Міністерство освіти і науки України.

Національний університет “Львівська Політехніка”

Кафедра ЕОМ

Звіт про виконання лабораторної роботи №1
“ Розробка мобільних додатків**”**
на тему: «Flutter - Додавання інтерактивного поля вводу»

Виконав: ст. групи КІ-405

Петрович В.А.

Прийняв: Ремінний О.А.

Львів – 2025

Мета: Ознайомитися з основами розробки мобільних застосунків за допомогою фреймворку Flutter, навчитися налаштовувати середовище розробки, створювати тестовий проєкт, додавати інтерактивні елементи інтерфейсу та керувати станом застосунку за допомогою Stateful віджетів. Закріпити навички роботи з лінтером, системою контролю версій Git та автоматичною перевіркою коду.

Теоретичні відомості

Flutter — це фреймворк з відкритим вихідним кодом, розроблений компанією **Google**, який використовується для створення кросплатформних мобільних, веб та десктопних застосунків з єдиною базою коду, написаною мовою **Dart**.

Основна перевага Flutter полягає у швидкому розробленні інтерфейсу завдяки використанню **віджетів** — основних будівельних блоків інтерфейсу користувача.

---

1. Віджети (Widgets)

У Flutter усе є віджетом — текст, кнопка, контейнер, поле вводу чи навіть сам застосунок. Віджети поділяються на два основні типи:

- **StatelessWidget** — віджети без стану, які не змінюються під час роботи програми (наприклад, текст, іконка, статичний контейнер).
- **StatefulWidget** — віджети зі станом, які можуть змінювати свій вигляд або поведінку в процесі виконання (наприклад, поле вводу, лічильник, форма).

---

2. Керування станом (State Management)

Стан (State) — це інформація, яка описує поточний вигляд або поведінку віджета.
У **StatefulWidget** зміна стану здійснюється за допомогою методу **setState()**, який оновлює інтерфейс після зміни даних.
Приклад:

```
setState(() {
  counter++;
});
```

3. Інтерактивні елементи

Інтерактивні віджети дозволяють користувачеві взаємодіяти з інтерфейсом.
До них належать:

- **TextField** — поле для введення тексту;
- **ElevatedButton**, **IconButton**, **FloatingActionButton** — кнопки з різними стилями;
- **GestureDetector** — розпізнає жести користувача (тап, свайп тощо).

---

4. Лінтер (Linter)

Linter — це інструмент для автоматичного аналізу коду, який допомагає дотримуватися стилістичних та структурних правил.
У Flutter/Dart правила лінтингу задаються у файлі **analysis_options.yaml**, який визначає стандарти, помилки та попередження, що слід перевіряти під час компіляції.

---

5. Git та CI/CD

Для командної роботи і зручної перевірки завдань використовується **GitHub** та міні-пайплайн (наприклад, файл **validation.yaml**) — це елемент CI/CD, який автоматично перевіряє код після створення Pull Request. Це дозволяє контролювати якість коду, уникати помилок і підтримувати єдиний стиль проєкту.

---

6. Dart — мова програмування Flutter

Dart — об’єктно-орієнтована, типізована мова, створена Google. Вона поєднує синтаксис, схожий на Java, та можливість швидкої компіляції в нативний код, що забезпечує високу продуктивність Flutter-застосунків.

Завдання

1. **Налаштувати середовище розробки Flutter:**
   - Встановити **Flutter SDK** згідно з офіційною інструкцією.
   - Налаштувати середовище розробки (наприклад, **Visual Studio Code** або **Android Studio**).
   - Перевірити коректність встановлення за допомогою команди: flutter doctor
2. **Створити тестовий проєкт:**
   - У терміналі виконати команду:

     flutter create my_project
   - Перейти до створеного каталогу та запустити додаток командою:

     flutter run
3. **Додати інтерактивне поле вводу:**
   - Реалізувати **TextField** для введення тексту користувачем.
   - Зберігати введені дані у змінній стану (**State**) за допомогою **StatefulWidget**.
4. **Здійснити обробку введених даних:**
   - Реалізувати дію, яка змінює стан інтерфейсу в залежності від введеного тексту.
5. **Використати Stateful віджети:**
   - Використати метод **setState()** для оновлення даних у реальному часі при зміні введення користувача.
   - Забезпечити динамічну зміну елементів інтерфейсу.
6. **Налаштувати лінтер:**
   - Створити у корені проєкту файл **analysis_options.yaml**.
   - Скопіювати у нього правила лінтера з наданого репозиторію.
   - Усунути всі попередження, які показує лінтер.
7. **Налаштувати автоматичну перевірку коду:**
   - У корені проєкту створити папку **.github/workflows**.
   - Додати файл **validation.yaml** із заданими параметрами.
   - Переконатися, що після коміту код успішно проходить перевірку.
8. **Створити Pull Request (ПР):**
   - Завантажити код на GitHub у власний репозиторій.
   - Створити окрему гілку для завдання.
   - Зробити **Pull Request**, додати коментар із **ПІБ**, **посиланням на репозиторій** та **скріншотами** роботи застосунку.

Виконання завдання

1. **Налаштування середовища Flutter**

Було встановлено **Flutter SDK** згідно з офіційною документацією за посиланням https://flutter.dev/docs/get-started/install.
Для розробки використано середовище **Visual Studio Code** з установленими плагінами *Flutter* та *Dart*.

Команда перевірки встановлення:

flutter doctor

підтвердила коректність налаштування середовища.

```
Microsoft Windows [Version 10.0.26100.6899]
(c) Корпорація Майкрософт. Усі права захищені.

C:\Users\Володя>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.35.6, on Microsoft Windows [Version 10.0.26100.6899], locale uk-UA)
[√] Windows Version (Ь ©Єа®6®дв Windows 11 Pro 64-bit, 24H2, 2009)
[√] Android toolchain - develop for Android devices (Android SDK version 36.1.0)
[√] Chrome - develop for the web
[√] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.13.3)
[√] Android Studio (version 2025.1.4)
[√] Connected device (3 available)
[√] Network resources

• No issues found!
```

---

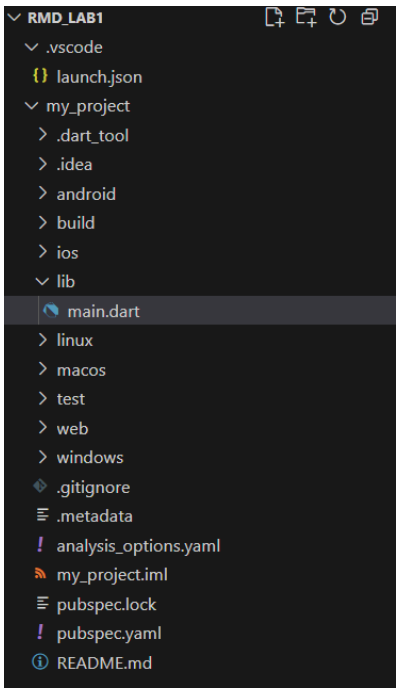
2. **Створення тестового проєкту**

У терміналі було виконано команди:

flutter create my_project

cd my_project

flutter run



3. **Додавання інтерактивного поля вводу**

Для реалізації завдання у файлі **lib/main.dart** до основного віджета було додано поле **TextField** для введення тексту користувачем. Використано **StatefulWidget**, який дозволяє зберігати введене значення у змінній стану та реагувати на зміни вмісту поля.

```
TextField(
              controller: _controller,
              onChanged: _onInput,
              style: const TextStyle(color: Colors.white, fontSize: 18),
              decoration: InputDecoration(
                filled: true,
                fillColor: Colors.grey[850],
                hintText: 'Введи: fire, aqua, calm, storm, party, sleep...',
                hintStyle: const TextStyle(color: Colors.white70),
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(12),
                  borderSide: const BorderSide(color: Colors.white),
                ),
              ),
            ),
```

---

4. **Реалізація логіки обробки введених даних**

Було створено лічильник counter, значення якого змінюється в залежності від введеного тексту:

```
void _onInput(String value) {
    _effectTimer?.cancel();
    _symbolTimer?.cancel();

    setState(() {
      switch (value.toLowerCase()) {
        case 'fire':
          _appBarTitle = '🔥 Fire Mood';
          _appBarColor = Colors.red.shade700;
          _currentSymbols = ['⋆', '✦', '✧', '🔥'];
          _startFireEffect();
          break;
        case 'aqua':
          _appBarTitle = '💧 Aqua Mood';
          _appBarColor = Colors.cyan.shade700;
          _currentSymbols = ['💧', '❋', '⍰', '•'];
          _startAquaEffect();
          break;
..........
```

5. **Оновлення стану застосунку**

Для оновлення інтерфейсу використано метод:

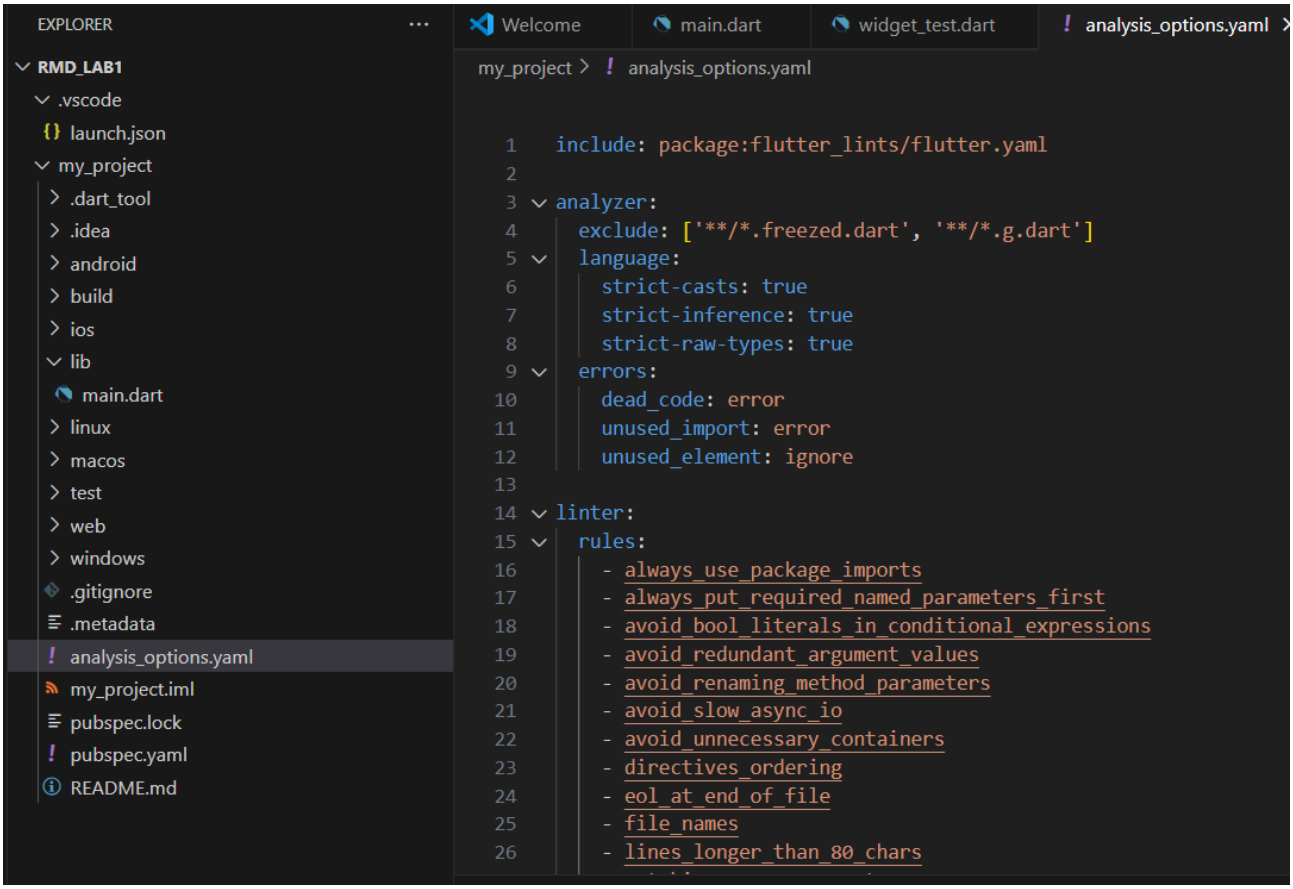
```
setState(() { … }
```

Це дозволяє динамічно змінювати вигляд застосунку без його перезапуску.

---

6. **Налаштування лінтера**

У кореневій директорії створено файл **analysis_options.yaml**.



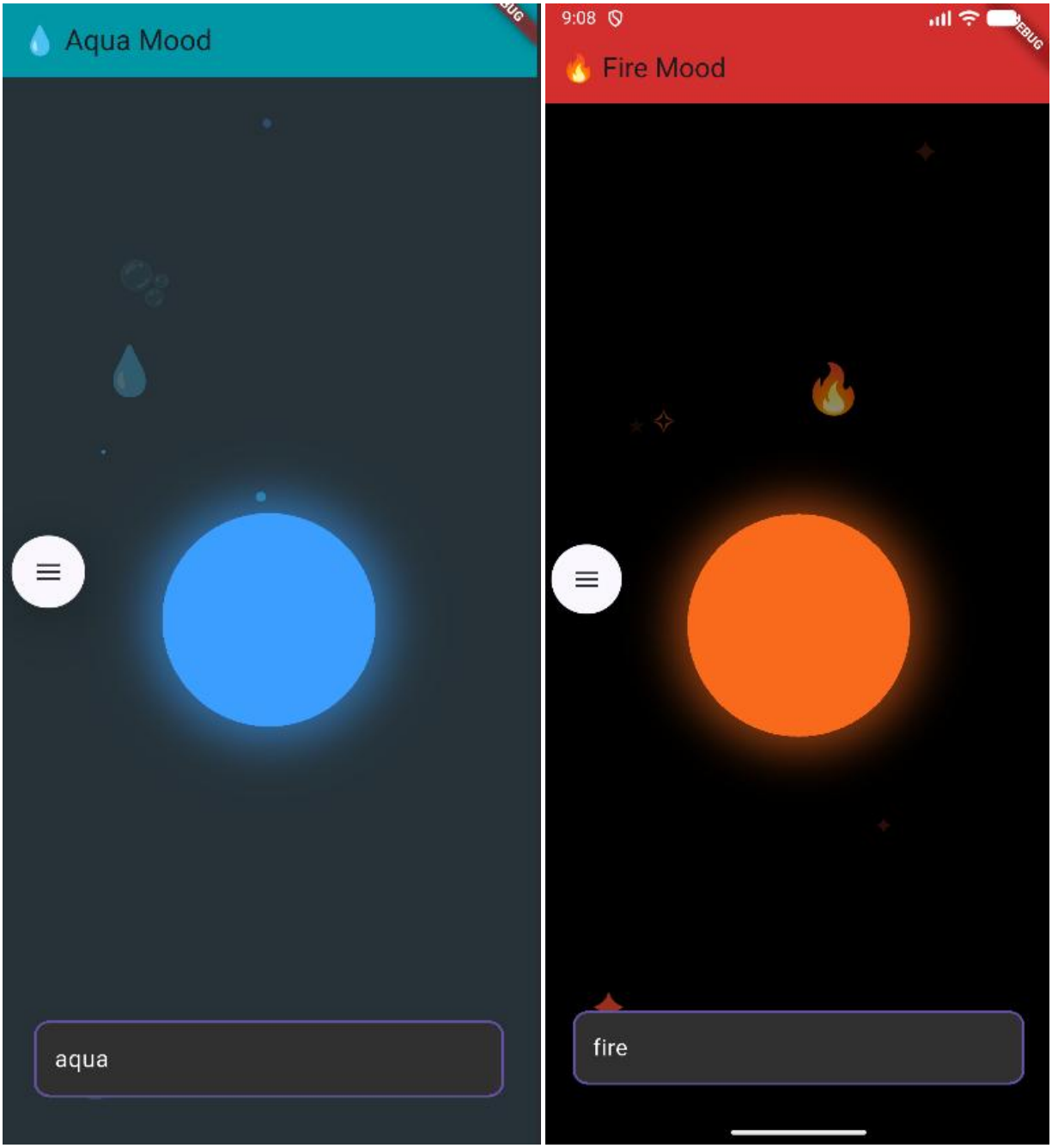
У нього було вставлено правила перевірки коду з наданого репозиторію. Після запуску перевірки усі попередження лінтера були виправлені.

---

8. **Публікація проєкту**

Після завершення роботи код було завантажено у власний репозиторій GitHub. Створено окрему гілку для лабораторної, зроблено **Pull Request**, додано коментар із ПІБ, посиланням на док і скріншотами роботи застосунку.

Скріншоти роботи програми



Висновок: У ході виконання лабораторної роботи було ознайомлено з основами розробки мобільних застосунків на фреймворку Flutter та мовою програмування Dart.

Було виконано налаштування середовища розробки, створено тестовий проєкт і реалізовано інтерактивний інтерфейс із полем введення даних та динамічним оновленням стану за допомогою Stateful віджетів.