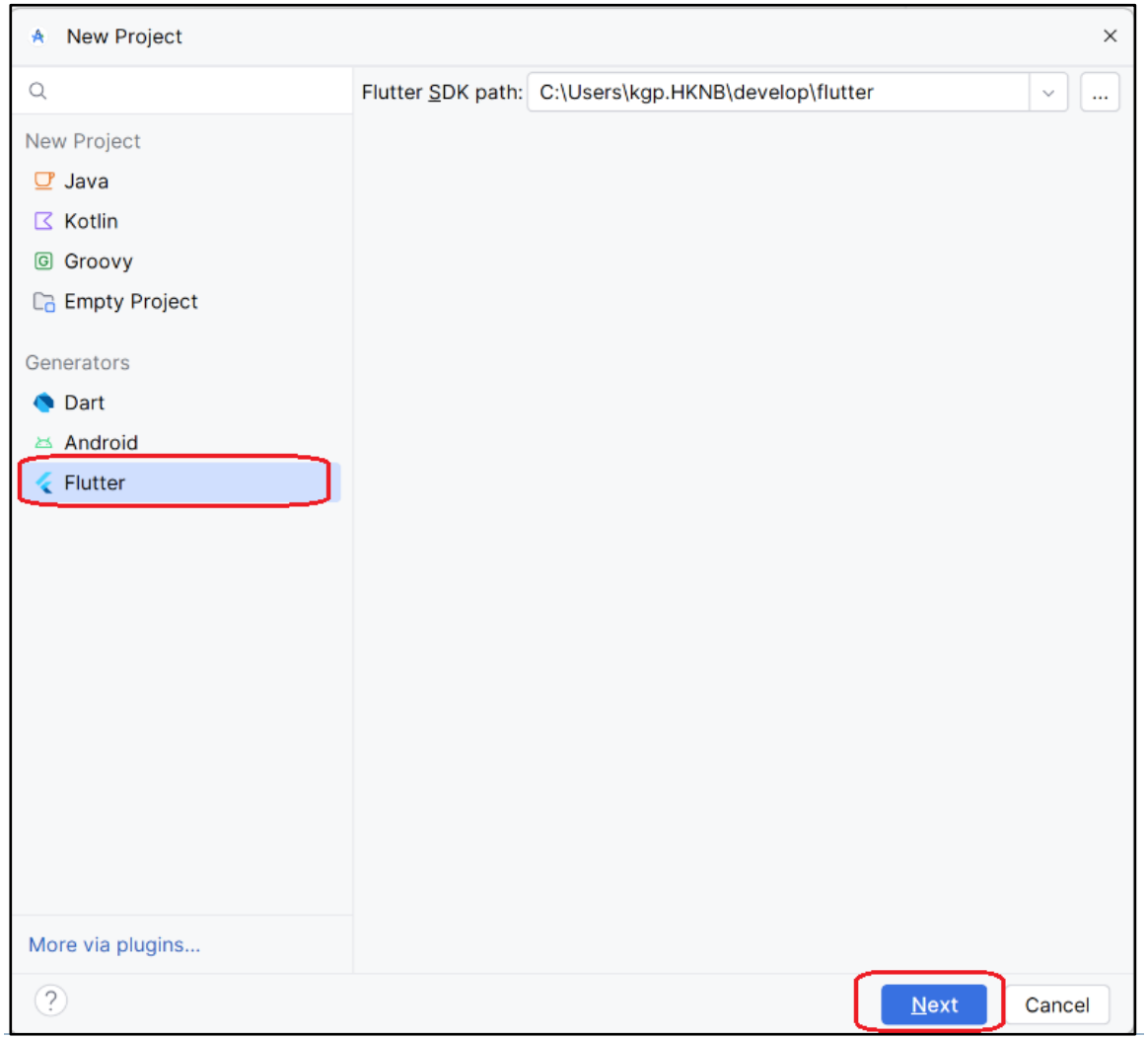
Команди Flutter – створення релізів



- **flutter build apk** - створення інсталяційного файлу .apk для Android.
- **flutter build appbundle** - створення файлу .aab (обов'язковий формат для публікації в Google Play).
- **flutter build ios** - підготовка збірки для iOS (виконується на Mac перед фінальним архівом у Xcode).
- **flutter build web** - компіляція проєкту в HTML/JS/CSS для розміщення на хостингу.



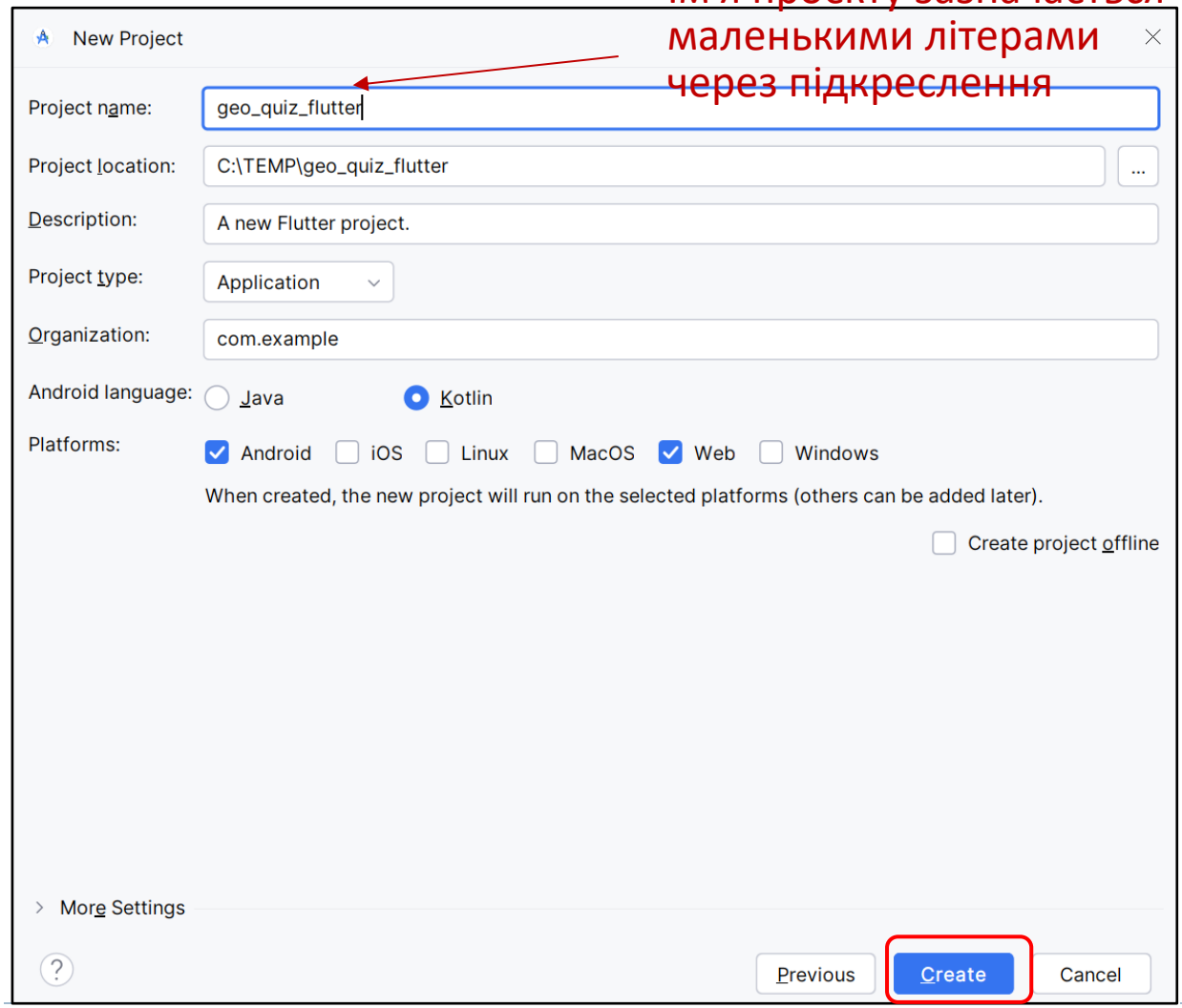
Створення проєкту в Android Studio



File - New – New Flutter Project…

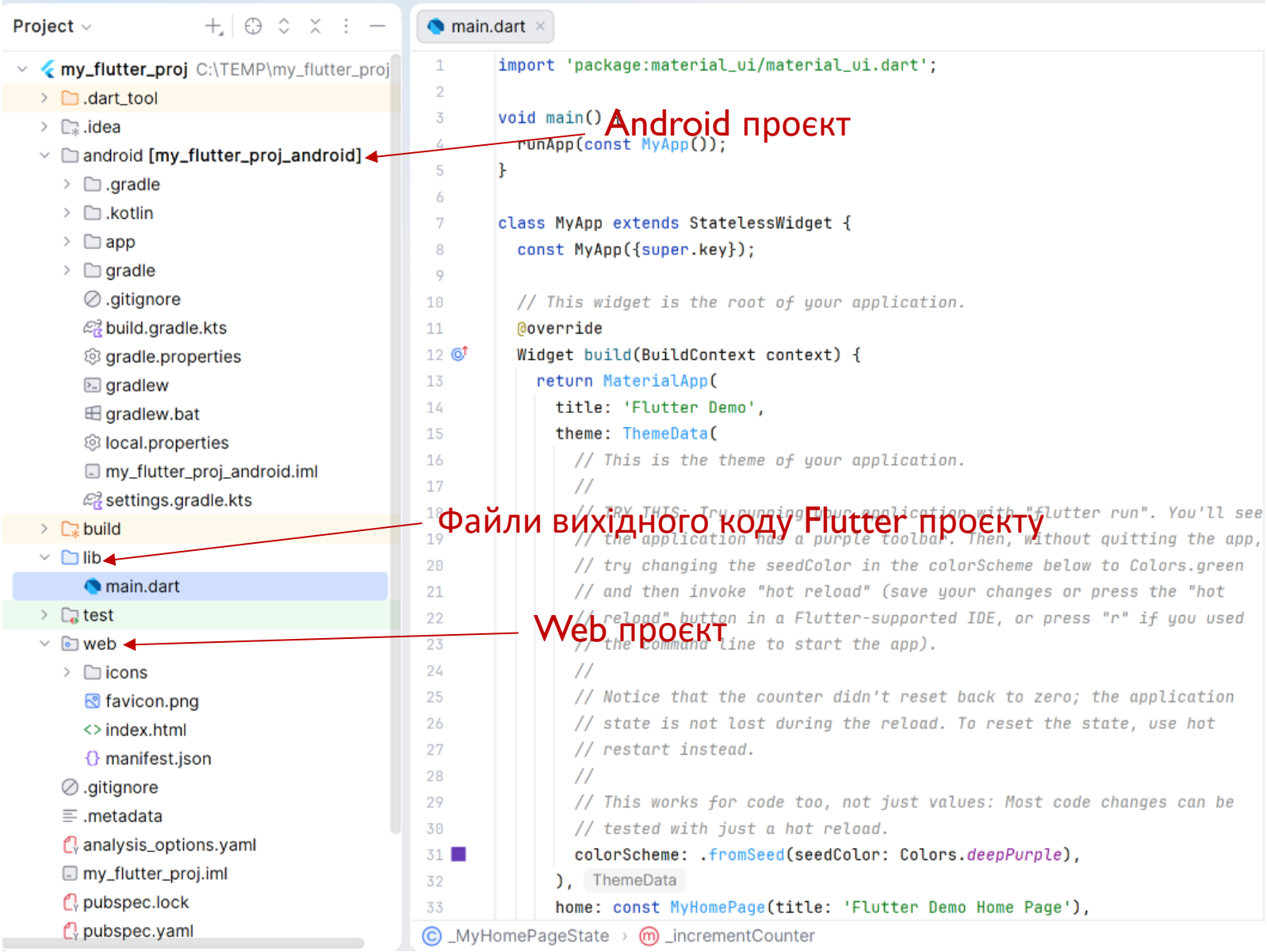
Створення проєкту в Android Studio

Ім'я проєкту зазначається маленькими літерами через підкреслення



File - New – New Flutter Project…

Структура Flutter проєкту в Android Studio



Обрання цільової платформи

Цільова платформа	Необхідна ОС для розробки	Обов'язкове програмне забезпечення (Інструменти)
Android	Windows, macOS або Linux	• Android Studio або VS Code • Android SDK та Command-line Tools • Java Development Kit (JDK) • Android Emulator або реальний Android-пристрій
iOS	Тільки macOS	• Xcode (актуальна версія з Mac App Store) • Менеджер залежностей Cocoapods (sudo gem install cocoapods) • Симулятор iOS або реальний iPhone/iPad
Web	Windows, macOS або Linux	• Браузер Google Chrome (для налагодження та дебагу) • Будь-яка IDE (VS Code, Android Studio)
Windows	Тільки Windows (10 або 11)	• Visual Studio (не VS Code!) з компонентом "Desktop development with C++" • Інструменти CMake та MSVC v142/v143
macOS	Тільки macOS	• Xcode (актуальна версія) • Менеджер залежностей Cocoapods
Linux	Тільки Linux (eg. Ubuntu)	• Інструменти збірки: clang, cmake, ninja-build, pkg-config • Бібліотеки: libgtk-3-dev, liblzma-dev

Головна функція та клас застосунку

```
import 'package:material_ui/material_ui.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

@override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(
                        seedColor: Colors.deepPurple),
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
...
```

Пакет з основними компонентами і функціями Flutter: runApp, StatelessWidget, StatefulWidget, State, MaterialApp, ThemeData, BuildContext, ...

Точка входу застосунку

Створює незмінний об'єкт MyApp на етапі компіляції

MyApp - це віджет без стану

Може передавати опціональний параметр key суперкласу, за яким Flutter може відслідковувати ідентичність віджета у дереві віджетів

Повертає кореневий віджет застосунку, що відповідає правилам Material - перевизначення методу StatelessWidget

Тема застосунку

Визначає стартовий віджет застосунку

Повинна бути визначена хоча б одна з властивостей: **home, routes, onGenerateRoute,** or **builder**

Головна функція та клас застосунку

```
...
class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});
  final String title;
  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;
  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }
...
```

MyHomePage - це віджет зі станом

Обов'язковий параметр – альтернатива ? – null safety

Перевизначення методу StatefulWidget

Клас стану віджету

Знак підкреслення робить член класу приватним (на рівні файлу (бібліотеки))

Обробник події

Цей виклик setState повідомляє фреймворку Flutter, що в цьому стані щось змінилося, що змушує його повторно запустити метод build нижче, щоб дисплей міг відобразити оновлені значення. Якщо ми змінимо _counter без виклику setState(), то метод build не буде викликаний.



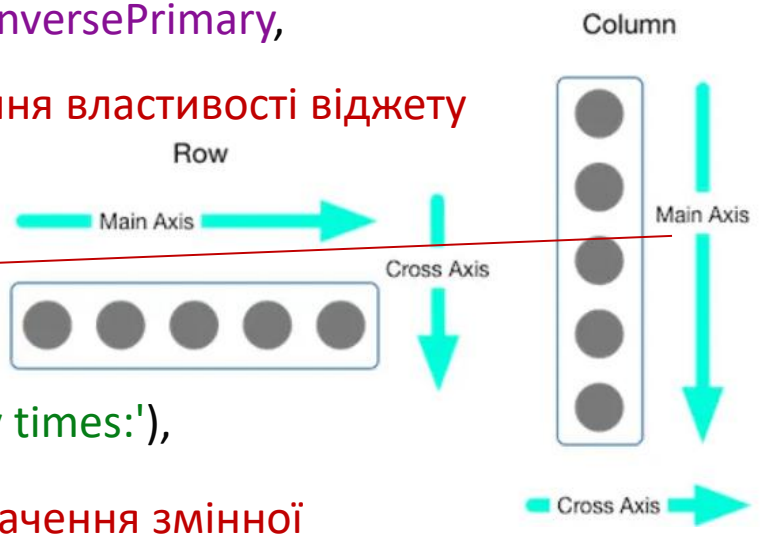
Створення проєкту в Android Studio

```
@override
Widget build(BuildContext context) {
 return Scaffold(
  appBar: AppBar(
   backgroundColor: Theme.of(context).colorScheme.inversePrimary,
   title: Text(widget.title),
  ),
  body: Center(
   child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: <Widget>[
     const Text('You have pushed the button this many times:'),
     Text(
      '$_counter',
      style: Theme.of(context).textTheme.headlineMedium,
     ),        ],       ),     ),
  floatingActionButton: FloatingActionButton(
   onPressed: _incrementCounter,
   tooltip: 'Increment',
   child: const Icon(Icons.add),
 ), );  }}
```

Перевизначення методу State - цей метод автоматично запускається при змінах стану

Повертає користувацький віджет - екран застосунку

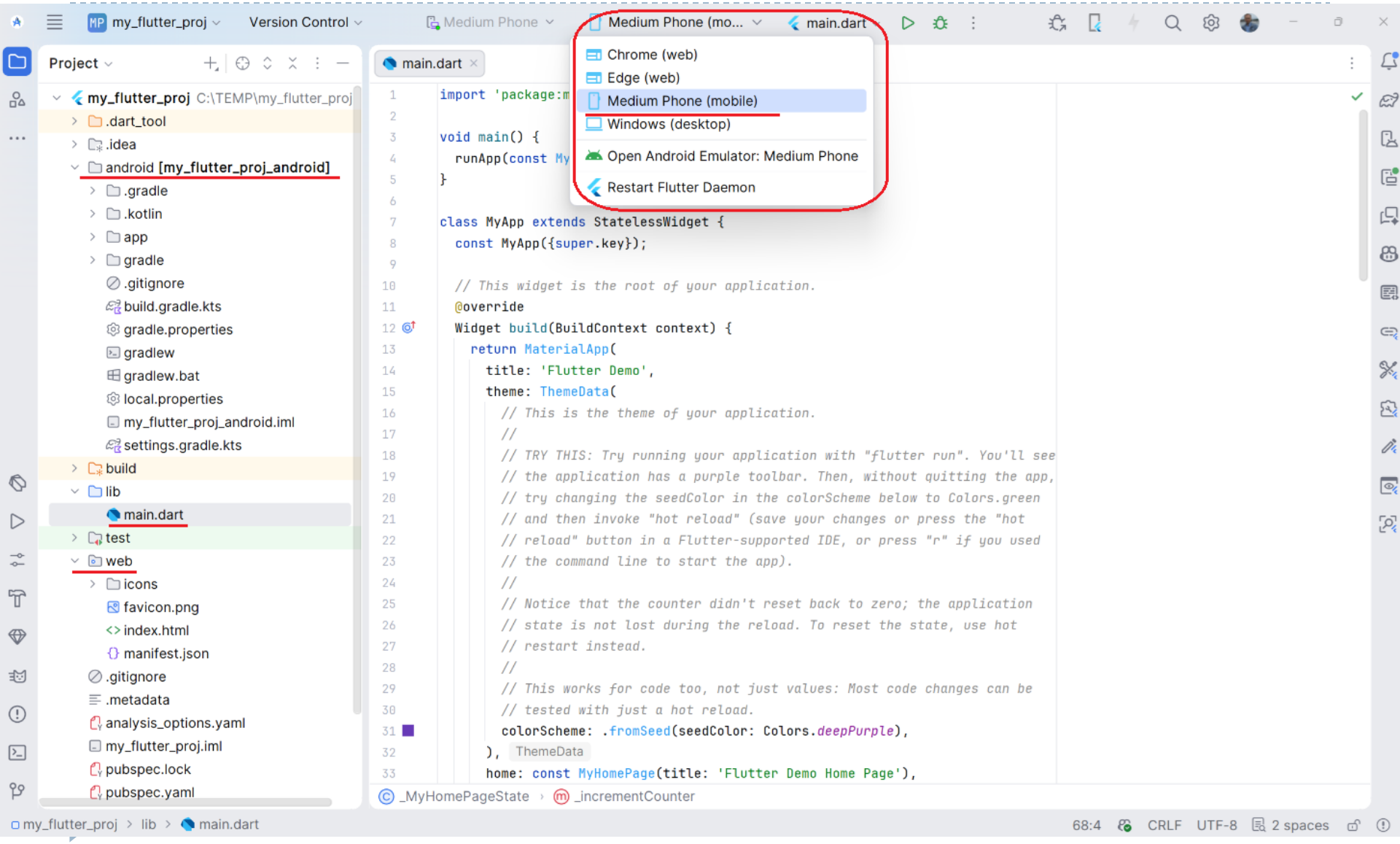
Використовує значення властивості віджету MyHomePage



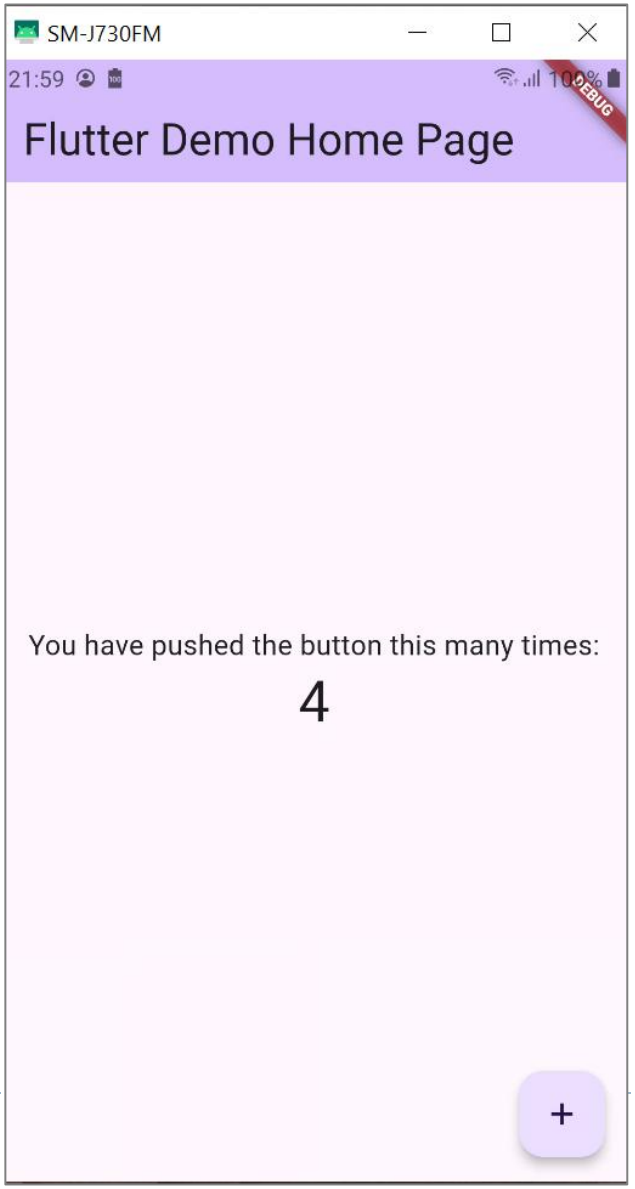
Отримання зі стану значення змінної

Виклик зі стану обробника події

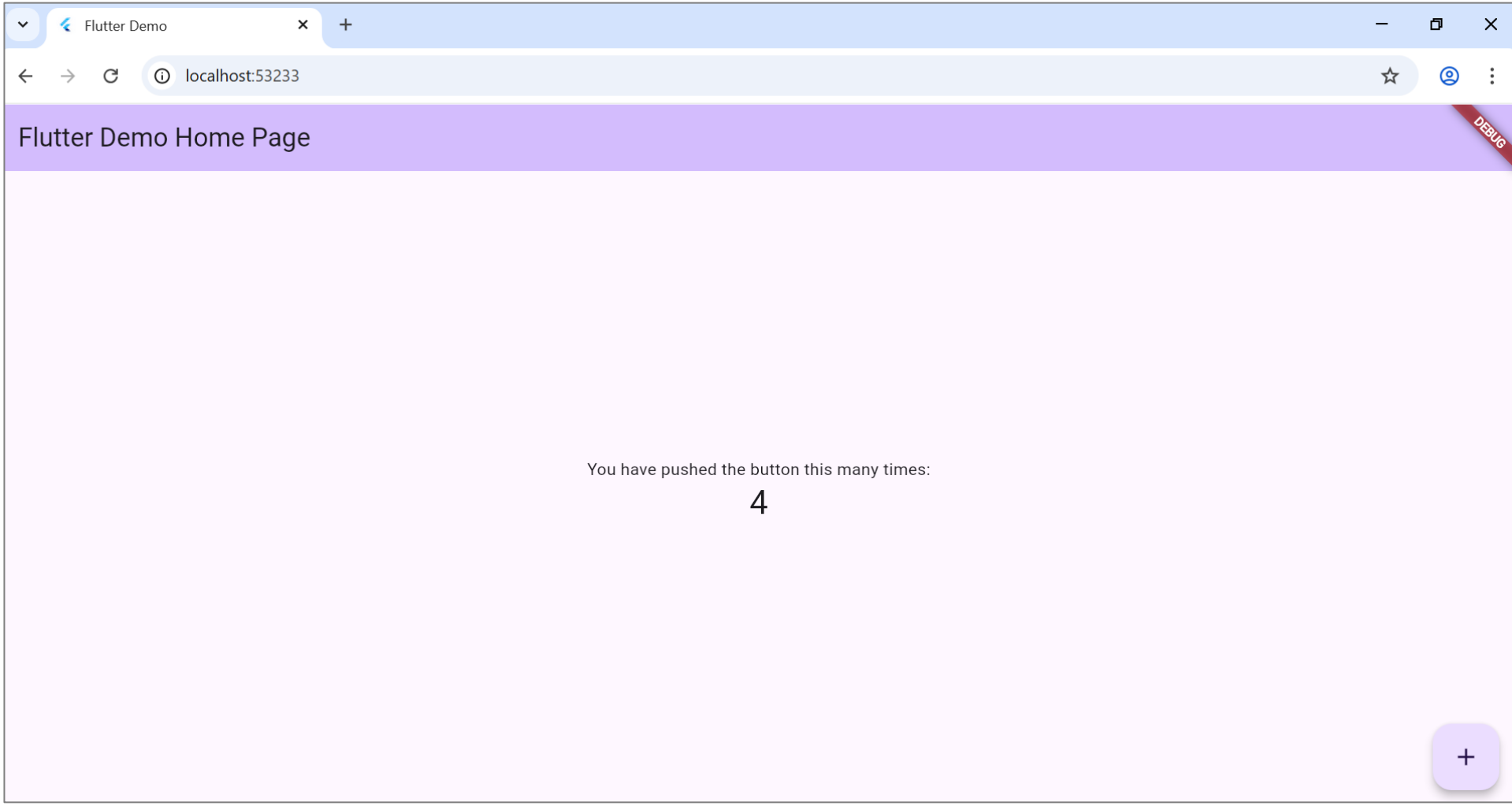
Запуск застосунку в Android



Запуск застосунку на Android пристрії

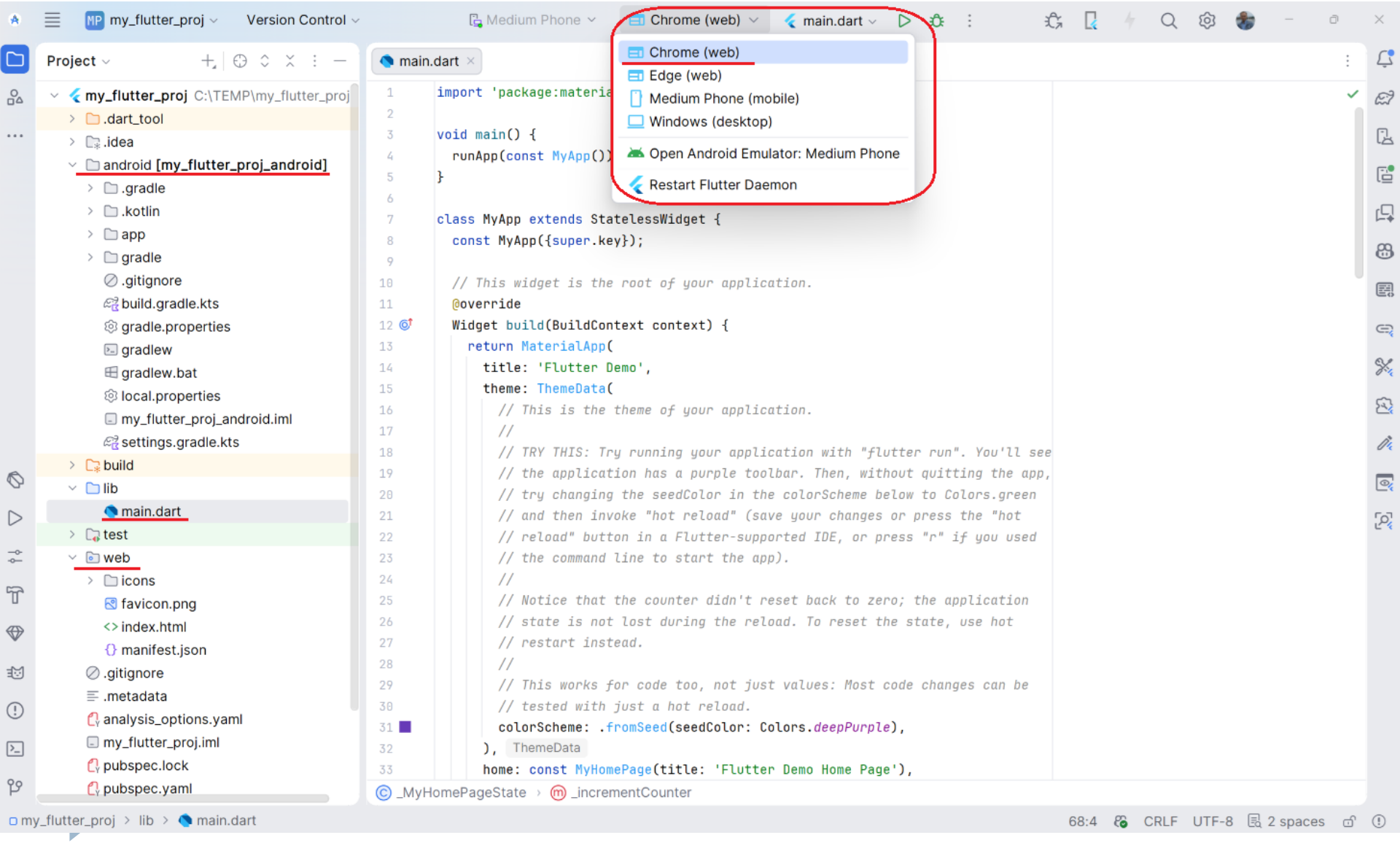


Запуск застосунку в браузері

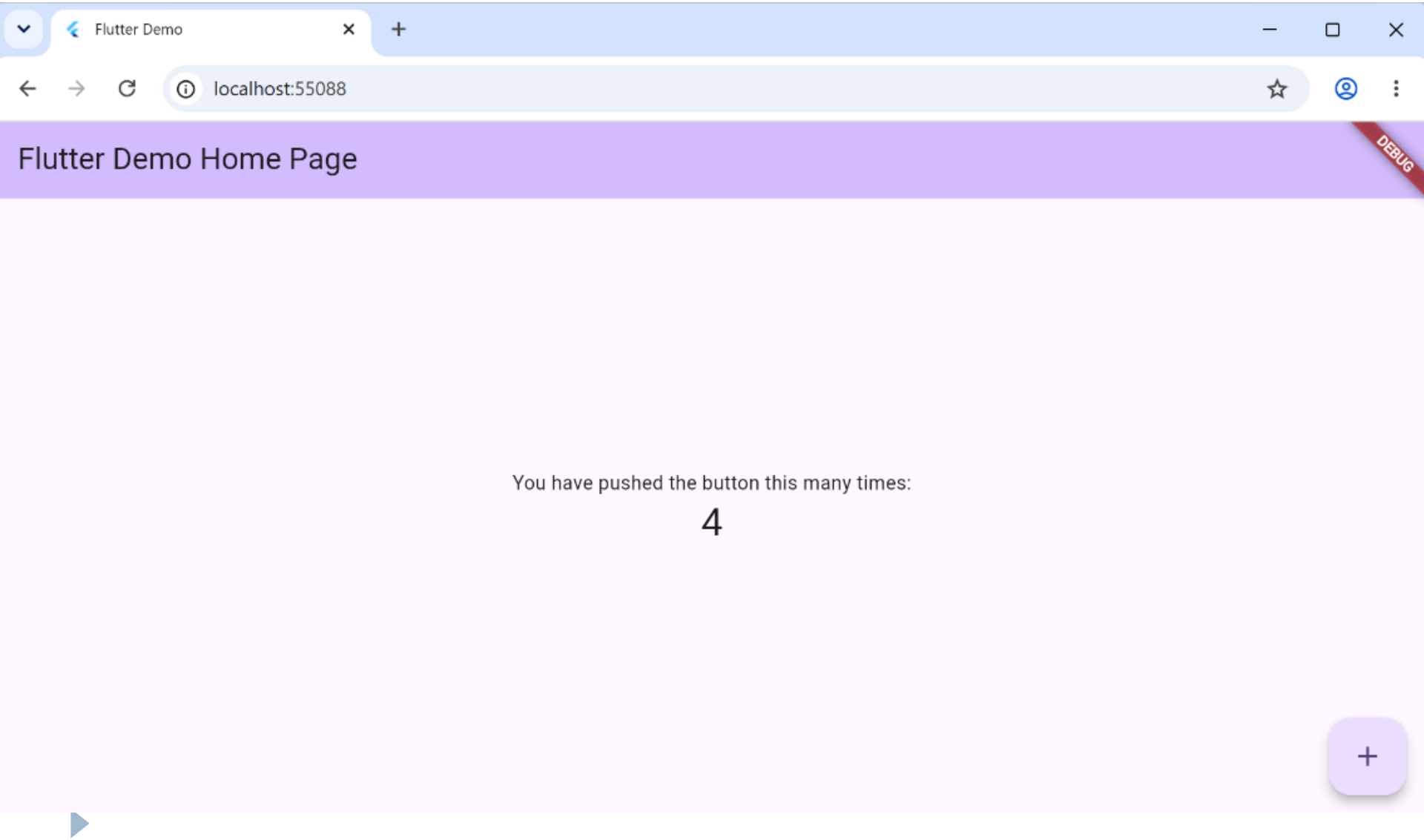




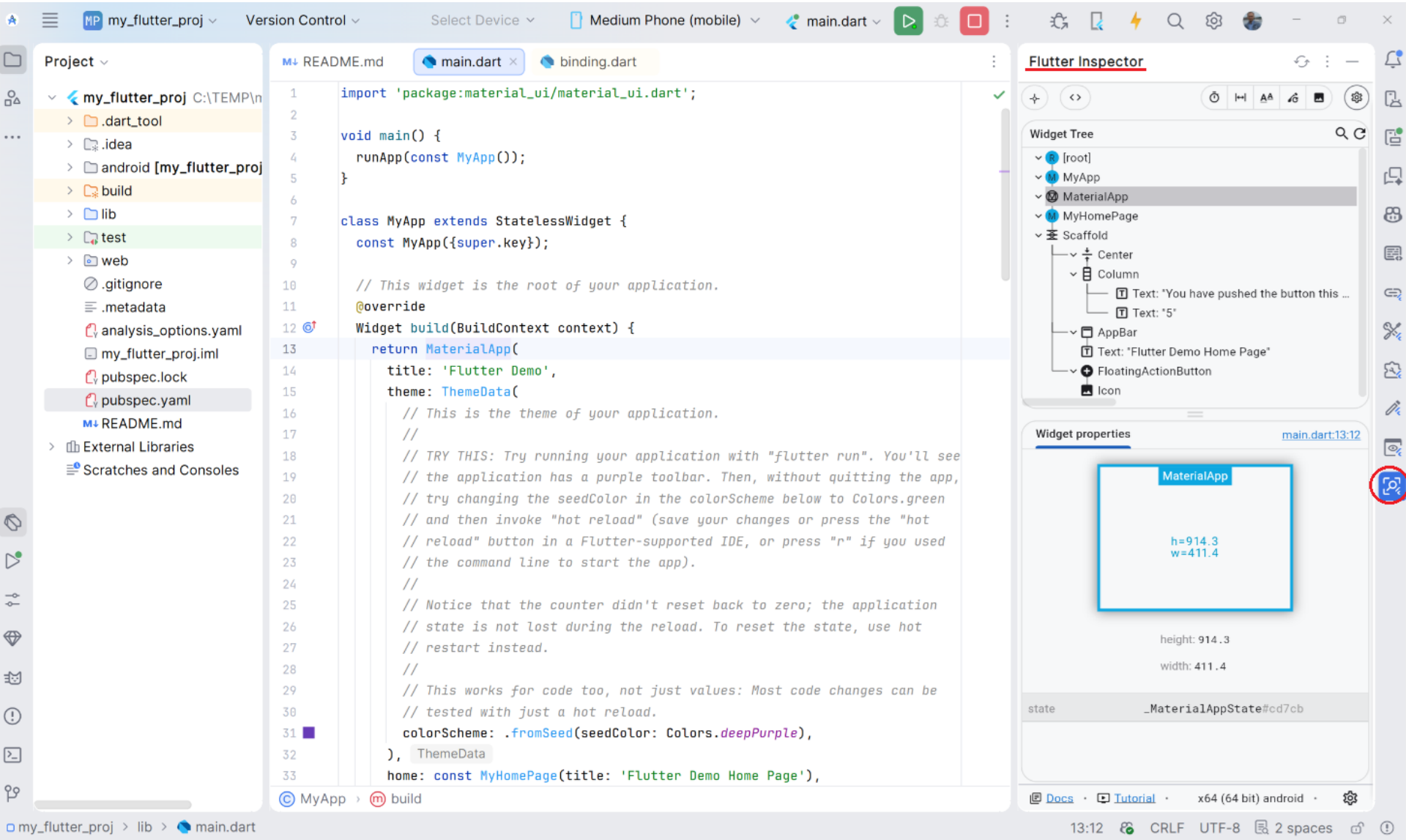
Запуск застосунку у Web



Запуск застосунку у Web



Перегляд дерева віджетів у Flutter Inspector



Flutter Widgets



- Віджети Flutter – це будівельні блоки інтерфейсу користувача.
- **Основні типи віджетів**:
  - **StatelessWidget:** Ці віджети не змінюють своїх властивостей під час роботи програми. Їхня поведінка визначається лише конфігурацією, яку вони отримують під час створення (наприклад, текст у Text або іконка у Icon).
  - **StatefulWidget:** Ці віджети можуть змінювати свої властивості динамічно протягом роботи програми. Вони мають об'єкт стану (State), який зберігає всю інформацію, що може змінюватися (наприклад, стан перемикача, текст у полі введення).



Flutter Widgets

- **Категорії віджетів**:
  - **Віджети компонування (Layout widgets):** Віджети, що керують розміщенням та позиціонуванням інших віджетів.
    - Column - розміщує віджети в стовпці,
    - Row - розміщує віджети в рядку,
    - Stack - накладає віджети один на одного,
    - Container - для додавання полів, відступів, меж та кольору та створення складених інтерфейсів,
    - Expanded - заповнює все доступне у поточному лейауті місце
    - Positioned - розміщує дочірній віджет з позиціонуванням відносно поточного лейауту.



Flutter Widgets

- **Категорії віджетів**:
  - **Віджети компонування (Layout widgets):** Віджети, що керують розміщенням та позиціонуванням інших віджетів.
  - **Віджети інтерфейсу користувача (UI widgets):** Віджети, які відображають дані або надають користувачеві певну функціональність.



Flutter Layout widgets

```
import 'package:material_ui/material_ui.dart';

void main() {
  runApp(const LayoutApp());
}

class LayoutApp extends StatelessWidget {
  const LayoutApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
      appBar: AppBar(title: const Text('Flutter Layouts')),
      body: Column(
        children: [
          Center(
            child: Text(
              'Hello World',
              style: TextStyle(color: Colors.black, fontSize: 24),
            ),
          ), ...
```

LayoutApp - це віджет без стану

Повинна бути визначена хоча б одна з властивостей: **home, routes, onGenerateRoute,** or **builder**

Розміщує дочірні віджети по вертикалі

Розміщує дочірні віджети по вертикалі

Розміщує один дочірній віджет по центру батьківського лейаута

lib/edu/layouts.dart

Flutter Layout widgets

...

```
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Icon(Icons.star, color: Colors.red, size: 30),
    SizedBox(width: 10),
    Icon(Icons.star, color: Colors.green, size: 30),
    SizedBox(width: 10),
    Icon(Icons.star, color: Colors.blue, size: 30),
  ],
),
Container(
  color: Colors.blue,
  height: 100,
  child: const Center(
    child: Text(
      'Container Example',
      style: TextStyle(color: Colors.white, fontSize: 24),
    ),
  ),
), ...
```

Розміщує дочірні віджети по горизонталі

Створює складений лейаут

Flutter Layout widgets

...

Розміщує дочірні віджети у всьому доступному просторі батьківського лейауту

```
Expanded(
 child: Row(
  children: [
   Container(
    padding: const EdgeInsets.only(left: 10, right: 10),
    color: Colors.green,
    width: 120,
    child: const Center(
     child: Text(
      'Sidebar',
      style: TextStyle(color: Colors.white, fontSize: 18),                    ),                    ), ),
   Expanded(
    child: Container(
     color: Colors.grey[200],
     child: const Center(
      child: Text(
       'Main Content Area',
       style: TextStyle(fontSize: 20),                    ),                    ),                    ),                    ),                ],  ), ),
```

...

Flutter Layout widgets

...

```
Stack(
  children: [
    Container(
      width: double.infinity,
      height: 150,
      color: Colors.orange,
    ),
    Positioned(
      top: 50,
      left: 50,
      child: Container(
        width: 100,
        height: 100,
        color: Colors.purple,
        child: const Center(
          child: Text(
            'Stacked Box',
            style: TextStyle(color: Colors.white),
          ),
        ),
      ),
    ),
  ],
),
```

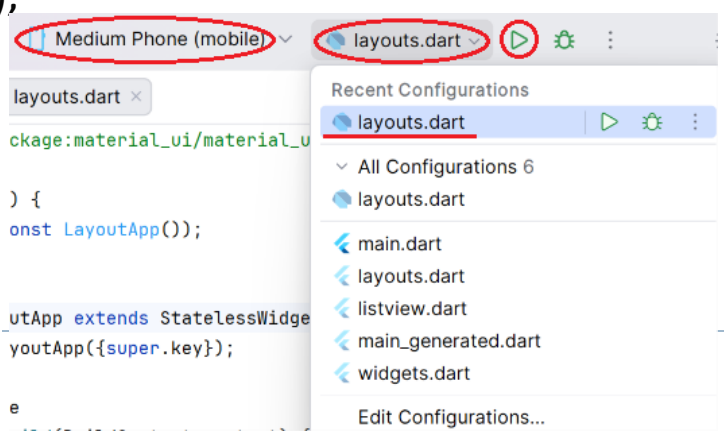
Розміщує дочірні віджети один поверх іншого

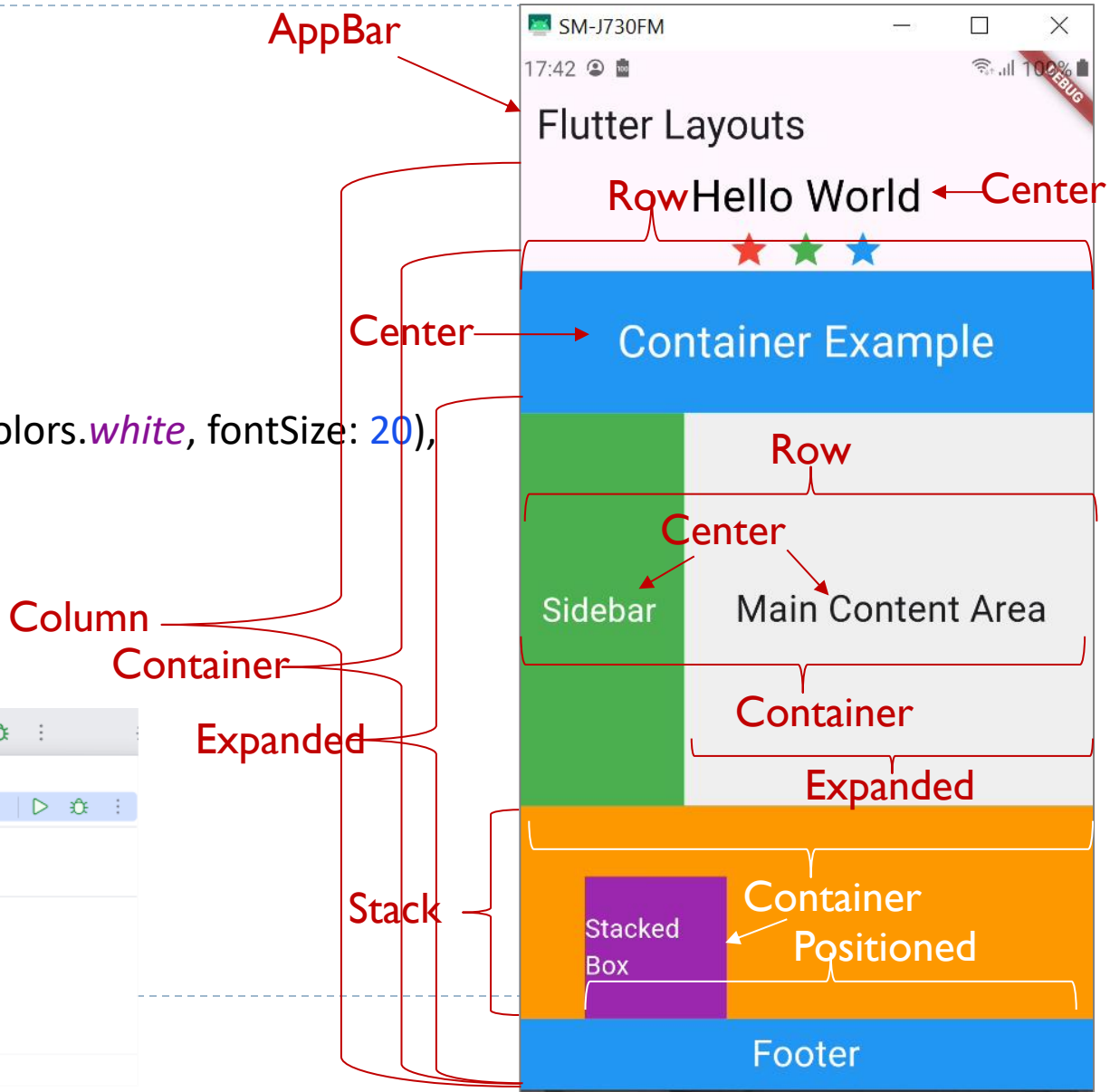
Розміщує один дочірній віджет поверх батьківського віджету з позиціонуванням відносно батьківського

...

Flutter Layout widgets

```
...
      Container(
        color: Colors.blue,
        height: 50,
        width: double.infinity,
        child: const Center(
          child: Text(
            'Footer',
            style: TextStyle(color: Colors.white, fontSize: 20),
          ),
        ),
      ),
    ],
  ),
),
);
}
}
```





Flutter UI Widgets



- **Віджети інтерфейсу користувача (UI widgets):** Віджети, які відображають дані або надають користувачеві певну функціональність.
  - Text: Відображає текст.
  - Image: Відображає зображення.
  - Icon: Відображає іконку.
  - Button: Відображає кнопку.
  - TextField: Поле для введення тексту.
  - Checkbox: Чекбокс.
  - Radio: Радіокнопка.
  - Slider: Повзунок.



Flutter UI Widgets

```
import 'package:material_ui/material_ui.dart';

void main() {
  runApp(const WidgetsDemoApp());
}

class WidgetsDemoApp extends StatelessWidget {
  const WidgetsDemoApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Stateful Widgets Demo',
      theme: ThemeData(useMaterial3: true),
      home: const WidgetsDemoPage(title: 'Stateful Widgets Demo'),
    );
  }
}
...
```

WidgetsDemoApp - це віджет без стану



lib/edu/widgets.dart

Flutter UI Widgets

```
...
class WidgetsDemoPage extends StatefulWidget {
  const WidgetsDemoPage({super.key, required this.title});
  final String title;
  @override
  State<WidgetsDemoPage> createState() => _WidgetsDemoPageState();
}

class _WidgetsDemoPageState extends State<WidgetsDemoPage> {
  int _counter = 0;
  bool _isFavorite = false;
  bool _showImage = true;
  void _incrementCounter() {
    setState(() {_counter++; });
  }
  void _toggleFavorite() {
    setState(() { _isFavorite = !_isFavorite; });
  }
...
```

WidgetsDemoPage - це віджет зі станом

Стан віджету

змінні, що визначають стан стандартних віджетів

обробник події кліку по кнопці ElevatedButton

обробник події кліку по кнопці IconButton

Flutter UI Widgets

```
...
 void _reset() {
  setState(() {
    _counter = 0;
    _isFavorite = false;
    _showImage = true;
  });
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
      actions: [
        IconButton(
          tooltip: 'Reset',
          onPressed: _reset,
          icon: const Icon(Icons.refresh),
        ),
      ],
    ), ...
```

обробник події кліку по кнопці IconButton в AppBar

виклик обробника події кліку по кнопці IconButton в AppBar

Flutter UI Widgets

```
...
    body: Padding(
      padding: const EdgeInsets.all(16),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text('Counter: $_counter', style: Theme.of(context).textTheme.headlineMedium),
          const SizedBox(height: 12),
          ElevatedButton.icon(
            onPressed: _incrementCounter,
            icon: const Icon(Icons.add),
            label: const Text('Increment'),
          ),
          const SizedBox(height: 24),
...
```

виклик обробника події кліку по кнопці ElevatedButton



Flutter UI Widgets

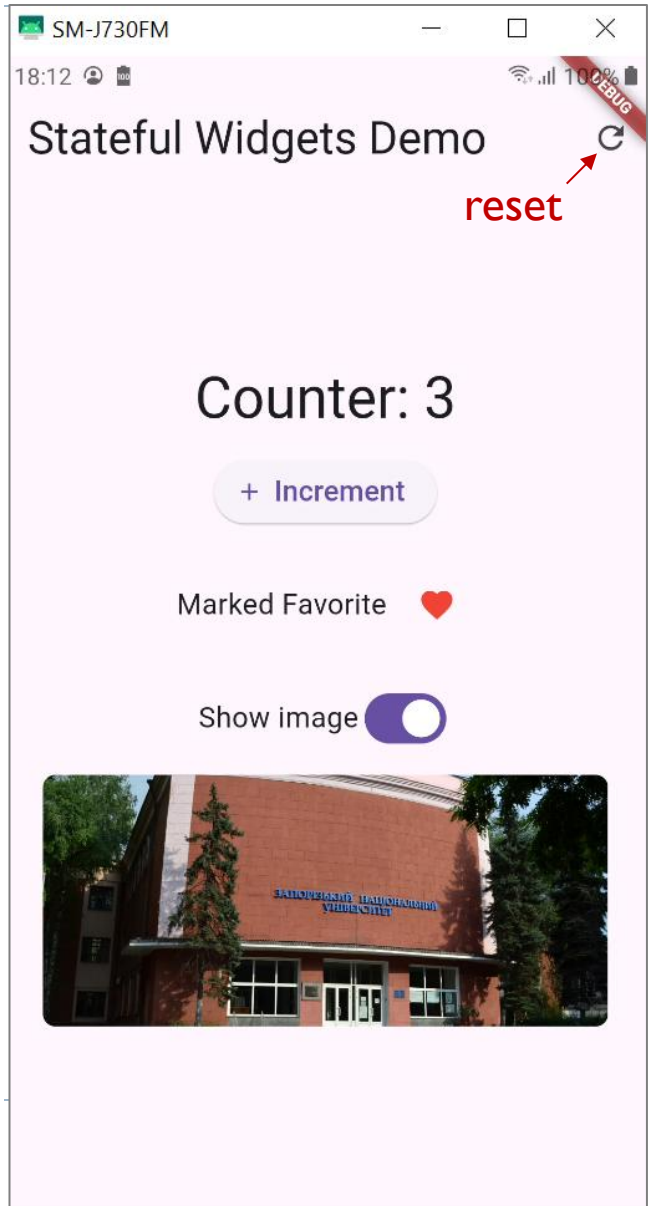
```
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Text(_isFavorite ? 'Marked Favorite' : 'Not Favorite'),
    const SizedBox(width: 8),
    IconButton(
      onPressed: _toggleFavorite,
      icon: Icon(
        _isFavorite ? Icons.favorite : Icons.favorite_border,
        color: _isFavorite ? Colors.red : null,
      ),          ),          ],          ),
const SizedBox(height: 24),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    const Text('Show image'),
    Switch(
      value: _showImage,
      onChanged: (v) => setState(() => _showImage = v),
    ),          ],          ), ,,,
```

виклик обробника події кліку по кнопці IconButton

виклик обробника події не зі стану

Flutter UI Widgets

```
...
        const SizedBox(height: 12),
        if (_showImage)
          ClipRRect(
            borderRadius: BorderRadius.circular(8),
            child: Image.network(
              'https://www.znu.edu.ua/
               gallery/2019/02/
               photo-3603-photo-3603-adminkorpus.jpg',
              height: 160,
              fit: BoxFit.cover,
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```



Flutter UI Widgets - ListView

```
import 'package:material_ui/material_ui.dart';

void main() {
  runApp(const ListViewApp());
}

class ListViewApp extends StatefulWidget {
  const ListViewApp({super.key});

  @override
  State<ListViewApp> createState() => _ListViewAppState();
}

class _ListViewAppState extends State<ListViewApp> {
  final List<Map<String, dynamic>> _items = [
    {'icon': Icons.map, 'title': 'Map'},
    {'icon': Icons.photo_album, 'title': 'Album'},
    {'icon': Icons.phone, 'title': 'Phone'},
    {'icon': Icons.contacts, 'title': 'Contacts'},
    {'icon': Icons.settings, 'title': 'Settings'},
  ];
...
```

ListViewAp - це віджет зі станом

Стан віджету

значення мапи визначається динамічно

lib/edu/listview.dart

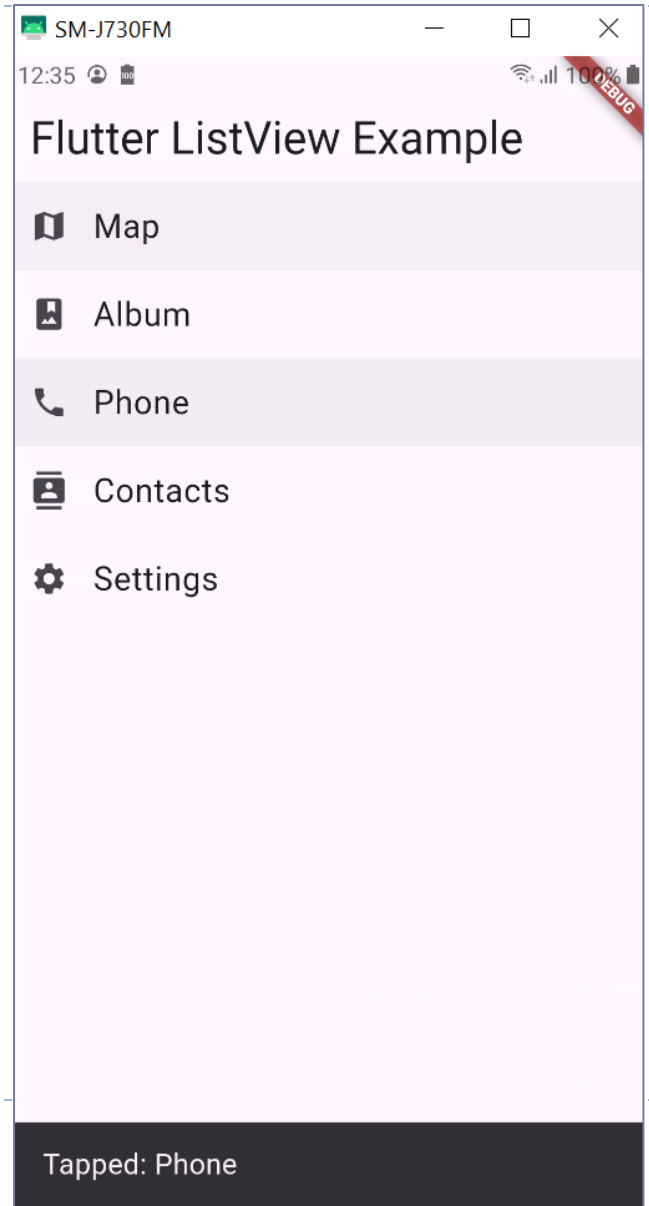
Flutter UI Widgets - ListView

```
...
void _showMessage(BuildContext ctx, String text) {
  final messenger = ScaffoldMessenger.of(ctx);
  messenger.hideCurrentSnackBar();
  messenger.showSnackBar(
    SnackBar(content: Text(text), duration: Duration(seconds: 2)),
  );  }
@override
Widget build(BuildContext context) {
  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(title: const Text('Flutter ListView Example')),
      body: ListView.builder(
        itemCount: _items.length,
        itemBuilder: (context, index) {
          final item = _items[index];
          return ListTile(
            leading: Icon(item['icon'] as IconData),
            title: Text(item['title'] as String),
            onTap: () => _showMessage(context, 'Tapped: ${item['title']}'),
          );        },      ),    );  }}
```

обробник події кліку по елементу списку

побудова списку

Flutter UI Widgets - ListView



Flutter UI Widgets - GridView

```
import 'package:material_ui/material_ui.dart';
void main() {
  runApp(const GridViewApp());
}
class GridViewApp extends StatefulWidget {
  const GridViewApp({super.key});
  @override
  State<GridViewApp> createState() => _GridViewAppState();
}
class _GridViewAppState extends State<GridViewApp> {
  List<int> _values = List<int>.generate(12, (i) => i + 1);
  void _squareValues() {
    setState(() {
      _values = _values.map((v) => v * v).toList();
    });
  }
...
```

GridViewAp - це віджет зі станом

Стан віджету

обробник події кліку по кнопці



lib/edu/gridview.dart

Flutter UI Widgets - GridView

```
...
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   home: Scaffold(
    appBar: AppBar(title: const Text('Flutter GridView Example')),
    body: Padding(
     padding: const EdgeInsets.all(8.0),
     child: GridView.builder(
      gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
       crossAxisCount: 3,
       crossAxisSpacing: 4.0,
       mainAxisSpacing: 4.0,
      ),
      itemCount: _values.length,
      itemBuilder: (context, index) {
       final value = _values[index];
       final colorIndex = (value % 9 + 1) * 100;
...
```

побудова таблиці

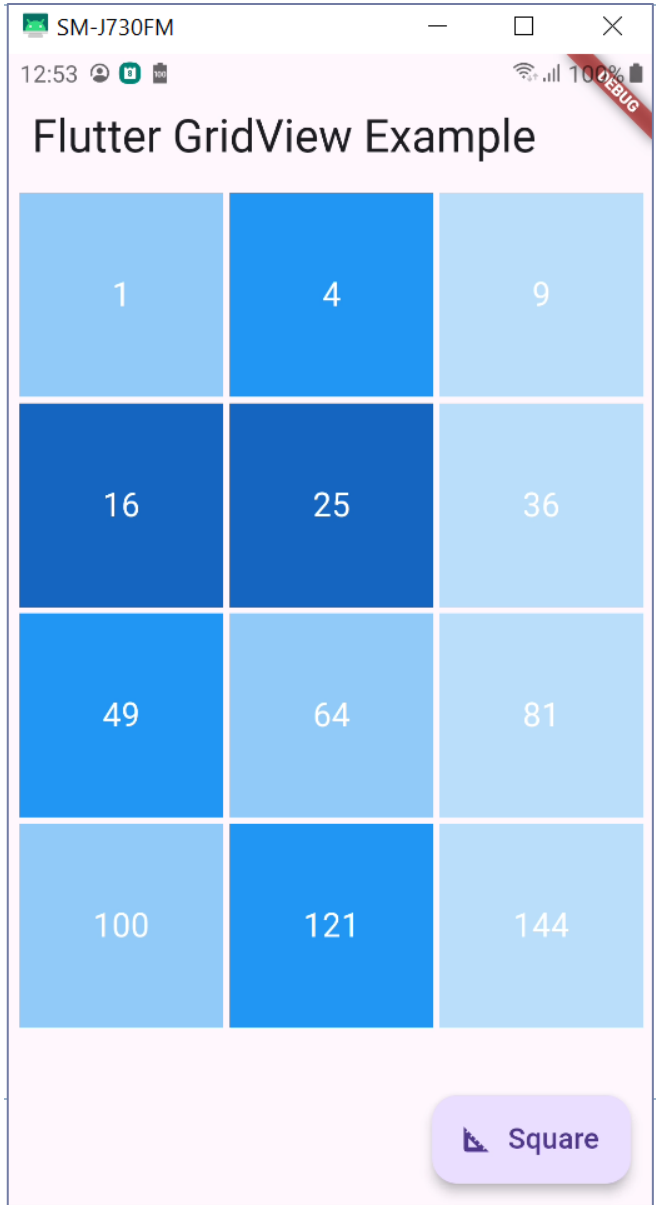


Flutter UI Widgets - GridView

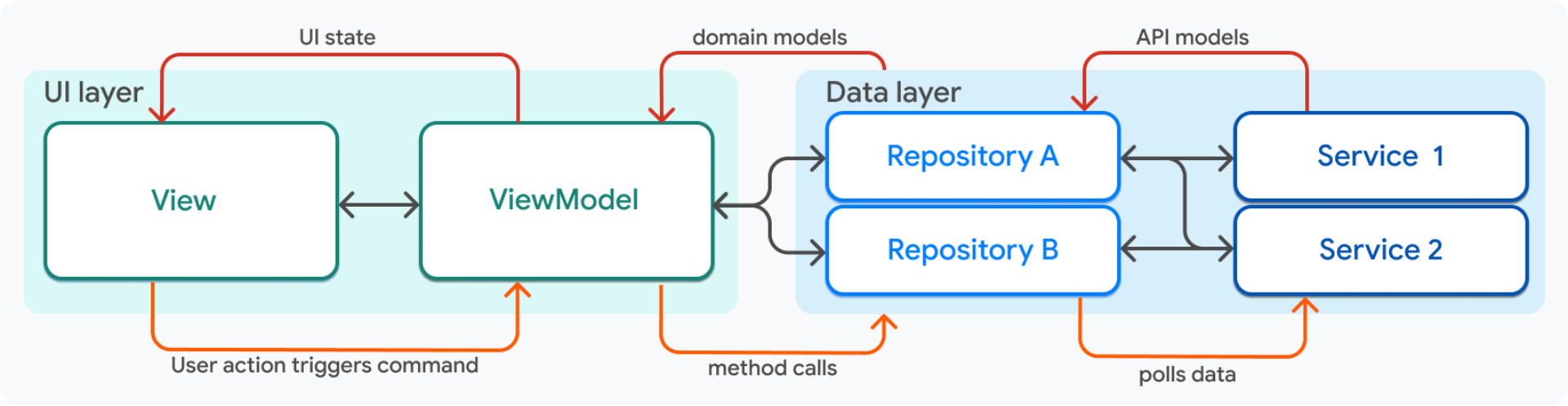
```
...
        return Container(
          color: Colors.blue[colorIndex.clamp(100, 900)],
          child: Center(
            child: Text(
              '$value',
              style: const TextStyle(color: Colors.white, fontSize: 16),
            ),
          ),
        );
      },
    ),
  ),
  floatingActionButton: FloatingActionButton.extended(
    onPressed: _squareValues,
    icon: const Icon(Icons.square_foot),
    label: const Text('Square'),
  ),
 ),
);
}}
```

побудова таблиці

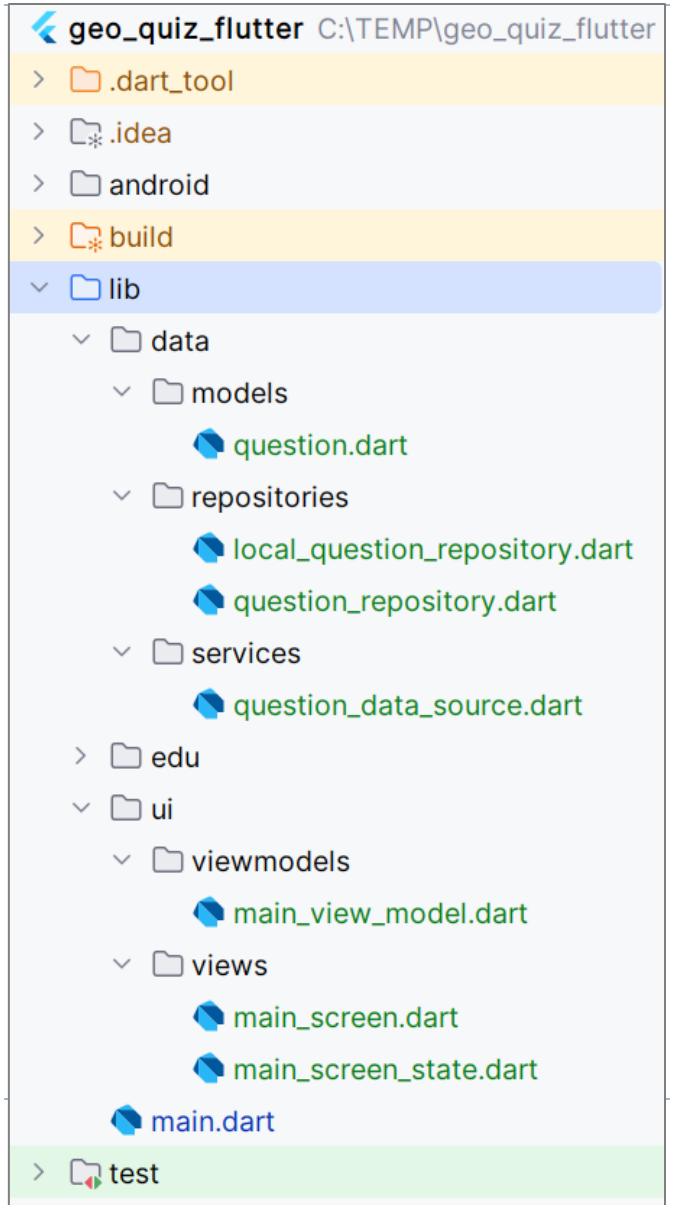
Flutter UI Widgets - GridView



Flutter - рекомендована архітектура застосунку



Flutter - рекомендована архітектура застосунку



Flutter - навчальний застосунок - додання залежності **uuid** до **pubspec.yaml**

```
name: geo_quiz_flutter

description: "A new Flutter project."

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

# The following defines the version and build number for your application.
version: 1.0.0+1

environment:
  sdk: ^3.14.0

dependencies:
  flutter:
    sdk: flutter
  uuid: ^4.5.2 # ^  - is equivalent to >=4.5.2 < 5.0.0.
  cupertino_icons: ^2.0.0
  material_ui: ^1.11.0

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^6.0.0

flutter:
  uses-material-design: true
```

додана залежність для генерації id питання

Flutter - навчальний застосунок - шар даних - модель



lib\data\models\question.dart

```
import 'package:uuid/uuid.dart';

class Question {
  final String id;
  final String text;
  final bool answer;

  Question({String? id, required this.text, required this.answer})
    : id = id ?? _generateId();

  Question copyWith({String? id, String? text, bool? answer}) {
    return Question(
      id: id ?? this.id,
      text: text ?? this.text,
      answer: answer ?? this.answer,
    );  }

  @override
  bool operator ==(Object other) =>
      identical(this, other) ||
      other is Question && runtimeType == other.runtimeType && id == other.id;

  @override
  int get hashCode => id.hashCode; }  ...
```

ініціалізатор - виконується перед конструктором, генеруючи значення id

перевизначення equals та hashcode для порівняння за значенням id

Flutter - навчальний застосунок - шар даних - модель



lib\data\models\question.dart

```
...
final Uuid _uuid = Uuid();
String _generateId() => _uuid.v4();
```

екземпляр генератора значень id

генерування значень id



Flutter - навчальний застосунок - шар даних - сервіс



lib\data\services\question_data_source.dart

```dart
import '../models/question.dart';

class QuestionDataSource {
  final List<Question> _questions = [
    Question(text: 'Канберра - це столиця Австралії', answer: true),
    Question(text: 'Тихий океан більший, аніж Атлантичний океан', answer: true),
    Question(text: 'Суецький канал поєднує Червоне море та Індійський океан',
             answer: false),
    Question(text: 'Витік річки Ніл знаходиться в Єгипті', answer: false),
    Question(text: 'Амазонка є найдовшою річкою у Америці', answer: true),
    Question(text: 'Озеро Байкал є найстарішим та найглибшим прісноводним озером',
             answer: true),
  ];

  /// Returns an unmodifiable view of the questions to prevent external mutation.
  List<Question> getQuestions() => List.unmodifiable(_questions);
}
```



Flutter - навчальний застосунок - шар даних - репозиторій (абстракція)



lib\data\repositories\question_repository.dart

```dart
import '../models/question.dart';

abstract class QuestionRepository {
  List<Question> getQuestionBank();
  Question getQuestionByIndex(int index);
  int getQuestionBankSize();
}
```



Flutter - навчальний застосунок - шар даних - репозиторій (реалізація)



lib\data\repositories\local_question_repository.dart

```dart
import '../models/question.dart';
import '../services/question_data_source.dart';
import '../repositories/question_repository.dart';

class LocalQuestionRepository implements QuestionRepository {
  final QuestionDataSource _dataSource;

  LocalQuestionRepository([QuestionDataSource? dataSource])
      : _dataSource = dataSource ?? QuestionDataSource();

  @override
  List<Question> getQuestionBank() => _dataSource.getQuestions();

  @override
  Question getQuestionByIndex(int index) {
    final list = _dataSource.getQuestions();
    return list[index];
  }

  @override
  int getQuestionBankSize() => _dataSource.getQuestions().length;
}
```



Flutter - навчальний застосунок - шар UI - View - MainScreenState

lib\ui\view\main_screen_state.dart

```
class MainScreenState {
  final int currentIndex;
  final List<Question> questionList;

  const MainScreenState({
     this.currentIndex = 0,
     List<Question> questionList = const [],
}) : questionList = List.unmodifiable(questionList);
  });

  MainScreenState copyWith({int? currentIndex, List<Question>? questionList}) {
    return MainScreenState(
      currentIndex: currentIndex ?? this.currentIndex,
      questionList: questionList ?? this.questionList,
    );
  }
}
```

Стан головного екрану, що зберігатиметься у ViewModel

Створення константи часу компіляції - незмінного пустого списку та привласнення посилання на нього властивості класу в ініціалізаторі

Стан може змінюватися тільки через копіювання

Flutter - навчальний застосунок - шар UI - ViewModel



lib\ui\viewmodels\main_view_model.dart

```
import 'package:flutter/foundation.dart';
import 'package:material_ui/material_ui.dart';
import '../views/main_screen_state.dart';
import '../../data/models/question.dart';
import '../../data/repositories/question_repository.dart';

class MainViewModel extends ChangeNotifier {
  final QuestionRepository questionRepository;
  MainScreenState _mainScreenState;
  MainScreenState get mainScreenState => _mainScreenState;
  MainViewModel({required this.questionRepository})
      : _mainScreenState = MainScreenState(
    currentIndex: 0,
    questionList: questionRepository.getQuestionBank(),
  ),
  Question getQuestionByIndex(int index) {
    return questionRepository.getQuestionByIndex(index);
  }
...
```

ChangeNotifier забезпечує просту спостережувану реалізацію, на яку може підписатися інтерфейс користувача. При зміні стану ViewModel викликає метод ChangeNotifier notifyListeners().

Стан зберігається у ViewModel



Flutter - навчальний застосунок - шар UI - ViewModel



lib\ui\viewmodels\main_view_model.dart

```
...
 String getQuestionText(int index) {
  final question = questionRepository.getQuestionByIndex(index);
  return question.text;
 }
 String onAnswerButtonClick(bool userAnswer) {
  final currentQuestion =
                    questionRepository.getQuestionByIndex(_mainScreenState.currentIndex
 );
  final correct = userAnswer == currentQuestion.answer;
  return correct ? 'Правильно' : 'Неправильно';
 }
 void onNextQuestionButtonClick() {
  final size = questionRepository.getQuestionBankSize();
  if (size == 0) return;
  _mainScreenState = _mainScreenState.copyWith(
   currentIndex: (_mainScreenState.currentIndex + 1) % size,
  );
  notifyListeners();
 }
}
```

Flutter - навчальний застосунок - шар UI - View - MainScreen



lib\ui\view\main_screen.dart

```
import 'package:material_ui/material_ui.dart';
import '../viewmodels/main_view_model.dart';

class MainScreen extends StatelessWidget {
  final MainViewModel viewModel;

  const MainScreen({super.key, required this.viewModel});

  @override
  Widget build(BuildContext context) {
    return AnimatedBuilder(
      animation: viewModel,
      builder: (context, _) {
        final questionText = viewModel.getQuestionText(
          viewModel.mainScreenState.currentIndex,
        );
        return Scaffold(
          appBar: AppBar(
            backgroundColor: Theme.of(context).colorScheme.primaryContainer,
            title: const Text('Geo Quiz'),
            centerTitle: true,
          ), ...
```

Віджет без стану

AnimatedBuilder виконує callback конструктора щоразу, коли наданий у параметрі animation Listenable(ChangeNotifierm наприклад) змінюється. Він використовується для керування оновленнями інтерфейсу користувача з анімацій або інших об'єктів Listenable без необхідності керування setState у StatefulWidget.

Flutter - навчальний застосунок - шар UI - View - MainScreen



...

```
body: SingleChildScrollView(
 child: SizedBox(
  height: MediaQuery.of(context).size.height,
  child: Center(
   child: Padding(
    padding: const EdgeInsets.symmetric(horizontal: 16.0),
    child: Column(
     mainAxisAlignment: MainAxisAlignment.center,
     crossAxisAlignment: CrossAxisAlignment.center,
     children: [
      Text(
       questionText,
       textAlign: TextAlign.center,
       style: Theme.of(context).textTheme.titleMedium,
      ),
      const SizedBox(height: 16),
```

lib\ui\view\main_screen.dart
обчислює висоту області контенту
без AppBar

...



Flutter - навчальний застосунок - шар UI - View - MainScreen



lib\ui\view\main_screen.dart

```
...
SizedBox(
  height: 80,
  width: 200,
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    crossAxisAlignment: CrossAxisAlignment.center,
    children: [
      ElevatedButton(
        onPressed: () {
          _showAnswerFeedback(context, true);
        },
        child: const Text('Так'),
      ),
      const SizedBox(width: 16),
      ElevatedButton(
        onPressed: () {
          _showAnswerFeedback(context, false);
        },
        child: const Text('Ні'),
      ),
    ],
  ),
),
const SizedBox(height: 12),
```

користувацька функція, визначена нижче

Flutter - навчальний застосунок - шар UI - View - MainScreen



lib\ui\view\main_screen.dart

```
..
            const SizedBox(height: 12),
            ElevatedButton(
              onPressed: () {
                viewModel.onNextQuestionButtonClick();
                debugPrint(
                  'MainScreen: currentIndex = ${viewModel.mainScreenState.currentIndex}',
                );
              },
              child: const Text('Далі'),
            ),          ],          ),          ),          ),          ),          ),          );      },    );  }

void _showAnswerFeedback(BuildContext context, bool userAnswer) {
  final message = viewModel.onAnswerButtonClick(userAnswer);
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      duration: const Duration(seconds: 1),
      content: Text(message, textAlign: TextAlign.center),
      behavior: SnackBarBehavior.floating,
    ),
  );  }}
```

Flutter - навчальний застосунок - головний файл програми

lib\main.dart

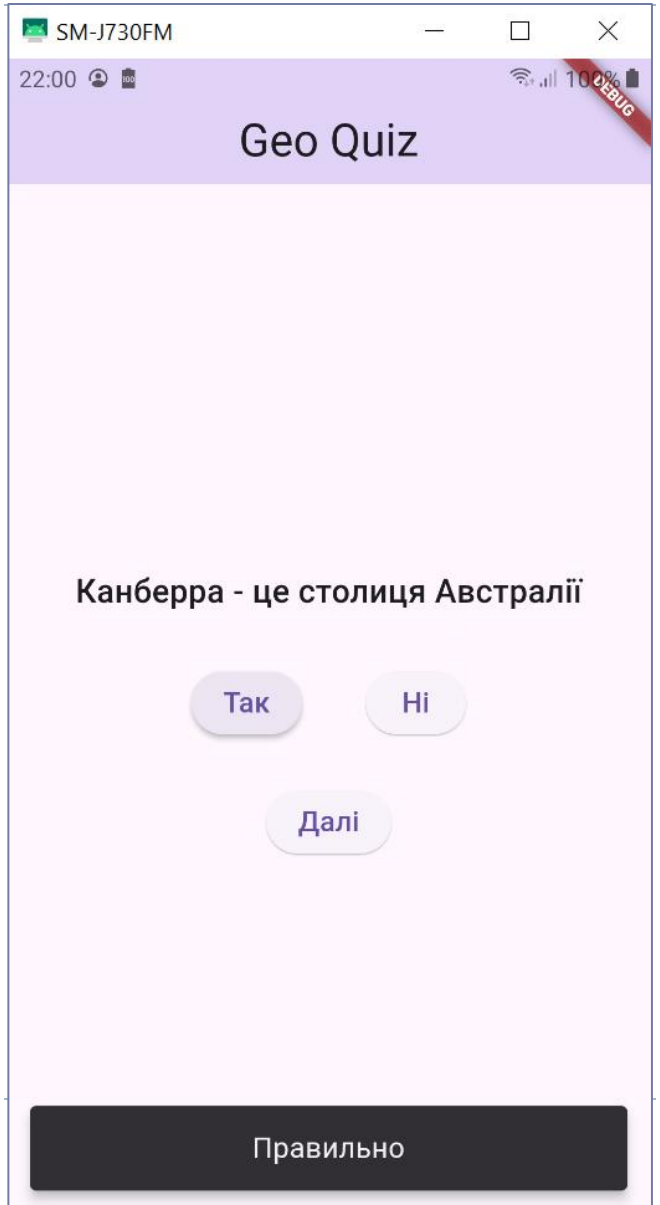
```
import 'package:material_ui/material_ui.dart’;
import 'data/repositories/local_question_repository.dart';
import 'ui/viewmodels/main_view_model.dart';
import 'ui/views/main_screen.dart';

void main() {
  final viewModel = MainViewModel(
    questionRepository: LocalQuestionRepository(),
  );
  runApp(MyApp(viewModel: viewModel));
}

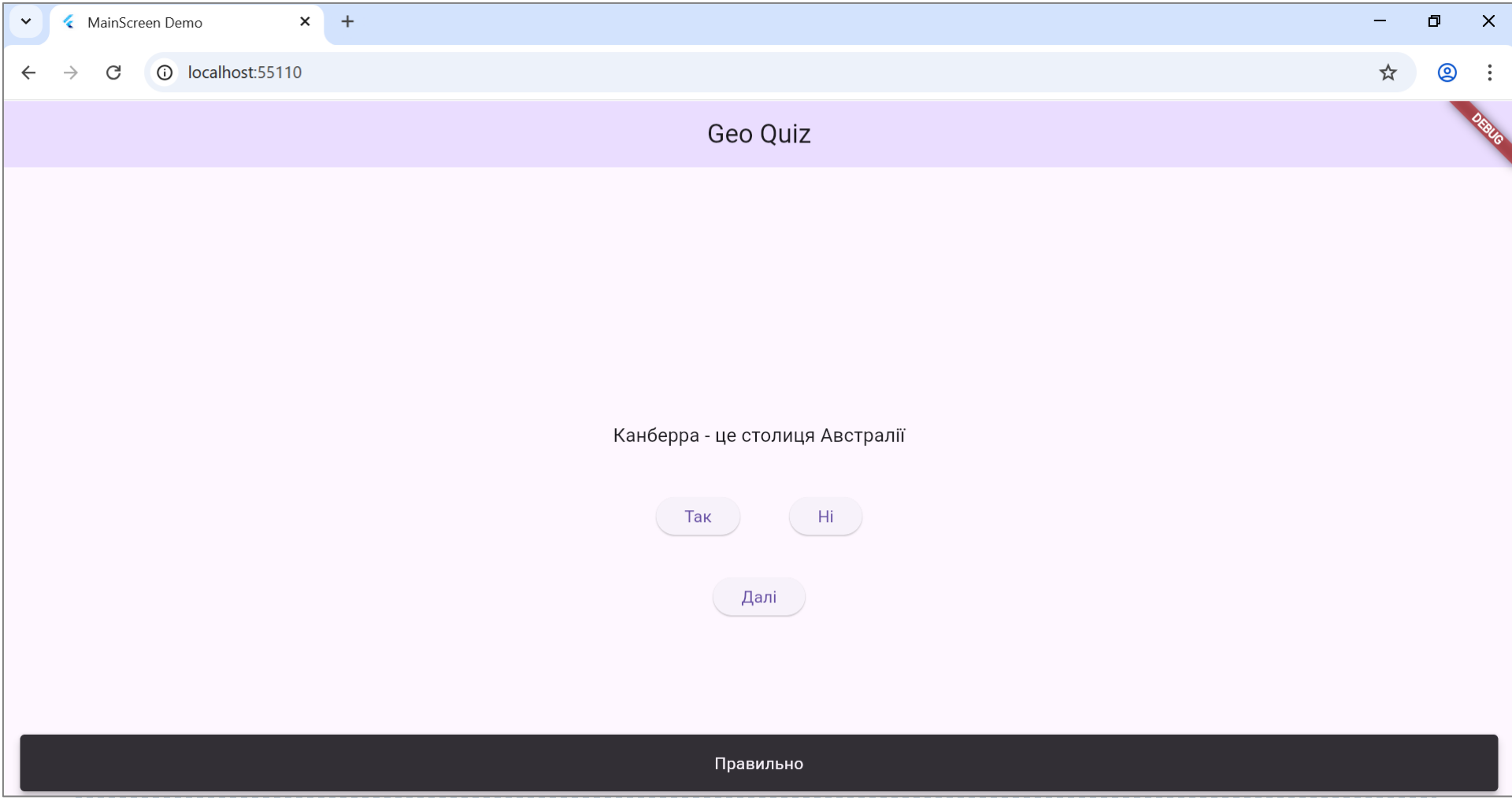
class MyApp extends StatelessWidget {
  final MainViewModel viewModel;
  const MyApp({super.key, required this.viewModel});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'MainScreen Demo',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: MainScreen(viewModel: viewModel),
    );  }}
```

Flutter - запуск навчального застосунку - Android



Flutter - запуск навчального застосунку - Chrome



Інтеграція з API

- Використання бібліотек, як-от http для роботи з мережевими запитами.
- Проста реалізація GET-запиту:

```
import 'package:material_ui/material_ui.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
...
```



Інтеграція з API

```
...
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('API Integration')),
        body: const ApiExample(),
      ),
    );
  }
}

class ApiExample extends StatefulWidget {
  const ApiExample({super.key});

  @override
  _ApiExampleState createState() => _ApiExampleState();
}
...
```



Інтеграція з API

```
 ...
class _ApiExampleState extends State<ApiExample> {
  String data = 'Loading...';

  @override
  void initState() {
    super.initState();
    fetchData();
  }

  Future<void> fetchData() async {
    final response =
        await
http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts/
1'));
    if (response.statusCode == 200) {
      setState(() {
        data = jsonDecode(response.body)['title'];
      });
...
```

Інтеграція з API

```
...
    } else {
      setState(() {
        data = 'Error loading data';
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Center(child: Text(data));
  }
}
```



Навігація між екранами

- Використання Navigator для переходу між екранами.
- Приклад реалізації:

```
import 'package:material_ui/material_ui.dart’;

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
…
```



Навігація між екранами

```
…
      routes: {
        '/': (context) => const FirstScreen(),
        '/second': (context) => const SecondScreen(),
      },
    );
  }
}

class FirstScreen extends StatelessWidget {
  const FirstScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('First Screen')),
      body: Center(
        child: ElevatedButton(
…
```

Навігація між екранами

```
...
          onPressed: () {
            Navigator.pushNamed(context, '/second');
          },
          child: const Text('Go to Second Screen'),
        ),
      ),
    );
  }
}
 ...
```



Навігація між екранами

```
...
class SecondScreen extends StatelessWidget {
  const SecondScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Screen')),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: const Text('Back to First Screen'),
        ),
      ),
    );
  }
}
```

Перспективи розвитку Flutter

- **Активна підтримка Google:** Регулярні оновлення та вдосконалення.
- **Зростаюча популярність:** Велика спільнота розробників і багатий вибір плагінів.
- **Нові можливості:**
  - Flutter для web-додатків і десктопів.
  - Інтеграція з Fuchsia OS (майбутня операційна система Google).
- **Попит на ринку:**
  - Зростання вакансій для Flutter-розробників.
  - Використання великими компаніями (Alibaba, BMW, Google Ads).
- **Гнучкість та масштабованість:**
  - Інструмент для створення застосунків будь-якої складності.
  - Підтримка різних платформ з єдиним кодом.