Методические указания к Практической работе №2
по дисциплине «Разработка кроссплатформенных мобильных приложений»
Тема работы: Виджеты. Дерево виджетов. Stateless Widget

План практической работы:

- Знакомство с основным перечнем часто применяемых виджетов: Scaffold, AppBar, Column, Row, Center, Divider, Container, SizedBox, Padding, Text, Icon
- Знакомство со Stateless Widget
- Выполнение практической работы №2 в соответствии с заданием и выбранной темой

Последовательность выполнения практической работы:

1. Знакомство с основным перечнем часто применяемых виджетов

Понятие Widget

Основой интерфейса во Flutter являются объекты класса Widget. Основная идея заключается в том, что пользовательский интерфейс строится из виджетов. Виджеты описывают, как должно выглядеть их представление с учетом их текущей конфигурации и состояния. Когда состояние виджета изменяется, виджет перестраивает свое описание, которое фреймворк сравнивает с предыдущим описанием, чтобы определить минимальные изменения, необходимые в базовом дереве рендеринга для перехода из одного состояния в другое.

Минимальное приложение Flutter вызывает runApp() функцию с виджетом MyApp. Функция runApp() берет данный виджет и делает его корнем дерева виджетов.

MyApp является основным виджетом, который будет содержать всю верстку приложения. Основной функцией для отрисовки наполнения собственных виджетов является build. Данная функция должна возвращать класс-наследник класса Widget. В классе MyApp данная функция возвращает экземпляр виджета MaterialApp, который позволяет задавать темы для управления стилем приложения при помощи свойства theme, а также имеет параметр home для указания центрального виджета, который будет отрисовываться при запуске приложения.

При написании приложения обычно создаются новые виджеты, которые являются подклассами либо, StatelessWidget либо StatefulWidget, в зависимости от того, управляет ли виджет каким-либо состоянием. Основная задача виджета реализовать build() функцию, которая описывает виджет с точки зрения других виджетов более низкого уровня. Фреймворк строит эти виджеты по очереди, пока процесс не достигнет нижнего предела в виджетах, представляющих базовый элемент RenderObject, который вычисляет и описывает геометрию виджета.

Основные виджеты

Flutter поставляется с набором мощных базовых виджетов, из которых обычно используются следующие:

1. **Scaffold**

   Базовый каркас экрана приложения.
   Пример:

```
Scaffold(
  appBar: AppBar(title: Text('Главная')),
  body: Center(child: Text('Привет!')),
)
```

2. **AppBar**

   Верхняя панель приложения (заголовок, кнопки, иконки).
   Обычно используется внутри `Scaffold`.
   Пример:

```
AppBar(
  title: Text('Заголовок'),
)
```

3. **Column**

   Располагает дочерние виджеты вертикально (один под другим).

   Пример:

```
Column(
  children: [
    Text('Верх'),
    Text('Низ'),
  ],
)
```

4. **Row**

   Располагает дочерние виджеты горизонтально (в ряд).

   Пример:

```
Row(
  children: [
    Icon(Icons.star),
    Text('Избранное'),
  ],
)
```

5. **SizedBox**

   Виджет для задания размера или отступа.

   Если задать `width/height` → фиксированный размер.

   Если без параметров, можно использовать как пустое пространство.

   Пример:

```
SizedBox(height: 20) // отступ по вертикали
```

6. **Divider**

   Горизонтальная линия-разделитель.

   Часто используется в списках.

   Пример:

```
Divider(
  color: Colors.grey,
  thickness: 2,
)
```

7. **Padding**

   Добавляет внутренние отступы вокруг дочернего виджета.

   Можно задать равномерный (`EdgeInsets.all`) или разный (`EdgeInsets.only`, `EdgeInsets.symmetric`).

```
Padding(
  padding: EdgeInsets.all(16),
  child: Text('Текст с отступом'),
)
```

8. **Text**

   Отображает текст на экране.

   Можно менять стиль (шрифт, цвет, размер).

   Пример:

```
Text(
  'Привет, Flutter!',
  style: TextStyle(fontSize: 20, color: Colors.blue),
)
```

8. **Icon**

   Виджет для отображения иконки (например, из набора Material Icons).

   Найти иконки можно на сайте Google Fonts.

   Пример:

```
Icon(Icons.home, size: 30, color: Colors.green)
```

Дерево виджетов (Widget Tree)

Во Flutter всё — это виджеты: текст, иконки, кнопки, отступы, колонки, даже сам экран (`Scaffold`). Когда мы создаем интерфейс, Flutter выстраивает их в иерархию, которая называется деревом виджетов:

- У каждого виджета есть родитель (тот, внутри кого он находится).
- И могут быть дочерние элементы (те, что он содержит).

– В итоге весь интерфейс приложения представляется как одно большое дерево, где корень — это runApp(MyApp()), а дальше идут вложенные виджеты.

Дерево виджетов помогает видеть структуру интерфейса, позволяет удобно отлаживать приложение (Flutter Inspector в DevTools показывает это дерево), помогает правильно группировать виджеты и задавать им отступы, размеры, выравнивание.

2. StatelessWidget

StatelessWidget — это виджет во Flutter, который не имеет собственного состояния (**.**

– Он строится один раз при создании.
– Его внешний вид и данные не меняются со временем, пока не пересоздастся сам объект.
– Если нужно отобразить что-то постоянное (например, текст, иконку, кнопку), обычно используют StatelessWidget.

Когда использовать StatelessWidget

– Отображение статических элементов: иконки, заголовки, кнопки, картинки.
– Если логика и данные приходят извне и не меняются внутри самого виджета.
– Для простых компонентов интерфейса.

Во Flutter все StatelessWidget наследуются от класса StatelessWidget, который, в свою очередь, является наследником Widget**.**

Пример приложения:

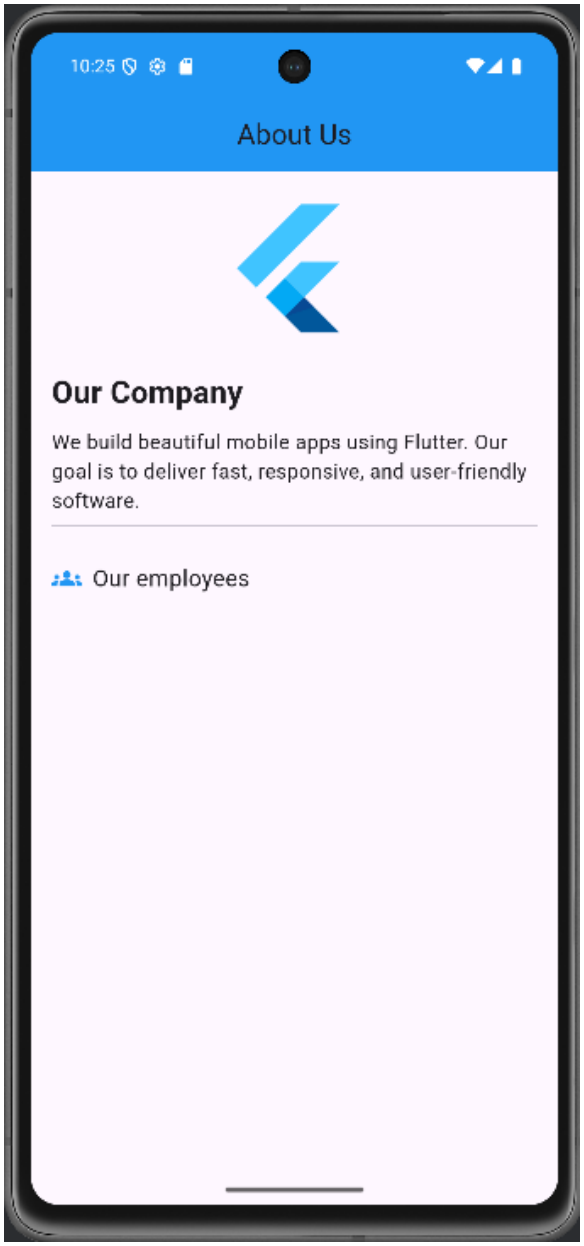

Рисунок 1 – Приложение About Us

Программный код приложения:

```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: AboutPage());
  }
}

class AboutPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('About Us'),
        centerTitle: true,
        backgroundColor: Colors.blue,
      ), //AppBar
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Center(
              child: Image.network(
                'https://img.icons8.com/color/256/flutter.png',
                width: 120,
                height: 120,
              ), //Image.network
            ), //Center
            SizedBox(height: 20),
            Text(
              'Our Company',
              style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
            ), //Text
            SizedBox(height: 10),
            Text(
              'We build beautiful mobile apps using Flutter. '
              'Our goal is to deliver fast, responsive, and user-friendly
software.',
              style: TextStyle(fontSize: 16),
            ), //Text
            Divider(),
            SizedBox(height: 20),
            Row(
              children: [
                Icon(Icons.groups, color: Colors.blue), // иконка слева
                SizedBox(width: 8), // отступ между иконкой и текстом
                Text('Our employees', style: TextStyle(fontSize: 18)),
              ],
            ), //Row
          ],
        ), //Column
      ), //Padding
    ); //Scaffold
  }
}
```

Рисунок 2 – Дерево виджетов приложения

3. Задание на практическую работу

1) Выбрать тему (предметную область (ПО)) из предложенных ниже или свою и согласовать ее с преподавателем
2) Создать приложение для выбранной предметной области согласно примерному макету.
3) В отчет по практической работе внести программный код приложения, скриншот дерева виджетов приложения, скриншот результата.

Темы (предметные области (ПО)

1. Языки программирования
2. СУБД
3. Мобильные операционные системы
4. Форматы данных
5. Типы тестирования
6. Сетевые протоколы
7. Облачные хранилища
8. Алгоритмы сортировки
9. Форматы изображений
10. Почтовые протоколы
11. Кроссплатформенные мобильные фреймворки
12. BI-инструменты
13. Социальные сети
14. Интернет-магазины
15. Графические редакторы
16. Музыкальные стриминги
17. Игровые движки
18. Антивирусы
19. Онлайн-кинотеатры
20. Процессоры

21. Видеокарты
22. Виды компьютерных игр
23. Форматы видеофайлов
24. Форматы аудиофайлов
25. Компьютерные сети
26. Виды программных приложений
27. Носители информации
28. Электронные устройства
29. Прикладное программное обеспечение
30. Системное программное обеспечение
31. Архиваторы
32. Архитектурные паттерны

Примерный макет приложения